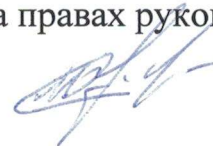


Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет»

На правах рукописи



Умницын Михаил Юрьевич

**МЕТОД АНАЛИЗА ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННОЙ
СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ ПОЛУНАТУРНОГО
МОДЕЛИРОВАНИЯ**

05.13.01 – Системный анализ, управление и обработка информации
(информационные технологии и промышленность)

05.13.19 – Методы и системы защиты информации, информационная безопасность

ДИССЕРТАЦИЯ

на соискание ученой степени

кандидата технических наук

Научный руководитель:

доктор технических наук, доцент

Садовникова Наталья Петровна

Научный консультант:

кандидат технических наук

Никишова Арина Валерьевна

Волгоград – 2019

ОГЛАВЛЕНИЕ

| | |
|---|----|
| ВВЕДЕНИЕ | 4 |
| 1 АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К АНАЛИЗУ ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ | 12 |
| 1.1 Анализ защищенности информационных систем как этап управления ИБ ... | 12 |
| 1.2 Подходы к моделированию информационных систем | 15 |
| 1.3 Подходы к моделированию злоумышленных воздействий..... | 21 |
| 1.3.1 Методы графологического построения сценариев злоумышленников..... | 24 |
| 1.3.2 Метод верификации моделей..... | 31 |
| 1.4 Постановка задачи исследования | 32 |
| 1.5 Вывод по первой главе | 33 |
| 2 МЕТОД АНАЛИЗА ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ..... | 36 |
| 2.1 Формализованное описание метода анализа защищенности информационной системы..... | 36 |
| 2.2 Спецификация модели информационной системы | 36 |
| 2.2.1 Рекомендации по упрощению модели информационной системы..... | 42 |
| 2.2.2 Методика сбора сведений об информационной системе..... | 43 |
| 2.3 Спецификации модели злоумышленника..... | 47 |
| 2.3.1 Структура библиотеки сценариев злоумышленников | 59 |
| 2.3.2 Выбор инструментального средства верификации модели..... | 64 |
| 2.4 Оценка защищенности информационной системы | 68 |
| 2.5 Вывод по второй главе..... | 69 |
| 3 МНОГОАГЕНТНАЯ СИСТЕМА ДЛЯ АНАЛИЗА ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ | 70 |
| 3.1 Многоагентная модель для моделирования злоумышленных воздействий на информационную систему | 70 |
| 3.1.1 Программный каркас Procedural Reasoning System | 76 |
| 3.1.2 Контролирующий механизм Belief-Desire-Intention..... | 80 |

| | | |
|-------|--|-----|
| 3.1.3 | Выбор среды разработки многоагентной системы..... | 90 |
| 3.2 | Архитектура многоагентной системы анализа защищенности ИС | 93 |
| 3.2.1 | Архитектура подсистемы построения модели злоумышленника | 97 |
| 3.2.2 | Модуль имитации злоумышленных воздействий..... | 101 |
| 3.2.3 | Механизм взаимодействия модели Belief-Desire-Intention с информационной системой | 110 |
| 3.2.4 | Коммутатор | 112 |
| 3.2.5 | Модуль виртуальных компонентов..... | 114 |
| 3.2.6 | Шлюз связи с информационной системой | 120 |
| 3.3 | Выводы по третьей главе..... | 122 |
| 4 | РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ..... | 123 |
| 4.1 | Разработка методики проведения экспериментальных исследований..... | 123 |
| 4.2 | Описание структуры ИС для проведения экспериментальных исследований | 124 |
| 4.3 | Проведение экспериментальных исследований..... | 128 |
| 4.4 | Результаты экспериментальных исследований..... | 144 |
| | ЗАКЛЮЧЕНИЕ | 147 |
| | СПИСОК ЛИТЕРАТУРЫ..... | 149 |
| | Приложение А | 165 |
| | Приложение Б | 169 |
| | Приложение В..... | 171 |

ВВЕДЕНИЕ

Актуальность темы исследования. Увеличение масштабов информационных систем (ИС, здесь и далее под информационной системой понимается совокупность содержащейся в базах данных информации и обеспечивающих ее обработку информационных технологий и технических средств [1]) с динамическим изменением их топологии, иерархии входящих в них элементов, повышение сложности множества информационных проектов, увеличение доли распределенных ИС, как и в корпоративном, так и в общедоступном секторе, многие другие факторы увеличивают число потенциальных уязвимостей и возможных атак на самые различные элементы ИС. В результате совокупность данных факторов оказывают крайне негативное влияние на эффективность и актуальность существующих моделей анализа защищенности, требуя пересмотра существующих подходов, инновационных исследований и разработки более эффективных, отвечающих современным реалиям, технологий и методов анализа и оценки защищенности, способных в полной мере охватывать как архитектуру современных ИС, так и процессы, протекающие в них [2-4].

Современные атаки сильно отличаются от того, какими они были десять лет назад. Сегодня на смену одиночкам, зачастую самоучкам, пришли группы подготовленных профессионалов, выполняющие многошаговые скоординированные распределенные атаки, со сложной организацией, сложным процессом их реализации, множеством целей атаки [5-6].

На сегодняшний день сложилась ситуация, когда существующие методы оценки защищенности по сути дела не способны отразить реальные показатели защищенности системы, однако от данных, казалось неадекватных, моделей до сих пор не отказываются, предлагая расширить возможности до необходимых критериев путем добавления недостающей функциональности или исправления серьезных ошибок без должной переработки системы, что затрудняет дальнейшее развитие.

Некоторые из таких методов со временем становятся настолько сложными, что их использование становится очень затруднительным, а то и вовсе они остаются лишь на бумаге, не имея какой-либо возможности в реализации.

Сложная зависимость между различными факторами в таких моделях, обычно, представляется каким-либо вероятностным процессом. При этом во внимание не принимается тот факт, что даже незначительное изменение связи между какими-либо факторами может привести к кардинальному изменению закона распределения этого процесса. Другими словами, такая модель в любой момент может оказаться неадекватной, и к результатам её работы стоит относиться с определённой долей недоверия [7-11].

Другим краеугольным камнем является то, что множество методов оценки защищенности систем не обладают достаточно формализованным описанием моделируемых процессов и механизмов, структур данных, их представляющих, что приводит к невозможности их использования для целого класса современных систем [12].

При проведении оценки защищённости информационной системы от того или иного вида угроз безопасности информации может возникнуть необходимость экспериментального подтверждения полученных расчётных результатов. Однако реализация эксперимента, в котором атаки реальны, а не заменены модельными, сопряжена со значительными финансовыми затратами и может привести к нарушению правильной работы информационной системы [13]. Необходимо минимизировать возможные отказы элементов ИС, приводящие к снижению производительности, потере данных, нарушению бизнес-процессов, правовых обязательств и пр. Вместе с тем необходимо обеспечить требуемую полноту и достоверность полученных в результате анализа оценок.

Поскольку на данном этапе имитационное моделирование (в т.ч. системная динамика, дискретно-событийное моделирование) не способно полностью заменить натурный эксперимент, предлагается совместить модельную и натурную методики, классифицировав такой подход как полунатурный.

Многоагентный подход к моделированию, активно развивающийся в последнее время, может быть выходом из сложившейся ситуации [15-16]. Поскольку в настоящее время такие модели находятся у самых истоков своего развития и не всегда способны адекватно имитировать тот или иной аспект информационного противостояния, предлагается совместить модельную и натурную методики, классифицировав такой подход как полунатурный. Его основу составляет многоагентное моделирование, а те компоненты модели, свойства которых изучены плохо, либо неадекватно моделируются, заменяются реальными объектами.

Степень разработанности темы. Проблеме моделирования ИС для анализа надежности функционирования и защищенности посвящены работы Волковой В.Н., Козлова В.Н., Ажмухамедова И.М., Левиной Т.М., Макуновой А.А., Путнина В.И., Зеленкова Ю.А., Иванова А.К. и др. В работах Котенко И.В., Cohen P.R., Dharmalingam J.M. и др. представлены подходы к анализу защищенности на основе многоагентного моделирования.

Процесс анализа защищенности ИС исследуются в работах Липатникова В.А., Машкиной И.В., Оладько В.С., Баранова Е.К., Тищенко Е.Н., Арькова П.А. и др. Основным недостатком существующих подходов анализа защищенности является высокая вероятность отказов компонентов ИС при инструментальном анализе, высокая сложность, снижение полноты при экспертном анализе защищенности. В работах Пестерева П.В., Леоновой Н.Л., Garro A., Аксенова К.А. и др. предлагается совместить модельную и натурную методики, классифицировав такой подход как полунатурный. Подобный подход применяется этими авторами при исследованиях систем специального назначения, таких как летательные и космические аппараты. Предлагается расширить применение этого подхода и для решения задачи анализа защищенности информационных систем.

Объект исследования – информационная система.

Предмет исследования – анализ защищенности ИС в процессе управления информационной безопасностью.

Цель исследования – повышение эффективности процесса анализа защищенности информационных систем за счет применения новых подходов к моделированию данного процесса, позволяющих снизить количество отказов ИС и повысить полноту анализа уязвимостей.

Полнота анализа уязвимостей понимается как отношение числа обнаруженных уязвимостей к числу уязвимостей, полученных с помощью сканеров уязвимостей на этапе сбора исходных данных.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) Провести системный анализ процессов управления информационной безопасностью, связанных с анализом защищенности ИС.
- 2) Исследовать существующие подходы к построению моделей ИС.
- 3) Разработать метод построения модели ИС для анализа защищенности.
- 4) Разработать алгоритмическое обеспечение для моделирования состояний ИС в процессе анализа защищенности.
- 5) Разработать метод анализа защищенности ИС в процессе управления ИБ.
- 6) Разработать систему поддержки принятия решений для решения задач управления информационной безопасностью на этапе анализа защищенности.
- 7) Провести экспериментальные исследования для обоснования эффективности предложенных методов и моделей.

Научная новизна:

- 1) Метод построения модели ИС, на основе теоретико-множественного подхода, который отличается от известных универсальным способом декомпозиции ИС (п.8 паспорта специальности 05.13.01).
- 2) Специальное алгоритмическое обеспечение для системы анализа защищенности ИС на основе многоагентного моделирования, отличающееся способом распределения ролей агентов (п.5 паспорта специальности 05.13.01).
- 3) Метод анализа защищенности ИС, который в отличие от известных позволяет реализовать принцип полунатурности эксперимента (п.3 паспорта специальности 05.13.19).

4) Модель многоагентной системы для поддержки принятия решений в процессе управления информационной безопасностью, в которой реализован новый способ взаимодействия агентов злоумышленника и агентов ИС (п.9 паспорта специальности 05.13.19).

Теоретическая значимость работы

Модели и алгоритмы, предложенные для совершенствования процесса анализа защищенности ИС, развивают теоретические положения в области теоретико-множественного описания сложных систем в процессе разработки их моделей, а также позволяют выявляться закономерности изменения состояний ИС с учетом внешних воздействий.

Практическая значимость работы

Разработана система для поддержки принятия решений при решении задач управления ИБ на этапе анализа защищенности в любых организациях и предприятиях, независимо от сложности эксплуатируемых в них ИС.

Теоретические и практические результаты, полученные в рамках диссертационного исследования, внедрены в компанию ООО «Инжиниринговый центр РЕГИОНАЛЬНЫЕ СИСТЕМЫ».

Методы исследования: системный анализ, теоретико-множественный анализ, многоагентное моделирование, полунатурное моделирование, методы формальной верификации моделей, методы проектирования и реализации программных средств, методы оценки защищенности информации.

Положения, выносимые на защиту:

1) Метод построения модели ИС, позволяющий упростить процесс описание компонентов за счет применения унифицированной структуры.

2) Метод анализа защищенности ИС, позволяющий комбинировать уязвимости и за счет этого уточнять список актуальных угроз.

3) Модель многоагентной системы, позволяющая за счет полунатурности при работе с критическими ресурсами, не нарушать функционирование ИС, сохраняя полноту результатов анализа.

4) Система поддержки принятия решений для решения задач управления информационной безопасностью на этапе анализа защищенности.

Степень достоверности включенных в исследование научных положений, теоретических выводов, практических рекомендаций обусловлена корректным применением указанных методов исследования, подтверждается проведенными экспериментальными исследованиями и практическим применением результатов диссертационного исследования, что подтверждено актами о внедрении.

Апробация научных результатов. Основные результаты работы докладывались на X Международной научно-практической конференции «Информационная безопасность» (Таганрог, 2008), II, IV Региональной научно-практической конференции «Проблемы обеспечения информационной безопасности в регионе» (Волгоград, 2009, 2011), VI Межрегиональной научно-практической конференции «Проблемы модернизации региона в исследованиях молодых ученых» (Волгоград, 2010), I-VI Всероссийской научно-практической конференции «Актуальные вопросы информационной безопасности регионов в условиях глобализации информационного пространства» (Волгоград, 2012-2017), VI сессии научной школы-практикума молодых ученых и специалистов «Технологии высокопроизводительных вычислений и компьютерного моделирования: технологии eScience» (г. Санкт-Петербург, 2013), награжден дипломом I степени, II международной молодежной научной конференции YSC-2013 «High Performance Computing and Simulation» в рамках конференции ICCS-2013 (Barcelona, Spain, 2013), XIX Пленуме учебно-методического объединения по образованию в области информационной безопасности (Таганрог, 2015), Всероссийской молодежной научной школе-конференции по проблемам информационной безопасности (Волгоград, 2017), SMART-2017 и SMART-2018 (Moradabad, India, 2017, 2018), VII Всероссийской научно-практической конференции «Актуальные вопросы информационной безопасности регионов в условиях перехода России к цифровой экономике» (Волгоград, 2018).

Публикации. По теме диссертации опубликовано 14 работ, в т.ч. 4 – в реферируемых журналах, рекомендованных ВАК по специальностям, 1 – в

зарубежном издании, индексируемом в WebOfScience, 2 свидетельства о регистрации программ для ЭВМ. Без соавторов опубликовано 5 работ.

Объем и структура диссертации. Диссертация состоит из введения, четырех глав, заключения, списка литературы из 150 наименований и 3 приложений общим объемом 8 страниц. Основной текст изложен на 164 страницах, включает 13 таблиц и 24 рисунков.

В первой главе проведен системный анализ процессов управления информационной безопасностью связанных с исследованием защищенности ИС, рассмотрены методы анализа защищенности, выявлены их недостатки. Исследованы подходы к описанию моделей ИС. Выбран теоретико-множественный подход к моделированию ИС для решения задачи анализа защищенности. Проведен анализ многоагентных моделей применяемых для анализа состояний ИС. Обоснована необходимость создания нового метода анализа защищенности ИС с использованием полунатурного моделирования. Определены требования к разрабатываемым моделям и методам.

Во второй главе приведен метод анализа защищенности ИС с использованием полунатурного моделирования, включающий этапы построения модели, построения модели злоумышленника, моделирования сценариев злоумышленных воздействий с использованием многоагентного подхода, оценки защищенности ИС. Описано получение библиотеки сценариев злоумышленных воздействий на ИС за счет выполнения верификация модели ИС с учетом сформированной модели злоумышленника. Дано понятие защищенности информационной системы и определен метод получения оценки защищенности.

В третьей главе представлена разработанная модель многоагентной системы для анализа защищенности ИС. Предложен подход к обеспечению полунатурности. При функционировании системы сообщения агента могут быть перенаправлены агентам в блок виртуальных компонентов ИС, моделирующим компоненты информационной системы, или перенаправлены на реальные узлы информационной системы.

В четвертой главе приведены экспериментальные исследования разработанных моделей и методов и разработанной системы поддержки принятия решений по управлению информационной безопасностью. Анализируются результаты, полученные при проведении экспериментальных исследований. Оценивается эффективность разработанного метода и реализующей его системы.

В приложения приведены:

- Описание библиотек сценариев в виде графов (Приложение А).
- Свидетельства о государственной регистрации программы для ЭВМ (Приложение Б).
- Акт о внедрении результатов диссертационного исследования (Приложение В).

1 АНАЛИЗ СОВРЕМЕННЫХ ПОДХОДОВ К АНАЛИЗУ ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННЫХ СИСТЕМ

1.1 Анализ защищенности информационных систем как этап управления ИБ

Система защиты информации, как и сама информационная система — сложная многокомпонентная система. Поэтому действующее законодательство предписывает внедрять системы управления информационной безопасностью. [2] Процесс управления информационной безопасностью (рисунок 1) включает следующие шаги:

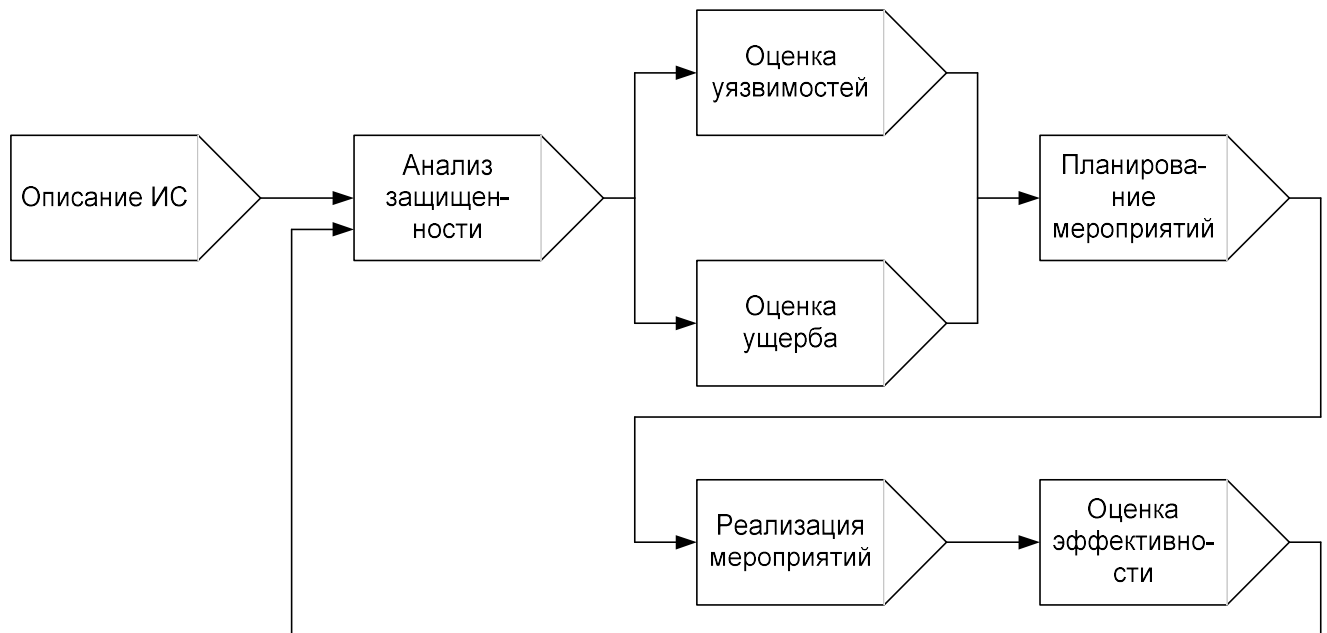


Рисунок 1 — Цикл управления информационной безопасностью

- 1) анализ и описание ИС и ее компонентов;
- 2) проведение анализа защищенности ИС;
- 3) полученные по результатам анализа защищенности оценки актуальных для ИС уязвимостей и ущерба от их реализации являются основанием для принятия решений по планированию мероприятий защиты информации в ИС;
- 4) запланированные мероприятия реализуются в системе защиты информации ИС;

5) проводится оценка эффективности принятых решений по совершенствованию системы защиты информации, которая требует вновь выполнить этап анализа защищенности ИС.

Защищенность является одним из основных показателей качества функционирования ИС, также как и производительность, надежность и т.д. [17].

Под защищенностью ИС в работе понимается способность системы противостоять несанкционированному доступу к её ресурсам, их искажению, разрушению или нарушению их функционирования посредством эксплуатации их уязвимостей.

Анализ защищенности ИС проводится для определения необходимости корректировки системы защиты информации (СЗИ) путем установки, добавления или замены некоторых средств защиты информации в связи с наличием в ИС не перекрытых уязвимостей, эксплуатация которых может позволить нарушить ее функционирование или конфиденциальность, доступность или целостность, информации, находящейся в ней. Выявленные не перекрытые уязвимости могут свидетельствовать не только об отсутствии некоторого средства защиты, но и о неправильности или неэффективности его функционирования, что может потребовать его дополнительной настройки или замены [18].

Основу анализа защищенности ИС составляет формирование перечня не перекрытых уязвимостей. Для этого применяется две группы методов:

- методы экспертной оценки;
- методы инструментальной оценки.

Методы экспертных оценок базируются на понятии профиля защиты. Профиль защиты представляет собой совокупность требований к функциям, выполняемым СЗИ и позволяющим достичь требуемого уровня защищенности. Получение оценки защищенности базируется на анализе активов организации, уязвимостей компонентов ИС и возможностей их эксплуатации. Для проведения подобного анализа строятся различные модели ИС и ее компонентов. При этом обеспечить полноту проводимого анализа очень сложно в связи со сложностью ИС,

поэтому и перечень не перекрытых уязвимостей компонентов ИС с высокой вероятностью не является полным [19].

Методы инструментальной оценки включают в себя пассивное и активное тестирование системы защиты с помощью дополнительных программных средств. При пассивном тестировании анализируются настройки операционных систем и приложений компонентов ИС на соответствие шаблонам и с применение списков проверки. При активном тестировании моделируется поведение потенциального злоумышленника. Производятся попытки эксплуатации уязвимостей. Подобные методы способны выявить большинство уязвимостей ИС. Однако могут нарушить функционирование ИС или вызвать сбой ее компонентов [20].

Тогда процесс анализа защищенности ИС можно представить в виде BPMN-диаграммы (рисунок 2).

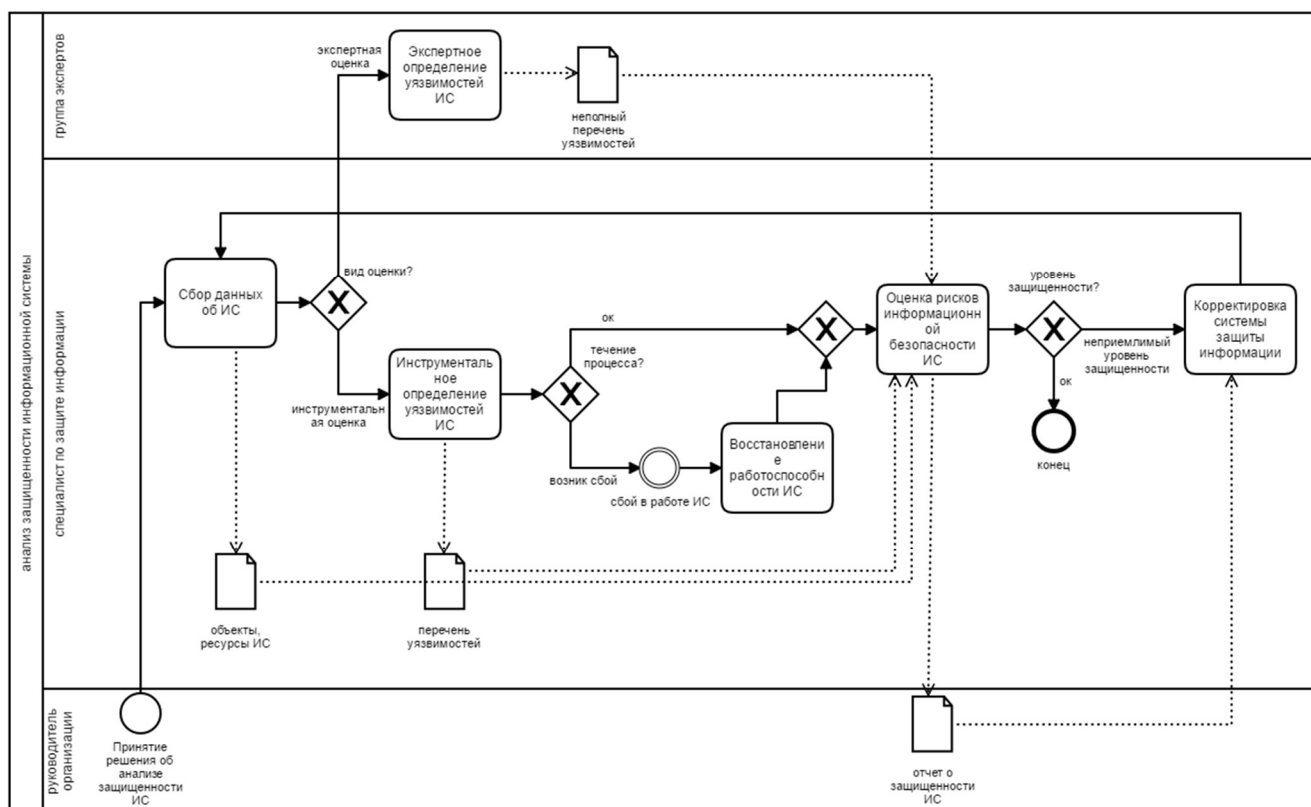


Рисунок 2 — BPMN-диаграмма процесса анализа защищенности ИС

Анализ показал следующие недостатки существующих подходов к анализу защищенности ИС:

1) Можно получить полный перечень не перекрытых уязвимостей в компонентах ИС, но при этом есть риск нарушить функционирование ИС или ее компонентов.

2) Можно получить перечень не перекрытых уязвимостей в компонентах ИС, построив ее модель и проведя ее анализ, но в связи со сложностью ИС, данный перечень будет не полный.

Для решения обозначенной проблемы предлагается объединить эти два подхода к анализу защищенности ИС. При этом используются и результаты инструментальной оценки, и результаты анализа построенной модели ИС.

1.2 Подходы к моделированию информационных систем

Абстрактное описание ИС и протекающих в ней процессов можно получить, используя различный математический аппарат. Однако по мере усложнения ИС сложность взаимодействия подсистем возрастает настолько, что лишь имитация позволяет получить удовлетворительные результаты с точки зрения решения задач анализа и управления.

Одним из основных понятий при моделировании является понятие состояния системы.

При рассмотрении ИС разделяется все пространство переменных описывающих ее, на три группы: входные переменные и воздействия, представляющие внешнюю по отношению к системе информацию; выходные переменные, характеризующие некоторые стороны функционирования системы и являющиеся ее реакцией на вход; переменные состояния, характеризующие динамическое поведение системы.

Имитационное моделирование (в широком смысле) – процесс конструирования модели реальной системы и постановка эксперимента на этой модели с целью либо понять поведение системы, либо оценить (в рамках накладываемых ограничений) различные стратегии, обеспечивающие функционирование этой системы.

Имитационное моделирование (в узком смысле) – представление динамического поведения системы посредством продвижения ее от одного состояния к другому в соответствии с хорошо известными операционными правилами (алгоритмами) [21].

В общем случае в имитационной модели ИС условно выделяются модель объекта управления, модель системы управления, модель внутренних случайных возмущений. Входы модели управляемого объекта подразделяются на контролируемые неуправляемые $X = (x_1, x_2, \dots, x_n)$, контролируемые управляемые $U = (u_1, u_2, \dots, u_m)$ и возмущения $E = (e_1, e_2, \dots, e_q)$. Система характеризуется выходными переменными $Y = (y_1, y_2, \dots, y_l)$. Управление U , в свою очередь, является выходом модели системы управления, а возмущения E – выходом модели внутренних возмущений.

Информационная система находится в одном из допустимых состояний, представляющих собой вектор $C = (c_1, c_2, \dots, c_s)$, $C \in \Omega$, где Ω - пространство возможных (не только допустимых) состояний системы. Пространство возможных состояний совсем необязательно должно быть привязано к реальной физической системе.

На ИС воздействует множество злоумышленников, которое также можно представить как некоторую сложную систему, имеющую аналогичное формальное описание. Эта сложная система может быть распределенной внешней и внутренней в отношении к ИС. Как правило, в ИС, выделяется подсистема защиты информации, цель которой не допустить перехода ИС в пространство недопустимых состояний. Исходя из данной концепции, злоумышленное воздействие можно рассматривать как действия, способные (и ставящее своей целью) перевести систему в такое состояние из пространства Ω , которое не является допустимым для ее нормального функционирования.

Воздействие, оказываемое на ИС, так или иначе, влияет на нее, а соответственно и на информацию, циркулирующую в ней. В нормативных документах, регламентирующих деятельность в области ИБ, не регулируется

понятие «злоумышленного воздействия». Но исходя из того, что «злоумышленное воздействие на ИС» эквивалентно «несанкционированному воздействию на информацию» [22] в контексте преднамеренного нарушения, злоумышленное воздействие в данной работе определяется как преднамеренное воздействие на защищаемую информацию с нарушением установленных прав и (или) правил доступа, приводящее к утечке, искажению, подделке, уничтожению, блокированию доступа к информации, а также к утрате, уничтожению или сбою функционирования носителя информации.

Имитационное моделирование позволяет заранее выбрать эффективные алгоритмы функционирования ИС и множества злоумышленников, спрогнозировать ход процессов функционирования, получать новые данные о ней и протекающих в ней процессах, проанализировать переходы в недопустимые состояния. В то же время в имитационной модели в полной мере реализуются алгоритмы управления и логика управления.

Выделяют два направления имитационного исследования сложных систем: натурные испытания и моделирование [23].

Натурные испытания подразделяются на научный эксперимент, комплексные испытания, производственный эксперимент и подразумевает проведение исследования на реальном объекте или его части в нормальном режиме с привлечением инструментальных средств, с последующей обработкой результатов эксперимента, получения обобщенных характеристик и определение границ устойчивости процессов.

Моделирование отличается от натурных испытаний тем, что исследование проводится на установках, которые сохраняют природу явлений и обладают физическим подобием. Исследование заключается в оценке поведения некоторых характеристик при создаваемых возмущениях внешней среды. Также, помимо привлечения моделей вместо реальных компонент, оно может протекать в «псевдореальном» масштабе времени.

Объединение данных двух подходов называется полунатурным моделированием.

Если имитационная модель ИС включает в себя систему управления в полном объеме, то имеет место полунатурное моделирование, когда команды системы управления поступают на модель ИС вместо непосредственного управления элементами ИС, инициируя процесс моделирования, а сообщения, вырабатываемые в модели, возвращаются в систему управления. Данный подход можно интерпретировать как моделирование со слоем абстракции над ИС. Такой подход позволяет не допустить реализации разрушающего воздействия на исследуемую систему.

Включение в модель системы управления ИС в полном объеме, при корректности ее реализации в самой ИС, позволяет гарантировать выполнение закона необходимости разнообразия Эшби. Выполнение данного требования гарантирует целостность подсистем сложной системы [24].

Полунатурное моделирование (полунатурные испытания) – это комплекс мероприятий, который предназначен для исследования поведения объекта в условиях близких к реальным. При полунатурном моделировании одна часть объекта представлена реальными физическими элементами, а другая часть элементов – их имитационными моделями [23].

Данный подход к моделированию сложных систем позволяет повысить адекватность результатов за счет приближения эксперимента к условиям, близким к исследованию на реальной системе.

Согласно [21, 25], среди подходов к имитационному моделированию выделяют:

- системную динамику;
- дискретно-событийное (процессное) моделирование;
- агентное моделирование.

Агентное моделирование используется для исследования децентрализованных систем. Данные модели применяются для получения представления о глобальных правилах и законах функционирования системы, общем поведении системы, исходя из предположений об индивидуальном, частном

поведении ее отдельных активных объектов и взаимодействии этих объектов в системе.

Дискретно-событийное моделирование – подход к моделированию, предлагающий абстрагироваться от непрерывной природы событий и рассматривать только основные события моделируемой системы.

Первые два вида подхода оперируют в основном с дискретными моделями.

Системная динамика – парадигма моделирования, где модель строится на основе причинных связей и глобальных влияний одних параметров на другие во времени. По сути, такой вид моделирования более всех других парадигм помогает понять суть происходящего выявления причинно-следственных связей между объектами и явлениями. Приоритетно оперирует с вероятностными моделями.

По отношению к уровням абстракции моделей (высокий (макроуровень, стратегический уровень); средний (мезоуровень, тактический уровень); низкий (микроуровень, оперативный уровень)) данные подходы классифицируются следующим образом. Системная динамика предполагает наивысший уровень абстракции. В данном подходе индивидуальные объекты заменяются их классами. Дискретно-событийное моделирование работает в низком и среднем диапазоне абстракции моделируемой системы. Агентное моделирование может применяться практически на любом уровне и в любых масштабах.

Одним из достоинств агентного подхода является то, что он может агрегировать отдельные черты подходов дискретно-событийного моделирования и системной динамики. Все его действия, изменения его состояний, поведение и сценарий его работы можно выразить логическими и математическими моделями.

Современные ИС требуют распределенной системы управления, в которой существует большое число локальных подсистем управления, зачастую принимающих самостоятельные решения на основе знаний и механизмов логического вывода. Такие подсистемы образуют некоторое сообщество, объединяющее их общими целями и использующее общее множество ограниченных ресурсов для достижения этих целей. Исходя из того, что множество злоумышленников также является сложной системой, то все это способствует

развитию многоагентных систем, как перспективного подхода имитационного моделирования.

Многоагентные системы (МАС) — это системы, состоящие из множества агентов, которые потенциально могут взаимодействовать друг с другом [26]. Агенты способны действовать в среде, которая, в свою очередь влияет на их поведение [23].

Достоинства многоагентного подхода в области противоборства злоумышленников и системы безопасности подробно представлены в [27, 28]. Данный подход исследует системы, состоящие из множества объектов (в том числе децентрализованные системы); позволяет получить представление об общем поведении системы, исходя из предположений об индивидуальном поведении её отдельных активных объектов.

Агентный подход позволяет моделировать многошаговые распределенные воздействия групп злоумышленников [29] на ИС. При этом каждый злоумышленник в команде представляет отдельную сущность – агента, – или их совокупность, способными к коммуникации, тем самым позволяя произвести разнородное деление по уровню знаний, обязанностям в команде, эмоциональному и физическому состоянию [30, 31]. Выделение интеллектуальных способностей позволяет говорить об их обучаемости, т.е. способности логического вывода новых знаний и следствий.

В [32] выделяют четыре основных типа злоумышленных воздействий, ранжированных по количеству злоумышленников и атакуемых объектов. Многоагентный подход, позволяя моделировать скоординированные злоумышленные воздействия любого типа, является весьма гибким инструментом для представления злоумышленных воздействий, безотносительно их природы. Также одним из наиболее важных достоинств многоагентного подхода является то, что поведение отдельного агента зачастую определяется простой дискретной моделью.

1.3 Подходы к моделированию злоумышленных воздействий

Любая система обладает компонентами и свойствами, причем эти свойства зачастую могут определяться внешней средой. Одним из краеугольных камней моделирования системы является определение ее состава – совокупности образующих систему элементов и связей между ними. Все структурные схемы имеют нечто общее, и это побуждает рассматривать их как особый объект математических исследований [33]. При абстрагировании от содержательной стороны структуры, оставляя только общие элементы ее описание можно свести к графу, где вершины – это элементы произвольной природы, а ребра (дуги, потоки) – связи между ними, или процессы их трансформации. Данный подход позволяет отразить различия между элементами или связями, наделяя их любым свойством. Графы могут изображать любые структуры (если не накладывать ограничений на пересекаемость ребер), причем некоторые виды структур имеют особенности, важные для практики.

Для моделирования злоумышленных воздействий модель ИС представляется в виде множества взаимосвязанных компонент (элементов), функции которых определяются процессами, происходящих в них и описываются множеством состояний. При этом сами компоненты стоит рассматривать не как одноранговые отношения, а как иерархическую структуру.

Недопустимыми состояниями будет совокупность возможных состояний отдельных компонентов, сочетание которых можно рассматривать как успешное злоумышленное воздействие.

Злоумышленные воздействия инициализируются злоумышленниками, обладающими целями и подразумевающими наличие сценариев по их достижению. Модель сценария злоумышленника будет напрямую зависеть от формы представления ИС.

Сценарий агентов-злоумышленников можно условно представить двумя компонентами: содержательными рассуждениями, описывающими логическую последовательность действий, событий или возможные варианты решения

проблемы, упорядоченные по времени; и результатами выполнения этапов сценария с выводами на их основе. Сценарий зависит как от формы представления, так и от методов его получения и составления с использованием. Сценарий является предварительной информацией, на основе которой проводится дальнейшая работа по прогнозированию или разработке вариантов агентов.

Множество построенных сценариев формируют библиотеку сценариев агентов и являются прототипом действий множества агентов-злоумышленников при имитационном моделировании злоумышленных воздействий.

Сценарий злоумышленника опирается на выбранную в каждом конкретном случае модель нарушителя, определяя «кем» может быть злоумышленник (внешним или внутренним), каковы его функциональные возможности, знания, умения, способности к кооперации.

Исследования в области информационной безопасности по синтезу сценариев злоумышленника предлагают ряд подходов по представлению действий злоумышленника.

Наиболее часто встречаемым является этапный подход представления сценариев. Но, данный подход описывает злоумышленное воздействие лишь поверхностно, без детализации действий, зачастую в контекстно-свободной форме.

Также, особое место занимают формальные топологические методы, базирующиеся на дискретных моделях, представляемых в виде конечных автоматов [34]:

- деревья и графы атак;
- представления поведения модели сложной системы.

Различие между данными методами весьма размыто. В большинстве случаев, оно заключается лишь в том, каким образом представляется ИС при построении сценариев, т.е. учитывается ли ее собственное поведение.

Такие модели пригодны для описания атак на широкий спектр ИС. Каждый шаг атаки в рассматриваемых моделях оценивается некоторыми критериями. Последовательность шагов может быть определена на основании критериев выбора для каждого шага.

Граф атак является расширением дерева атак, которое учитывает наличие циклов в графе. Также отмечается, что узлы графа чаще представляют не концептуальные действия, а узлы сети, процессы программ, конфигурационные файлы, участки кода и т.д. В обоих случаях переход между узлами определён рядом правил. Целью воздействия является достижения определенных вершин. Зная условия их достижения, проводят анализ защищённости исследуемой ИС.

Основное назначение этих моделей — отображение топологии уязвимостей, анализ необходимых критериев для их эксплуатации.

В методе, основанном на анализе поведения сложной системы, применяется декларативный подход представления модели сложной системы. Т.к. данный подход абстрагируется от алгоритмов системы, замещая его представлением результатов работы алгоритмов, то переложение построенных сценариев агентов с моделируемых на реальные компоненты становится менее сложной задачей. Такие методы позволяют исследовать не только известные злоумышленные воздействия, но и обнаружить ранее не выявленные.

Согласно [35] модели атак можно классифицировать по 3 группам:

- простейшие модели (этапные модели, деревья атак);
- классические модели (графы атак, системы на основе правил, формальные грамматики, ISM);
- интеллектуальные модели (логическое программирование, вероятностные модели, когнитивные карты, эволюционные модели).

Рассмотренные выше подходы относятся к 1 и 2 группе методов.

К общим недостаткам моделей всех групп можно отнести ручную генерацию спецификаций моделей исследуемых компонентов или непосредственно самих сценариев злоумышленника. Вторым недостатком некоторых моделей (характерна для графов атак) – избыточность.

Абстрагируясь от детализации данных подходов, и анализируя саму концепцию, данные подходы представляют собой ту или иную форму представления поведения сложной системы и ассоциации нарушителей.

В данных методах сценарий нарушителя определяется условиями, при которых выполняется дестабилизация системы: вершина графа состояний системы — цель, маршрут ее достижения — последовательность изменений значений свойств компонент.

Соответственно, сценарий одного нарушителя, описывающий последовательность его действий в информационной системе, представляется каким-либо участком графа. В зависимости от метода построения сценариев интерпретация графа может несколько отличаться. Каждое действие изменяет состояния ИС. Одно действие может быть как отдельной полноценной атакой, так и ее этапом.

1.3.1 Методы графологического построения сценариев злоумышленников

Построение сценария тесно связано с используемыми методами и инструментами. Злоумышленники строят свои сценарии исходя из своего представления об ИС. Выделяют множество способов графологического построения сценариев злоумышленников. Множество методов построения сценариев можно сгруппировать по используемым средствам. Попытка такой классификации данных методов была предложена в [36, 37]. В данных работах выделяются следующие методы построения сценариев злоумышленника:

- интеллектуальные экспертные системы;
- автоматическое доказательство теорем;
- вероятностные подходы;
- логические подходы;
- верификация моделей.

Данные методы охватывают большинство существующих подходов к построению сценариев злоумышленников. Разработка собственных подходов не рассматривается. Согласно [36], разработка собственного метода сопряжена с потенциально слабой формализацией и высоким числом ошибок и недоработок. К тому же придется провести ряд сопутствующих разработок, чтобы учесть и

обосновать все факторы. Данный подход критикуется по причине того, что гораздо проще воспользоваться уже существующими исследованиями в данной области, а не пытаться создать свою систему, аналогичную существующим.

И хотя все методы могут быть использованы для выявления отношений между уязвимостями, часть из них достаточно сложно реализуема или не имеет свободно доступной реализации. Также, в данных методах их авторами предъявляются разные требования к модели ИС.

Среди основных критериев оценки выделяются:

- наличие инструментов (фреймворков или языков программирования), позволяющих реализовывать данный подход;
- возможность представления модели ИС;
- возможность представления модели злоумышленников;
- полнота результатов – максимизация числа исследованных вариантов.

В последнее время получили признание и распространение подходы с использованием автоматизированных интеллектуальных экспертных систем принятия решений. Построение сценария основано на поиске соответствий множеств правил и фактов.

Однако, обычно, их отличает сложный синтаксис языка, хотя существуют попытки внедрить более синтаксически и семантически простые диалекты (как пример – диалект JGSAW для системы JESS). Сложностью данного подхода является то, что он требует знаний в области искусственного интеллекта и высокой квалификации пользователя системы. Однако, в сегодняшнем виде, класс данных систем позволяют легко решать задачи управления сложными системами и иногда включают в себя средства имитационного моделирования.

Любое отдельно взятое действие, описываемое в таких системах, можно обобщить двумя понятиями:

- требования (описывают необходимое изначальное состояние системы - предусловие);
- достижения (последствия выполнения некоторого действия - постусловие).

Соответственно результатом будет перевод системы в другое состояние посредством некоторого действия, возможного лишь в данных условиях.

Наиболее известное из теоретических исследований в предметной области были проведены на базе среды Java ExpertSystem Shell (JESS) [38, 39].

Еще один недостаток таких систем – они представляют все возможные сопоставления между записями фактов и переходов. Т.е. в таких системах смоделировать конкретного злоумышленника становится невозможным и единственный способ – обрабатывать полученные результаты. Также в данном подходе не учитывается собственное поведение анализируемой системы.

Автоматическое (машинное) доказательство теорем на сегодняшний день рассматривают лишь как «возможно перспективный» метод и он не получил широкого распространения. В последнее время его стали рассматривать в виде симбиоза с системами верификации моделей, подразумевая доказательство теорем на модели системы. Согласно [40], доказательства этого типа могут применяться к любым аксиоматическим теориям первого порядка. При этом существует множество методов доказательства теорем. Зачастую они используются при строгой формализации объекта исследования в терминах аналитической алгебры, линейных пространств, теории групп. Среди методов доказательства разрешимости теорий первого порядка выделяют: методы элиминации кванторов, методы интерпретации, методы семантических таблиц, методы теории автоматов, методы теории моделей. Часто, в данном подходе используются логики высших порядков. Цели злоумышленника в данном методе формулируются в виде теоремы, доказательство которой и определяет ее выполнимость.

Вероятностные модели в большинстве своем основаны на марковских и полумарковских моделях. ИС задается множеством состояний в виде матрицы смежности. Элементы данной матрицы – вероятности перехода системы в следующее состояние. В полумарковских моделях дополнительно выделяется матрица того же порядка функций распределения времени. Результатом анализа таких моделей являются вероятности достижения требуемого состояния за указанное число шагов.

В информационной безопасности вероятностные модели нашли широкое применение по причине проработанного математического аппарата [41, 42]. Однако в данных моделях даже незначительное изменение связи между какими-либо факторами может привести к кардинальному изменению закона распределения этого процесса. Другими словами, такая модель в любой момент может оказаться неадекватной, и к результатам её работы стоит относиться с определённой долей недоверия. Цели злоумышленника в таких моделях формулируются следующим образом: «возможно ли достичь указанного состояния за n -ое число шагов» [43].

Также, выделяют логические подходы к построению графов атак. Зачастую, такие подходы основаны на декларативном построении сценария на основе логических выводов. Одним из таких подходов в области ИБ является система автоматической идентификации и классификации уязвимостей компьютерной сети MulVAL [44]. Модель MulVAL основывается на логическом программировании и описывается с помощью языка DataLog, являющегося диалектом и подмножеством языка Prolog. Реализует данный язык среда IDE XSB.

В данной модели выделяются следующие виды объектов-вершин графа атак – вершины вывода (правило вывода) и вершины фактов (правила – предикаты, определяющие значения и вид прототипа действия). Семантическое значение этапа сценария заключается в логическом следовании «Конъюнкт \rightarrow Факт», что позволяет определить, когда выбранный предикат имеет значение «истина». Под конъюнктом понимается логический вывод.

Для того чтобы граф атак содержал все возможные сценарии злоумышленных воздействий, необходимо оценить все возможные маршруты обхода вершин и записи шагов сценария в процессе злоумышленного воздействия.

В данных моделях существует несколько видов вершин. Одним вершинам соответствуют логические операторы: «И», «ИЛИ»; другие представляет собой непосредственные действия злоумышленников, и выражаются в виде термов с аргументами. Дуги обычно обозначают значения аргументов. Подобную идею также реализует подход, предложенный в [45].

Одно из основных достоинств данного метода – быстрое действие и малое требование к ресурсам ПЭВМ. Полученный граф атак позволяет наглядно оценить самые опасные сценарии действий злоумышленника. Также, описание графа является семантически понятным.

К недостаткам следует отнести то, что данный метод оценивает лишь возможные заранее заложенные программные атаки, и то, что очень сложно описать исходную структуру представлений о системе. Т.е. во многом, построенный сценарий с помощью такого метода отражает лишь последовательность действий злоумышленника, но не отражает, что происходит с системой на каждом этапе его действий, что вызовет трудности при применении полунатурной модели ИС. Соответственно и цели формулируются лишь как возможность выполнить указанный набор уязвимостей.

Существует метод с использованием систем верификации моделей и, в частности, символьных верификаторов моделей. Данный подход позволяет провести статическое исследование модели на корректность. Наиболее близким по смыслу подходом является автоматное программирование [46].

Согласно [37], впервые данную технику в ИБ применили в 2000 году в [47], используя немодифицированную систему CMU SMV для анализа защищенности сети. Но настоящее развитие данное направление получило после работы [48] в которой применили модифицированную систему верификации NuSMV. Основное достоинство данного подхода – формальность: если анализируемая модель системы признана безопасной, то это означает, что все варианты атак были рассмотрены, и вопросов касательно полноты анализа не возникает.

Модель информационной системы в таких моделях разбивается на классы. Каждый класс представляется множеством взаимосвязанных состояний. Для каждого состояния указывается условия перехода в следующее состояние. Для реализации данного подхода используется модель Крипке.

Цели злоумышленника формируются с помощью темпоральных логик по отношению к состояниям исследуемой системы.

В частности, именно символьная верификация моделей на сегодняшний день де-факто считаются ориентиром для будущих разработок в данной области по следующим причинам [42]:

- базируется на существующих разработках framework'ов;
- обладают широкой масштабируемостью;
- позволяют оперировать формализованными утверждениями;
- результаты анализа исчерпывающи (позволяют идентифицировать все существующие сценарии атаки), в тоже время и лаконичны (идентифицируются только достижимые цели).

Данный подход уникален тем, что он не использует императивный подход при моделировании действий злоумышленника, а представляет собой моделирование поведения информационной системы в случае злоумышленного воздействия. Состояния, приводящие к нарушению работы системы, являются целью злоумышленника, а маршрут их достижения – их сценарием. Смена состояний – последствия влияния на систему злоумышленника.

В таблице 1 приведен результат сравнения методов графологического построения сценариев злоумышленников.

С учетом применения полунатурного подхода, метод верификации моделей является наиболее предпочтительным к использованию, т.к. позволяет решить совместно задачи моделирования ИС и построения сценариев злоумышленников. При этом полученный результат может быть легко применен на реальных компонентах.

Таблица 1 – Сравнение методов графологического построения сценариев злоумышленников

| Подход | Используемые инструменты | Способ задания модели ИС | Цели злоумышленника | Полнота результатов | Концепция модели |
|--------------------------------------|--|---|-----------------------------------|---|---|
| Экспертные системы | JESS (JIGSAW), CLIPS, SkyBox Security | Отражается только начальное состояние | Не выражены в явном виде | Все возможные сценарии | Строится фиксированное начальное состояние. системы, затем применяются правила вывода |
| Автоматическое доказательство теорем | SAM, Aura, Otter | Алгебраическая, геометрическая, в терминах линейных пространств, теории групп | В виде теорем (логика предикатов) | Ограниченное число сценариев | Стоится только последовательность действия злоумышленника. |
| Вероятностный подход | Марковские, полумарковские процессы. | Матрица переходов | Число шагов, состояние | Все возможные сценарии | Стоится модель функционирования компонентов с указанием недопустимых состояний. |
| Логический подход | DataLog, Prolog, математическая логика | Отсутствует, либо элементы ИС | Логические формулы | Ограниченное число сценариев | Стоится только последовательность действия злоумышленника. |
| Верификация моделей | CMU SMV, NuSMV, SPIN | Модель Крипке | Темпоральные логики | Ограниченное число сценариев или все возможные сценарии | Стоится модель функционирования компонентов с указанием недопустимых состояний. |

В отличие от других методов он позволяет учесть поведение модели ИС. ИС представляется совокупностью состояний, где целью злоумышленника является достижение некоторых из них. [49]

Также, метод изначально был разработан для проверки корректности моделируемой системы и предполагает ее автоматическую проверку, а также генерации сценариев злоумышленников для всех анализируемых случаев нарушений поведения ИС.

1.3.2 Метод верификации моделей

Верификатор моделей – это инструментальное средство, предназначенное для проверки того, что модель системы удовлетворяет спецификации, заданной CTL (Compute Tree Logic) или LTL (Linear Temporal Logic)[50]. В общем случае, любая символьная система верификации моделей опирается на следующие принципы [50]:

- моделируется система с конечным числом состояний на специализированном языке;
- описываются требования и определения к модели с помощью темпоральной логики CTL/LTL;
- внутренне представление основывается на BDD (Bynary Decision Diagram);
- верификатор автоматически проверяет описание на истинность условий, в противном случае генерирует контрпример.

Третье свойство – опциональное, в основном является отличительной особенностью символьного верификатора моделей. Существует более широкий спектр верификаторов моделей, в которых используются самые разнообразные подходы, что объясняет их успешное применение в разных областях: в реагирующих системах – в системах, постоянно взаимодействующих с окружением [51], при анализе параллельных и последовательных систем и алгоритмов.

Данный подход разрабатывался как альтернатива имитационному моделированию. Отличия заключается в том, что проводится анализ не одного компонента или траектории, а совокупность всех возможных сценариев поведения модели, что позволяет выявить новые элементы для дальнейшего имитационного моделирования. Что самое важное, сама модель зачастую для обоих подходов практически совпадает.

В верификаторе логическая модель ИС представляет собой символическое представление пространства всех возможных состояний исследуемой системы Ω . Т.е. процесс построения модели можно представить следующим способом. Вначале строится модель с конечным числом состояний, требования к ней (проверяемые свойства) оформляются на языке темпоральной логики, после чего данная структура подается на вход верификатора. Его задача – проверить требования к модели и в случае невыполнения требований сформировать контрпример, который указывает последовательность состояний модели, при которой нарушается проверяемое требование.

Самым сложным этапом данного подхода является описание и преобразование требований к системе, к модели, и обратное преобразование контрпримера, сформированного для модели, к контрпримеру для системы.

При всем существующем многообразии реализаций систем верификации, отличаются они друг от друга языком описания модели и алгоритмами проверки требований.

1.4 Постановка задачи исследования

Необходимо разработать метод анализа защищенности ИС. Система S представляется множеством своих ресурсов Res : $S = \langle Res \rangle$. Каждый ресурс res_i , принадлежащий множеству ресурсов Res , характеризуется совокупностью не перекрытых уязвимостей Vul_i . При этом в процессе анализа могут быть реализованы деструктивные воздействия B_i на ресурс. Тогда ресурс ИС описывается кортежем: $res_i = \langle Vul_i, B_i \rangle$.

Критериями качества анализа защищенности ИС являются:

- 1) полнота проведенного анализа;
- 2) количество деструктивных воздействий на ресурсы ИС.

Под полнотой анализа уязвимостей понимается отношение числа найденных уязвимостей к числу уязвимостей, полученных с помощью сканеров уязвимостей на этапе сбора исходных данных.

Для улучшения качества анализа защищенности ИС необходимо:

1) максимизировать количество выявленных в результате анализа не перекрытых уязвимостей: $\max_{Res} |Vul|$, где $Vul = \cup_i Vul_i$.

2) минимизация количества деструктивных воздействий на ресурсы ИС в процессе анализа: $\min_{Res} |B|$, где $B = \cup_i B_i$.

Для проведения анализа защищенности ИС создается множество агентов ИС A , которые ставятся в соответствие ресурсам ИС: $Res \rightarrow A$, и агентов злоумышленников I , выполняющие действия по эксплуатации уязвимостей. При этом ресурсы ИС делят на критически важные ресурсы и другие ресурсы: $Res = \{Res_{кр}, Res_{др}\}$. По аналогии с ними агенты делятся на виртуальные и реальные: $A = \{A_v, A_p\}$. Причем: $Res_{кр} \rightarrow A_v, Res_{др} \rightarrow A_p$. В этом случае можно пренебречь количеством деструктивных воздействий на ресурсы, которым соответствуют виртуальные агенты. Тогда количество деструктивных воздействий на ресурсы ИС можно представить в виде множества $\hat{B} = \cup_j B_j | res_j \in Res \setminus Res_{кр}$. Однако применение реальных агентов позволяет потенциально выявлять большее количество не перекрытых уязвимостей, что не позволяет заменить все ресурсы ИС на виртуальные.

1.5 Вывод по первой главе

Проведен анализ процесса управления информационной безопасностью, в результате которого было выявлено, что защищенность, надежность, производительность являются одними из основных показателей качества

функционирования ИС. Наиболее важный этап управления – этап анализа защищенности, результат которого является основанием для принятия решений по планированию мероприятий защиты информации в ИС.

Определены недостатки существующих подходов к анализу защищенности информационных систем. Экспертные методы анализа защищенности являются трудоемкими и могут выявлять не все угрозы информационной безопасности, а значит, выдают неадекватную оценку защищенности. Инструментальные методы анализа защищенности обладают большей полнотой, но могут нарушать функционирование информационной системы. Поэтому требуется предложить метод, который бы объединял достоинства обоих подходов и минимизировал недостатки.

Информационная система рассматривается как сложная система. Основная концепция построения моделей сложных систем – представление их в виде множеств допустимых и недопустимых состояний.

Лучший подход имитационного моделирования сложных систем – многоагентное моделирование. Оно позволяет получить представление об общем поведении системы, исходя из предположений об индивидуальном поведении её отдельных активных объектов. Данный подход позволяет применять дискретный подход моделирования к элементам системы.

Для повышения точности результатов моделирования используется полунатурное моделирование, тем самым реализуется исследование поведения объекта в условиях близких к реальным.

Для представления сценариев злоумышленных воздействий выбираются методы, использующие теорию конечных автоматов, — графы атак.

Лучшим методом для построения множества сценариев агентов-злоумышленников на модели состояний информационной системы является метод верификации моделей. Достоинства данного метода:

— позволяет представить модель ИС в виде модели состояний (модель Крипке).

- позволяет использовать темпоральные логики CTL или LTL для представления целей злоумышленника;
- позволяет построить сценарии агентов-злоумышленников как посредством верификации (проверки соответствия) модели ИС при заданных целях злоумышленника (условиях, выраженных темпоральными логиками).
- проводить адаптацию модели к изменениям внешней среды.

Сформулирована задача на исследование, состоящая в максимизации количества выявляемых не перекрытых уязвимостей и минимизации количества отказов в работе компонентов ИС.

2 МЕТОД АНАЛИЗА ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Формализованное описание метода анализа защищенности информационной системы

Предлагаемый метод анализа защищенности реализует следующую последовательность шагов:

1) Построение модели ИС. Выбранный метод верификации моделей предъявляет определенные требования к описанию ИС, как пространства всех возможных состояний. Требования к модели ИС (проверяемые свойства) оформляются на языке темпоральной логики.

2) Построение модели злоумышленника. Определяются цели злоумышленника, и выполняется верификация модели ИС. При верификации проверяются требования к модели, и в случае невыполнения требований формируется контрпример, который указывает последовательность состояний модели, при которой нарушается проверяемое требование. В результате формируется библиотека сценариев злоумышленника.

3) Моделирование сценариев злоумышленных воздействий с использованием многоагентного подхода. Формируются агенты ИС и агенты злоумышленника. Для обеспечения полунатурности агенты ИС разделяют на реальные и виртуальные.

4) На основе результатов моделирования оценивается защищенность ИС.

2.2 Спецификация модели информационной системы

Подход к моделированию сложных систем на основе верификации достаточно апробирован. Например, в работах [52, 53] приведено несколько примеров успешного применения средств верификации применительно к поиску ошибок в алгоритмах. Именно в данной области он получил наиболее широкое применение как средство автоматизации отладки программ, однако он успешно

применяется и для проверки логических схем (верификаторы VHDL), и для протоколов, и для смешанных (гибридных) систем на базе сетей Петри (DSSZ-МС на базе BDD-Мс) [47]. Хотя, если коснуться исторических аспектов разработки данной методики, то основное ее применение виделось как замена (или дополнение) имитационному моделированию при решении задач корректного проектирования безопасных систем [54].

Широкий спектр решаемых задач требует адаптации данного подхода для каждого конкретного случая. В контексте применимости анализа сложных систем (информационных систем) наиболее применимыми видятся верификаторы общего назначения.

Задача любого средства верификации – скрыть от пользователя используемый математический аппарат. Для этой цели верифицируемая модель описывается на специальном языке (SMV, VHDL, Promela, Simulink/Stateflow, языках программирования С, С++, Java), предполагающем иерархическое представление модели с разделением ее на классы и объекты. В каждом классе уже определяются свойства и множество их возможных значений.

Такое описание упрощает построение модели и предъявление спецификаций к ней.

Моделируемая система описывается множеством состояний – мгновенного описания системы, в котором зафиксированы значения переменных компонент в некоторый момент времени. Под изменением понимается перевод системы из состояния до определенного действия (предусловие) в состояние, после его совершения (постусловие).

Для формализации данных представлений предлагается использовать темпоральную логику, которая является языком формулирования утверждений, использующих понятия времени и модель Крипке, как разновидность графа переходов. Модель Крипке состоит из множества состояний, множества переходов между состояниями и функции, которая помечает каждое состояние набором свойств, принимающих истинное значение в данном состоянии [55].

Модель ИС представляется тройкой объектов:

$$M_{\text{ИС}} = \langle \text{Obj}, \text{Vul}, \text{Res} \rangle \quad (1)$$

где $\text{Obj} = \{\text{obj}_i\}$ – множество объектов информационной системы, $\text{Vul} = \{\text{vul}_i\}$ – множество уязвимостей, $\text{Res} = \{\text{res}_i\}$ – множество ресурсов ИС, найденных при исследовании информационной системы. Множество объектов ИС представляется следующим образом:

$$\text{Obj} = \langle \text{Host}, \text{Component} \rangle, \quad (2)$$

где $\text{Host} = \{\text{host}_i\}$ – множество хостов ИС, $\text{Component} = \{\text{component}_j\}$ – множество компонентов ИС, функционирующих в ней.

Тогда каждый компонент в терминах модели Крипке представляется как:

$$\text{component}_j = (S, S_0, R, L). \quad (3)$$

Хост в терминах модели Крипке определяется как объединение компонент.

В терминах модели Крипке двойка множеств $\langle V, D \rangle$ проецируется в двойку:

$$\langle \text{Property}, \text{Value} \rangle, \quad (4)$$

где Property – множество переменных системы, Value – домен интерпретации переменных (значений переменных). Под двойкой $\langle \text{Property}, \text{Value} \rangle$ понимаются все свойства и значения объектов ИС. Состояние s информационной системы, при $\langle V = \text{Property}, D = \text{Value} \rangle$, в модели Крипке определяется как:

$$s(v) = d. \quad (5)$$

Процессы функционирования (поведение) ИС обозначаются f и определяются логикой изменений значений свойств.

Тогда, в терминах спецификации ИС модель элементы модели Крипке определяются следующим образом:

— Множество состояний ИС S определяется множеством соответствий $\langle \text{Property}, \text{Value} \rangle$.

— Множество начальных состояний S_0 определяется подмножеством начальных значений $\text{Value}_0 \subseteq \text{Value}$. В качестве начального состояния может выступать любое значение из множества возможных значений свойств. Перебор всех возможных сочетаний начальных состояний позволяет реализовать все

возможные начальные условия, а соответственно проверить все варианты поведения модели.

— Множество переходов R определяются логикой изменений значений свойств и обозначаются литерой f . Во множество f учитывается не только логика изменений значений свойств, но и отношения между объектами информационной системы. [56]

Архитектура информационной системы предполагает иерархию входящих в нее объектов, с учетом взаимосвязей между ними.

Каждый элемент $host_i$ определяется подмножеством компонентов $Component'$ и их взаимным функционированием, определенных в информационной системе:

$$host_i = \langle id, Component', f \rangle, \quad (6)$$

где id – идентификатор (имя) хоста, $Component' \subseteq Component$, функция f накладывает ограничения на функционирование компонентов внутри хоста.

Компонент информационной системы определяется как:

$$component_j = \langle id, Component', f, Property' \rangle, \quad (7)$$

где id – идентификатор (имя) компонента, $Component'$ – множество дочерних компонентов, f – функция изменений значений свойств, $Property'$ – множество атомарных (неделимых) свойств i -ого компонента.

Тогда, в предлагаемых терминах состояние s описывается как множество всех возможных соответствий между всеми свойствами ИС и мгновенными значениями этих свойств.

$$s(host_i. [component_k]. component_j. property_{ijk}) = (value_{ijkw}). \quad (8)$$

Уязвимость можно трактовать как недопустимое изменение состояния ИС. В терминах модели Крипке уязвимость определяется как:

$$vul_i = R(s', s'') = (SF, PF), \quad (9)$$

где s' или SF – состояние ИС, описывающее предусловия, а s'' или PF – постусловие.

Расширенное описание уязвимости можно представить как:

$$vul_i = \langle id, desc, action, SF, PF, V_{exploit} \rangle, \quad (10)$$

где id – идентификатор уязвимости, $desc$ – описание уязвимости, $action$ – правила эксплуатации уязвимости, SF – множество предусловий, PF – множество постусловий, $V_{exploit}$ – сложность эксплуатации уязвимости.

Множества SF и PF имеют одинаковую структуру и определяются как множество записей вида:

$$\langle host_i. [component_k]. component_j. property_{ijk} = value_{ijkw} \rangle. \quad (11)$$

В разрабатываемом описании множество предусловий может быть пустым. [57]

Уязвимости проецируются на некоторое подмножество объектов информационной системы. Всем уязвимостям Vul в соответствие ставится оценка их критичности.

Под ресурсами Res понимается циркулирующая в ИС информация. Ресурсы непосредственно не моделируются. В модели ИС они представляются как множество состояний, в которых производится несанкционированное воздействие по отношению к информации, циркулирующей в информационной системе: базам данных, документам, правам доступа, настройкам служб.

Ресурс определяются как:

$$res_i = \langle id, S', C, R_d \rangle, \quad (12)$$

где id – идентификатор (название) ресурса, S' – подмножество состояний, в которых нарушитель осуществляет доступ к ресурсу, C – стоимость ресурса (количественная оценка ущерба), в случае несанкционированного доступа к ресурсу, R_d – допустимый риск. Одним из примеров такого подмножества может быть тройка состояний (s_k, s_d, s_c) , где состояния обозначают нарушение конфиденциальности, доступности и целостности соответственно. В соответствие к этой тройке ставится тройка ущерба (c_k, c_d, c_c) . [58]

Интерпретация спецификации модели ИС приведена на рисунке 3.

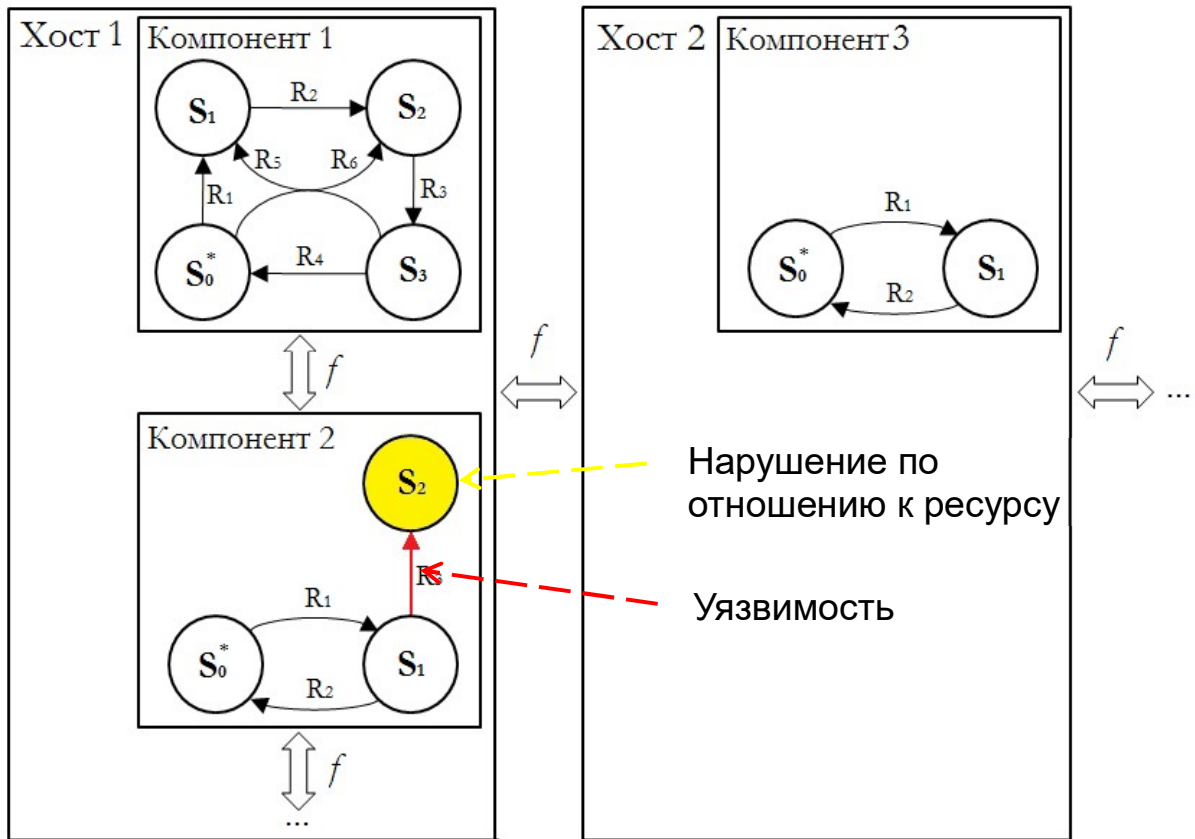


Рисунок 3 – Интерпретация спецификация модели ИС

Модель ИС представляется множеством взаимосвязанных хостов. Внутри каждого хоста выделяют группу компонент, поведение каждой из которых представляется моделью Крипке. В компоненты закладываются уязвимости, позволяющие получить доступ к тому или иному ресурсу. При этом процессы функционирования компонент могут быть взаимосвязаны между собой.

К примеру, ошибка, связанная с переполнением буфера в компоненте СУБД удаленного хоста позволяет выполнить произвольный код и получить, к примеру, права администратора на удаленной машине. В результате злоумышленник может получить доступ ко всем обрабатываемым и хранящимся базам данным.

Данное представление модели ИС является более удобным для применения в инструментальных средствах, и в тоже время, оперирует терминами, понятными для специалиста по информационной безопасности.

2.2.1 Рекомендации по упрощению модели информационной системы

Зачастую, подходы, основанные на использовании полных графов, являются ресурсоемкими, и чрезмерное увеличение описания информационной системы без должной декомпозиции ее представления способно привести к «комбинаторному взрыву» при построении графа сценариев. Согласно [47], для повышения эффективности решений может использоваться агрегация подобных хостов Host как для улучшения наглядности графа атак, так и для повышения производительности. По результатам разведки зачастую вполне возможна ситуация, когда одна или несколько уязвимостей будет присуща множеству хостов. Соответственно на этих хостах будет подобная схема компонентов. Простейшая агрегация заключается в замене группы идентичных хостов на один хост. В случае, если компоненты мало отличаются между хостами, то при объединении хостов можно объединить и их компоненты. Однако, такое вмешательство нежелательно, вследствие неочевидности последствий изменений.

При исследовании информационной системы, состоящей из множества хостов, соединенных маршрутизаторами, задачу можно упростить тем, что в каждой подсети выделять только по одному хосту одного типа. В случае громоздкости, можно рассматривать только финальную часть атаки, без учета устройств, через которые будет проходить сетевой трафик атакующего злоумышленника.

Другой метод заключается в построении ограниченного графа, используемого исключительно для ответа на следующие вопросы: какие хосты могут быть скомпрометированы нарушителем с некоторого хоста и какое минимальное множество уязвимостей позволит нарушителю достигнуть его цели? Т.е. одним из способов упрощения расчетов графа – анализ связности отдельных компонентов и анализа их влияния друг на друга, а так же их возможности эксплуатации независимо друг от друга. Если одна уязвимость не зависит от другой, то нет смысла описывать их совместно, тем самым усложняя модель.

Следствием из данного метода является то, что часть сценариев можно разбить на несколько этапов. К примеру, сценарий атаки сервера с удаленного внешнего хоста можно разбить на два этапа: получение опорной точки внутри информационной системе и непосредственно атаку сервера (или атаку до шлюза, и атаку после прохождения шлюза).

Также неизвестным параметрам системы можно присваивать некоторые значения по умолчанию. Т.е. при невозможности уточнить эти параметры наиболее оптимальным видится выставление значений, которые соответствуют реальным элементам системы при ее развертывании

2.2.2 Методика сбора сведений об информационной системе

Качество и полнота собранных сведений о структуре информационной системы, ее компонентах, процессах, уязвимостях, в первую очередь будут зависеть от качества проведения разведки интересующей информационной системы. На процесс разведки непосредственным образом влияет неформальная модель злоумышленника.

Разведка заключается в сборе сведений об объектах Obj информационной системы, уязвимостях Vul и ресурсах Res. Перечень уязвимых программно-аппаратных средств и принцип их анализа приведен в [54].

В результате разведки формируются следующие сведения:

- Перечень найденных уязвимостей Vul, их описание, список компонентов, связанных с их реализацией, оценка критичности.
- Перечень ресурсов Res информационной системы, перечень возможных нарушений для каждого ресурса и оценка критичности.
- Перечень хостов Host информационной системы.
- Перечень компонент Component информационной системы, связанных с уязвимостями. Также, анализу подвергаются взаимосвязи между компонентами.

К компонентам относятся:

- сетевая топология;

- программы и службы;
- устройства хранения информации, прочие устройства;
- перечень аккаунтов пользователей на хосте, в сети;
- процедуры авторизации пользователя в системе;
- и др.

На основе анализа можно попытаться спрогнозировать, какие из элементов защиты потенциально могут содержать бреши, как их обойти предполагаемому злоумышленнику, предположить какие средства анализа выбрать в том или ином конкретном случае. Также для анализа следует использовать результаты анализа специфики предприятия, бизнес-процессов, подходов работы с информацией ограниченного доступа, должностные регламенты лиц, имеющих высокий уровень полномочий в ИС.

Исходя из [55], необходимо разделение разведки на два этапа: внешнюю и внутреннюю.

Внешняя разведка заключается в сборе сведений из общедоступных источников.

Провести внешнюю разведку зачастую достаточно сложно. Большинство анализируемых элементов связано со сбором сведений по сетевым уязвимостям публичных ресурсов. Организации, имеющие в своем активе информацию ограниченного доступа, обычно не имеют прямого подключения к глобальным ресурсам, соответственно становится невозможным провести данный вид разведки в полной мере.

С другой стороны, под внешнюю разведку попадают действия, выполнение которых подразумевает наличия пользовательских или гостевых прав доступа – это сетевые информационные ресурсы свободного доступа.

Такие действия не требуют повышения сетевой активности пользователя, использования специализированного программного обеспечения или действий, которые могут показаться подозрительными для системного администратора. В эту же категорию попадает пассивный сбор информации о сетевом трафике.

Результатом внешней разведки являются прототипы компонентов (т.к. не всегда можно однозначно идентифицировать компонент) без их детализации, и множество уязвимостей, связанных с данными компонентами, выявление которых весьма не сложно. Выявление полного множества хостов Host зачастую невозможна, т.к. не все хосты можно будет определить.

Предлагается дополнять внешнюю разведку внутренней.

В ходе внутренней разведки специалисты действуют с точки зрения гостя или рядового сотрудника организации (в зависимости от типа ИС), администратора. Сбор сведений производится из внутренних источников организации.

Составляется перечень доступных информационных ресурсов, типы доступа к ним, предпринимается попытка построения топологической схемы ИС (выявления всего множества хостов Host), прорабатываются пути проникновения.

Внутренняя разведка подразумевает уже более активные способы сбора информации. Здесь корректно использовать различные инструментальные средства анализа защищенности [59, 60], такие как X-Spider, Nessus, OpenVAS, nmap, семейство пакетов построения топологии и анализа сети 10-Strike, мониторы процессов. Сюда же можно отнести различные утилиты по сканированию портов, построению топологии сети, удаленному определению конфигурации хостов, их настроек, перечня открытых портов, анализу политики безопасности. Результатом их работы является указание конкретных уязвимостей со ссылками на их более детальное описание. Также производится анализ разрешенной пользователю документации, регламентов, приказов и писем, методик, циркулирующих внутри организации.

Внутренняя разведка позволяет выявить уязвимости Vul, не обнаруженные на этапе внешней разведки. [61]

На основании анализа процессов организации, найденных уязвимостей, строятся модели компонент и описываются процессы их функционирования. Построение моделей заключается в выявлении всех зависимостей f между ними, определение вложенности компонент, определении множеств *Property* и *Value*.

Стоит отметить, что построенные спецификации будут несколько отличаться от моделей, представленных на языке SMV.

Исходя из модели злоумышленника, формируются его цели и множество начальных состояний моделей компонентов системы. Цели злоумышленника формируются исходя из перечня ресурсов в ИС и найденных уязвимостях. Злоумышленник будет стремиться получить несанкционированный доступ к ресурсу, нарушить его целостность, конфиденциальность или доступность.

Исходя из принципа разумной достаточности, очевидно, что моделировать все компоненты или процессы, не влияющие на защищенность, бессмысленно. Т.е. проблема кроется не в том, что невозможно будет построить нужную модель и/или она будет неверной, а в том, что изначально требуется определить перечень моделируемых компонентов ИС. Перечень моделируемых объектов определяется на основании модели злоумышленника. Дополнительно, для ее решения и требуется провести анализ найденных уязвимостей. Анализ проводится на основе списков CVE, классов CWE, NVD, метрик CVSS, бюллетеней Microsoft, отчетов инструментальных средств анализа защищенности, др. документов [62-66].

На первом шаге анализируются списки уязвимостей, и определяется, с какими компонентами она связана. Это позволит определить ряд предусловий, необходимых для ее эксплуатации, а также внутри какого процесса она функционирует. Также, для каждой уязвимости определяется ее результат эксплуатации злоумышленником, т.е. последствия в случае ее успешного использования (постусловия). Цель данного шага – определить состояния до и после эксплуатации уязвимости.

На втором шаге производится детализация правил эксплуатации уязвимости. Процесс поиска заключается в тщательном поиске существующих под данные уязвимости эксплойтов, описания эксплуатации в списках уязвимостей, и взаимосвязанных с ними документов и отчетов («баг-треков»).

Третий шаг заключается в оценке уязвимости. Его суть – поиск метрик данных уязвимостей. Количественную оценку уязвимости можно сформировать на

основании списков NVD и метрик CVSS. Их анализ позволит провести количественную оценку уязвимости с позиции сложности ее эксплуатации.

После того, как уязвимости проанализированы, необходимо определить, к каким нарушениям по отношению к ресурсам приводит эксплуатация выявленных уязвимостей.

По результатам разведки должно быть собрано необходимое количество сведений, показывающих основные недостатки существующей системы информационной безопасности, а также возможные пути проникновения в систему.

2.3 Спецификации модели злоумышленника

Библиотека сценариев – совокупность всех сценариев. Библиотека сценариев относительно модели ИС выражается в виде

$$Tr = \{tr_i\} = ver(Obj, tg(Vul, Res)), \quad (13)$$

где tr_i – i -ый сценарий в библиотеке, ver – функция верификации, проверяющая выполнимость спецификаций, описывающих множество целей, на модели объектов ИС Obj , функция $tg(Vul, Res)$ – функция выявления целей на основании найденных уязвимостей Vul в ИС и ресурсов Res .

Отдельный сценарий представляется в виде четверки:

$$tr_i = \langle id, S, R, P_{y.cц} \rangle, \quad (14)$$

где id – идентификатор сценария, S – множество состояний ИС в сценарии, R – множество дуг между состояниями, $P_{y.cц}$ – вероятность выбора данного сценария.

Сценарий агента-злоумышленника интерпретируется как последовательность действий, необходимых для реализации злоумышленного воздействия на определенный ресурс. Очевидно, что число идентичных или даже похожих сценариев для разных ИС будет достаточно мало. Этот факт связан с различием архитектуры ИС и составляющих их компонентов, а также различиями в используемых моделях нарушителя.

Полученные сценарии являются ориентированным подмножеством состояний системы, которое можно интерпретировать в виде графа (рисунок 4).

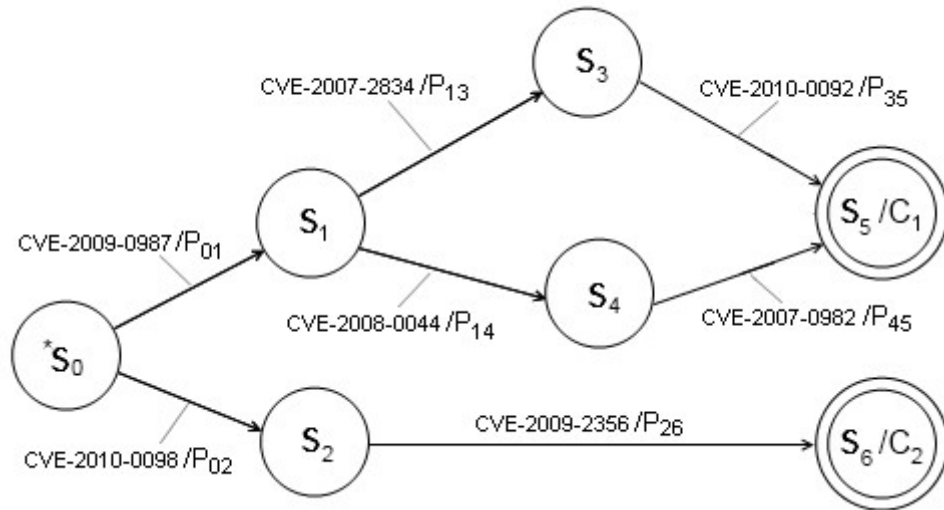


Рисунок 4 – Граф библиотеки сценариев агентов-злоумышленников

Множество сценариев имеет двухуровневое описание:

- вершины графа s представляют собой состояния ИС;
- дуги (множество переходов R). Определяют действия, необходимые для изменения состояния. [67]

Маршрут из начального состояния в конечное – это последовательность уязвимостей, описывающих стратегический план агентов-злоумышленников. Множество маршрутов из начальной вершины в конечную вершину можно назвать множеством сценариев.

Вершины графа содержат описания состояний ИС, создающих предпосылки к эксплуатации уязвимостей, задаваемых дугами, выходящими из этих состояний. Дуги же обозначают действия, необходимые для достижения данного плана и являются тактическими планами агентов-злоумышленников. Пометка дуг может быть различной.

Такое представление позволяет моделировать реализацию многошаговых атак, в которых эксплуатируемые уязвимости зависят от нескольких компонент, разнесенных по разным хостам. Данная спецификация позволяет говорить о частичной унифицированности архитектуры ИС, а также о ее распределенности.

Сценарии злоумышленника выбираются из существующей библиотеки сценариев. Реализация злоумышленных воздействий осуществляется агентами.

Пример: пусть есть граф из двух вершин – состояний s' и s'' . Они идентичны, за исключением свойства активности компонента антивируса *status*. В первом случае оно имеет значение «включен», во втором – «выключен». Тогда целью злоумышленника будет выключение антивируса для возможности совершения каких-то других целей. Дуга между двумя состояниями обозначает действие, заключающееся в остановке защиты антивирусного средства на указанном хосте. К примеру, это возможно, если пользователь знает пароль, установленный администратором антивирусной защиты.

Любую уязвимость можно представить в виде предусловия, необходимого для ее реализации, и постусловий – результата реализации уязвимости. Предусловием и постусловием являются состояния ИС. Таким образом, специалисту необходимо будет определить компоненты ИС, на который воздействует агент-злоумышленник, определить их начальное, необходимое для атаки, и конечные состояния. Эти данные могут быть получены из результатов анализа уязвимости. Каждая уязвимость определяется через изменение одно или нескольких состояний компонент. Вследствие эксплуатации уязвимостей злоумышленник может получить доступ к одному или нескольким ресурсам. После установления всего перечня сочетаний состояний и их зависимостей они выражаются как требования к модели, выполнение которых необходимо проверить. Суть процесса построения требований заключается в нахождении способа представления целей злоумышленника в терминах темпоральных логик. В связи с иерархической компоновкой модели сложной системы CTL/LTL спецификации стоит предъявлять только в блоке SPEC (для CTL) или LTLSPEC (для LTL) в модуле Main. Синтаксис одной цели (элемента) в условии будет иметь вид (листинг 1).

Листинг 1 – Синтаксис формулирования целей злоумышленника

SPEC

⟨предикат⟩

$(host_i.[component_k].component_j.property_{ijk} \langle \text{оператор} \rangle value_{ijkw})$

Данное условие описывает цель злоумышленника. Цель формулируется, как нарушение по отношению к какому-либо ресурсу. В дополнительных случаях помимо получения доступа к ресурсу в цель вносится факт эксплуатации той или иной уязвимости. Такое условие позволит проверить, позволяет ли данная уязвимость получить доступ к требуемому ресурсу.

Согласно спецификации, модель уязвимости представляется следующей шестеркой объектов: идентификатор, описание, правила эксплуатации, предусловие, постусловие, сложность эксплуатации.

Идентификатор формируется на основании множества именованных уязвимостей в списках CVE, классах CWE или др. документах.

Описание определяется множеством идентификаторов уязвимости и представляет собой собранные неформализованные сведения о ней в различных источниках.

Правила эксплуатации формируются на основании множества алгоритмов и эксплойтов. Они публикуются на сайте CWE и в базах эксплойтов. Форма представления может быть различной (логическая последовательность действий на декларативном языке, программный код, т.д.), но достаточной для программной реализации агента.

Сложность эксплуатации определяется на основании метрик CVSS. Предусловия и постусловия имеет одинаковую спецификацию. Из всего множества объектов, описывающих уязвимость, только они находят отражения в модели ИС. В некоторой степени они определяют ее, т.к. закладываются в нее и соответственно влияют на ее поведение. В некоторых случаях оправдана разработка модели ИС только на основании анализа уязвимостей.

Множество предусловий может быть пустым. Значения свойств в описаниях предусловий и постусловий могут совпадать, если для эксплуатации уязвимости это критично.

Предусловие любой уязвимости включает в себя начальные значения свойств компонент задействованных хостов. Постусловие включает в себя значения свойств компонент, в случае успешного выполнения атаки.

Множества предусловий и постусловий уязвимости описываются следующим образом [68] (листинг 2).

Листинг 2 – описание уязвимостей.

предусловия:

```
hosti:
    componentj:
        propertyk=valuek1;
```

постусловия:

```
hosti:
    componentj:
        propertyk=valuek2;
```

После построения модели ИС на языке SMV производится формирование целей злоумышленника. Злоумышленник может преследовать цели, не связанные с непосредственной эксплуатацией уязвимости. Т.е. в данном случае уязвимости представляют собой лишь инструмент достижения цели.

Дополняя вышесказанное, целями злоумышленников могут быть не только получение доступ к ресурсам ИС, но и нарушение функционирования ИС. При этом спецификация объявления целей злоумышленника в терминах темпоральной логики не меняется.

Формирование целей в логиках CTL или LTL предполагает реализацию двух подходов [69]:

- спецификации строятся, исходя из проверки условия, что модель ИС всегда находится в допустимых состояниях (не достигает недопустимых состояний);
- спецификации строятся, исходя из проверки условия, что модель ИС достигает недопустимое состояние.

В первом случае контрпример (если таковой существует) покажет произвольный сценарий злоумышленника, выводящий систему из множества допустимых состояний. Во втором случае, верификатор, в случае успешного выполнения условия, просто подтвердит истинность утверждения. Поэтому, для генерации сценария злоумышленника, необходимо над результатом формулы выполнить отрицание (т.е. перед формулой поставить знак «!»).

Для представления целей злоумышленника логикой CTL предлагается использовать в условиях следующие предикаты:

— AG. Свойство, следующее за данным квантором, всегда выполняется во всех последующих состояниях. Если существуют, сценарии, где в каком-либо состоянии оно перестает выполняться (хотя бы один раз), то верификатор его предложит в качестве контрпримера.

— AF. Свойство, следующее за данным квантором, выполняется в во всех последующих состояниях хотя бы один раз. Если существует сценарии, где оно не выполнилось никогда, то верификатор его предложит в качестве контрпримера.

— EG. Существует хотя бы одна последовательность состояний, в которой свойство, следующее за данным квантором, выполняется во всех последующих состояниях.

— EF. Существует хотя бы одна последовательность состояний, в которой свойство, следующее за данным квантором, выполнится хотя бы раз.

Предикаты AG, AF применяются для safe-подхода. К примеру, можно проверить ситуацию, что пользователь, не авторизованный в системе, никогда не сможет авторизоваться как администратор (всегда будет неавторизованным).

Предикаты EF, EG удобно применять для unsafe-подхода. Ими удобно проверять выполнимость целей злоумышленника. Пусть цель злоумышленника – права администратора. То данные предикаты позволяют проверить, существуют ли в ИС такие уязвимости, которые позволяют злоумышленнику получить права администратора. Если перед предикатом поставить отрицание, то контрпример и будет иллюстрировать такой сценарий.

Наиболее используемыми являются предикаты AG и EF. Предикаты EG и AF используются совместно с предыдущими двумя.

Для представления целей логикой LTL предлагается использовать предикаты:

— F. Данное свойство выполнится хотя бы в одном последующем состоянии.

— G. Данное свойство выполняется всегда во всех последующих состояниях.

— U. Данный предикат позволяет сформировать требование, что первую часть маршрута выполнялось одно свойство, а вторую часть другое.

В логике LTL два различия между видами подходов определяется значениями проверяемых свойств. В каждом из подходов они будут разные.

После того, как модель ИС построена, и объявлены все спецификации, необходимо выполнить процесс верификации ИС. Его результатом является множество сценариев злоумышленника.

Если в ходе их построения возникают ошибки, то необходимо пересмотреть модель и цели злоумышленника, т.к. возможно они построены неверно.

Тогда верификация модели ИС при заданной модели злоумышленника будет заключаться в построении множества контрпримеров – библиотеки сценариев агентов-злоумышленников.

Верификация модели темпоральной логики выглядит следующим образом: пусть имеется модель Крипке $M=(S,R,L)$ и формула темпоральной логики f , тогда требуется найти множество $\{s \in S \mid M, s \models f\}$. Данную задачу и решает верификация моделей.

Процесс верификации модели в CTL можно описать следующим образом. Пусть задана модель Крипке $M=(S,R,L)$ и требуется определить, на каких состояниях из S выполняется CTL-формула f . Работа алгоритма заключается в приписывании каждому состоянию s множества $label(s)$ тех подформул формулы f , которые становятся истинными в состоянии s . Первоначально $label(s)$

совпадает с $L(s)$, Далее совершается ряд шагов, где на шаге i обрабатываются подформулы, в которых глубина вложенности CTL-операторов равна $i-1$. После обработки подформулу добавляют к разметке всех тех состояний, в которых она истинна. По окончании работы алгоритма соотношение $M, s \models f$ имеет место в том и только том случае, когда $f \in \text{label}(s)$. Аналогичным образом описывается алгоритм верификации логики LTL. Сложность алгоритма верификации моделей для CTL или LTL с явным представлением состояний имеет линейную сложность, как по размеру графа модели, так и по длине формулы, и равно $O(|f|(|S|+|R|))$ для произвольной формулы f .

В процессе верификации модели ИС проверяется выполнение всех возможных целей злоумышленников. Однако сценарии строятся исходя из заданных начальных условий. Модель ИС потенциально может содержать большее число уязвимостей, чем будет выявлено по результатам верификации.

Для решения задачи построения модели ИС в терминах разработанной спецификации необходимо иметь сведения о:

- объектах ИС;
- уязвимостях ИС;
- ресурсах ИС;
- модели злоумышленника.

При построении модели ИС учитываются только те компоненты, которые необходимы для реализации найденных уязвимостей.

Структура отдельного модуля в общем случае представлена в листинге 3 [70].

Листинг 3 – Структура модуля SMV

MODULE имя

VAR

Список переменных.

ASSIGN

Начальные значения переменных.

Преобразования переменных.

(Описание модели Крипке).

DEFINE

Переопределение имен переменных.

FAIRNESS

Ограничение справедливости.

TRANS

Преобразования переменных.

SCEC

CTL-спецификация.

LTLSPEC

LTL-спецификация.

Модуль MODULE Main является вершиной иерархии и представляет собой абстракцию сетевой инфраструктуры ИС. Каждый компонент ИС можно вынести в отдельный модуль, что упростит разработку модели ИС.

Начало каждого модуля обозначается служебным словом MODULE. Имена остальных модулей выбираются пользователем. Описание пользовательских модулей удобней выносить в отдельный файл или располагать вслед за описанием модуля Main. В модули можно передавать параметры. Также существует конструкция, позволяющая выполнить возвращение значения.

Переменные данного модуля – множество хостов Host, множество флагов Vul (выполнение постусловий уязвимости) и множество флагов получения доступа к ресурсам Res. При построении данного множества следует пользоваться правилами понижения сложности модели. В данный модуль выносятся флаги, относящиеся к свойствам всей ИС – факт известности паролей пользователей домена и др.

Множество хостов подразумевает синхронную композицию, т.е. за каждый шаг системы процессы на хостах выполняются одновременно, а не пошагово друг за другом.

Каждый $host_i$ выделяется в отдельный дочерний модуль, в котором производится описание компонент множества Component. Сами компоненты также, как и в предыдущем случае, не детализируются. Каждый компонент представляется отдельным классом (модулем). Его экземпляры включаются в состав хоста.

Синтаксис объявления модуля: `MODULE <Obj>`.

Такой подход позволяет создать, к примеру, модуль (класс) компонента Antivirus или User, после чего внести его во все хосты как экземпляр. В случае, если один компонент зависит от другого (существует зависимость f), то можно вынести эту зависимость в вышележащий модуль.

Компоненты внутри хоста предполагают интерлинговую композицию, т.е. за один шаг системы будет выполнен только 1 компонент хоста, после чего управление перейдет к следующему компоненту. При объявлении компонентов обязательно следует служебное слово process.

Компонент определяется как совокупность дочерних компонент, свойств Property, где каждому элементу этого множества в соответствие ставится множество значений Value. Каждое свойство процесса может состоять из нескольких доступных состояний. К примеру, для компонента User одно из свойств будет LogAs, а множество его значений определяться множеством возможных состояний данной переменной {none,guest,user,root}.

Компонент представляет собой абстракцию некоторого реально существующего в ИС объекта. К компонентам относятся:

- устройства (портативные устройства хранения данных, USB-ключи, программно-аппаратные компоненты СЗИ);
- программы (антивирусные средства, СУБД, удаленные средства администрирования, Web-сервера и сайты);
- модель пользователя (процедуры авторизации, флаги знания паролей учетных записей, права доступа пользователя).

Свойства компонент и их значения могут быть различными: «вкл/выкл», «get/put» и пр.

Множество хостов, компонентов, свойств и перечня их значений идентичен. Они определяются сразу за объявлением модуля MODULE в блоке VAR.

Синтаксис их объявления представлен в листинге 4.

Листинг 4 – Синтаксис объявления объектов

VAR

```
<имя_хоста> : hosti;
<имя_компонента> : process componentj;
propertyk : {valuekw};
```

Затем производится выделение начального состояния S_0 , задающего режим работы ИС при инициализации. Начальные значения каждого свойства задаются в блоке ASSIGN и имеют следующий синтаксис, представленный в листинге 5.

Листинг 5 – Синтаксис присвоения начальных значений

ASSIGN

```
init(propertyk) := valuekw;
```

Начальные состояния непосредственным образом определяются из модели злоумышленника, т.к. начальное состояние ИС определяют начальное состояние всех сценариев. Изменение начального состояния ИС приводит к изменению модели злоумышленника.

Процесс функционирования – логика изменения значений свойств. Описание процессов функционирования производится только для модулей компонент. Процессами являются множество функций f .

Функция f представляет собой описание процессов и представляется в виде:

— Логических ограничений «если-то», накладываемых на функционирования компонент и описывающих логику процессов компонент (изменений значений свойств). Реализуются блоком ASSIGN. Имеет 2 вида синтаксиса (листинг 6).

Листинг 6 – Синтаксис описания изменений состояний

ASSIGN

```
next(propertyk) := valuekw;
```

ASSIGN

```

next(propertyk): =
case
propertym ⟨оператор⟩ valuemn [⟨оператор⟩ ...] : {valuekw};
esac;

```

— Множество условий в виде двоек $\langle \textit{property}_k, \textit{value}_{kw} \rangle$, выполнение которых (истинность) невозможно, ограничивающих переходы R и влияющих на поведение модели. Реализуются блоком FAIRNESS (листинг 7).

Листинг 7 – Синтаксис контрольных ограничений

FAIRNESS

```

propertyk ⟨оператор⟩ valuekw [⟨оператор⟩ ... ];

```

— Множества логических и арифметических выражений, значения которых пересчитываются на каждой итерации модели и определяют значения свойств (переменных). Реализуется блоком DEFINE.

— Множества заданных переходов модели компонента, происходящих каждый раз, когда выполняется указанное условие. Реализуется блоком TRANS.

— Порядком выполнения компонент и процессов – синхронным или асинхронным. Реализуется в виде служебного слова process для переменных модуля, или служебной комбинацией FAIRNESS running. [71]

Важной особенностью является то, что один процесс может непосредственно влиять на функционирование другого процесса. Это реализуется с помощью передачи одних компонент или ее свойств в другой компонент в виде аргументов модуля. Однако, такая возможность должна реализовываться для всего класса, а не для отдельного экземпляра.

Данную возможность можно использовать следующим образом. Если в ИС выделяют сервера и клиентские машины, то в серверную машину можно передать в качестве аргументов экземпляры клиентов. В результате данной операции, клиентские процессы могут управляться серверными процессами. К примеру,

централизованное отключение сетевого антивируса приведет к отключению клиентских антивирусов.

Для анализа модели в каждый компонент необходимо вводить свойство Step. Его значения задаются чисел от 0 до n. Значение n не следует делать большим. На каждом шаге данная переменная итерируется. По ее изменениям определяется, который компонент выполнялся на текущем шаге модели.

Для более полного представления процессов рекомендуется вводить в комментарии описание возможных действий, необходимых для перевода ИС из состояния в состояния. Комментарии к переходам системы оставляется в строке, в которой описан переход. Лучше всего их выражать в виде термов: уязвимостей, действий Connect(), Login(), Read(), Write(), и т.д.

Факт успешной эксплуатации уязвимости представляется в виде флага. Флаг принимает истинное значение тогда и только тогда, когда условие, описывающее постусловие данной уязвимости истинно.

Получение доступа к какому-либо ресурсу ИС представляется в виде отдельного состояния ИС. Каждому нарушению в соответствии определяется флаг $property_k$, значения которого могут принимать либо истинное (TRUE), либо ложное значение (FALSE). Тогда эксплуатация той или иной уязвимости будет изменять значения нарушений для некоторых ресурсов в истинное состояние.

Предложенная модификация позволяет уменьшить трудоемкость моделирования действий злоумышленника. Также, при разработке такой модели становится прозрачной архитектура ИС.

2.3.1 Структура библиотеки сценариев злоумышленников

Для описания полученных сценариев библиотеки агентов предполагается использование языка разметки XML [72-74]. Использование такого представления является наиболее удобным для последующей его программной обработки.

Библиотека сценариев включает в себя:

— множество вершин S;

- множество дуг R ;
- множество характеристик сценариев.

Множество вершин представляет собой множество мгновенных состояний ИС.

Множество дуг определяют взаимосвязь между вершинами и хранят действия, необходимые для изменения состояния.

Сценарий представляют собой некоторую последовательность вершин (s_0, \dots, s_n) .

Для хранения каждой секции в XML-спецификации библиотеки сценариев необходимо выделить корневые элементы.

Т.к. число состояний в библиотеки сценариев будет меньше возможных состояний ИС, и, исходя из того, что она формируется на основе сценариев, построенных верификатором, то более удобным представлением является объединение множества вершин и множества дуг в единую структуру, где с каждой вершиной ассоциируются множество исходящих из нее дуг. Такое представление можно ассоциировать как строку матрицы смежности.

Отдельная вершина в сценарии выделяется корневыми элементами (листинг 8).

Листинг 8 – Описание вершины.

```
<node id="">
</node>
```

где атрибут id – идентификатор вершины.

Внесение всех вершин сценариев формируют полное описание библиотеки сценариев агентов.

Отдельная дуга выделяется корневыми элементами (листинг 9)

Листинг 9 – Описание дуги.

```
<path next-node="">
</path>
```

где атрибут `next-node` хранит идентификатор `id` следующей вершины. Для каждой вершины описывается множество исходящих дуг. Множество исходящих дуг объединяются корневыми элементами `<paths></paths>`. Описание состояний производится между корневыми элементами `<state></state>`.

Описание состояния `state` включает в себя следующую композицию иерархически вложенных элементов:

- хост;
- компоненты хоста;
- свойства компонент и их значения.

В XML данное описание, отображающее состояние одного хоста, представлено на листинге 10.

Листинг 10 – Представление хоста и вложенных компонент.

```
<host name="">
  <component name="">
    <property name="" value="" />
  </component>
</host>
```

Одному состоянию ИС соответствует одна вершина в библиотеке сценариев.

Атрибут `name` хоста содержит имя хоста, используемое в модели ИС. Хост описывается множеством своих компонентов. Атрибут `name` компонентов отображает имя компонента, используемое в модели ИС. Каждый компонент описывается множеством своих свойств и их значениями. Имя свойства заносится в атрибут `name`, значение свойства в атрибут `value`.

Дуга описывается двумя элементами:

- множеством описаний `description`, описывающих эту дугу;
- множеством действий `instruction`, необходимых для перехода в следующее состояние.

Описание содержит `description` названия уязвимости (если есть), или ее пояснение и может быть пустым. Каждое именование или пояснение заносится между тегами `<note></note>`.

Действия `instruction` представляют собой последовательность шагов, выполнение которых переведет ИС в следующее, указанное в атрибуте «`next-node`» состояние. Каждый шаг описывается между тегами `<step></step>`. В качестве шагов выступает программный код или логическая последовательность действий, выраженная в терминах.

Результирующее описание одного узла графа сценария выглядит следующим образом (листинг 11).

Листинг 11 – Результирующее описание одного узла графа.

```
<node id="">
  <state>
    <host name="">
      <component name="">
        <property name="" value="" />
      </component>
    </host>
  </state>
  <paths>
    <path next-node="">
      <descriptions>
        <note></note>
      </descriptions>
      <instructions>
        <step></step>
      </instructions>
    </path>
    ...
  </paths>
</node>
```

Одно такое описание вершины, по сути, хранит всю строку матрицы смежности.

Каждый сценарий в библиотеки обозначается корневым элементом `path` и имеет свой идентификатор `id` (`<path id="">`). Тегам `<target></target>` описывается цель злоумышленника. Между тегам `<risk></risk>` указывается риск по данному сценарию. Данное значение используется для выбора приоритетного сценария из библиотеки. Маршрут сценария описывается вложенными в него тегам `<step></step>`, содержимое которых представляет собой идентификатор вершины `node id`. Каждый сценарии содержит поле `target`, в которых указывается цель злоумышленника.

Результирующее описание заголовка сценария (листинг 12).

Листинг 12 – Результирующее описание заголовка сценария.

```
<path id="">
    <target></target>
    <risk></risk>
    <step></step>
</path>
```

В библиотеке сценариев маршруты сценариев и состояния информационной системы выделены в группы.

Маршруты всех построенных сценариев записываются между корневыми элементами `<scenarios_paths></scenarios_paths>` в заголовке библиотеки сценариев.

После заголовка между тегам `<rows></rows>` следует описание вершин сценариев. [75]

Такое представление сценариев позволяет проводить над библиотекой сценариев любые операции (добавление новых сценариев; удаление сценариев; изменение существующих сценариев) в автоматизированном режиме. Также, данное представление упрощает анализ сценариев и их использование для построения множеств агентов, реализующих их.

2.3.2 Выбор инструментального средства верификации модели

Для проведения задачи верификации необходимо выбрать инструментальное средство, предназначенное для проверки того, что система переходов с конечным числом состояний удовлетворяет требованиям, заданным на языке темпоральной логики.

Требования к верификаторам формулируются такими же, как и к любому программному обеспечению, расширяя их специфическими для верификаторов критериями сравнения, исходя из позиционирования на рынке и особенностей той или иной реализации. Предлагаются следующие параметры сравнения:

- язык описания модели;
- язык спецификации модели (вид используемых логик);
- интерфейс;
- лицензия;
- особенности верификатора;
- используемые разработки (данный критерий связан с тем, что часто один верификатор является развитием какого-либо другого верификатора).

Среди исследуемых верификаторов выбраны следующие [70,76-79]:

- CMU SMV (первый верификатор общего назначения);
- Cadence SMV (разработка на базе предыдущего, активно поддерживается);
- NuSMV (результат глубокой переработки CMU SMV, символьный верификатор);
- FSAP/NuSMV-SA (независимое улучшение NuSMV);
- SPIN (один из наиболее популярных верификаторов);
- PRISM (вероятностный верификатор).

Рассматривая их по порядку, можно отметить, что CMU SMV – первый символьный верификатор, один из верификаторов общего назначения. Его достоинством является то, что он изначально поддерживал полноценную генерацию контрпримера, обладал встроенным специализированным языком

описания модели. На сегодняшний день безнадежно устарел. Вторым его недостатком является то, что на ряде моделей он имел свойство входить в бесконечный цикл. Консольный. Разработка университета Карнеги-Меллон. Не обновляется.

Верификатор Cadence SMV является дальнейшим улучшением CMU SMV. Поддерживается до сих пор. К сомнительным аспектам стоит отнести, что нигде не заявлено о решении проблем с заикливанием. Имеет графический интерфейс. Разработка Национальной лаборатории Лоуренса Беркли. Основная проблема, связанная с выбором используемого программного обеспечения – его доступность. Данный верификатор - закрытая разработка, распространяется по лицензии FUSP (Full Usage of Sub-Channel). Поддерживается и обновляется.

NuSMV – символьный верификатор, являющийся глубокой переработкой CMU SMV. В него была добавлена возможность использовать язык SAT для представления моделей. Также, для решения проблем верификации (заикливания), ее ускорения, в нем реализована концепция BDD. Распространяется свободно с открытым исходным кодом. Интерфейс консольный. Совместная разработка ITC-IRST, университета Карнеги-Меллон, университета города Генуя, университета города Тренто. Поддерживает расширения. Существует независимая графическая реализация. Поддерживается и обновляется. Обладает наиболее мощными средствами по реализации модели Крипке и формированию условий на языке темпоральной логики.

FSAP / NuSMV-SA – серьезная доработка NuSMV v.2. Во много раз улучшена скорость работы, внесено множество дополнений в виде возможности использования паттернов, классов сообщений, логического анализа, введена поддержка XML, улучшен механизм BDD. Обладает графическим интерфейсом. Однако распространяется только по подписке для членов проекта ESACS/ISAAC.

SPIN – пожалуй, это самый известный и самый распространенный верификатор моделей. Применяется для анализа программного кода, хотя возможно применение для моделей общего характера. Язык описания модели – Promela. Обладает графическим интерфейсом. К его достоинствам можно отнести

еще очень высокую производительность, превосходящую SMU SMV на сотни порядков. Однако не всегда способен генерировать контрпример. В последних версиях, в подавляющем большинстве случаев, лишь способен сообщить об истинности или ложности формул. Также, в основном позиционируется как средство верификации C/C++ программ (где его недостатки в целом нивелируются из-за особенностей представления программ – контрпример в этом случае просто бесполезен). Разработка лабораторий Белл авторов ОС Unix. В настоящее время разрабатывается отдельной компанией.

PRISM – вероятностный верификатор, разработанный в Оксфордском университете. Поддерживает верификацию Марковских цепей с дискретным и непрерывным временем, вероятностных автоматов и вероятностях временных автоматов с расширением всех вышеперечисленных моделей функциями стоимости и вознаграждения. Для описания моделей используется встроенный язык PRISM. Средство регулярно обновляется и поддерживается. Однако данный верификатор узкоспециализирован для исследований только вероятностных моделей – предлагаемые средства не подходят для анализа дискретных моделей.

В таблице 2 приведен результат сравнения верификаторов моделей.

Функционально самым мощным верификатором можно назвать FSAP/NuSMV-SA и PRISM, но первый он является закрытой внутренней разработкой, второй – узкоспециализированным вероятностным верификатором. Поэтому наилучшим выбором видится использование верификатора NuSMV. При некоторых недостатках, он наилучшим образом подходит для решения данных задач.

Единственным серьезным недостатком данной системы является отсутствие встроенной поддержки графического интерфейса пользователя. В тоже время, NuSMV обладает графическим расширением gNuSMV, а консольный вариант полноценно интегрируется в среду Eclipse и способен функционировать из-под нее. Также данный верификатор позволяет подключать дополнительные модули расширения [80].

| Продукт | Язык описания модели | Язык спецификации модели | Интерфейс | Лицензия | Особенности | Используемая разработка |
|-----------------|----------------------|--------------------------|-----------------------------|-------------------------|--|-------------------------|
| CMU SMV | SMV | CTL | Консольный | GNU GPL | Подвержен заикливанию | - |
| Cadence SMV | SMV | CTL | Графический | FUSC | Подвержен заикливанию | CMU SMV |
| NuSMV | SMV / SAT | CTL/LTL | Консольный / Графический | LGPL v2 | Поддерживает расширения | CMU SMV |
| FSAP / NuSMV-SA | SMV / SAT / XML | CTL/LTL | Графический | FUSC / Проприетарная | - | NuSMV |
| SPIN | Promela | LTL | Графический | GNU GPL / FUSC | Невозможна генерация контрпример | - |
| PRISM | PRISM Language | PCTL, CSL, LTL, PCTL* | Консольный / Графический | GNU GPL | Вероятностный, поддержка Марковских цепей вероятностных автоматов, поддерживает расширения | - |

Таблица 2 – Сравнение программных средств верификации моделей

Для решения прикладных задач по автоматизированному формированию сценариев действий агентов по полученным результатам работы SMV используется высокоуровневый язык программирования C#.

2.4 Оценка защищенности информационной системы

Для каждой выявленной не перекрытой уязвимости vul_j определен уровень критичности в соответствии с методикой Common vulnerability scoring system calculator version 2. Для каждого ресурса ИС res_i в соответствии с экспертной оценкой владельца ИС определена критичности i -го ресурса ИС. [81,82]

Тогда защищенности i -го ресурса от j -ой уязвимости Sec_{ij} будет определяться в соответствии с таблицей 3.

Таблица 3 — Защищенность i -го ресурса ИС от эксплуатации j -ой уязвимости

| Уровень критичности j -ой уязвимости | Уровень критичности i -го ресурса ИС | | |
|--|--|---------|---------|
| | низкий | средний | высокий |
| низкий | высокая | высокая | средняя |
| средний | высокая | средняя | низкая |
| высокий | средняя | низкая | низкая |

Тогда, зная защищенность всех ресурсов ИС от всех не перекрытых уязвимостей, можно определить защищенность всей ИС Sec :

$$Sec = \min_{ij} Sec_{ij}. \quad (15)$$

Защищенность ИС соответствует наименьшей защищенности ее ресурсов, т.к. злоумышленник может эксплуатировать именно ту уязвимость, которая и определяет наименьший уровень защищенности ресурсов ИС, и получить несанкционированный доступ к ресурсу, исказить, разрушить или нарушить его функционирование.

2.5 Вывод по второй главе

Приведено описание предложенного метода анализа защищенности ИС на основе полунатурного моделирования и многоагентного подхода.

Определена спецификация модели ИС. Предлагается использовать темпоральную логику, которая является языком формулирования утверждений, использующих понятия времени и модель Крипке, как разновидность графа переходов.

Даны рекомендации по упрощению построенной модели ИС. Предложена методика сбора исходных данных об ИС, необходимых для формирования модели.

Определена спецификация модели злоумышленника. Для формирования модели необходимо выполнить верификацию модели ИС и выделенных целей злоумышленника. Верификация модели ИС при заданной модели злоумышленника будет заключаться в построении множества контрпримеров – библиотеки сценариев агентов-злоумышленников. Определена структура библиотеки сценариев злоумышленника.

Из существующих инструментальных средств, решающих задачу верификации, выбран символьный верификатор модели NuSMV, т.к. он обладает:

- наиболее мощным языком для представления модели ИС;
- автоматизированным построением сценариев при невыполнении условий темпоральной логики.

Определен порядок получения оценки защищенности ИС, основанный на полученных оценках защищенности отдельного ресурса системы от эксплуатации некоторой уязвимости.

3 МНОГОАГЕНТНАЯ СИСТЕМА ДЛЯ АНАЛИЗА ЗАЩИЩЕННОСТИ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Многоагентная модель для моделирования злоумышленных воздействий на информационную систему

Многоагентная модель состоит из множества интеллектуальных агентов, взаимодействующих между собой и с ИС (внешней средой). Агенты, отвечающие за решение определенного круга задач, объединяются в группы. Множества агентов, из которых строятся группы, не обязаны быть непересекающимися [83]. Каждый агент обладает ограниченным набором методов, определяющих сферу его влияния на внешнюю среду (рисунок 5).

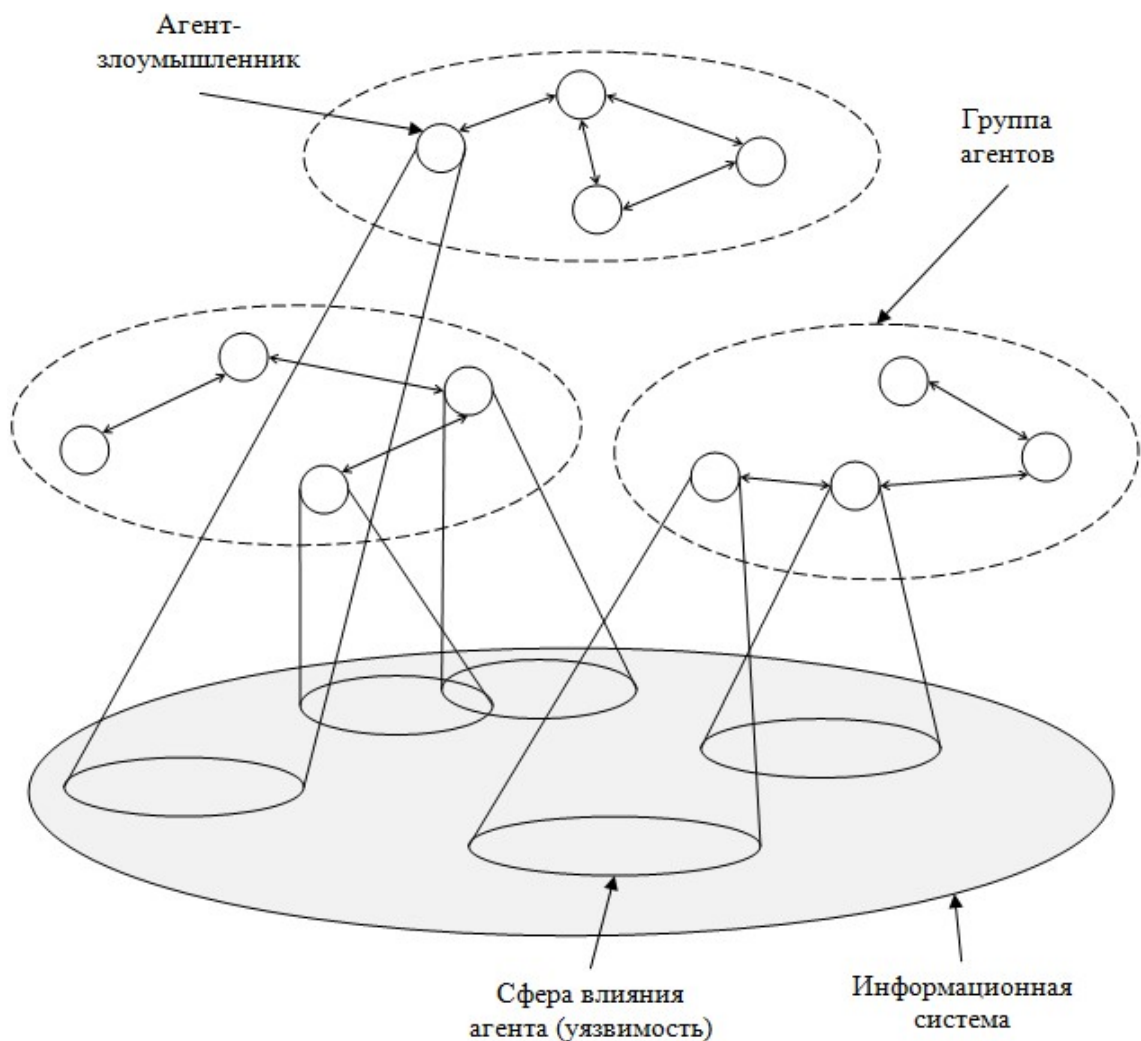


Рисунок 5 – Структура многоагентной модели

Интеллектуальный агент представляется в виде функции, проецирующей любой подходящий результат восприятия на действие, которое этот агент способен выполнить, либо в коэффициент, элемент обратной связи, функцию или константу, способную повлиять на дальнейшие действия. [84]

За прошедшие десятилетия появилось множество моделей, пытающихся описать поведение интеллектуального агента. Первой фундаментальной работой в этой области стала Belief-Desire-Intention, затем возникли такие модели как Soar, TouringMachines, InteRRaP, Cougaar и PECS [85 - 91].

Анализ указанных моделей предлагается выполнить на основе критериев:

— возможность имитации рационального поведения злоумышленника (под рациональностью понимается обеспечение максимального выигрыша при минимальных затратах ресурсов нарушителя [85]);

— возможность имитации деятельности группы злоумышленников;

— масштабируемость.

Soar является узкоспециализированной моделью, представляет собой псевдоформальный язык и подходит лишь для решения задач когнитивной эргономики. С помощью Soar можно ответить на вопрос, сколько шагов потребуется нарушителю, чтобы реализовать некоторое злоумышленное воздействие на информационную систему, однако получить подробное императивное описание этих шагов нельзя [87].

Агент, построенный в соответствии с моделью TouringMachines, представляет собой конечный автомат, переходы между состояниями которого контролируются специальным механизмом, имитирующим рациональное мышление. Данная модель подходит для решения простейших задач имитации человеческого поведения, с её помощью можно воссоздать деятельность злоумышленника, атакующего информационную систему. Имитация деятельности группы злоумышленников с помощью TouringMachines затруднительна, поскольку модель не специфицирует принципы взаимодействия агентов и не уделяет внимания социальной активности агента-нарушителя [88].

InteRRap, в отличие от TouringMachines, делает акцент на социальном поведении агента: он не обособлен, а функционирует в социальной среде и способен делиться своими знаниями и целями с другими агентами-нарушителями. Это позволяет имитировать деятельность групп злоумышленников. Однако сама модель имеет сложную многослойную структуру и плохо масштабируется: незначительное увеличение количества нарушителей ведет к серьезному увеличению требований к вычислительным ресурсам аппаратного обеспечения для работы модели [89].

Модель Cougaar предназначена для решения задач планирования процессов управления какой-либо системой. Из-за специфичного предназначения модели использовать её для имитации злоумышленных воздействий на информационную систему невозможно [90].

PECS базируется на идеях, выдвинутых InteRRap, и обладает теми же недостатками: плохой масштабируемостью. Акцент в этой модели сделан на персонализацию внутреннего мира агентов, что позволяет упростить моделирование злоумышленных групп, имеющих определённую иерархическую структуру, т.е. когда одни злоумышленники управляют действиями других [91].

Интеллектуальные агенты, построенные согласно модели Belief-Desire-Intention (далее BDI), также имитируют рациональное мышление злоумышленников. Однако, в отличие от анализировавшихся выше моделей, структура BDI-агента проста, её практическая реализация не требовательна к вычислительным ресурсам. Это говорит о хорошей масштабируемости результирующей многоагентной системы. Кроме того, BDI уделяет значительное внимание социальному аспекту. Всё это позволяет имитировать деятельность больших групп злоумышленников.

Таким образом, BDI лучше всего удовлетворяет заявленным выше критериям. Далее указанная модель анализируется более детально.

BDI, в отличие от конкурирующих подходов, великолепно проработана на фундаментальном уровне, что позволило создать её практически безупречное логико-математическое представление, а в последствии — реализовать язык

программирования для описания работы компонентов многоагентной системы — AgentSpeak [92-94].

Основой практически всех общепризнанных подходов к построению и функционированию многоагентных систем является BDI. Из внушительного списка программных комплексов, построенных на её основе, особого внимания заслуживают Procedural Reasoning System (RPS), Distributed Multi-Agent Reasoning System (dMARS), а также — среды разработки JACK Intelligent Agents и Jason.

Модель BDI, появившаяся в середине восьмидесятых годов прошлого века, реализует концептуальные принципы философской теории Майкла Бретмана о целевом программировании человеческого сознания [92,95].

Согласно этой теории, в основе механизма, побуждающего субъект к некоторому действию, лежат три основных ментальных аспекта: убеждения, желания и намерения.

Применительно к интеллектуальному агенту-злоумышленнику убеждения могут быть интерпретированы как совокупность знаний об окружающей действительности, т.е. состоянии исследуемой информационной системы. Желания — как множество потенциальных целей агента-злоумышленника. Намерения — как совокупность сценариев по реализации желаний, т.е. атаке системы.

Идея разработки сложной программной системы в рамках терминов убеждение, желание, намерение представляет собой ключевой момент модели BDI, концептуальные основы которой были заложены Иовом Шохамом в работе, посвящённой агентно-ориентированному программированию [96].

Агентно-ориентированный подход можно расценивать как одну из форм «пост-декларативного» программирования [97]. Классическая процедурная разработка программного обеспечения предполагает пошаговое описание каждого действия, которое способна выполнять система. Ввиду этого процедурная разработка сложных систем является нетривиальной задачей. Декларативное программирование (пример — язык Пролог) пытается бороться с указанной проблемой, описывая какво нечто, а не как его создать. Цель такого подхода

заключается в минимизации контроля за процессом работы системы: системе объясняется некоторая задача, которую необходимо решить, после чего встроенный в язык механизм контроля самостоятельно определяет, что нужно сделать, чтобы это решение найти [97, 98].

Парадокс заключается в том, что для эффективного использования декларативного подхода необходимо хорошо знать устройство механизма контроля, что явно идёт в разрез с главной целью рассматриваемой парадигмы [99].

Цель агентно-ориентированного подхода к построению систем та же, что и у декларативного: определить набор целей и позволить контролирующему механизму самостоятельно решить, как их достичь. Достоинство агентной методики в том, что контролирующий механизм представляет собой совокупность интеллектуальных агентов, структура и принципы работы которых основаны на интуитивном восприятии человеком таких понятий, как убеждение, желание и намерение. Таким образом, для понимания работы контролирующего механизма в агентно-ориентированном языке требуется минимум усилий, поэтому агентно-ориентированный подход и относят к разряду «пост-декларативных».

Функционирование интеллектуального агента с некоторым набором убеждений, желаний и намерений определяется следующей формулой:

$$f(B, D, Res) = I, I: \lim_{Res \rightarrow \min_{\forall I} Res} g(B, D, I, Res) = \max_{\forall I} g(B, D, I, Res), \quad (16)$$

где B — множество убеждений (знания агента-злоумышленника о состоянии информационной системы), D — множество желаний (цели агента-злоумышленника по эксплуатации уязвимостей информационной системы), I — множество намерений (сценарии эксплуатации уязвимостей), f — рациональность (функция, обеспечивающая максимальный выигрыш (g) при минимальных затратах ресурсов (Res) агента).

Процесс использования интеллектуальным агентом формулы (16) называется рассуждением (*practical reasoning*) [65]. Рассуждение состоит из двух этапов:

- взвешивание (*deliberation*);
- определение пути достижения намерения (*means-ends reasoning*).

В процессе взвешивания агент, основываясь на текущих убеждениях и желаниях, выбирает намерение. Важным свойством намерения является то, что, будучи выбранным, оно непосредственно влияет на дальнейший ход рассуждения.

Например, до тех пор, пока агент убеждён, что выбранное им намерение достижимо и актуально, он не будет совершать действия, противоречащие текущему намерению. Говоря проще, если злоумышленник намерен извлечь из ИС некоторую информацию с помощью заранее внедрённой закладки, он не будет совершать действия по удалению этой закладки.

Намерения тесно связаны с убеждениями агента о предполагаемом будущем. В частности, намерение совершить что-либо говорит о том, что агент верит в принципиальную осуществимость выбранного действия и, более того, в то, что в благоприятных условиях это действие будет успешно совершено. С другой стороны, агент также может верить и в то, что нечто непредвиденное может помешать реализации намерения. Возвращаясь к примеру с закладкой: если злоумышленнику удалось внедрить в ИС закладку, то у него появляется убеждение в возможности извлечения интересующей информации. В то же время злоумышленник может обладать убеждением, что какой-либо компонент защиты ИС может заблокировать внедренную закладку до того, как информация будет получена.

На втором этапе агент решает, как реализовать выбранное намерение, используя тот набор действий, которые он способен выполнять в окружающей среде. В области изучения проблем искусственного интеллекта этот этап более известен как планирование [100]. Планирование позволяет достичь заданной цели путём перебора различных вариантов.

Планирование относится к классу PSPACE-полных задач, т.е. его сложность выше, чем у NP-сложных задач [101,102]. В виду высокой вычислительной сложности подобные алгоритмы планирования не могут быть использованы в системах, предполагающих функционирование в масштабах реального времени.

Альтернативные подходы, предназначенные для использования на втором этапе, предполагают отказ от планирования с нуля в процессе функционирования агента.

Примером одного из таких подходов является разработка так называемой библиотеки частичных планов [100]. Идея заключается в том, что на этапе создания многоагентной системы разработчик подготавливает некоторый набор реакций агента на происходящие события. Задача агента — связать имеющиеся в библиотеке последовательности действий воедино, чтобы достичь заданной цели. Данный метод обладает меньшей гибкостью, но он хорошо зарекомендовал себя на практике [100, 103].

Применительно к защите информации, указанный подход предполагает, что специалист должен описать основные этапы эксплуатации каждой уязвимости, а также научить агента выявлять взаимосвязь между различными уязвимостями. После этого встроенный механизм контроля агента самостоятельно будет генерировать, и реализовывать различные векторы атак на исследуемую ИС.

3.1.1 Программный каркас Procedural Reasoning System

Одной из лучших реализаций BDI и модели рассуждений агента в виде программного каркаса является Procedural Reasoning System (PRS) [104]. PRS — прародитель агентно-ориентированного языка AgentSpeak, являющегося передовым воплощением BDI. Анализ структуры PRS направлен на выявление основ, которыми руководствовались разработчики AgentSpeak.

Процесс анализа архитектуры PRS удобно представить в итеративной форме. В первом приближении управляющий цикл программы агента-злоумышленника содержит следующие важные элементы:

- слежение за состоянием окружающей среды (информационной системы), корректировка убеждений на основе полученной информации;
- взвешивание, выбор приоритетного намерения (какую угрозу стоит реализовать);

- построение плана достижения намерения (как реализовать выбранную угрозу);
- выполнение плана (атака системы).

При детальном рассмотрении данной структуры выявляется ряд недостатков. Поведение агента-злоумышленника жёстко привязано к однажды выбранному намерению, т.е. сформированный план будет выполняться до тех пор, пока намерение не будет реализовано. Если в процессе выполнения плана во внешней среде произойдут события, делающие выбранное намерение недостижимым или бессмысленным, то агент никак на это не отреагирует.

Управляющий цикл должен быть сбалансирован таким образом, чтобы агент-злоумышленник, выполняя поставленную задачу, оставался в курсе происходящих в ИС событий и был способен корректировать план своих действий.

Псевдокод, приведённый в листинге 13, учитывает обнаруженные недостатки [105]. Этот же алгоритм реализует и злоумышленник: зондирование, подбор инструментария, выявление самого слабого звена в составе ИС, построение сценария атаки.

Листинг 13 — Основной цикл программы агента-злоумышленника

```

1. В = <начальные убеждения агента>
2. I = <начальные намерения агента>
3. цикл пока агент_запущен ()
4.     d = получить_данные_с_сенсоров ();
5.     В = обновить_убеждения (В, d);
6.     D = выбрать_желания (В, I);
7.     I = выбрать_намерения (В, D, I);
8.     p = сформировать_план (В, I, Ас);
9.     цикл пока не (выполнен (p) или успех (I, В)
10.        или невозможно (I, В))
11.         x = взять_первый_элемент (p);
12.         выполнить (x);
13.         p = удалить_первый_элемент (p);
14.         d = получить_данные_с_сенсоров ();
15.         В = обновить_убеждения (В, d);

```

16. если пересмотреть (I, B)
17. D = выбрать_желания (B, I);
18. I = выбрать_намерения (B, D, I);
19. кц если
20. если не соответствует (p, I, B)
21. p = сформировать_план (B, I, Ac);
22. кц если
23. кц цикл
24. кц цикл

Цикл управления начинается с того, что агент получает от сенсоров информацию о состоянии ИС (строка 4). Далее эта информация используется для корректировки текущих убеждений агента.

На шестой строке агент определяет набор потенциальных целей, используя обновлённые убеждения и текущие намерения. После этого выполняется взвешивание, выбирается следующее намерение и строится план атаки на ИС.

Вложенный цикл (строки 9 - 23) отвечает за реализацию плана достижения намерения. В идеальном случае агент последовательно берет из плана одно действие и выполняет его. Так продолжается до тех пор, пока план не опустеет, т.е. все действия будут выполнены.

Однако после того как выполнение очередного действия завершено, агент делает паузу (строки 14 - 15), чтобы считать новую информацию с сенсоров и обновить свои убеждения. После этого (строка 16) агент спрашивает себя, не пришло ли время обновить намерения? Если ответ положительный, повторяется процедура взвешивания, корректируются намерения.

Наконец, независимо от того обновлялись ли намерения агента или нет, выполняется проверка актуальности плана на основании текущих убеждений и намерений. В случае неактуальности выполняется процедура повторного планирования (строка 21).

В PRS нет компонентов, отвечающих за планирование в масштабах реального времени. Применяется альтернативный подход: используется

библиотека частичных планов, разрабатываемых специалистом на этапе создания агента.

Каждый план содержит:

- цель — постусловие;
- контекст — предусловие;
- тело — последовательность выполняемых действий.

Цель определяет, для чего план предназначен. Контекст отвечает на вопрос, когда план может быть применён. Отличительная особенность тела плана заключается в том, что это не просто список действий, которые обязан выполнить агент: тело, помимо различных элементарных действий, может содержать множество подцелей. Если в определённом месте тела расположена некоторая подцель, то она должна быть достигнута до того, как будет выполнена оставшаяся часть плана — отчётливо видно применение декларативных принципов. Кроме того, в теле могут встречаться дизъюнктивные композиции целей («достичь *a* или достичь *b*»), циклы («достигать *a*, пока достигается *b*») и т.п.

При запуске агент располагает некоторым набором планов, набором убеждений о внешней среде, а также некоторой приоритетной целью. Убеждения в RPS представляют собой литералы, записанные в предикативной форме.

После запуска приоритетная цель помещается в специальный стек, называемый стеком намерений. Он содержит все цели, ожидающие реализации. Затем выполняется просмотр библиотеки планов на предмет наличия плана, постусловие которого совпадает с целью, находящейся на вершине стека намерений. Выбираются только те планы, контексты которых истинны для текущего набора убеждений. Если окажется, что существуют несколько подходящих планов, для исполнения будет выбран тот, в метаданных которого установлен более высокий приоритет. Исполнение плана может включать внесение новых целей в стек намерений и, как следствие, поиск в библиотеке планов, подходящих для этих целей.

Анализ программного каркаса RPS показывает, что это средство может с успехом применяться для имитации поведения различных злоумышленных

сущностей безотносительно к природе их возникновения. RPS одинаково хорошо применим как для имитации поведения одушевленного злоумышленника, так и для имитации поведения программной закладки.

3.1.2 Контролирующий механизм Belief-Desire-Intention

Контролирующий механизм BDI является неотъемлемой частью декларативного агентно-ориентированного языка AgentSpeak. Поскольку AgentSpeak наследует и развивает идеи RPS, то он может быть применен для решения задач имитации поведения злоумышленника. Достоинством языка является инкапсуляция, сокрытие большинства деталей управляющего механизма BDI [106].

AgentSpeak относится к классу контекстно-свободных формальных грамматик. Его синтаксис удобно описать, используя формы Бэкуса — Наура [107-109]:

$$\begin{aligned}
 ag & ::= bs \ ps \\
 bs & ::= b_1 \dots b_n \quad (n \geq 0) \\
 ps & ::= p_1 \dots p_n \quad (n \geq 1) \\
 p & ::= te : ct \leftarrow h \\
 te & ::= +at \quad | \quad -at \quad | \quad +g \quad | \quad -g \\
 ct & ::= ct_1 \quad | \quad \emptyset \\
 ct_1 & ::= at \quad | \quad \neg at \quad | \quad ct_1 \wedge ct_1 \\
 h & ::= h_1 ; \emptyset \quad | \quad \emptyset \\
 h_1 & ::= a \quad | \quad g \quad | \quad u \quad | \quad h_1 ; h_1 \\
 at & ::= P(t_1, \dots, t_n) \quad (n \geq 0) \\
 at & ::= P(t_1, \dots, t_n)[s_1, \dots, s_m] \quad (n \geq 0, m > 0) \\
 s & ::= percept \quad | \quad self \quad | \quad id \\
 a & ::= A(t_1, \dots, t_n) \quad (n \geq 0)
 \end{aligned} \tag{2}$$

$$\begin{array}{l}
 g \quad := \quad !at \quad | \quad ?at \\
 u \quad := \quad +b \quad | \quad -at
 \end{array}$$

Во введённой грамматике ag используется для обозначения агента-злоумышленника, содержащего набор убеждений bs и библиотеку планов ps (сценариев злоумышленных воздействий). Нетерминал b характеризует отдельное убеждение (какое-то конкретное знание об ИС) и представляет собой литерал, записанный в предикативной форме. Символ p описывает некоторый план, состоящий из инициирующего события (обнаружена уязвимость в ИС) te , контекста ct (злоумышленник констатирует наличие у него инструментария для эксплуатации уязвимости) и тела (сценария для эксплуатации уязвимости) h (символ \emptyset обозначает пустое тело, т.е. отсутствие сценария). Терм S используется для указания источника, сформировавшего убеждение: *percept* — убеждение сформировано на основании данных, полученных от сенсоров (злоумышленник сформировал знание, наблюдая за ИС); *self* — убеждение получено в результате логического вывода; *id* — убеждение получено от другого агента (знание получено злоумышленником от другого злоумышленника, нетерминал id приводится к имени одного из агентов системы). Нетерминалы a , g и u обозначают соответственно функцию действия (элементарный шаг, выполняемый агентом, для эксплуатации уязвимости), цель (описание состояния, в которое агент-злоумышленник должен перевести ИС) и функцию обновления убеждений.

Данная грамматика описывает упрощённую версию AgentSpeak, содержащую ключевые элементы PRS. В частности, работа с метаданными сведена лишь к аннотации источника происхождения убеждения. Из рассмотрения исключены специфичные особенности AgentSpeak, которые не имеют особого значения для дальнейшего анализа семантики языка.

Формально многоагентная система представляется совокупностью конфигураций вида

$$\langle ag, C, M, T, s \rangle, \tag{17}$$

где ag — программа-агент (бот), обладающая набором убеждений bs и библиотекой планов ps ;

C — контекст агента, характеризуемый тройкой объектов $\langle I, E, A \rangle$, где I соответствует множеству намерений $\{i_1, i_2, \dots\}$, E — множеству событий $\{(te_1, i_1), (te_2, i_2), \dots\}$, A — множеству действий, запланированных для выполнения (каждое событие описывается парой (te, i) , где te — идентификатор события, i — намерение, инициировавшее наступление события te);

M — тройка объектов $\langle In, Out, SI \rangle$, где In соответствует множеству входящих сообщений, адресованных агенту ag , Out — множеству исходящих сообщений, отправляемых агентом ag другим агентам-собеседникам, SI — множеству намерений, выполнение которых временно приостановлено (в это множество попадают намерения, дальнейшая реализация которых требует ответа от агента-собеседника);

T — это пятёрка объектов $\langle R, Ap, i, \varepsilon, \rho \rangle$, временно хранящих вспомогательную информацию в течение одного управляющего цикла работы агента. R содержит множество планов, предназначенных для обработки зарегистрированного события ε , Ap содержит подмножество элементов R , обладающих истинным контекстом; i и ρ — соответственно намерение, вызвавшее наступление события ε , и план из множества Ap , выбранный для обработки события ε ;

s идентифицирует текущий этап управляющего цикла, $s \in \{ProcMsg, SelEv, RelPl, ApplPl, SelAppl, AddIM, SellInt, ExecInt, ClrInt\}$. Соответствующая расшифровка для каждого этапа: обработка входящего сообщения, выбор события для обработки, поиск планов для обработки выбранного события, выбор плана с истинным контекстом, внесение нового намерения, а также выбор, реализация и удаление намерения.

Далее в целях удобочитаемости могут быть использованы обозначения:

— если C — контекст агента, то запись C_E обозначает ссылку на компонент E тройки объектов C (аналогичная запись справедлива для любой другой конфигурации);

— запись $i[p]$ обозначает ссылку на план p , находящийся на вершине стека намерений i ;

— если p — план вида $te : ct \leftarrow h$, то $TrEv(p) = te$, и $Ctxt(p) = ct$.

Интерпретатор языка AgentSpeak использует три специализированные функции выбора: s_ε , s_o , s_i — для выбора события из множества C_E , выбора плана из множества T_{Ap} и выбора намерения из множества C_I соответственно. Согласно спецификации языка, интерпретатор предоставляет стандартные реализации этих функций, однако разработчик многоагентной системы может переопределить их на этапе разработки, реализовав собственные функции выбора для каждого агента.

Для определения следующих функций вводится понятие логического следствия [110,111]. Базовый терм at_1 с множеством аннотаций $\{s_1, \dots, s_{1_n}\}$ является логическим следствием из множества базовых термов bs (обозначается $bs \mapsto at_1[s_1, \dots, s_{1_n}]$) тогда и только тогда, когда существует $at_2[s_{2_1}, \dots, s_{2_m}] \in bs$ такое, что $at_1\theta = at_2$ для некоторого унификатора θ , и $\{s_1, \dots, s_{2_m}\} \subseteq \{s_2, \dots, s_{2_m}\}$.

Некоторая подстановка θ является унифицирующей, если в результате её применения различные логические выражения становятся идентичными.

Например, $p(X)[agent_1]$ следует из убеждения $p(t)[agent_1, agent_2]$, т.к. переменная X может быть сведена к терму t , а множество аннотаций $\{agent_1\}$ является подмножеством $\{agent_1, agent_2\}$. В свою очередь, $p(X)[agent_1, agent_2]$ не следует из $p(t)[agent_1]$.

Для заданного множества планов ps и иницилирующего события te множество подходящих планов $RelPlans(ps, te)$ определяется правилом $\{(p, \theta) \mid p \in ps, te \mapsto TrEv(p)\theta\}$.

Например, план с инициирующим событием $+!p(X)[s]$ подходит для обработки события $\langle +!p(t)[s,t], \emptyset \rangle$, т.к. $p(t)[s,t] \mapsto p(X)[s]\theta$: переменная X приводится к значению t , а $\{s\} \in \{s,t\}$.

Для заданного множества подходящих планов R и множества убеждений bs множество применимых планов $AppPlans(bs,R)$ определяется правилом $\{(p,\theta\theta') \mid (p,\theta) \in R, bs \mapsto Ctxt(p)\theta\theta'\}$.

Подходящий план применим, если его контекст является следствием множества убеждений агента.

В AgentSpeak существует особая категория целей (в оригинале — test goals), предназначенных для извлечения информации из множества убеждений агента. Обращается такая цель с помощью специальной функции Test.

Для заданного множества убеждений bs и некоторой формулы at определена функция $Test(bs,at) = \{\theta \mid bs \mapsto at\theta\}$.

Как видно из определения, эта функция возвращает множество подстановок, обращающих at в следствие bs .

Далее определяется операционная семантика языка [112,113]. В процессе работы многоагентной системы агент может находиться в одном из девяти состояний (рисунок 6).

Состояние 1. После старта системы агент-злоумышленник имеет конфигурацию $\langle ag,C,M,T,ProcMsg \rangle$, компоненты C , M и T пусты. Рабочий цикл начинается с состояния ProcMsg, в котором осуществляется обработка сообщений, пришедших от других агентов-злоумышленников.

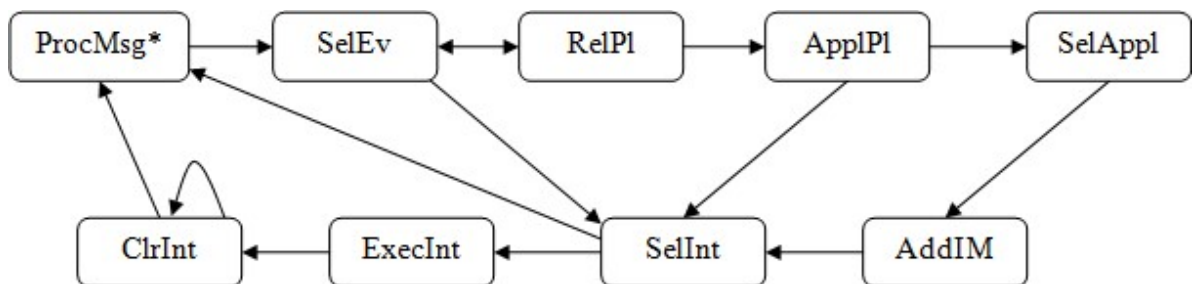


Рисунок 6 — Граф состояний агента

Состояние 2. Выбор события (event selection). Функция выбора события s_ε выбирает некоторое событие (произошедшее в исследуемой информационной системе) из множества E и помещает его в T_ε . Выбранное событие удаляется из E .

$$\frac{s_\varepsilon(C_E) = \langle te, i \rangle}{\langle ag, C, M, T, SelEv \rangle \rightarrow \langle ag, C', M, T', RelPl \rangle}, \quad (18)$$

где $C'_E = C_E \setminus \{\langle te, i \rangle\}$, и $T'_E = \langle te, i \rangle$.

Если оказывается, что зарегистрированных событий нет (состояние ИС не менялось), т.е. множество C_E пустое, то агент переходит в режим выбора приоритетного намерения $SelInt$.

$$\frac{C_E = \{ \}}{\langle ag, C, M, T, SelEv \rangle \rightarrow \langle ag, C, M, T, SelInt \rangle}. \quad (19)$$

Состояние 3. Поиск подходящих планов (relevant plans selection). В этом режиме агент осуществляет поиск подходящих для обработки события планов и размещает найденное множество в T_R .

$$\frac{T_\varepsilon = \langle te, i \rangle \quad RelPlans(ag_{ps}, te) \neq \{ \}}{\langle ag, C, M, T, RelPl \rangle \rightarrow \langle ag, C, M, T', ApplPl \rangle}. \quad (20)$$

Если оказывается, что нет планов, с помощью которых можно было бы обработать событие T_ε , то событие удаляется.

$$\frac{RelPlans(ag_{ps}, te) = \{ \}}{\langle ag, C, M, T, RelPl \rangle \rightarrow \langle ag, C, M, T, SelEv \rangle}. \quad (21)$$

Если агент обнаруживает, что не может найти в ps план для обработки некоторого события (в ИС обнаружена уязвимость, но агент-злоумышленник не знает, как её эксплуатировать), он может попросить других агентов-злоумышленников поделиться подходящим планом. Для этого в AgentSpeak предусмотрена мощная коммуникационная инфраструктура [114-116].

Состояние 4. Поиск применимых планов (applicable plans filtering). Попав в данное состояние, агент выбирает из множества T_R планы с истинным контекстом (в данном состоянии у агента-злоумышленника имеется несколько сценариев

эксплуатации выявленной уязвимости, и требуется выбрать тот сценарий, для исполнения которого имеется нужный инструментарий).

$$\frac{\text{AppPlans}(ag_{bs}, T_R) \neq \{ \}}{\langle ag, C, M, T, \text{ApplPl} \rangle \rightarrow \langle ag, C, M, T, \text{SelAppl} \rangle}, \quad (22)$$

где $T'_{Ap} = \text{AppPlans}(ag_{bs}, T_R)$.

Если таких планов нет (злоумышленник не располагает нужным инструментарием), агент прекращает обработку события и переходит в режим поиска приоритетного намерения.

$$\frac{\text{AppPlans}(ag_{bs}, T_R) = \{ \}}{\langle ag, C, M, T, \text{ApplPl} \rangle \rightarrow \langle ag, C, M, T, \text{SelInt} \rangle}. \quad (23)$$

Состояние 5. Выбор применимого плана (applicable plan selection). Выбор применимого плана из множества, полученного на предыдущем шаге, осуществляется с помощью специальной функции s_o , возвращающей идентификатор и унификатор плана (если существует несколько сценариев эксплуатации уязвимости, и оба сценария обеспечены необходимым инструментарием, агент-злоумышленник выберет тот сценарий, который позволит эксплуатировать уязвимость, затратив наименьшее количество ресурсов агента).

$$\frac{s_o(T_{Ap}) = (p, \theta)}{\langle ag, C, M, T, \text{SelAppl} \rangle \rightarrow \langle ag, C, M, T', \text{AddIM} \rangle}, \quad (24)$$

где $T'_p = (p, \theta)$.

Состояние 6. Внесение плана в сценарий работы агента (adding an intended means). События в AgentSpeak делятся на две категории: внешние и внутренние. Внешние события возникают в результате изменений, происходящих во внешней среде (ИС изменяет своё состояние).

$$\frac{T_\varepsilon = \langle te, \emptyset \rangle \quad T'_p = (p, \theta)}{\langle ag, C, M, T, \text{AddIM} \rangle \rightarrow \langle ag, C, M, T', \text{SelInt} \rangle}, \quad (25)$$

где $C'_I = C_I \cup \{[p\theta]\}$. Так как внешнее событие не имеет явной связи с намерениями агента, выбранный план просто помещается в контекст агента-злоумышленника.

Если же событие является внутренним, то его наступление явно инициировано некоторым намерением агента (например, агент предпринимает действия по эксплуатации уязвимости). В этом случае выбранный план помещается на вершину соответствующего стека намерений: $C'_I = C_I \cup \{i[p\theta]\}$.

Состояние 7. Выбор намерения (selection of intention). Выбор намерения осуществляется с помощью функции s_I . Если множество намерений не содержит элементов, то выполняется переход в режим обработки сообщений (агент-злоумышленник выбирает для последующего исполнения тот сценарий атаки, выполнение которого в данный момент принесет наибольший выигрыш).

$$\frac{C_I \neq \{ \} \quad s_I(C_I) = i}{\langle ag, C, M, T, SelInt \rangle \rightarrow \langle ag, C, M, T', ExecInt \rangle}, \quad (26)$$

где $T'_i = i$.

$$\frac{C_I = \{ \}}{\langle ag, C, M, T, SelInt \rangle \rightarrow \langle ag, C, M, T, ProcMsg \rangle}. \quad (27)$$

Состояние 8. Исполнение плана (execution of intended means). Попав в данное состояние, агент может выполнять четыре типа операций.

Первый тип — стандартное действие (action). Представляет собой команду, передаваемую исполнительным механизмам агента (агент-злоумышленник выполняет конкретный шаг для эксплуатации уязвимости в ИС в соответствии с выбранным сценарием атаки). Намерение, связанное с выполняемым действием, замораживается до тех пор, пока от исполнительного механизма не будет получен ответ об успехе или ошибке выполнения.

$$\frac{T_i = i[head \leftarrow a; h]}{\langle ag, C, M, T, ExecInt \rangle \rightarrow \langle ag, C', M, T', ClrInt \rangle}, \quad (28)$$

где $C'_A = C_A \cup \{a\}$, $T'_i = i[head \leftarrow h]$, и $C'_I = (C_I \setminus \{T_i\}) \cup \{T'_i\}$.

Второй тип — цель (achievement goal). Выполняя эту операцию, агент генерирует новое внутреннее событие, которое вносится в множество E . В последствии это событие может быть выбрано в режиме SelEv (выполнение

выбранного сценария атаки по каким-то причинам в данный момент не возможно, агент ставит перед собой задачу устранить эту причину).

$$\frac{T_i = i[\text{head} \leftarrow !at; h]}{\langle ag, C, M, T, \text{ExecInt} \rangle \rightarrow \langle ag, C', M, T, \text{ProcMsg} \rangle}, \quad (29)$$

где $C'_E = C_E \cup \{+!at, T_i\}$, $C'_I = C_I \setminus \{T_i\}$.

Третий тип — тестовая цель (test goal). Эта операция используется для извлечения информации из множества убеждений агента (агент-злоумышленник обращается к своим воспоминаниям). В случае успеха выполняется унификация параметров цели, и переменные получают значения из множества bs .

$$\frac{T_i = i[\text{head} \leftarrow ?at; h] \quad \text{Test}(ag_{bs}, at) \neq \{ \}}{\langle ag, C, M, T, \text{ExecInt} \rangle \rightarrow \langle ag, C', M, T', \text{ClrInt} \rangle}, \quad (30)$$

где $T'_i = i[(\text{head} \leftarrow h)\theta]$, $\theta \in \text{Test}(ag_{bs}, at)$, $C'_I = (C_I \setminus \{T_i\}) \cup \{T'_i\}$.

Если требуемые данные не удаётся получить из убеждений агента, генерируется внутреннее событие. Возможно, данные удастся получить в результате выполнения предусмотренного для этого случая плана.

$$\frac{T_i = i[\text{head} \leftarrow ?at; h] \quad \text{Test}(ag_{bs}, at) = \{ \}}{\langle ag, C, M, T, \text{ExecInt} \rangle \rightarrow \langle ag, C', M, T, \text{ClrInt} \rangle}, \quad (31)$$

где $C'_E = C_E \cup \{+?at, T_i\}$, $C'_I = C_I \setminus \{T_i\}$.

Четвёртый тип — обновление убеждений (updating beliefs). С помощью этой операции агент вносит или удаляет элементы, содержащиеся в множестве bs (агент-злоумышленник запоминает или забывает сведения об ИС). Операция генерирует внутреннее событие.

$$\frac{T_i = i[\text{head} \leftarrow +b; h]}{\langle ag, C, M, T, \text{ExecInt} \rangle \rightarrow \langle ag', C', M, T', \text{ClrInt} \rangle}, \quad (32)$$

где $ag'_{bs} = ag_{bs} + b[\text{self}]$, $C'_E = C_E \cup \{+b[\text{self}], \emptyset\}$, $T'_i = i[\text{head} \leftarrow h]$, и $C'_I = (C_I \setminus \{T_i\}) \cup \{T'_i\}$.

$$\frac{T_i = i[\text{head} \leftarrow -at; h]}{\langle ag, C, M, T, \text{ExecInt} \rangle \rightarrow \langle ag', C', M, T', \text{ClrInt} \rangle}, \quad (33)$$

где $ag'_{bs} = ag_{bs} + at[self]$, $C'_E = C_E \cup \{\langle -at[self], \emptyset \rangle\}$, $T'_i = i[head \leftarrow h]$, и $C'_I = (C_I \setminus \{T_i\}) \cup \{T'_i\}$.

Состояние 9. Сборка мусора (clearing intentions). В этом режиме агент избавляется от тех намерений, которые удалось реализовать (агент-злоумышленник деактивирует инструменты, использовавшиеся для атаки на ИС).

Если намерение выполнено полностью, то оно просто удаляется. После чего начинается новый цикл работы.

$$\frac{T_i = [head \leftarrow \emptyset]}{\langle ag, C, M, T, ClrInt \rangle \rightarrow \langle ag, C', M, T, ProcMsg \rangle}, \quad (34)$$

где $C'_I = C_I \setminus \{T_i\}$.

Если на вершине непустого стека намерений находится пустой план, то следует удалить этот план и связанную с ним цель, расположенную в нижележащем плане стека, а затем повторить цикл очистки.

$$\frac{T_i = i[head \leftarrow \emptyset]}{\langle ag, C, M, T, ClrInt \rangle \rightarrow \langle ag, C', M, T, ClrInt \rangle}, \quad (35)$$

где $C'_I = (C_I \setminus \{T_i\}) \cup \{k[(head' \leftarrow h)\theta]\}$, $i = k[head' \leftarrow g; h]$ и $g\theta = TrEv(head)$.

Если больше удалять нечего, агент начинает новый цикл работы.

$$\frac{T_i \neq [head \leftarrow \emptyset] \wedge T_i \neq i[head \leftarrow \emptyset]}{\langle ag, C, M, T, ClrInt \rangle \rightarrow \langle ag, C, M, T, ProcMsg \rangle}. \quad (36)$$

Структура и режимы работы агента проанализированы и формализованы. На основании результатов анализа можно формализовать базовые ментальные аспекты.

Определение 7. Некоторый литерал φ является убеждением агента $ag = \langle bs, ps \rangle$, обладающего контекстом C , тогда и только тогда, когда $bs \vdash \varphi$, т.е. $BEL_{\langle ag, C \rangle} \equiv bs \vdash \varphi$.

Определение 8. Некоторый агент $ag = \langle bs, ps \rangle$ с контекстом C обладает намерением φ тогда и только тогда, когда справедливо условие

$$\text{INTEND}_{\langle ag, C \rangle}(\varphi) \equiv \varphi \in \bigcup_{i \in C_I} agls(i) \vee \varphi \in \bigcup_{\langle te, i \rangle \in C_E} agls(i), \quad (37)$$

где

$$agls(i[p]) = \begin{cases} \{at\} \cup agls(i), & \text{если } p = +!at : ct \leftarrow h \\ agls(i), & \text{иначе} \end{cases}. \quad (38)$$

Объект φ является намерением агента, если он расположен в одном из стеков намерений, либо ассоциирован с одним из событий контекста.

Определение 9. Некоторый агент $ag = \langle bs, ps \rangle$ с контекстом C обладает желанием φ тогда и только тогда, когда справедливо условие

$$\text{DES}_{\langle ag, C \rangle}(\varphi) \equiv \langle +!\varphi, i \rangle \in C_E \vee \text{INTEND}_{\langle ag, C \rangle}(\varphi). \quad (39)$$

Т.е. φ является желанием тогда, когда ассоциировано с некоторой целью, находящейся во множестве внутренних событий, либо является намерением.

Таким образом, AgentSpeak является простым, но мощным средством разработки многоагентных систем.

3.1.3 Выбор среды разработки многоагентной системы

Анализ интегрированных сред разработки (Jadex [117], Jason [118], Jack Intelligent Agents [119], Jam [120], Spark [121] и 3APL [122], SPADE2 [123], Jade [124], BORIS [125]), предназначенных для создания многоагентных систем, работающих согласно модели BDI, основывался на экспертном сравнении ряда важных характеристик [126,127]:

- статус. Этот показатель определяет текущее состояние проекта: продолжается ли его разработка, как часто выполняется выпуск новых версий, стабильна ли последняя сборка;

- лицензия. Эта характеристика определяет правовой статус специалиста, использующего программный продукт, необходимость и величину финансовых затрат на приобретение и эксплуатацию продукта [128, 129];

- платформа. Этот показатель определяет универсальность среды разработки и создаваемого с её помощью программного продукта. Здесь

приводится перечень операционных систем, сред, виртуальных машин, в рамках которых среда разработки способна функционировать [130,131];

— интерпретатор AgentSpeak. Данный язык является передовым воплощением концепций BDI. Его наличие является обязательным критерием выбора среды;

— отладчик. Отладка существенно увеличивает эффективность разработки программного продукта. Наличие встроенного отладчика AgentSpeak — значительное преимущество [132,133];

— анализатор взаимодействий. Данное средство предназначено для анализа результатов работы многоагентного комплекса и построения диаграмм последовательности. Наличие подобного инструмента позволяет отказаться от разработки средств, визуализирующих процесс функционирования многоагентной системы [134,135].

Результаты анализа представлены в таблице 4.

Результаты показывают, что внимания заслуживают лишь две среды разработки: SPADE2, Jason Jason.

Почти для всех анализировавшихся продуктов характерно применение платформы Java или языка Python, что позволяет запускать созданную многоагентную систему практически на любой вычислительной машине с соответствующим интерпретатором.

Таблица 4 — Результаты анализа сред разработки многоагентных систем

| Наименование программного продукта | Статус | Лицензия | Языки и платформа | Используемый подход к разработке | Документация / Поддержка | Взаимодействие с внешней средой | Межагентная коммуникация в ЛВС |
|------------------------------------|--------------------------------------|---------------|--|----------------------------------|--------------------------|--|----------------------------------|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| Jadex | Стабильн., регулярно обнов. | GPL | Java / XML (поддержка Android-устройств) | Императивный | Частично | Да | Да (С BDI-совместимыми агентами) |
| Jason | Стабильн., регулярно обнов. | LGPL | AgentSpeak, Java | Декларативный Императивный | Есть | Да (Через расширения на Java) | Да (С BDI-совместимыми агентами) |
| Jack Intelligent Agents | Стабильн., регулярно обновл. | Проприетарная | Java | Императивный | Частично | Да | Да (BDI-совместимые агенты) |
| 3APL | Экспериментальная, регулярно обновл. | GPL | Java / 3APL | Императивный Декларативный | Частично | Да (Опосредовано, через расширения на Java) | Да (через расширения на Java) |
| SPADE2 | Стабильн., регулярно обновл. | GPL | Python 2.6 | Императивный | Частично | Да | Да (С BDI-совместимыми агентами) |
| Boris / Boris.NET | Стабильн., регулярно обнов. | - | Java, NET, Lisp | Императивный | Частично | Да | Да (через расширения) |
| NetLogo | Стабильн., регулярно обнов. | GPL | Java / Logo | Декларативный Императивный | Есть | Да (Опосредовано, через расширения на Java) | Да (через NetLogoHub) |
| JADE | Стабильн., регулярно обнов. | LGPL | Java | Императивный | Есть | Да | Да (через расширения) |

Среды Jade, Boris, SPADE2, Jason являются BDI-совместимыми, что позволяет организовывать гетерогенную многоагентную среду.

В среду Jason интегрирован мощный интерпретатор языка AgentSpeak. В среде Jack имеется собственный язык разработки. Оба языка являются интерпретируемыми, для их использования в многоагентную систему должна быть включена соответствующая исполняющая среда.

Jack и Jason, в отличие от конкурентов, обладают анализаторами взаимодействий агентов. Это средство реализовано в виде транслированной в байт-код библиотеки, в режиме реального времени следящей за активностью агентов. По результатам наблюдения возможно построение диаграммы последовательности. Предлагается использовать данное средство для визуализации процесса работы многоагентной системы.

Существенным достоинством среды Jason является разделение высокоуровневой модели агента (декларативной, реализуется на AgentSpeak) и низкоуровневой модели агента (императивной, Java). Данное разделение позволяет реализовывать агентов с различной реализацией поведения при одинаковом построении логики и интерфейсах.

Решающим критерием для выбора среды разработки является тип лицензии, согласно которой она распространяется. Jack — коммерческий продукт с проприетарной лицензией. Jason — проект с открытым исходным кодом, имеющий лицензию LGPL.

Для реализации многоагентной системы предлагается выбрать интегрированную среду разработки Jason.

3.2 Архитектура многоагентной системы анализа защищенности ИС

Для реализации предложенных методов и моделей предлагается включить в архитектуру многоагентной системы следующие модули: ядро, подсистему построения модели злоумышленника, коммутатор, модуль виртуальных компонентов информационной системы, графический интерфейс пользователя,

шлюз связи с информационной системой и модуль имитации злоумышленных воздействий. [136]

Исполняющая среда AgentSpeak — наиболее важный элемент виртуальных компонентов ИС, а также модуля имитации злоумышленных воздействий. Она является контейнером, в котором размещаются интеллектуальные агенты, и предназначена для интерпретации когнитивных программ, в первом случае реализующих различные функции моделируемых компонентов ИС, а во втором — имитирующей деятельность злоумышленников, атакующих систему. Сама исполняющая среда и прочие компоненты многоагентной системы реализованы на объектно-ориентированном языке Java.

Важным элементом исполняющей среды является коммуникационная шина, через которую агенты взаимодействуют друг с другом. Использование агентами шины осуществляется с помощью специальных языковых конструкций AgentSpeak. Факт существования шины инкапсулирован в эти конструкции и скрыт от агентов.

Посредством ядра многоагентной системы исполняющая среда AgentSpeak проецирует обращения интеллектуальных агентов, направленные к ИС, в вызовы соответствующих методов сенсоров и исполнительных механизмов, функционирующих непосредственно в ядре. Сама исполняющая среда является внешним по отношению к ядру объектом.

Графический интерфейс пользователя необходим для визуализации и управления процессом работы многоагентной системы. Основой модуля является анализатор взаимодействий, имеющий связь с ядром системы и с шиной исполняющей среды AgentSpeak. Анализатор следит за сообщениями, проходящими по шине, и на основе получаемых данных строит диаграмму последовательности, которая выводится на графический интерфейс пользователя и отображает, кто, когда, с кем и как взаимодействовал. Элементы ядра могут выводить в текстовую консоль интерфейса различные сообщения, предупреждающие о наступлении важных событий в ходе работы системы.

Агенты-злоумышленники взаимодействуют с исследуемой ИС через коммутатор, в котором указывается, какие компоненты ИС являются реальными, а какие имитируются. Если запрос адресуется реальному компоненту системы, то он передается в шлюз связи, отвечающий за доставку сообщений ИС. Если же запрос предназначен компоненту, наличие которого в системе лишь имитируется, то коммутатор отправляет этот запрос в блок пересылки сообщений, являющийся частью модуля виртуальных компонентов ИС. Таким образом, реализуется полунатурный подход.

Модуль виртуальных компонентов ИС включает в себя блок пересылки сообщений, множество сенсоров и исполнительных механизмов, функционирующих в ядре многоагентной системы, а также множество виртуальных компонентов ИС.

Модуль имитации злоумышленных воздействий и виртуальные компоненты ИС являются условными модулями. Фактически условные модули представляют собой множества интеллектуальных агентов, функционирующих в рамках общей исполняющей среды. Однако каждое такое множество агентов предназначено для решения строго определённого круга задач, что позволяет определить его как модуль.

На рисунке 6 приведена архитектура многоагентной системы. Вся система функционирует в виртуальной машине Java, является кроссплатформенной и может быть запущена на любой вычислительной машине с соответствующим набором библиотек времени выполнения. [137]

Графический интерфейс пользователя, исполняющая среда AgentSpeak являются компонентами комплекса Jason и представляют собой транслированные в байт-код библиотеки, не предназначенные для модификации.

Ядро, коммутатор, модуль имитации злоумышленных воздействий, модуль виртуальных компонентов ИС и шлюз связи разрабатываются с учётом особенностей целевой ИС и затем компонируются с указанными выше библиотеками.

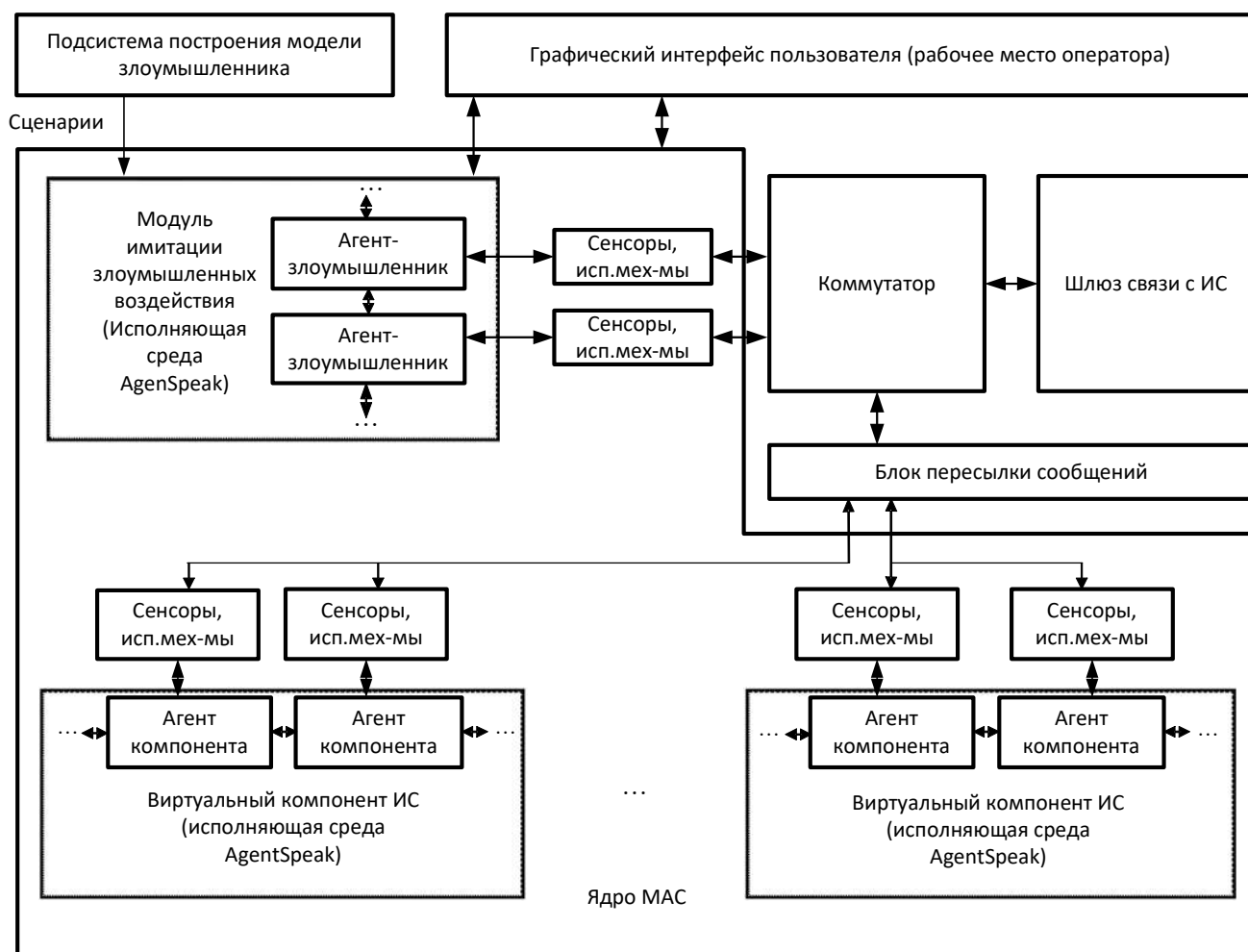


Рисунок 7 — Архитектура многоагентной системы анализа защищенности ИС

В комплект поставки среды Jason включён суперкласс Environment, что сводит реализацию ядра к простому определению того, какому агенту какие сенсоры и исполнительные механизмы соответствуют.

На вход модуля имитации злоумышленных воздействий подаётся множество сценариев злоумышленника, на основе которого строятся программы поведения интеллектуальных агентов, атакующих исследуемую ИС. Множество сценариев разрабатывается в первой части проекта и передаётся в данную часть в виде XML-файла, содержащего описание уязвимых состояний ИС и описания связей между этими состояниями. Файл получил название «библиотека сценариев злоумышленника».

Для каждой уязвимости в библиотеке задана критичность. Последовательность эксплуатируемых уязвимостей названа сценарием злоумышленника. Каждый сценарий завершается конечным состоянием, достигая

которого, нарушитель реализует несанкционированное воздействие на защищаемую информацию. Для каждого конечного состояния определена критичность того ресурса, на который воздействует злоумышленник. На основе указанных уровней критичности для каждого сценария определяется защищенность.

Многоагентная система, архитектура которой представлена на рисунке 7, имитирует злоумышленные воздействия, задаваемые библиотекой сценариев. Имитация необходима, потому что сценарии, задаваемые библиотекой, строятся с помощью математической модели, которая хоть и позволяет выявлять взаимосвязь между уязвимостями, однако не может учесть все нюансы конфигурации ИС. Сценарий, построенный этой моделью, может оказаться нереализуемым на практике. [138]

Таким образом, защищенность, задаваемая библиотекой, является теоретическим. Многоагентная система позволяет получить фактический размер ущерба.

3.2.1 Архитектура подсистемы построения модели злоумышленника

Библиотека сценариев состоит из множества данных записей вершин графа с отношением между вершинами и представляет собой объединение всех выявленных сценариев агентов-злоумышленников.

Формирование библиотеки производится на основании:

- перечня построенных сценариев;
- модели информационной системы, выраженной на языке SMV;

Архитектура подсистемы построения модели злоумышленника представлена на рисунке 8.

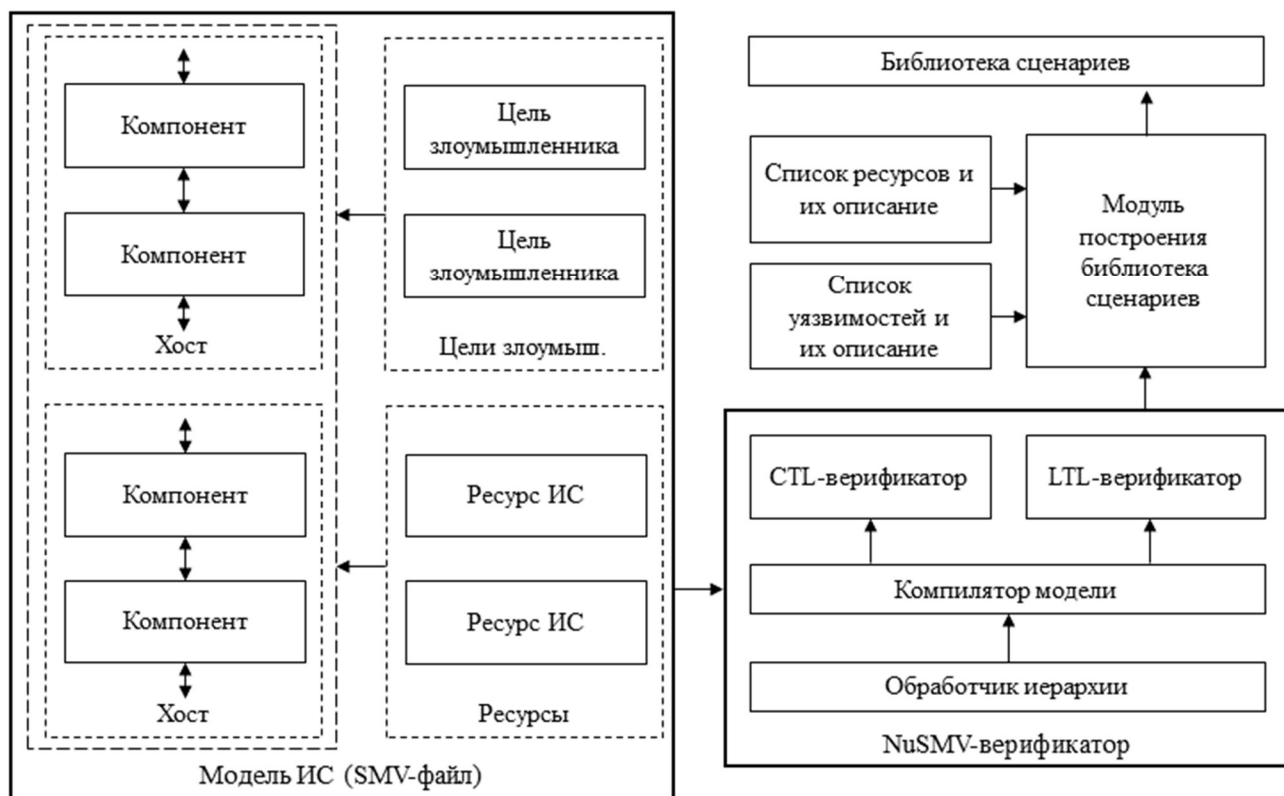


Рисунок 8 – Архитектура подсистемы построения модели злоумышленника

Модель ИС и цели злоумышленника формируются согласно спецификации и методике и компонуется в один или несколько SMV-файлов. Данные файлы представляют собой данные, необходимые для проведения верификации и построения сценариев.

Задачу верификации выполняет программное средство NuSMV. Каждому модулю верификатора соответствует одна команда. При этом порядок вызова сохраняется. Алгоритмы функционирования каждого блока скрыты от пользователя. Верификацию производят два отдельных модуля для каждого вида темпоральной логики. Результатом верификации является вывод на интерфейс пользователя множества построенных сценариев.

Управление процессом верификации производится либо посредством модуля «консольного ввода/вывода», либо посредством «графического пользовательского интерфейса gNuSMV». Однако вызов последнего также производится из консоли, которая способна перенаправлять ввод/вывод.

Формирование библиотеки сценариев на базе множеств построенных сценариев производит «модуль построения библиотеки сценариев». Библиотека строится на основании множества построенных сценариев, SMV-файла и списка уязвимостей.

Список уязвимостей хранит двойки $\langle id, P_y \rangle$, описывающих уязвимости. Данные сведения необходимы для определения защищенности от выполнения данного сценария.

Библиотека хранится в модуле «библиотеки сценариев» и доступна для анализа через консольный ввод/вывод.

Блок-схема алгоритма построения модели злоумышленника приведена на рисунке 9.

Алгоритм состоит из следующих этапов:

- 1) В блоке 1 происходит инициализация выполнения алгоритма.
- 2) В блоке 2 производится загрузка множества построенных сценариев и загрузка SMV-файла модели.
- 3) В блоке 3 проверяется, существуют ли не добавленные сценарии, и если такой сценарий существует, то управление передается блоку 4. Если сценариев для обработки не осталось, то управление переходит к шагу 10.
- 4) В блоке 4 в заголовок вносится идентификатор нового сценария.
- 5) В блоке 5 все состояния сценария конвертируются в XML-формат.
- 6) В блоке 6 производится проверка, остались ли необработанные состояния в выбранном сценарии. Если остались, то управление передается блоку 7. В противном случае, блоку 8.
- 7) В блоке 7 данное состояние обрабатывается – вносится в библиотеку сценариев и заголовок сценария. В библиотеку вносятся только те состояния, которые характеризуют выполнение уязвимостей или получение доступа к ресурсу. Внутренние изменения состояний ИС между фактами эксплуатации уязвимостей игнорируются.

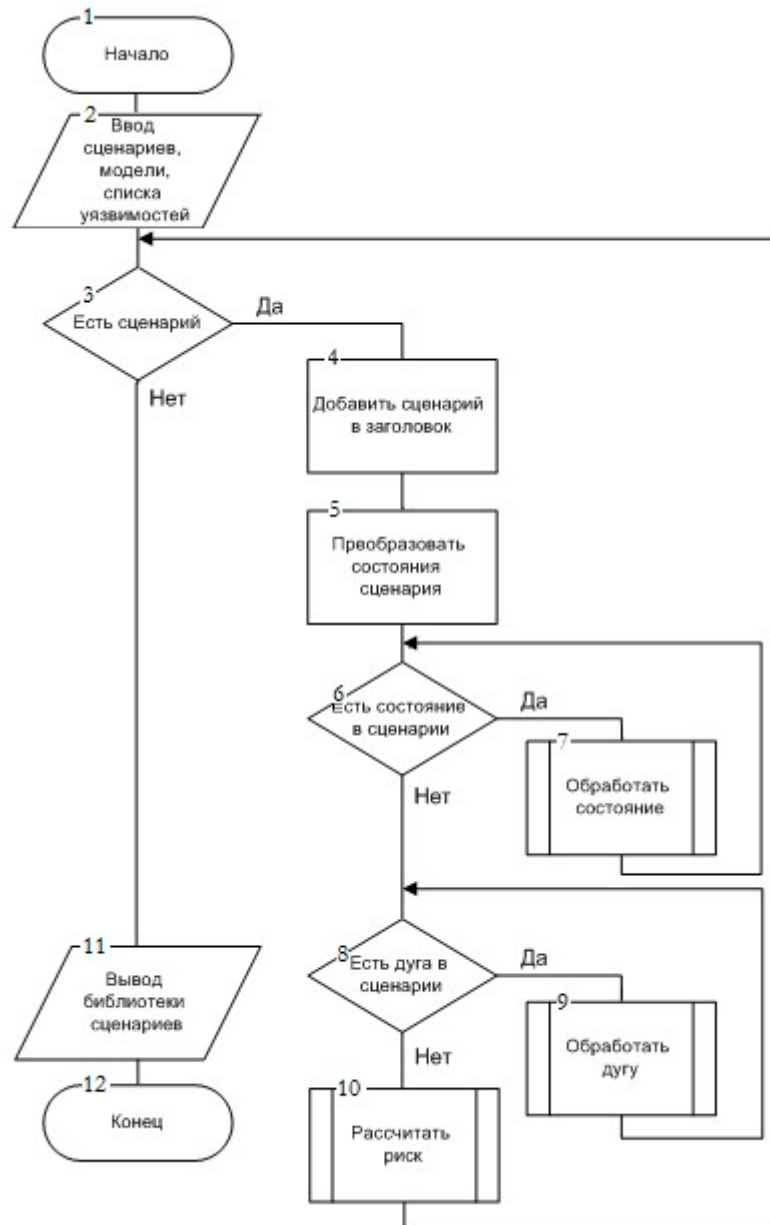


Рисунок 9 – Блок-схема алгоритма построения библиотеки сценариев

8) После того, как обработаны все вершины, в блоке 8, проверяется, остались ли необработанные дуги между двумя добавленным состояниями. Если остались, то управление передается блоку 9, в противном случае блоку 10.

9) В блоке 9 заносится связь между состояниями.

10) В блоке 10 производится расчет защищенности, согласно методике оценки защищенности ИС. После того как сценарий добавлен в библиотеку, управление передается к шагу 3, и действия повторяются.

11) В блоке 11 производится запись полученной библиотеки сценариев.

12) Блок 12 – завершение работы алгоритма.

Предлагаемая подсистема позволяет реализовать построение библиотеки сценариев агентов-злоумышленников в представленной XML-форме, посредством верификатора NuSMV.

3.2.2 Модуль имитации злоумышленных воздействий

Разработка данного модуля основывается на библиотеке сценариев. Библиотека содержит подробные описания различных путей эксплуатации выявленных в ИС уязвимостей. Пример графической интерпретация библиотеки приведён на рисунке 10.

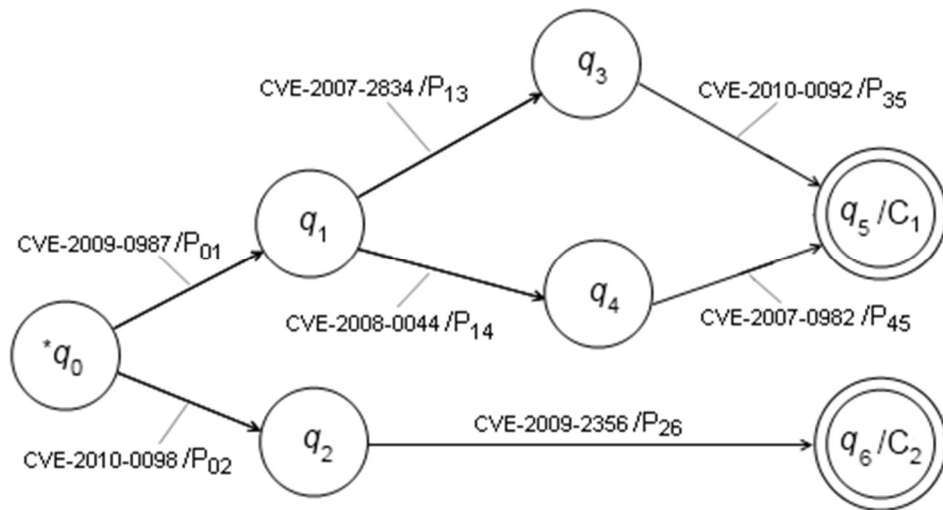


Рисунок 10 – Графическое представление библиотеки сценариев агентов-злоумышленников

Вершины графа содержат описания состояний ИС, создающих предпосылки к эксплуатации уязвимостей, задаваемых дугами, выходящими из этих состояний. Каждой дуге соответствует определённая бюллетень CVE и критичность уязвимости. В графе выделяются начальное (q_0) и конечные (q_5 , q_6) состояния, характеризующие соответственно исходное состояние ИС и состояния, в которых злоумышленные воздействия считаются реализованными. Под сценарием понимается ациклический путь, ведущий из начального состояния в одно из конечных. Для каждого конечного состояния определена критичность связанного

с ним информационного ресурса. На основе этой характеристики в процессе имитации воздействий выполняется расчет защищенности.

Для имитации злоумышленных воздействий, соответствующих заданной библиотеке сценариев, предлагается использовать два типа агентов:

— $\{ag^{PS_{strategy}}\}$ — множество агентов, которым известны сценарии воздействий на ИС (т.е. последовательность посещаемых вершин графа, стратегия); они выполняют мониторинг состояния ИС, определяют, какой вершине соответствует текущее состояние, и отдают приказы агентам второго множества на эксплуатацию уязвимостей для перевода системы в состояние, соответствующее одной из следующих вершин графа;

— $\{ag^{PS_{tactics}}\}$ — множество агентов, которым известны тактики эксплуатации уязвимостей (т.е. последовательность действий, приводящих к переходу в следующую вершину графа), но представления о стратегии злоумышленного воздействия на ИС они не имеют.

Агентов первого множества предлагается назвать ведущими, а второго — ведомыми.

Для управления ходом имитации, а также для оценки результатов злоумышленных воздействий предлагается ввести в многоагентную систему специального мастер-агента, обладающего сведениями о защищенности ресурса, связанного с каждым сценарием.

Таким образом, модуль имитации злоумышленных воздействий представляет собой совокупность одного мастер-агента и множеств ведущих и ведомых агентов-злоумышленников, когнитивные программы которых функционируют в рамках исполняющей среды AgentSpeak.

Принципы функционирования и структура мастер-агента определяются следующим образом:

— в множестве убеждений мастер-агента указываются сведения обо всех возможных сценариях злоумышленных воздействий на ИС, их характеристики

(защищенность для различных узлов сценария) и идентификаторы ведущих агентов, реализующих эти сценарии;

— мастер-агент определяет, какой сценарий будет реализовываться в данный момент: выбор выполняется в порядке возрастания защищенности;

— если ведущий агент сообщает об успешной реализации сценария, то из набора убеждений мастер-агента удаляются сведения обо всех альтернативных сценариях, ведущих к тому же конечному состоянию, а защищенность, ассоциированная с этим состоянием, добавляется к текущему обобщенному уровню защищенности;

— когда в базе убеждений мастер-агента не остается сведений о сценариях, имитация завершается, выводится результирующую защищенность.

Для реализации множества ведущих агентов предлагается следующая методика:

1) в библиотеке сценариев выделяется маршрут, ведущий из начальной вершины в одну из конечных. На рисунке 9 таких маршрутов три: (q_0, q_1, q_3, q_5) , (q_0, q_1, q_4, q_5) и (q_0, q_2, q_6) ;

2) для каждой дуги, входящей в выделенный на 1-ом шаге маршрут, реализуется запрос к ведомому агенту. В запросе указываются уязвимый компонент и тип уязвимости. За последующую эксплуатацию уязвимости отвечает ведомый агент, принявший запрос и уведомивший об этом ведущего агента-источника запроса;

3) для каждой входящей в маршрут вершины кодируется база убеждений ведущего агента. Как только набор убеждений, полученных от сенсоров агента, совпадет с одной из таких баз, выполняется рассылка соответствующего широковещательного запроса.

Блок-схема приведенного алгоритма представлена на рисунке 11 [139-141].

Для реализации множества ведомых агентов предлагается следующая последовательность шагов:

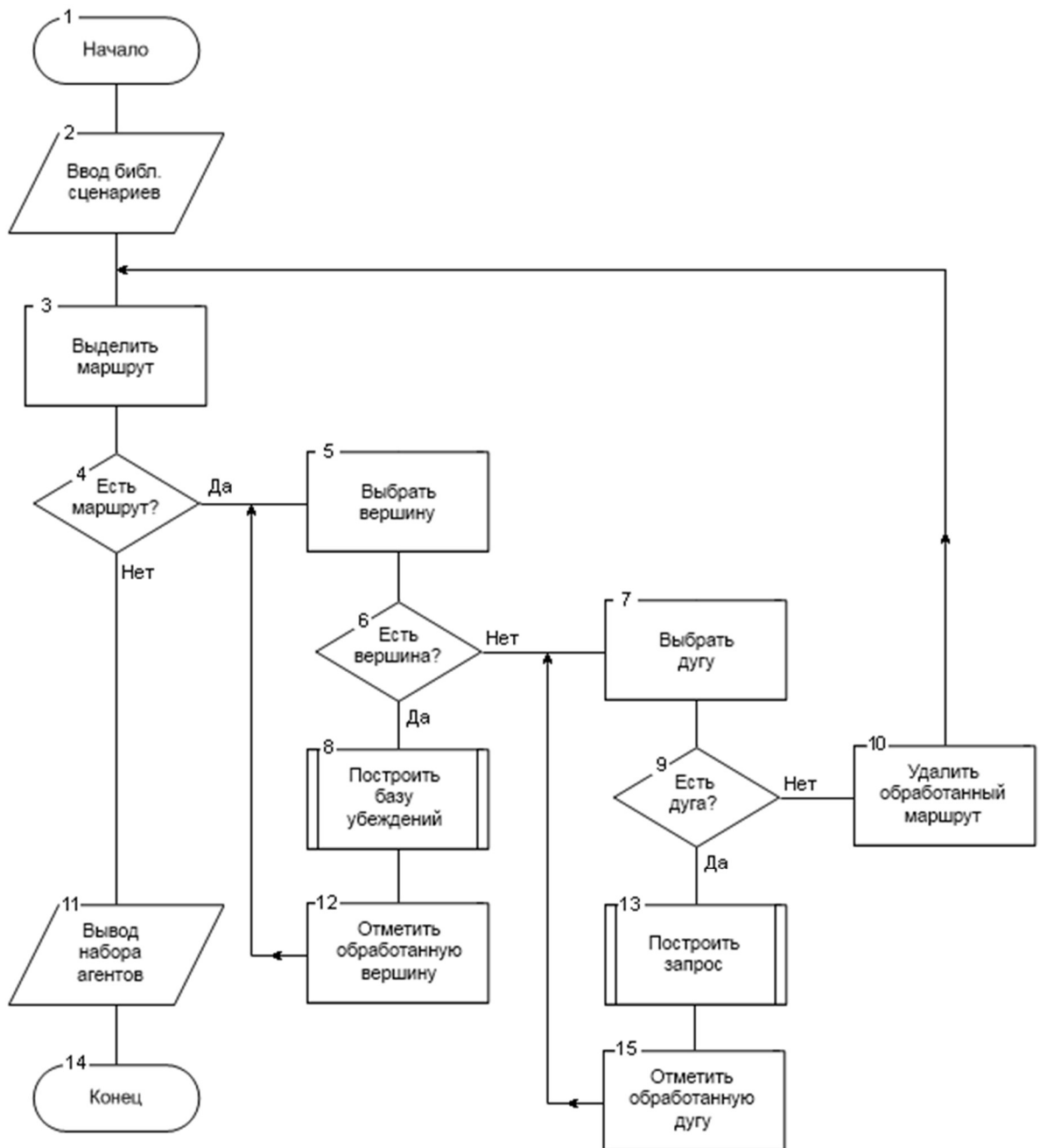


Рисунок 11 — Блок-схема алгоритма построения ведущих агентов-злоумышленников

- 1) в библиотеке сценариев выбирается дуга, для которой предстоит реализовать ведомого агента;
- 2) на основе CVE- и CWE-описаний, соответствующих дуге, выполняется декомпозиция процесса эксплуатации уязвимости в конечный автомат (декомпозиция выполняется экспертным методом);

3) полученные состояния автомата характеризуют различные этапы эксплуатации уязвимости и используются для определения множеств желаний и намерений агента.

4) на основе полученных множеств желаний и намерений выполняется кодирование агента-злоумышленника.

Пусть $M = (Q, \Sigma, \delta, q_0, F)$ — конечный автомат, где Q — множество состояний автомата (этапы атаки), Σ — входной алфавит (злоумышленные воздействия), δ — заданное таблицей отображение множества $Q \times \Sigma$ во множество Q (взаимосвязь этапов атаки), q_0 — начальное состояние автомата (состояние ИС, создающее предпосылку к началу атаки), а F — множество заключительных состояний [142]. Тогда упорядоченное множество $D = (\delta_{i_0}, \dots, \delta_{i_n})$ такое, что последовательность отображений $\delta_{i_0}, \dots, \delta_{i_n}$ переводит автомат из состояния q_0 в состояние $q_n \in F$, является множеством желаний агента, а упорядоченное множество входных данных $I = (\sigma_{j_0}, \dots, \sigma_{j_n})$, соответствующих отображениям $\delta_{i_0}, \dots, \delta_{i_n}$, является множеством намерений агента.

Блок-схема алгоритма реализации набора ведомых агентов приведена на рисунке 12.

Для примера, процесс запуска на уязвимой системе командной строки с помощью переполнения буфера можно декомпозировать в автомат, изображенный на рисунке 13.

Пусть в исследуемой ИС имеется компонент, в котором присутствует уязвимость типа «переполнение буфера», и известно, что эта уязвимость позволяет выполнить произвольный код.

Состояние q_0 . Уязвимый компонент запущен и готов к приему и обработке входных данных. Желание δ_0 злоумышленника заключается в том, чтобы разместить в памяти уязвимого компонента код, выполнение которого приведёт к запуску командной строки. Для этого злоумышленник выполняет намерение σ_0 , отправляя компоненту сообщение, размещающее в его памяти требуемый код.

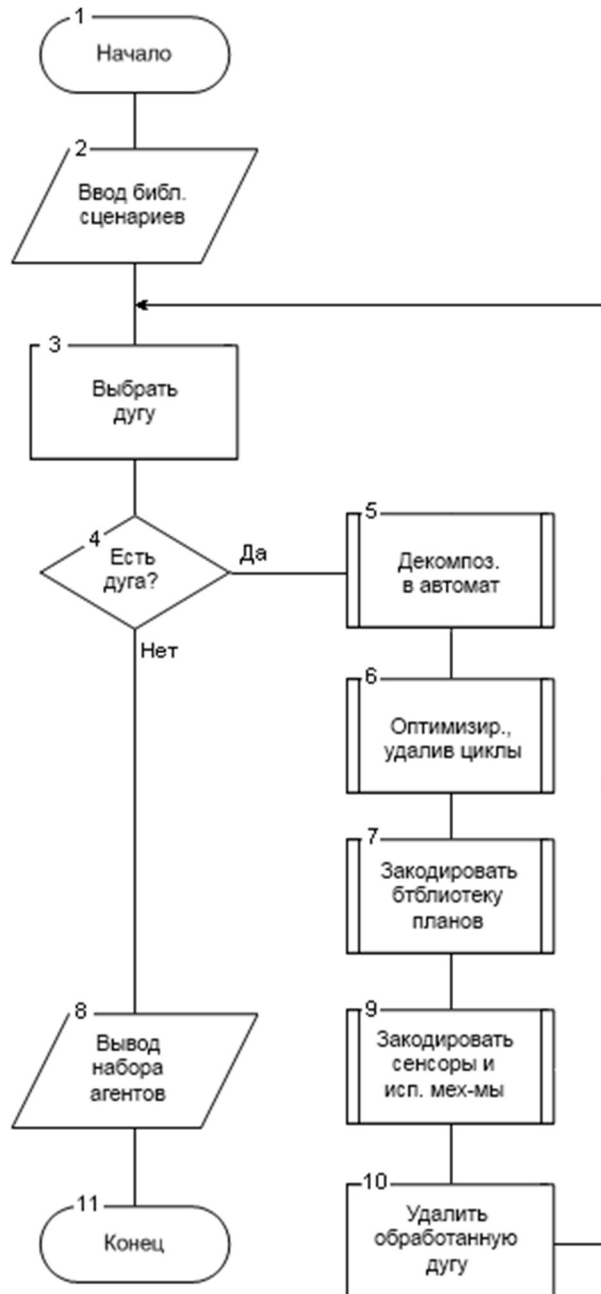


Рисунок 12 — Блок-схема алгоритма построения ведомых агентов-злоумышленников

Состояние q_1 . Код, запускающий командную строку, находится в памяти приложения. Теперь злоумышленник желает заставить приложение выполнить этот код (желание δ_1). Для этого он отправляет сообщение (намерение σ_1), обработка которого вызовет переход по адресу, начиная с которого расположен код, запускающий командную строку.

Состояние q_2 . Злоумышленнику не удалось сформировать верное сообщение, и при переполнении буфера возникла ошибка сегментации, что привело к остановке работы уязвимого компонента системы. Для продолжения атаки необходимо дождаться повторного запуска приложения и начать процесс сначала.

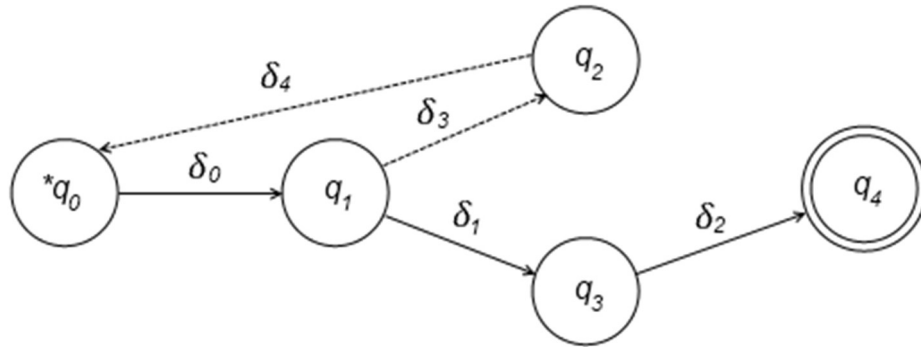


Рисунок 13 — Пример декомпозиции процесса эксплуатации уязвимости

Состояние q_3 . Переполнение использовано успешно, и приложение приступило к обработке кода, запускающего командную строку. Теперь желание δ_2 злоумышленника заключается в том, чтобы использовать запущенную на определённом порту командную строку. Для этого он выполняет намерение σ_2 , ожидая запуска командной строки.

Состояние q_4 . Заключительное состояние, командная строка запущена, злоумышленник получил контроль над системой.

В приведённом примере $D = (\delta_0, \delta_1, \delta_2)$ — множество желаний, а $I = (\sigma_0, \sigma_1, \sigma_2)$ — множество намерений. Дуги δ_3 и δ_4 желаниями не являются, т.к. формируют цикл, переводящий автомат в начальное состояние.

Подобным образом может быть декомпозирован процесс эксплуатации любой программной уязвимости [143].

Для программной реализации ведомого агента требуется закодировать библиотеку планов, сенсоры и исполнительные механизмы.

Формирование библиотеки выполняется за несколько итераций. Сначала создается остов библиотеки. В примере с запуском командной строки через

переполнение буфера, простейший вариант остова может выглядеть, как представлено в листинге 14.

Листинг 14 — Остов библиотеки планов

```

 $\sigma_0$ :   connect (Component) ;
           !inject (shell_code) .
 $\sigma_1$ :   execute (shell_code) .
 $\sigma_2$ :   connect (shell) ;
           ?start (Component) ;
           disconnect (Component) .

```

После создания остова библиотеки необходимо сформировать множество иницирующих событий. Для создания первой части множества используется автоматная декомпозиция процесса атаки, для создания второй — множество целей из остова библиотеки планов. Результат приведён в листинге 15.

Листинг 15 — Множество иницирующих событий

```

 $q_0$ :   +start (Component)
 $q_1$ :   +injected
 $q_2$ :   +unexecuted
 $q_3$ :   +executed
 $\sigma_0$ : +!inject

```

Далее необходимо закодировать планы, соответствующие второй части множества иницирующих событий. Пример представлен в листинге 16.

Листинг 16 — Расширение библиотеки планов

```

 $\sigma_0$ :   ?has (Component, shell_code, Shell_code_id) ;
           inject (Shell_code_id) .

```

Остается объединить остов библиотеки планов и множество иницирующих событий в единый модуль. Результирующая простейшая когнитивная программа ведомого агента представлена в листинге 17.

Листинг 17 — Когнитивная программа агента-злоумышленника

```

+start (Component)
  <- connect (Component) ;

```

```

        !inject(shell_code).
+injected
        <- execute(shell_code).
+executed
        <- connect(shell);
        ?start(Component);
        disconnect(Component).
+unexecuted
        <- ?start(Component);
        disconnect(Component).
+!inject
        <- ?has(Component, shell_code, Shell_code_id);
        inject(Shell_code_id).

```

Суть разработки исполнительных механизмов заключается в реализации термов-действий в виде функций на языке Java. В примере выше таких термов четыре: `connect`, `inject`, `execute` и `disconnect`. Каждый из термов отвечает соответственно за установку связи с уязвимым компонентом системы, введение кода в адресное пространство компонента, передачу управления введённому коду и отключение от уязвимого компонента.

Такой терм обладает простой императивной реализацией, которая выполняется с использованием программного каркаса, предоставляемого средой Jason. Этот каркас носит название Environment Simulation Framework (ESF). Он инкапсулирует механизм вызова императивной реализации терма при обращении к нему в когнитивной программе агента. Таким образом, взаимосвязь различных этапов атаки описывается декларативно, а набор действий, составляющих процесс атаки, реализуется на привычном объектно-ориентированном языке.

Для передачи управления императивной реализации терма используется метод «executeAction» базового класса «Environment».

Сенсоры, отвечающие за формирование у агента убеждений, символизирующих о наступлении того или иного этапа атаки, также реализуются с

помощью указанного каркаса. Для этого используется метод «updatePercepts» базового класса «Environment».

3.2.3 Механизм взаимодействия модели Belief-Desire-Intention с информационной системой

Для имитации действий злоумышленников лучше всего подходит модель BDI, однако она позволяет решить не все задачи. В процессе исследования ИС на неё могут оказываться деструктивные воздействия, способные вывести атакуемые компоненты из строя или разрушить их. Если какой-либо из этих компонентов критичен для функционирования системы, подобный исход не приемлем.

Указанной проблемы можно избежать, если расширить BDI следующим образом: в случае, когда некоторое действие агента-злоумышленника носит разрушающий характер, компонент ИС, на который оно направлено, необходимо заменить моделью [144]. Для этого требуется наделить многоагентную систему, построенную согласно модели BDI, дополнительными функциональными модулями:

- модулем, отвечающим за взаимодействие агентов-злоумышленников с моделируемыми компонентами ИС;
- модулем, содержащим моделируемые компоненты ИС. [145]

Архитектура многоагентной системы с дополнительными модулями приведена на рисунке 14.

Взаимодействие агентов-злоумышленников с исследуемой информационной системой осуществляется через модуль, называемый коммутатором. Ему известно, какие компоненты системы являются реальными, а какие моделируются. Основываясь на этих сведениях, коммутатор пересылает запросы агентов-злоумышленников либо в модуль, содержащий модели компонентов информационной системы, либо самой информационной системе через шлюз связи с ней.

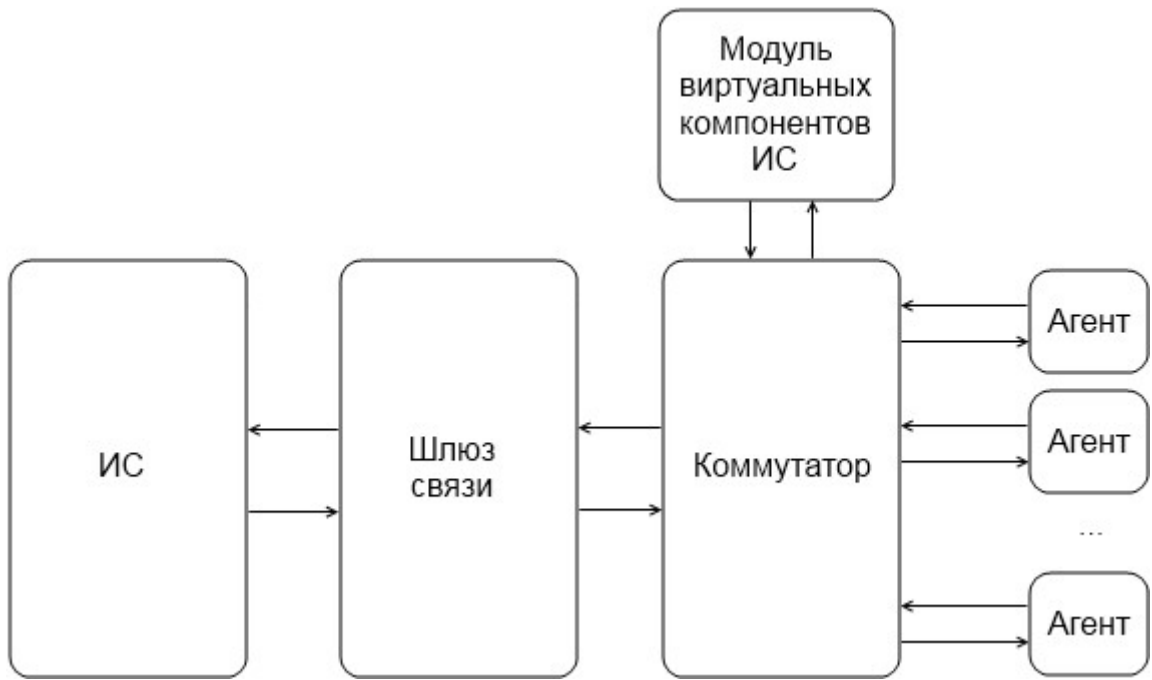


Рисунок 14 — Архитектура полунатурной многоагентной системы

Поскольку такая многоагентная система включает в себя как реальную ИС, так и некоторый набор её моделируемых компонентов, то она классифицируется как полунатурная. [146]

Технический аспект взаимодействия агентов с ИС предполагает невозможность прямого обращения к требуемому компоненту системы непосредственно из когнитивной программы интеллектуальной сущности. Для этого агент должен делегировать задание на взаимодействие своим сенсорам или исполнительным механизмам.

Для решения указанной задачи предлагается ввести модуль, в котором размещаются сенсоры и исполнительные механизмы всех агентов. Данный модуль, называемый ядром, принимает задания от AgentSpeak-программ и для их реализации запускает выполнение соответствующих Java-программ, являющихся сенсорами или исполнительными механизмами агентов. Ядро и среда, в которой исполняются когнитивные программы агентов, являются разными модулями, функционирование которых взаимосвязано и взаимообусловлено. [147]

3.2.4 Коммутатор

В состав коммутатора предлагается включить следующие модули: таблицу коммутации, блок коммутации и коммуникационные порты.

Таблица коммутации представляет собой ассоциативный массив, хранящий сведения о том, какие компоненты ИС являются моделируемыми.

Через коммуникационные порты к коммутатору подключаются агенты-злоумышленники, входящие в состав модуля имитации злоумышленных воздействий, а также модуль виртуальных компонентов и шлюз связи с ИС.

Выделяются два типа портов: статические и динамические. Статические порты предназначены для соединения с модулем виртуальных компонентов и шлюзом связи с ИС. К динамическим портам подключаются агенты-злоумышленники. Количество статических портов коммутатора фиксировано. Они инициализируются при запуске коммутатора, в дальнейшем параметры таких портов остаются неизменными. Количество динамических портов варьируется в процессе работы коммутатора: при внесении в многоагентную систему нового агента-злоумышленника создаётся новый динамический порт. При удалении агента-злоумышленника выполняется редукция количества динамических портов.

Блок коммутации, используя информацию, расположенную в таблице коммутации, выполняет пересылку сообщений между портами коммутатора. Если поступившее сообщение предназначено моделируемому компоненту ИС, то оно передается на порт, соединенный с модулем виртуальных компонентов. В противном случае передача сообщения выполняется на статический порт, подключенный к шлюзу связи с ИС.

На рисунке 15 представлена результирующая структура коммутатора.

Алгоритм работы коммутатора представлен на блок-схеме, изображённой на рисунке 16, и включает следующие шаги:

- 1) в блоке 2 выполняется приём сообщения от агента-злоумышленника (например, это может быть сетевой пакет);

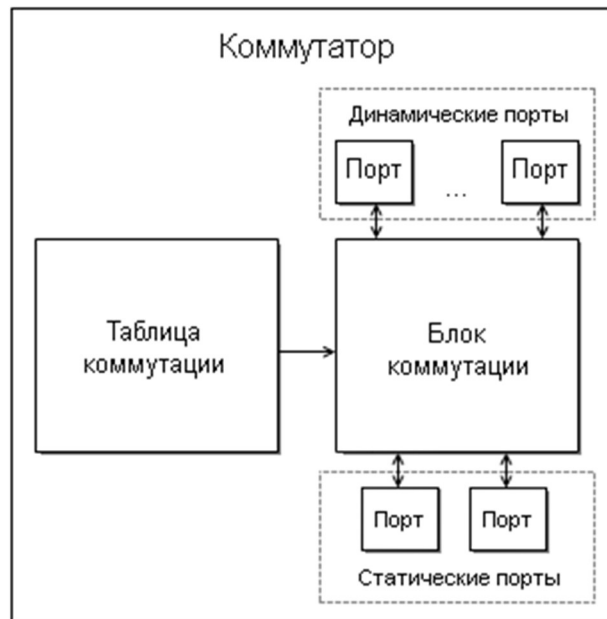


Рисунок 15 — Структура коммутатора

- 2) в блоке 3 из принятого сообщения извлекается идентификатор компонента информационной системы, которому адресовано сообщение; выполняется поиск информации для этого идентификатора в таблице коммутации;
- 3) в блоке 4 выполняется анализ полученного идентификатора; если в таблице коммутации указано, что компонент, которому адресуется сообщение, является моделируемым, то выполняется переход к блоку 5, иначе — к блоку 6;
- 4) если осуществлён переход к блоку 5, то сообщение выводится в модуль виртуальных компонентов информационной системы;
- 5) в блоке 7 выполняется ожидание сигнала от модуля виртуальных компонентов; сигнал содержит результат обработки сообщения агента-злоумышленника;
- 6) в блоке 8 выполняется обработка полученного сигнала; если сигнал сообщает о невозможности предотвратить поступление сообщения в ИС, выполняется переход к блоку 9, иначе — к блоку 10.

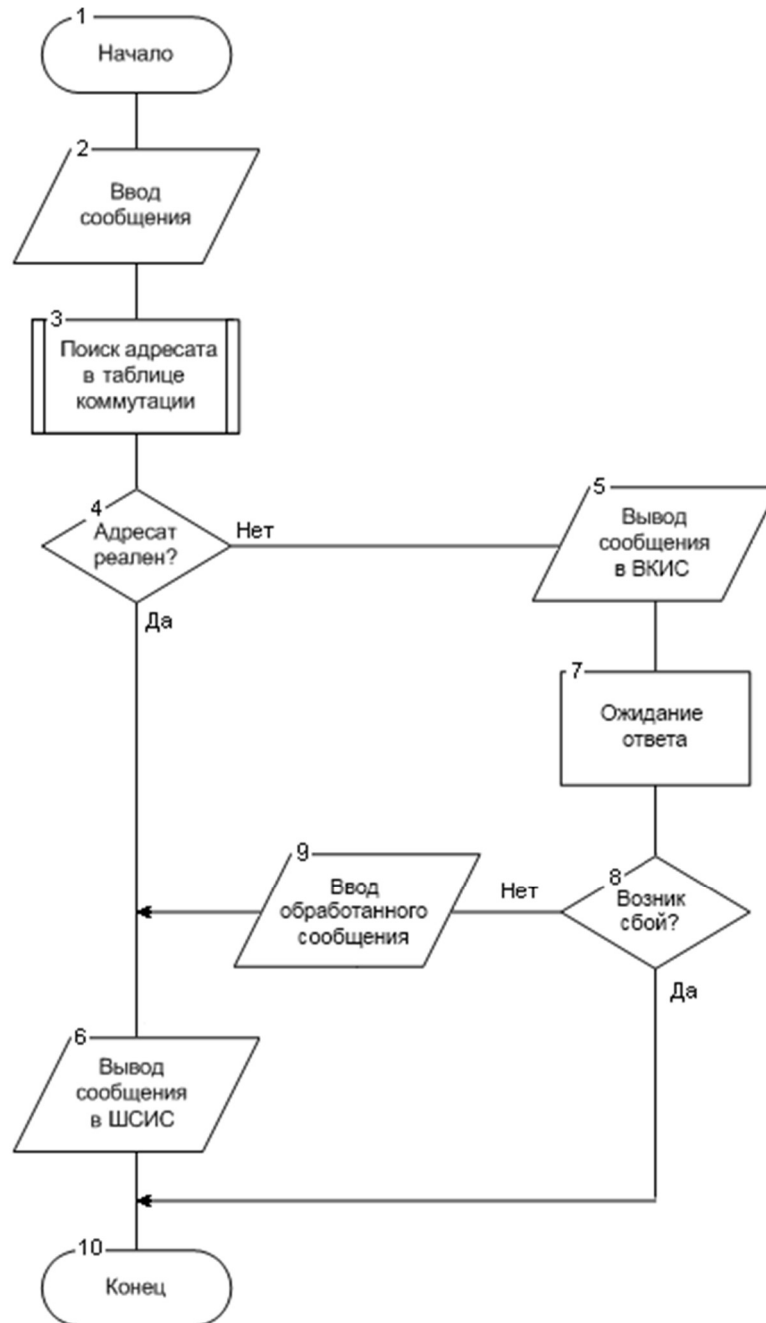


Рисунок 16 — Блок-схема алгоритма работы коммутатора

3.2.5 Модуль виртуальных компонентов

Модуль виртуальных компонентов ИС является основой для обеспечения полунатурного подхода при имитации злоумышленных воздействий. Он содержит модели тех элементов ИС, на которые в процессе имитации могут выполняться деструктивные, разрушающие воздействия.

В состав модуля включены: блок пересылки сообщений и набор виртуальных компонентов информационной системы.

Блок пересылки предназначен для передачи сообщений, полученных от коммутатора, всем подключенным к нему виртуальным компонентам. Виртуальные компоненты подключаются к блоку в том порядке, в котором бы они обрабатывали сообщения, если бы на самом деле присутствовали в системе. Результирующая структура модуля приведена на рисунке 17. [148]

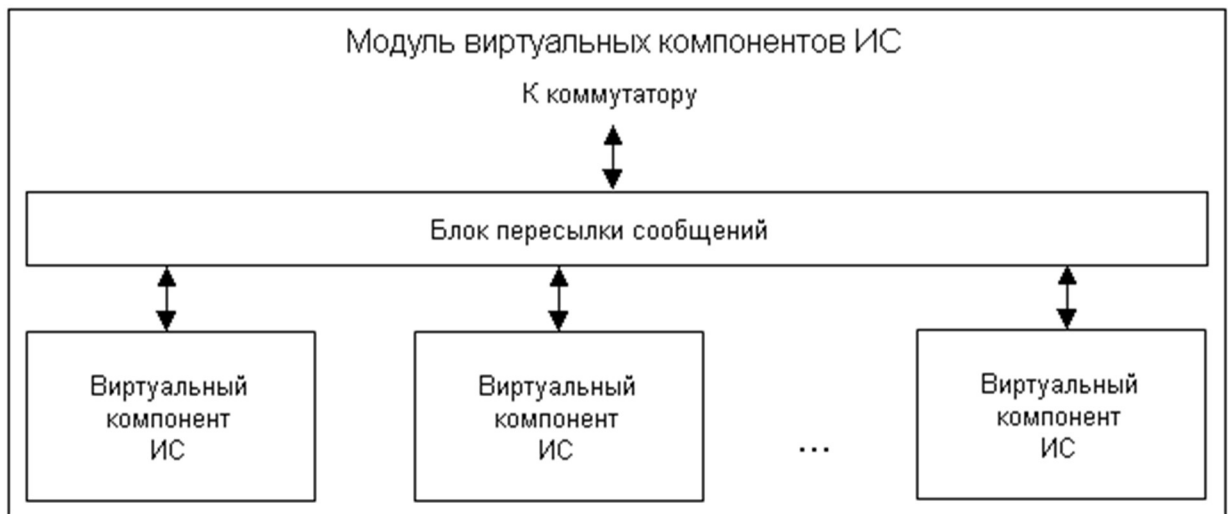


Рисунок 17 — Структура модуля виртуальных компонентов информационной системы

Алгоритм работы модуля представлен на блок-схеме, изображённой на рисунке 18, и включает следующие шаги:

1) в блоке 2 выполняется прием сообщения от агента-злоумышленника; сообщение поступает в блок пересылки из коммутатора;

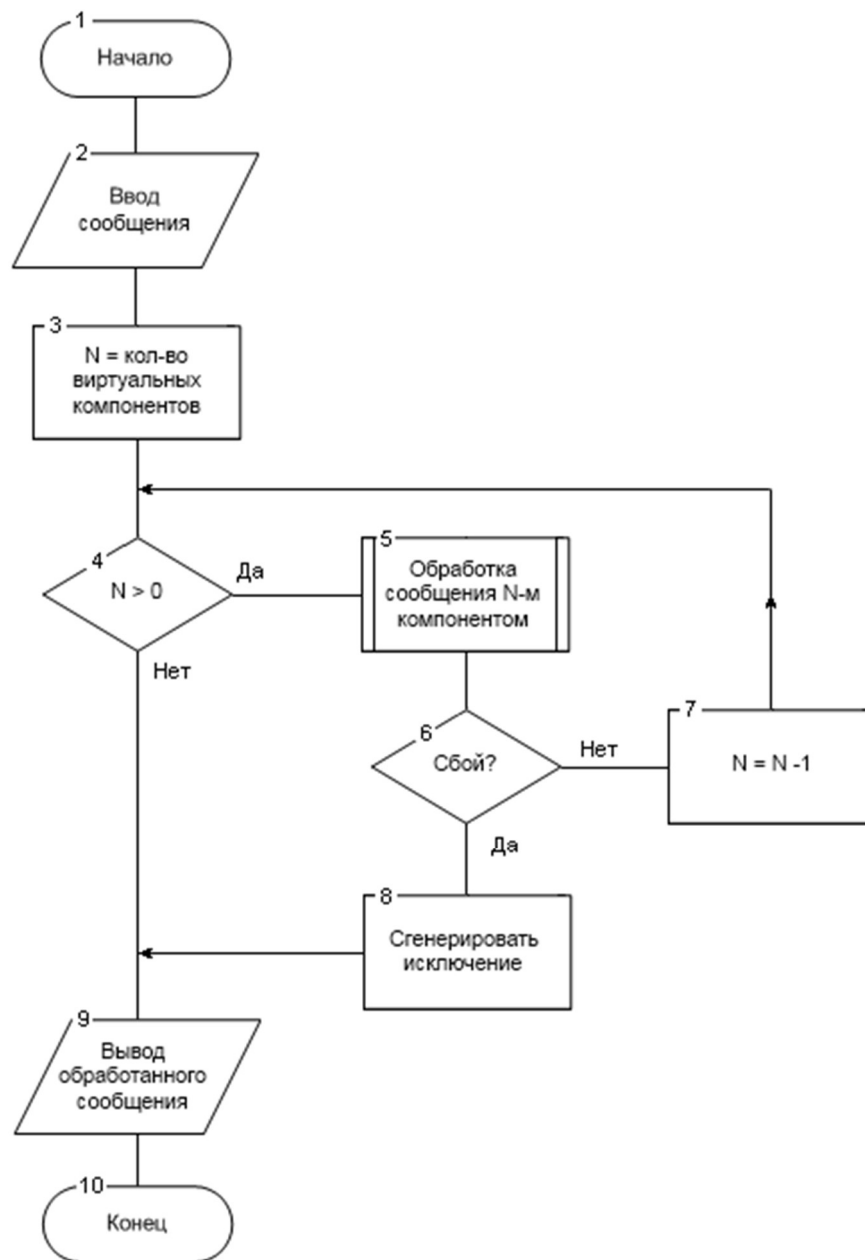


Рисунок 18 — Блок-схема алгоритма работы модуля виртуальных компонентов ИС

- 2) в блоке 3 определяется количество виртуальных компонентов ИС; полученное значение заносится в переменную N;
- 3) в блоке 4 проверяется значение переменной N; если значение больше нуля, значит, не все виртуальные компоненты обработали сообщение, выполняется переход к блоку 5; иначе — к блоку 9;

- 4) в блоке 5 право обработать буфер, содержащий сообщение агента-злоумышленника, передаётся виртуальному компоненту N ИС; ограничений на возможные операции с содержимым буфера нет;
- 5) в блоке 6 проверяется результат работы виртуального компонента; если зафиксирована ошибка, возникшая в процессе обработки сообщения, и компонент не справился со своей защитной задачей, выполняется переход к блоку 8, иначе — к блоку 7;
- 6) в блоке 7 уменьшается значение параметра N, т.к. сообщение было успешно обработано очередным виртуальным компонентом;
- 7) в блоке 8 генерируется исключение безопасности, которое будет перехвачено коммутатором; коммутатор, получив такое исключение, поймет, что виртуальная подсистема защиты не справилась с предотвращением проникновения сообщения агента-злоумышленника в реальную ИС;
- 8) в блоке 9 сообщение, обработанное всеми (или частью из-за возникновения ошибки) виртуальными компонентами, выводится обратно в коммутатор, после чего либо передаётся реальной ИС, либо возвращается агенту-злоумышленнику.

Для реализации моделей компонентов ИС предлагается подход, основанный на использовании введенных ранее типов агентов: ведомых и ведущих. Ведущие агенты реализуют логику работы компонента, а ведомые — множество элементарных функций, выполняемых компонентом.

Перед проведением многоагентной декомпозиции компонента осуществляется его предварительное исследование. Целью исследования является сбор информации для построения конечного автомата, содержащего множество элементарных функций компонента. Пример такого автомата приведён на рисунке 19.

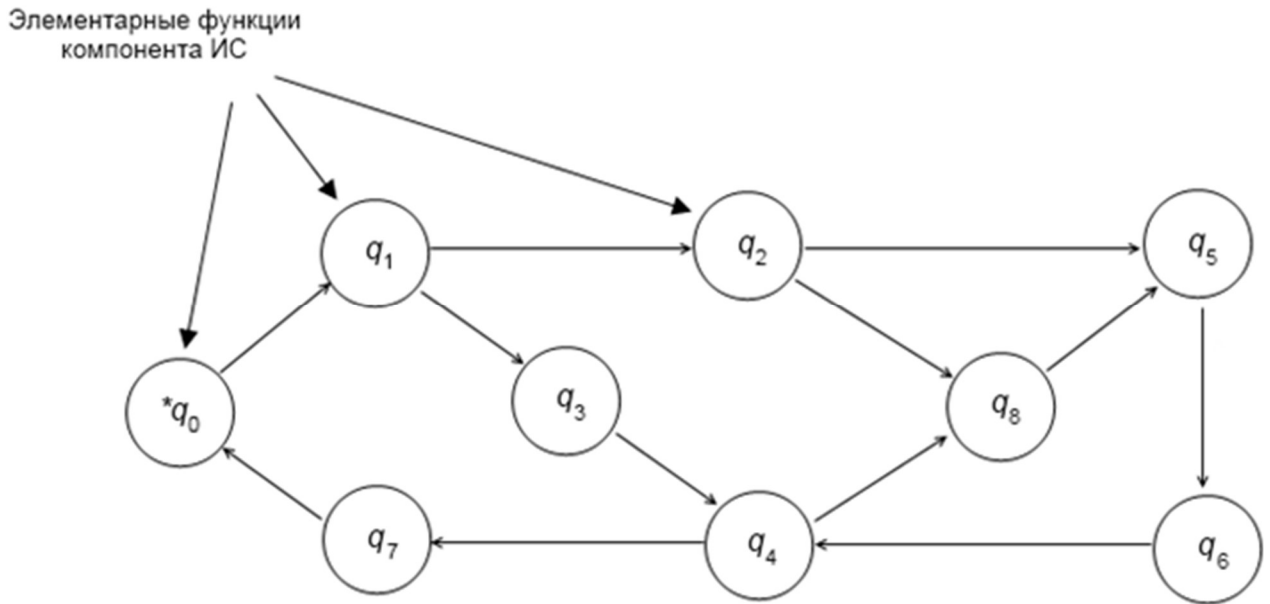


Рисунок 19 — Пример автоматной декомпозиции процесса работы некоторого компонента ИС

Состояния q_n автомата содержат элементарные функции, выполняемые компонентом системы (например, перехват сетевых пакетов, проверку каких-либо параметров в заголовках пакетов, блокирование некорректных пакетов и т.д.). Автоматное описание процесса работы лучше всего удовлетворяет многоагентному подходу, т.к. позволяет указать частные связи элементарных функций без необходимости определения их глобальных влияний на функционирование всего компонента.

После завершения автоматной декомпозиции для реализации многоагентной модели компонента применяется алгоритм, блок-схема которого приведена на рисунке 20. Алгоритм включает в себя следующие шаги:

- 1) в блоке 2 выполняется ввод автоматной декомпозиции процесса работы компонента ИС;
- 2) в блоке 3 выбирается ещё не обработанное состояние автомата;
- 3) в блоке 4 проверяется, удалось ли выбрать необработанное состояние; если удалось, выполняется переход в блок 5, иначе — в блок 9;

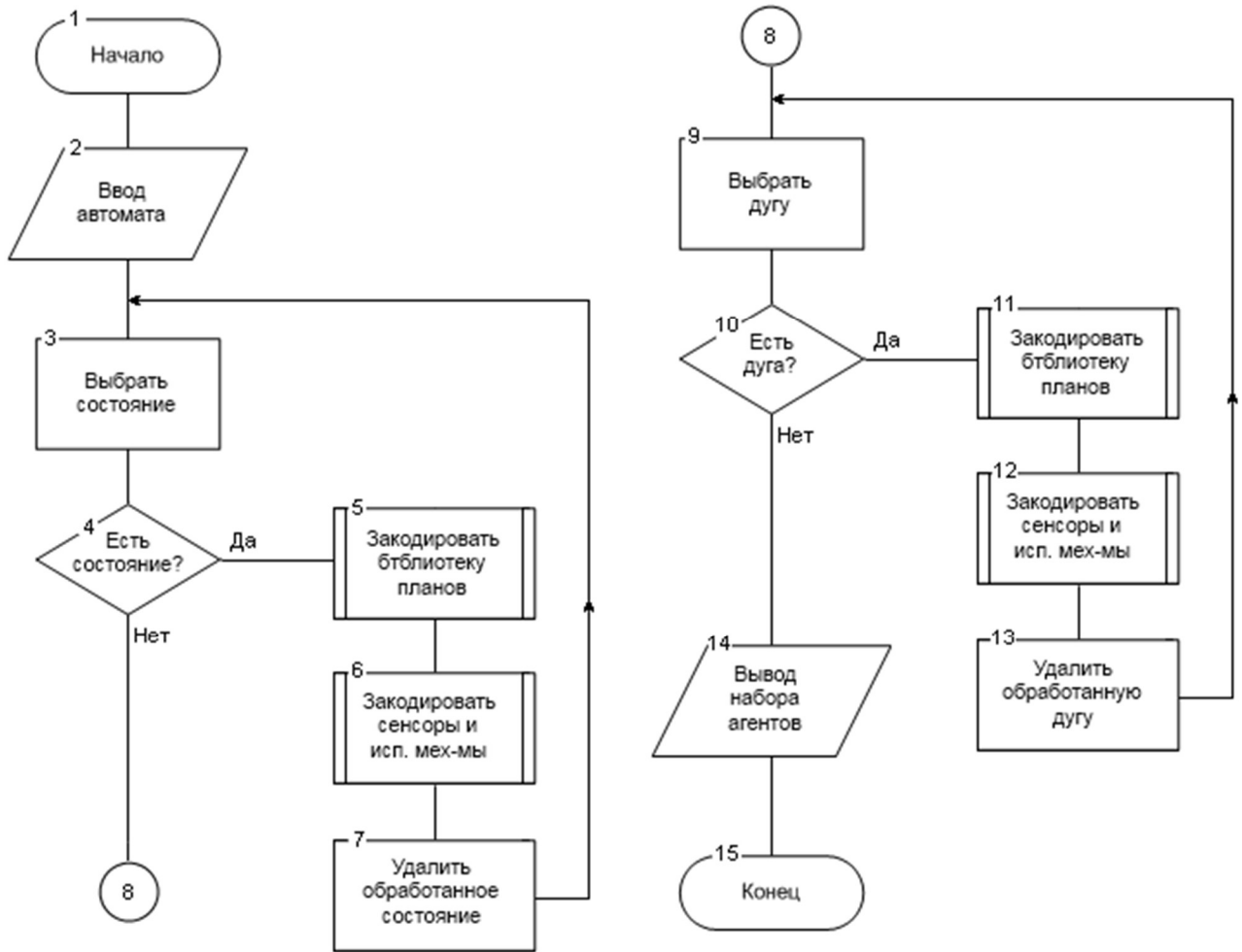


Рисунок 20 — Блок-схема алгоритма реализации многоагентной модели некоторого компонента информационной системы

- 4) в блоках 5 и 6 осуществляется кодирование ведомого агента в соответствии с описанием выбранного состояния (при кодировании используются языки AgentSpeak и Java);
- 5) в блоке 7 обработанное состояние удаляется из автомата, выполняется переход к блоку 3;
- 6) в блоке 9 выбирается ещё не обработанная дуга автомата;
- 7) в блоке 10 проверяется, удалось ли выбрать дугу; если удалось, выполняется переход к блоку 11, иначе — к блоку 14;
- 8) в блоках 11, 12 выполняется кодирование ведущего агента (примечание: каждая следующая итерация дополняет одного и того же ведущего

агента; в случаях, когда структура агента получается слишком сложной, возможно введение дополнительного ведущего агента);

9) в блоке 13 из автомата удаляется обработанная дуга, выполняется переход к блоку 9;

10) в блоке 14 выводится полученное множество агентов компонента.

3.2.6 Шлюз связи с информационной системой

Шлюз связи предназначен для пересылки сообщений, идущих от агентов-злоумышленников к ИС и обратно: модуль играет роль моста, связующего многоагентную систему с ИС.

Реализация модуля зависит от того, каким образом развернута инфраструктура для запуска агентов-злоумышленников. Если выбрана распределённая архитектура, и агенты функционируют на вычислительных машинах, имеющих сетевую связь с целевой ИС, то шлюз реализуется как сетевой мост, дублирующий сообщения, идущие от агентов-злоумышленников, в сеть [149].

Алгоритм работы шлюза представлен на блок-схеме, изображённой на рисунке 21, и включает следующие шаги:

1) в блоке 2 выполняется ввод сообщения от агента-злоумышленника;

2) в блоке 3 проверяется, является ли сообщение запросом на создание нового подключения к какому-либо компоненту ИС; если является, выполняется переход к блоку 4, иначе — к блоку 5;

3) в блоке 5 проверяется, является ли сообщение запросом на отправку каких-либо данных компоненту ИС; если является, выполняется переход к блоку 6, иначе — к блоку 7;

4) в блоке 7 проверяется, является ли сообщение запросом на получение каких-либо данных от компонента информационной системы; если является, выполняется переход к блоку 8, иначе — к блоку 9;

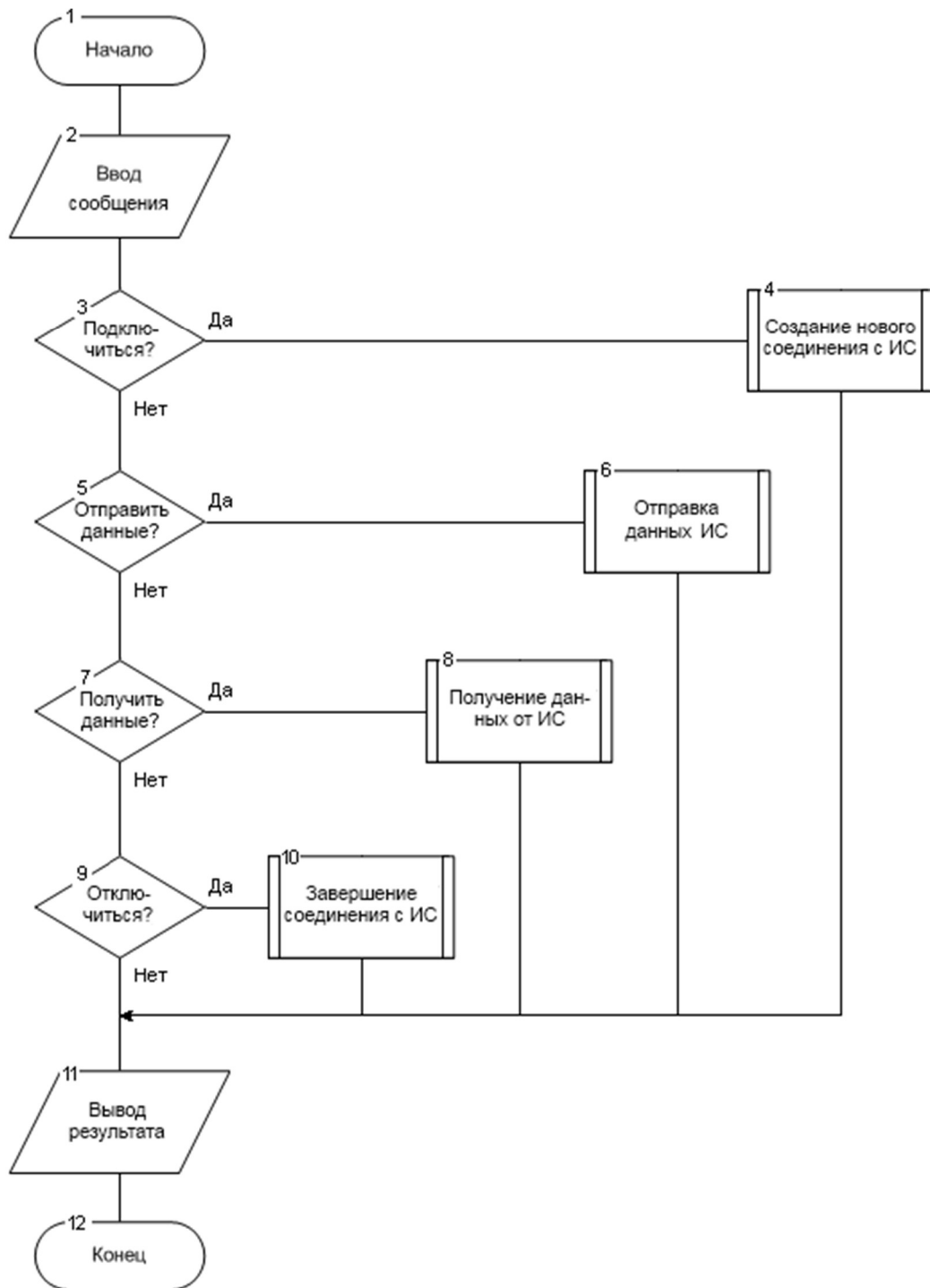


Рисунок 21 — Блок-схема алгоритма работы шлюза

5) в блоке 9 проверяется, является ли сообщение запросом на закрытие соединения с компонентом ИС; если является, выполняется переход к блоку 10, иначе — к блоку 11;

6) в блоке 12 результат взаимодействия с ИС возвращается агенту-злоумышленнику, инициировавшему запрос.

3.3 Выводы по третьей главе

Анализ моделей, определяющих структуру и функционирование интеллектуальных агентов, показал, что для такой прикладной области, как защита информации, лучшей моделью является BDI. С помощью этой модели можно имитировать как поведение злоумышленников, так и функционирование компонентов ИС.

Для реализации модели BDI выбран язык AgentSpeak. Данный язык является интерпретируемым, для его использования в многоагентную систему должна быть включена исполняющая среда, в режиме реального времени транслирующая когнитивные программы интеллектуальных агентов.

Разработана архитектура многоагентной системы для анализа защищенности ИС.

Реализован модуль имитации злоумышленных воздействий. Созданы алгоритмы реализации агентов-злоумышленников.

Для того чтобы обезопасить критичные компоненты ИС от вывода из строя в процессе имитации злоумышленных воздействий, необходимо использовать полунатурный подход, заключающийся в моделировании этих критичных компонент. Реализация такого подхода предполагает внедрение в многоагентную систему дополнительных функциональных модулей: коммутатора и модуля, содержащего моделируемые компоненты исследуемой системы.

Для обеспечения возможности взаимодействия интеллектуальных агентов с ИС необходим модуль, называемый ядром, в котором размещаются сенсоры и исполнительные механизмы агентов.

Реализован модуль виртуальных компонентов ИС. Создан алгоритм реализации виртуальных компонентов ИС.

Реализован шлюз связи с информационной системой.

4 РЕЗУЛЬТАТЫ ЭКСПЕРИМЕНТАЛЬНЫХ ИССЛЕДОВАНИЙ

4.1 Разработка методики проведения экспериментальных исследований

Цель экспериментальных исследований – оценить эффективность предложенных моделей и методов для оценки защищённости ИС.

Необходимо проанализировать адекватность построенных сценариев поведения злоумышленника, а так же сравнить результаты анализа защищённости в полунатурном и натурном режиме.

Проведение экспериментальных исследований подразумевает выполнение следующих шагов:

1) Описание структуры ИС для проведения экспериментальных исследований;

2) Анализ информационной системы, включающий в себя этапы:

2.1) Этап анализа состава ИС, который заключается в сборе данных об ИС. На данном этапе производится выявление перечня хостов и выделение компонентов информационной системы. Определяются ресурсы информационной системы, критичные с точки зрения нарушения их конфиденциальности, доступности, целостности.

2.2) Этап определения уязвимостей, который заключается в экспертном определении уязвимостей (исходя из анализа состава ИС), так и в инструментальном определении, без попыток их эксплуатации.

2.3) Этап выявления потенциальных целей злоумышленника.

3) Построение модели информационной системы.

4) Формализация целей злоумышленника с использованием темпоральных логик CTL и LTL.

5) Построение сценариев злоумышленника посредством верификации модели информационной системы при заданных целях злоумышленника. Формирование библиотеки сценариев в виде XML-файла.

- 6) Определение критических объектов, выделение реальных и виртуальных хостов и компонентов ИС.
- 7) Реализация агентов-злоумышленников под заданные сценарии в соответствии с построенной библиотекой сценариев, построение МАС.
- 8) Выполнение имитации с использованием виртуальных компонентов исследуемой системы, оценка нанесённого ущерба.
- 9) Выполнение имитации без включения виртуальных компонентов исследуемой системы, оценка нанесенного ущерба.
- 10) Сравнение результатов, полученных на 9-ом и 10-ом шагах. Проводится оценка количества отказов компонентов информационной системы.
- 11) Оценка защищенности исследуемой системы для двух режимов.

4.2 Описание структуры ИС для проведения экспериментальных исследований

Для проведения экспериментальных исследований выбрана часть ИС ФГАОУ ВО «Волгоградского государственного университета», включающая в себя общеуниверситетские сервисы и хосты и компоненты института приоритетных технологий. Исследуемая подсеть выделена в диапазоне 10.10.7.0\24.

В ИС выделяются следующие хосты и их группы:

- IP: 10.10.7.1. Шлюз доступа к ресурсам управления информатизации и телекоммуникаций ВолГУ. Проxy-сервер SQUID.
- IP: 10.10.7.2. ОС: Шлюз доступа к ресурсам управления информатизации и телекоммуникаций ВолГУ. Управляемый коммутатор HP IX.
- IP: 10.10.7.120. АРМ лаборатории технических средств защиты информации. На хосте развернута операционная система Windows XP SP3.
- IP: 10.10.7.121. WiFi-роутер ASUS RT-12. Версия ПО: 2.0.1.
- IP: 10.10.7.123. АРМ сотрудников деканата. ОС: Windows XP SP3, предоставлен удаленный доступ к системе 1С:Университет, 1С:Документооборот.

- IP: 10.10.7.124. АРМ сотрудников деканата. ОС: Windows XP SP3, предоставлен удаленный доступ к системе 1С:Документооборот.
- IP: 10.10.7.125. АРМ сотрудников деканата. ОС: Windows 7 Professional SP1 x64, предоставлен удаленный доступ к системе 1С:Университет, выполняется работа на сайте ВолГУ в системе 1С:Битрикс.
- IP: 10.10.7.144. Управляемый коммутатор Cisco Catalyst 2950 (C2950-I6Q4L2-M), Версия ПО: 12.1(9)EA1. Доступно подключение по протоколу telnet.
- 10.10.7.176 – 10.10.7.178. АРМ преподавательской кафедры информационной безопасности института приоритетных технологий. ОС: Windows XP SP3. Предоставлен удаленный доступ к системе 1С:Университет, 1С:Документооборот. Входят в домен INFSEC.
- IP 10.10.7.179. АРМ преподавательской кафедры информационной безопасности института приоритетных технологий. ОС: Windows XP SP3. Предоставлен удаленный доступ к системе 1С:Университет, 1С:Документооборот, предоставлен удаленный доступ по протоколу RDP, предоставлен общий доступ к сетевым ресурсам, принтеру Canon LBP-810. Входит в домен INFSEC.
- IP 10.10.7.180. АРМ преподавательской кафедры информационной безопасности института приоритетных технологий. ОС: Windows 7 Professional SP1 x64. Предоставлен удаленный доступ к системе 1С:Университет, 1С:Документооборот. Входит в домен INFSEC.
- IP: 10.10.7.181. ОС: Windows Server 2008 R2. Является сервером, выполняющим роль контроллера домена. На сервере развернуты: Active Directory, DNS, DHCP, служба терминалов. Домен организации: INFSEC. FQDN-имя: server.infsec.volsu.ru.
- IP 10.10.7.182. Маршрутизатор Cisco 2600. Маршрутизатор предоставляет доступ к сети ВолГУ для 10 АРМ с IP-адресами 192.168.10.100–192.168.10.110, недоступными снаружи. Версия ПО: 12.2(11).
- IP 10.10.7.183. Сервер для развертывания средств виртуализации. ОС: VMWare ESXi 5.5. Для данного сервера зарезервированы IP 10.10.7.184 – 10.10.7.194.

— IP: 10.10.7.184. ОС: Windows XP SP3. Специально сформированная уязвимая виртуальная машина. Содержит web-сервер Apache 2.4.0, пакет DVWA с отключенной авторизацией.

— IP: 10.10.7.186. ОС: Linux. Специально сформированная уязвимая виртуальная машина. Содержит web-сервер Apache 2.4.0, специально разработанный Web-сайт с 2 уязвимостями SQL-injection (GET, POST), 1 XSS-уязвимостью.

Структура исследуемой ИС изображена на рисунке 22.

Топология сети ВолГУ. Корпус К. Институт приоритетных технологий.

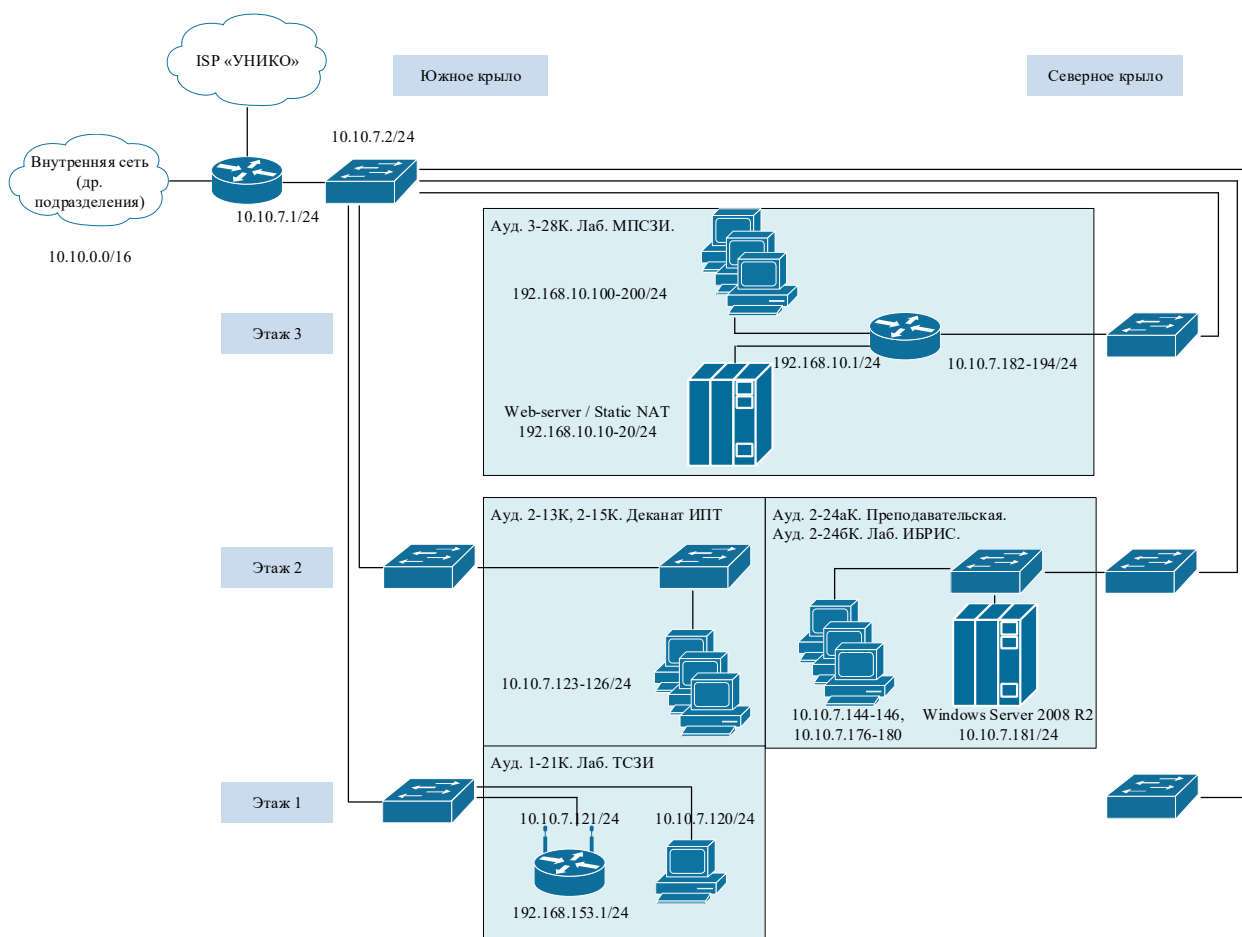


Рисунок 22 – Структура исследуемой ИС

Для анализа и выявления уязвимостей в информационной системе использовались сканеры уязвимостей: [150]

— Nmap 7.70 – сканер уязвимостей общего назначения. Для каждого вышеуказанного IP-адреса сканировалось 1000 наиболее известных портов (запуск

с параметром «--top-ports 1000»), включен режим выявления сервисов (-sV), включен режим выявления уязвимостей с использованием расширения vulnersCom («--script vulners.nse»).

- Nessus 7.3.4 Pro (Trial version) – сканер уязвимостей общего назначения. Запуск производился в режиме по умолчанию;
- Sqlmap – сканер для выявления и эксплуатации SQL-инъекций. Запуск производился в режиме минимального риска для исследуемой машины («--risk=1 -level=1»);
- XSSTracer – средство для выявления и эксплуатации XSS-инъекций;
- XSSPy – средство для выявления и эксплуатации XSS-инъекций;
- EternalScanner – средство для массового выявления уязвимости MS17-10;
- Nemesida Scanner – сканер уязвимостей для анализа защищенности web-ресурсов.

Цели злоумышленника формулируются из целей анализа защищенности следующим образом:

- 1) Вызвать отказ в обслуживании:
 - Контролера домена INFSEC;
 - Гипервизора VMWare ESXi.
- 2) Получить привилегированный доступ к:
 - Сетевому оборудованию;
 - Контроллеру домена INFSEC;
 - АРМ пользователей, выполняющих административную работу в системах «1С:Университет», «1С:Документооборот КОРП», «1С:Битрикс»;
- 3) Получить доступ к информации, содержащейся на:
 - Виртуализируемых АРМ (web-серверах).

Анализ защищенности проводится двумя методами:

- 1) методом «черного ящика» (с подключением к сети, без предоставления реквизитов доступа к ресурсам);

2) методом «серого ящика» (с предоставлением ограниченного санкционированного доступа к ресурсам).

Данные условия определяют исходные знания нарушителя и его возможности, и соответственно, будут определять порядок его действий.

В качестве исходных знаний и точек подключения злоумышленника рассматриваются:

— доступ с подключением к подсети Wi-Fi IP: 10.10.7.121 без предоставления реквизитов доступа;

— доступ с подключением к подсети Wi-Fi, IP: 10.10.7.121 с предоставлением реквизитов доступа группы «Student» к домену INFSEC;

— доступ с подключением к коммутатору 10.10.7.2 из подсети 10.10.7.0/24 с IP: 10.10.7.250 без предоставления реквизитов доступа;

— доступ с подключением к коммутатору 10.10.7.2 из подсети 10.10.7.0/24 с IP: 10.10.7.250 с предоставлением реквизитов доступа группы «Student» к домену INFSEC.

4.3 Проведение экспериментальных исследований

При анализе вышеуказанной подсети с использованием сканеров уязвимостей были выявлены следующие уязвимости, представленные в таблице 5.

Таблица 5 – Выявленные уязвимости ИС

| п/п | Уязвимость | Уровень критичности | Уязвимые хосты, компоненты, ресурсы | Описание уязвимости | Уязвимость позволяет реализовать цель злоумышленника | | |
|-----|--------------------------|----------------------|--|--|--|---|---|
| | | | | | 1 | 2 | 3 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | CVE-2008-4250 (MS08-067) | Высокий (CVSS: 10.0) | 10.10.7.120 10.10.7.123 10.10.7.124 10.10.7.176 10.10.7.177 10.10.7.178 10.10.7.179 10.10.7.184 | Уязвимость ОС позволяет получить удаленный доступ с максимальными привилегиями к компьютеру жертвы. Данная уязвимость легко эксплуатируется. | — | + | + |

Продолжение таблицы 5

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|------------------------|--|---|---|---|---|
| 2 | CVE-2017-0143 (EternalBlue, MS17-010) | Высокий (CVSS: 9.3) | 10.10.7.125 10.10.7.180 | EternalBlue - эксплойт для Windows, который эксплуатирует уязвимость MS17-010 от 14 марта 2017 года. С помощью этого эксплойта нарушитель может получать удаленный доступ к компьютеру и выполнять произвольные программы. Использование данного эксплойта стало причиной массовых эпидемий вирусов шифровальщиков WannaCry и Petya. | — | + | — |
| 3 | NetBios/LLMNR/wpad спуфинг CWE-284 | Высокий (CVSS: 9.8) | 10.10.7.176 10.10.7.177 10.10.7.178 10.10.7.179 10.10.7.180 10.10.7.181 | Суть атаки заключается в том, что атакующий отвечает на каждый широковещательный NetBios/Browser/LLMNR-запрос своим ip адресом. Такие запросы посылаются каждой Windows-рабочей станцией, если та при разрешении имени в адрес использует короткие имена, без указания полного домена (например, ad вместо ad.infsec.volsu.ru). В результате, обманутая ложным ответом, рабочая станция может выполнить подключение к сервису на хосте атакующего. В ряде случаев, в зависимости от запрашиваемого удаленного сервиса, это влечет за собой отправку хэша пароля | — | + | — |
| 4 | CVE-2017-9788 | Средний (CVSS: 6.4) | 10.10.7.184 10.10.7.186 | CVE-2017-9788 – Уязвимость модуля mod_auth_digest веб-сервера Apache HTTP Server, позволяющая нарушителю получить доступ к конфиденциальной информации или вызвать отказ в обслуживании | — | — | + |

Продолжение таблицы 5

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|----------------|---------------------|----------------------------|---|---|---|---|
| 5 | CVE-2017-15715 | Средний (CVSS: 6.8) | 10.10.7.184 10.10.8.186 | CVE-2017-15715 - В функции <code>ap_rgetline_core()</code> скрипта <code>server / protocol.c</code> существует возможность обхода ACL из-за неправильной обработки выражений <code><FilesMatch></code> . Удаленный злоумышленник может обойти ограничения по загрузке файла. | — | — | + |
| 6 | CVE-2018-11763 | Средний (CVSS: 4.9) | 10.10.7.184 10.10.8.186 | CVE-2018-11763 – уязвимость веб-сервера Apache HTTP Server, которая связана с недостаточной проверкой входных данных. Эксплуатация уязвимости может позволить нарушителю, действующему удалённо, вызвать отказ в обслуживании путем исчерпания лимита одновременно открытых соединений через непрерывную отправку кадров максимального размера с типом <code>SETTINGS</code> . | — | — | — |
| 7 | CVE-2015-7547 | Средний (CVSS: 6.8) | 10.10.7.183 | Уязвимость существует из-за ошибки переполнения стекового буфера в функции <code>getaddrinfo()</code> библиотеки <code>glibc</code> . В VMware ESXi 5.5 используется уязвимая версия библиотеки, позволяющая выполнить удаленное выполнение произвольного кода. Удаленный пользователь может выполнить произвольный код на целевой системе с использованием контролируемого злоумышленником | — | — | — |

Продолжение таблицы 5

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|---------------|--|---|--|---|---|---|
| | | | | доменного имени, контролируемого злоумышленником DNS-сервера или атаки «человек-в-середине». | | | |
| 8 | CVE-2017-4925 | Низкий (CVSS: 2.1) | 10.10.7.183 | Уязвимость в VMware ESXi 5.5, связанная с разыменованием нулевого указателя, возникающим в момент обработки программой RPC-запросов гостевой системы. Таким образом, обладая лишь привилегиями обычного пользователя, злоумышленник сможет вызвать отказ виртуальной машины. | + | — | — |
| 9 | CWE-307 | Средний (CVSS: 7,6. Вектор CVSS: AV:N/AC:H/Au:N/C:C/I:C/A:C/E) | 10.10.7.125 10.10.7.144 10.10.7.179 10.10.7.181 10.10.7.182 | Возможность подключиться к сервисам любого IP-адреса. Возможность брутфорс атаки на сервисы RDP (10.10.7.179, 10.10.7.181). Возможность атаки bruteforce на интерфейс web-сайта ВолГУ «1С-Битрикс» (10.10.7.125) Возможность брутфорс-атаки по протоколу Telnet на сетевое оборудование (10.10.7.144, 10.10.7.182) | — | + | — |
| 10 | CVE-1999-0024 | Средний (CVSS: 5.0) | 10.10.7.181 | Данная уязвимость DNS-серверов позволяет выполнить «отравление кэша» DNS-сервера, посредством отправки DNS-ответов на запросы с предсказуемыми идентификаторами запроса DNS ID. Злоумышленник может навязать ложные DNS-записи. | + | — | — |

Продолжение таблицы 5

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|---------|--|----------------------------|---|---|---|---|
| 11 | CWE-650 | Высокий (CVSS: 7.0. Вектор CVSS: AV:N/AC:L/Au:N/C:P/I:P/A:P) | 10.10.7.184 | Некорректная настройка Web-сервера Apache, связанная с доверием используемых HTTP-методов на стороне сервера. Возможность использования метода HTTP PUT. Метод используется для загрузки данных, которые сохраняются на сервере по предоставленному пользователем URL-адресу. Злоумышленник может загрузить вредоносное содержимое в приложение. | — | — | + |
| 12 | CWE-514 | Высокий (CVSS: 9.3. Вектор CVSS: AV:N/AC:M/Au:N/C:C/I:C/A:C) | 10.10.7.184 | Получение скрытого (недекларированного) канала передачи данных в следствии эксплуатации иных уязвимостей. | — | — | + |
| 13 | CWE-89 | Средний (CVSS: 5.4. Вектор CVSS: AV:N/AC:H/Au:N/C:C/I:N/A:N) | 10.10.7.184 10.10.7.186 | Множественные уязвимости, связанные с SQL-инъекциями в web-приложении. Уязвимость связана с возможностью модификации (расширения) SQL-запроса, посредством манипуляции с отправляемыми на web-сервер данными. Данная уязвимость связана с отсутствием проверки вводимых пользователем данных. | — | — | + |
| 14 | CWE-80 | Низкий (CVSS: 2.1. Вектор CVSS: AV:N/AC:H/Au:S/C:P/I:N/A:N) | 10.10.7.184 10.10.7.186 | Множественные XSS-уязвимости, связанные с отсутствием проверки вводимых пользователем данных, которые могут быть интерпретированы как элементы web-страницы (управляющие символы web- | — | — | + |

Продолжение таблицы 5

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|----|---------------|--|----------------------------|---|---|---|---|
| | | | | страницы). Злоумышленник может внедрить дополнительные элементы в web-страницу | | | |
| 15 | CWE-288 | Средний (CVSS: 5.9. Вектор CVSS: AV:A/AC:H/Au:S/C:C/I:N/A:C) | 10.10.7.121 | Обход аутентификации. В SOHO-оборудовании ASUS в прошивках 2.0.x. Для аутентификации Web-интерфейса требуется предоставить логин и пароль, однако возможно получить привилегированный доступ к настройкам оборудования (в т.ч. к логину и паролю администратора, хранящимися в открытом виде). | — | + | — |
| 16 | CVE-2001-0537 | Высокий (CVSS: 9.3) | 10.10.7.144 10.10.7.182 | Возможность выполнения команд от имени привилегированного пользователя в оборудовании Cisco. HTTP-сервер в Cisco IOS с версии 11.3 по версию 12.2 при использовании локальной аутентификации позволяет выполнять произвольные команды посредством указания высокого уровня доступа в URL | — | + | — |

При анализе информационной системы, исходя из описания, выше были выделены компоненты хостов, определено наличие уязвимости на данном хосте. Дополнительно, в процессе анализа были определены возможности эксплуатации уязвимостей при выбранных начальных условиях сценариев. В зависимости от того, какие права доступа были предоставлены злоумышленнику в исследуемой информационной системе, а также в зависимости от точки подключения к сети, непосредственная эксплуатация уязвимостей была невозможна, что приведено в таблице 6.

Таблица 6 – Описание хостов и компонентов информационной системы

| Назначение хоста | Имя хоста в модели | IP-адрес | Количество компоненто в на хосте | Компоненты | Уязвимость компонента | Возможность эксплуатации уязвимости при начальном сценарии | | | |
|--------------------|--------------------|-------------|----------------------------------|---|-----------------------|--|---|---|---|
| | | | | | | 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
| АРМ пользователя | h120 | 10.10.7.120 | 3 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. | CVE-2008-4250 | — | — | + | + |
| WiFi-маршрутизатор | wifi121 | 10.10.7.121 | 3 | Workflow – описывает статус функционирования. Network – описывает подсети и функции маршрутизатора. DevMng – описывает функции маршрутизатора, процессы идентификации и аутентификации | CWE-288 | + | + | — | — |
| АРМ пользователя | h123 | 10.10.7.123 | 5 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации CIU – описывает права доступа к системе «IC:Университет» CID – описывает права доступа к системе «IC:Документооборот» | CVE-2008-4250 | — | — | + | + |

Продолжение таблицы 6

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---------------------|-------|-------------|---|--|---------------|---|---|---|---|
| АРМ пользователя | h124 | 10.10.7.124 | 4 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации CID – описывает права доступа к системе «1С:Документооборот» | CVE-2008-4250 | — | — | + | + |
| АРМ пользователя | h125 | 10.10.7.125 | 5 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. CIU – описывает права доступа к системе «1С:Университет» CIB – описывает доступ к системе «1С:Битрикс» | CVE-2017-0146 | — | — | + | + |
| | | | | | CWE-307 | + | + | + | + |
| Коммутатор | sw144 | 10.10.7.144 | 3 | Workflow – описывает статус функционирования. Network – описывает подсети и функции маршрутизатора. DevMng – описывает функции маршрутизатора, процессы идентификации и аутентификации | CWE-307 | + | + | + | + |
| | | | | | CVE-2001-0537 | + | + | + | + |
| АРМ пользователя | h176 | 10.10.7.176 | 6 | SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. WACLD – описывает права доступа в ОС Windows, процессы идентификации и аутентификации в домене INFSEC. | CVE-2008-4250 | — | — | + | + |
| | | | | | CWE-284 | — | — | + | + |

Продолжение таблицы 6

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---------------------|------|-------------|---|---|---------------|---|---|---|---|
| | | | | <p>C1U – описывает права доступа к системе «1С:Университет»</p> <p>C1D – описывает права доступа к системе «1С:Документооборот»</p> <p>Workflow – описывает статус функционирования</p> | | | | | |
| АРМ пользователя | h177 | 10.10.7.177 | 6 | <p>Workflow – описывает статус функционирования</p> <p>SW – описывает установленное ОС и ПО.</p> <p>WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации.</p> <p>WACLD – описывает права доступа в ОС Windows, процессы идентификации и аутентификации в домене INFSEC.</p> <p>C1U – описывает права доступа к системе «1С:Университет»</p> <p>C1D – описывает права доступа к системе «1С:Документооборот»</p> | CVE-2008-4250 | — | — | + | + |
| | | | | | CWE-284 | — | — | + | + |
| АРМ пользователя | h178 | 10.10.7.178 | 6 | <p>Workflow – описывает статус функционирования</p> <p>SW – описывает установленное ОС и ПО.</p> <p>WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации.</p> <p>WACLD – описывает права доступа в ОС Windows, процессы идентификации и аутентификации в домене INFSEC.</p> <p>C1U – описывает права доступа к системе «1С:Университет»</p> <p>C1D – описывает права доступа к системе «1С:Документооборот»</p> | CVE-2008-4250 | — | — | + | + |
| | | | | | CWE-284 | — | — | + | + |

Продолжение таблицы 6

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|---------------------|------|-------------|---|--|---------------|---|---|---|---|
| АРМ пользователя | h179 | 10.10.7.179 | 7 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. WACLD – описывает права доступа в ОС Windows, процессы идентификации и аутентификации в домене INFSEC. WShared – описывает права доступа и функции общих ресурсов. CIU – описывает права доступа к системе «1С:Университет» CID – описывает права доступа к системе «1С:Документооборот» | CVE-2008-4250 | — | — | + | + |
| | | | | | CWE-284 | — | — | + | + |
| | | | | | CWE-307 | + | + | + | + |
| АРМ пользователя | h180 | 10.10.7.180 | 6 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. WACLD – описывает права доступа в ОС Windows, процессы идентификации и аутентификации в домене INFSEC. CIU – описывает права доступа к системе «1С:Университет» CID – описывает права доступа к системе «1С:Документооборот» | CVE-2017-0146 | — | — | + | + |
| | | | | | CWE-284 | — | — | + | + |

Продолжение таблицы 6

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|------------------------------------|-------|-------------|---|---|----------------|---|---|---|---|
| Сервер домена INFSEC | s181 | 10.10.7.181 | 7 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. RAD – описывает роль AD. RDNS – описывает роль DNS RDHCP – описывает роль DHCP RTRDP – описывает роль удаленных терминалов. | CWE-284 | — | — | + | + |
| | | | | | CWE-307 | + | + | + | + |
| | | | | | CVE-1999-0024 | + | + | + | + |
| Маршрутизатор | r182 | 10.10.7.182 | 3 | Workflow – описывает статус функционирования. Network – описывает подсети и функции маршрутизатора. DevMng – описывает функции маршрутизатора, процессы идентификации и аутентификации | CWE-307 | + | + | + | + |
| Гипервизор VMWare ESXi | s183 | 10.10.7.183 | 2 | Workflow – описывает статус функционирования SWESXi – описывает ПО гипервизора. | CVE-2015-7547 | — | — | — | — |
| | | | | | CVE-2017-4925 | — | + | — | + |
| Виртуальный АРМ (web-сервер) | vh184 | 10.10.7.184 | 5 | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. WACL – описывает права доступа в ОС Windows, процессы идентификации и аутентификации. WA – описывает web-сервер Apache. DWVA – описывает пакет DVWA | CVE-2008-4250 | — | — | + | + |
| | | | | | CVE-2017-9788 | + | + | + | + |
| | | | | | CVE-2017-15715 | + | + | + | + |
| | | | | | CVE-2018-11763 | + | + | + | + |
| | | | | | CWE-650 | + | + | + | + |
| | | | | | CWE-514 | — | — | — | — |
| | | | | | CWE-89 | + | + | + | + |
| | | | | | CWE-80 | + | + | + | + |

Продолжение таблицы 6

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 |
|--|-------|-------------|---|---|----------------|----|----|----|----|
| Виртуальный АРМ (web-сервер) | vh186 | 10.10.7.186 | | Workflow – описывает статус функционирования SW – описывает установленное ОС и ПО. LACL – описывает права доступа в ОС Linux, процессы идентификации и аутентификации. WA – описывает web-сервер Apache. Web1 – описывает уязвимый сайт. | CVE-2017-9788 | + | + | + | + |
| | | | | | CVE-2017-15715 | + | + | + | + |
| | | | | | CVE-2018-11763 | + | + | + | + |
| | | | | | CWE-89 | + | + | + | + |
| | | | | | CWE-80 | + | + | + | + |
| Уязвимостей, подверженных эксплуатации при заданных начальных условиях | | | | | | 19 | 20 | 34 | 35 |
| из них, позволяющих выполнить цель злоумышленника 1 | | | | | | 1 | 2 | 1 | 2 |
| позволяющих выполнить цель злоумышленника 2 | | | | | | 7 | 7 | 21 | 21 |
| позволяющих выполнить цель злоумышленника 3 | | | | | | 5 | 5 | 6 | 6 |

Выделенные ресурсы информационной системы, интересующие злоумышленника для достижения целей приведены в таблице 7.

Таблица 7 – Связь ресурсов ИС и целей злоумышленника

| Цель злоумышленника | Детализация цели | Спецификация цели (имя проверяемого ресурса) | Спецификация детализированной цели (имя проверяемого ресурса) | Критичность ресурса |
|---|--|--|---|--|
| Вызвать отказ в обслуживании | Отказ контролера домена INFSEC | sres.dos | s181.dos | Высокая |
| | Отказ гипервизора VMWare ESXi | | s183.dos | Высокая |
| Получить привилегированный доступ | Доступ к сетевому оборудованию | sres.rootaccess | sw144.rootaccess wifi121.rootaccess r182.rootaccess | Высокая Средняя Высокая |
| | Доступ к контроллеру домена INFSEC | | s181.rootaccess | Высокая |
| | К АРМ пользователей, выполняющих административную работу в системах «ИС:Университет», «ИС:Документооборот КОРП», «ИС:Битрикс»; | | h120.rootaccess h123.rootaccess h124.rootaccess h125.rootaccess h176.rootaccess h177.rootaccess h178.rootaccess h179.rootaccess h180.rootaccess | Средняя Средняя Средняя Средняя Средняя Средняя Средняя Средняя |
| Получить доступ к информации, содержащейся в системах | Доступ к данным web-хостов | sres.infleak | vh184.infleak vh186.infleak | Низкая Низкая |

В зависимости от входных данных о злоумышленнике (исходные знания о системе, права доступа, точки подключения и пр.) были сформированы 4 варианта

библиотеки сценариев (Приложение А), фрагмент которых представлен на рисунке 23 и таблице 8.

Таблица 8 – Обобщенное описание сформированных библиотек сценариев

| Библиотека сценариев | Количество сценариев по цели злоумышленника | |
|----------------------|---|------------|
| | Цель | Количество |
| 1 | Вызвать отказ в обслуживании | 3 |
| | Получить привилегированный доступ | 16 |
| | Получить доступ к информации, содержащейся в системах | 6 |
| 2 | Вызвать отказ в обслуживании | 4 |
| | Получить привилегированный доступ | 16 |
| | Получить доступ к информации, содержащейся в системах | 6 |
| 3 | Вызвать отказ в обслуживании | 4 |
| | Получить привилегированный доступ | 26 |
| | Получить доступ к информации, содержащейся в системах | 6 |
| 4 | Вызвать отказ в обслуживании | 5 |
| | Получить привилегированный доступ | 26 |
| | Получить доступ к информации, содержащейся в системах | 6 |

Для имитационного моделирования было разработано 4 многоагентных системы (таблица 9).

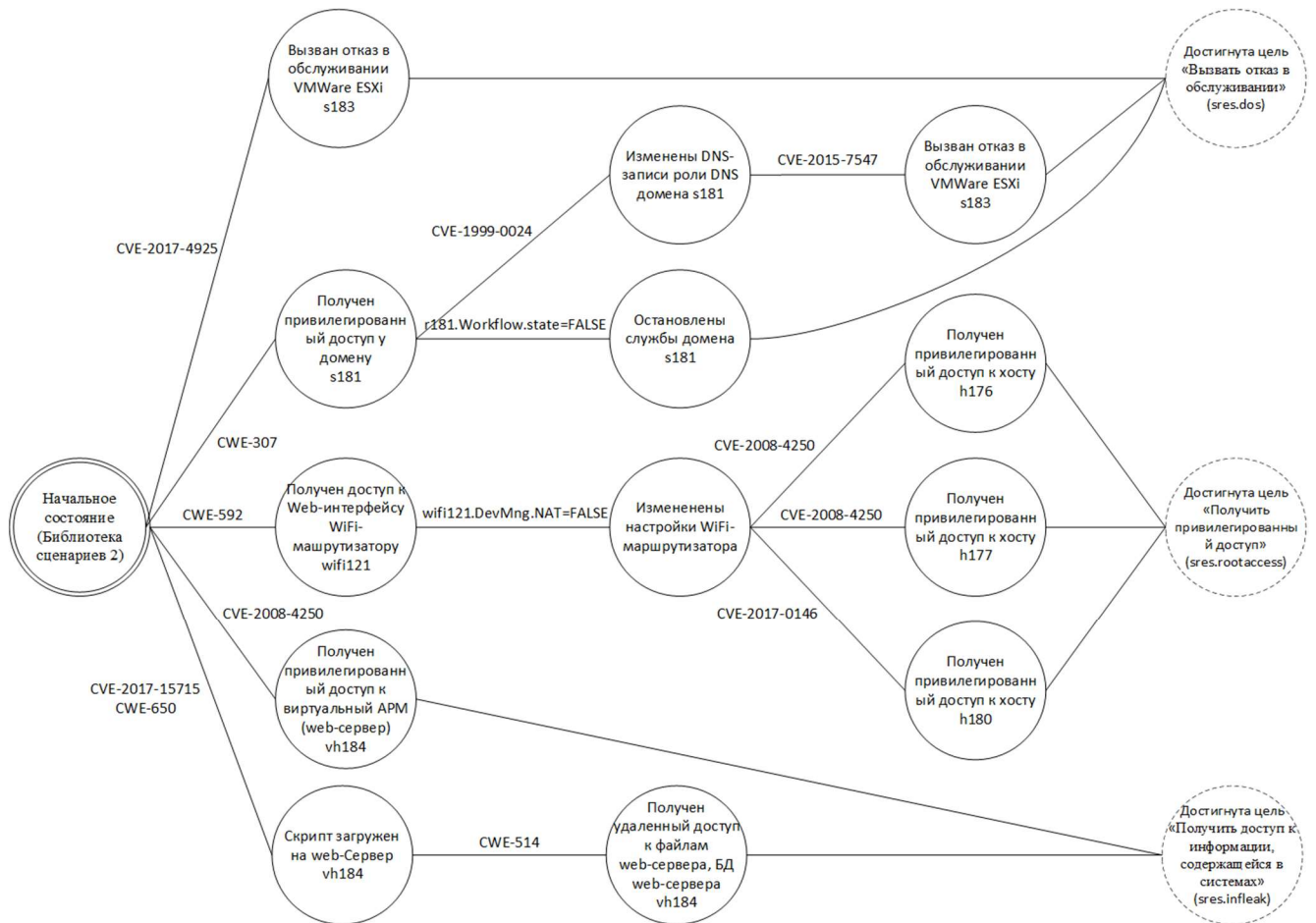


Рисунок 23 — Графическое представление фрагмента библиотеки сценариев

Таблица 9 – Обобщенное описание сформированных в соответствии с библиотеками сценариев многоагентных систем

| Библиотека сценариев | Количество | | |
|----------------------|----------------|-----------------|-----------------|
| | мастер-агентов | ведущих агентов | ведомых-агентов |
| 1 | 1 | 15 | 16 |
| 2 | 1 | 16 | 17 |
| 3 | 1 | 34 | 16 |
| 4 | 1 | 35 | 17 |

Количество уникальных агентов-злоумышленников (таблица 10): 1 уникальный мастер-агент, 37 уникальных ведущих агентов, 19 уникальных ведомых агентов.

Таблица 10 – Описание агентов многоагентных систем

| Тип агента | Имя файла агента в модели | Назначение агента |
|--------------------------|--|---|
| Мастер-агент | master.asl | Получение и вывод данных от ведущих агента |
| Ведущий агент | От attack_leader_1.asl до attack_leader_37.asl | Координация сценариев 1 – 37. |
| Ведомые агенты | attack_CVE-2008-4250.asl | Агент, эксплуатирующий уязвимость CVE-2008-4250 |
| | attack_CVE-2017-0143.asl | Агент, эксплуатирующий уязвимость CVE-2017-0143 |
| | attack_CWE-284.asl | Агент, эксплуатирующий уязвимость CWE-284 посредством NetBios-спуфинга и формирования ложных запросов. |
| | attack_CVE-2017-9788.asl | Агент, эксплуатирующий уязвимость CVE-2017-9788 |
| | attack_CVE-2017-15715.asl | Агент, эксплуатирующий уязвимость CVE-2017-15715 |
| | attack_CVE-2018-11763.asl | Агент, эксплуатирующий уязвимость CVE-2018-11763 |
| | attack_CVE-2015-7547.asl | Агент, эксплуатирующий уязвимость CVE-2015-7547 |
| | attack_CVE-2017-4925.asl | Агент, эксплуатирующий уязвимость CVE-2017-4925. Требует учетную запись пользователя домена INFSEC. |
| | attack_CWE-307_RDP_telnet_web.asl | Агент, эксплуатирующий уязвимость CWE-307. Выполняет перебор паролей по указанному протоколу: RDP, telnet, web. |
| | attack_CVE-1999-0024_poison.asl | Агент, эксплуатирующий уязвимость CVE-1999-0024 посредством прослушивания трафика DNS и навязывания ложного ответа. |
| | attack_CVE-1999-0024_AD.asl | Агент, эксплуатирующий уязвимость CVE-1999-0024 посредством изменения записи в AD от имени Администратора домена. |
| | attack_CWE-650.asl | Агент, эксплуатирующий уязвимость CWE-650 (загрузка PHP shell-кода R57Shell) |
| | attack_CWE-514.asl | Агент, эксплуатирующий уязвимость CWE-514 (взаимодействие с PHP-shell-кодом R57Shell) |
| | attack_CWE-89.asl | Агент, эксплуатирующий уязвимость CWE-89 (SQL-инъекции) |
| | attack_CWE-80.asl | Агент, эксплуатирующий уязвимость CWE-80 (XSS-инъекция) |
| | attack_CWE-288_asus.asl | Агент, эксплуатирующий уязвимость CVE-288 (обход аутентификации SOHO-маршрутизаторов ASUS) |
| attack_CVE-2001-0537.asl | Агент, эксплуатирующий уязвимость CVE-2001-0537 | |
| attack_shutdown.asl | Агент, выполняющий остановку служб, выключение APM | |
| attack_device_change.asl | Агент, выполняющий изменение настроек маршрутизатора | |

Для реализации агентов, имитирующих компоненты информационной системы, была определена структура агента, позволяющего обработать атакующее воздействие и вернуть результат.



Рисунок 24 — Структура агента в виде конечного автомата, описывающего процесс имитации воздействия на ИС

Процесс функционирования такого агента можно представить в виде конечного автомата (рисунок 24) и включает следующие состояния:

— S_0 — начальное состояние, в котором выполняется ожидание воздействие от агентов-злоумышленников; как только будет зафиксировано наличие запроса, этот пакет перехватывается, и выполняется переход в состояние S_1 ;

— S_1 — состояние, в котором анализируется перехваченный запрос; при поступлении запроса на получение результатов анализа выполняется переход в состояние S_2 ;

— S_2 — состояние, в котором формируется корректный ответ, совпадающий с ответом реального компонента информационной системы.

В результате предварительного анализа построенных сценариев было выявлено, что следующие сценарии, приведенные в таблице 11, вызывают нарушение работоспособности информационной системы.

Для минимизации нарушений работоспособности системы были реализованы агенты, имитирующие следующие компоненты информационной системы:

Таблица 11 – Описание сценариев, приводящих к отказу компонентов информационной системы

| № библиотеки сценариев | Цель | Количество сценариев, предположительно способное привести к отказу | Значения переменных модели, приводящие к отказу |
|------------------------|---|--|--|
| 1 | 2 | 3 | 4 |
| 1 | Вызвать отказ в обслуживании | 3 | s181.vuln.CVE-1999-0024 = TRUE, s181.vuln.CVE-1999-0024 = TRUE & s183.vuln.CVE-2015-7547 = TRUE, s181.Workflow.state = FALSE |
| | Получить привилегированный доступ | 7 | h120.vuln.CVE-2008-4250 = TRUE, h123.vuln.CVE-2008-4250 = TRUE, h124.vuln.CVE-2008-4250 = TRUE, h176.vuln.CVE-2008-4250 = TRUE, h177.vuln.CVE-2008-4250 = TRUE, h178.vuln.CVE-2008-4250 = TRUE, h179.vuln.CVE-2008-4250 = TRUE |
| | Получить доступ к информации, содержащейся в системах | 0 | – |
| 2 | Вызвать отказ в обслуживании | 4 | s181.vuln.CVE-1999-0024 = TRUE, s181.vuln.CVE-1999-0024 = TRUE & s183.vuln.CVE-2015-7547 = TRUE, s181.Workflow.state = FALSE, s181.WACLuser-=adm & s183.vuln.CVE-2017-4925 = TRUE |
| | Получить привилегированный доступ | 7 | h120.vuln.CVE-2008-4250 = TRUE, h123.vuln.CVE-2008-4250 = TRUE, h124.vuln.CVE-2008-4250 = TRUE, h176.vuln.CVE-2008-4250 = TRUE, h177.vuln.CVE-2008-4250 = TRUE, h178.vuln.CVE-2008-4250 = TRUE, h179.vuln.CVE-2008-4250 = TRUE |
| | Получить доступ к информации, содержащейся в системах | 0 | – |
| 3 | Вызвать отказ в обслуживании | 4 | s181.vuln.CVE-1999-0024 = TRUE, s181.vuln.CVE-1999-0024 = TRUE & s183.vuln.CVE-2015-7547 = TRUE, s181.Workflow.state = FALSE, s183.vuln.CVE-2017-4925 = TRUE |
| | Получить привилегированный доступ | 7 | h120.vuln.CVE-2008-4250 = TRUE h123.vuln.CVE-2008-4250 = TRUE h124.vuln.CVE-2008-4250 = TRUE h176.vuln.CVE-2008-4250 = TRUE h177.vuln.CVE-2008-4250 = TRUE h178.vuln.CVE-2008-4250 = TRUE h179.vuln.CVE-2008-4250 = TRUE |
| | Получить доступ к информации, содержащейся в системах | 0 | – |
| 4 | Вызвать отказ в обслуживании | 5 | s181.vuln.CVE-1999-0024 = TRUE, s181.vuln.CVE-1999-0024 = TRUE & s183.vuln.CVE-2015-7547 = TRUE, s181.Workflow.state = FALSE, s181.WACLuser-=adm & s183.vuln.CVE-2017-4925 = TRUE, s183.vuln.CVE-2017-4925 = TRUE |

Продолжение таблицы 11

| 1 | 2 | 3 | 4 |
|---|---|---|--|
| | Получить привилегированный доступ | 7 | h120.vuln.CVE-2008-4250 = TRUE h123.vuln.CVE-2008-4250 = TRUE h124.vuln.CVE-2008-4250 = TRUE h176.vuln.CVE-2008-4250 = TRUE h177.vuln.CVE-2008-4250 = TRUE h178.vuln.CVE-2008-4250 = TRUE h179.vuln.CVE-2008-4250 = TRUE |
| | Получить доступ к информации, содержащейся в системах | 0 | – |

1) Имитация выключения ресурса штатными командами операционной системы (sim_shutdown.acl).

2) Имитация выполнения эксплойта, эксплуатирующего уязвимость CVE-2008-4250 (sim_cve-2008-4250.acl). Возвращает реквизиты заранее подготовленных учетных записей администратора APM.

3) Имитация выполнения эксплойта, эксплуатирующего уязвимость CVE-2017-4925 (sim_cve-2017-4925.acl). Возвращает флаг об успешном выполнении эксплойта.

4) Имитация выполнения эксплойта, эксплуатирующего уязвимость CVE-2015-7547 (sim_cve-2015-7547.acl). Возвращает флаг об успешном выполнении эксплойта.

4.4 Результаты экспериментальных исследований

В таблице 12 приведены результаты оценки количества отказов компонентов ИС.

Таблица 12 – Оценка количества отказов компонентов ИС

| Библиотека сценариев | Цель | Всего сценариев | Количество потенциально возможных отказов | Количество отказов при реальном анализе защищенности | Количество отказов при полунатурном анализе защищенности |
|----------------------|-----------------------------------|-----------------|---|--|--|
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | Вызвать отказ в обслуживании | 3 | 3 | 3 | 0 |
| | Получить привилегированный доступ | 16 | 7 | 1 | 0 |
| | Получить доступ к информации, | 6 | 0 | 2 | 2 |

Продолжение таблицы 12

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|----|---|---|---|
| | содержащейся в системах | | | | |
| 2 | Вызвать отказ в обслуживании | 4 | 4 | 4 | 0 |
| | Получить привилегированный доступ | 16 | 7 | 1 | 0 |
| | Получить доступ к информации, содержащейся в системах | 6 | 0 | 2 | 2 |
| 3 | Вызвать отказ в обслуживании | 4 | 4 | 4 | 0 |
| | Получить привилегированный доступ | 26 | 7 | 1 | 0 |
| | Получить доступ к информации, содержащейся в системах | 6 | 0 | 2 | 2 |
| 4 | Вызвать отказ в обслуживании | 5 | 5 | 5 | 0 |
| | Получить привилегированный доступ | 26 | 7 | 1 | 0 |
| | Получить доступ к информации, содержащейся в системах | 6 | 0 | 2 | 2 |

При анализе причин отказов было выявлено, что при цели злоумышленника «получение привилегированного доступа» он эксплуатирует уязвимость CVE-2008-4250. Эксплуатация данной уязвимости может привести к отказу в связи с недостатками существующих эксплойтов. В ряде случаев для АРМ с установленной операционной системой Windows XP SP3 попытка проведения атаки приводит к «синему экрану смерти» (Blue Screen of Death).

При цели злоумышленника «Получить доступ к информации, содержащейся в системах» был получен отказ в обслуживании web-сервера Apache 2.4 (прекращение работы web-сервера). При анализе причин было выявлено, что в описании уязвимости на официальном сайте web-сервера отсутствовало описание данной проблемы.

По результатам проведенных экспериментальных исследований на основе построенной модели, была получена оценка уровня защищенности компонентов ИС.

Таблица 13 — Результаты оценки защищенности информационной системы

| Номер сценария | Описание сценария | Уровень критичности сценария | Принадлежность сценария библиотеке сценариев | | | |
|----------------|--|------------------------------|--|---|---|---|
| | | | 1 | 2 | 3 | 4 |
| 1. | Отказ в обслуживании домена INFSEC | Высокий | + | + | + | + |
| 2. | Отказ в обслуживании домена INFSEC | Высокий | + | + | + | + |
| 3. | Отказ в обслуживании VMWare ESXi | Высокий | + | + | + | + |
| 4. | Отказ в обслуживании VMWare ESXi | Высокий | - | - | + | + |
| 5. | Отказ в обслуживании VMWare ESXi | Высокий | - | + | - | + |
| 6. | Привилегированный доступ к сетевому оборудованию, маршрутизатор wifi121 | Средний | + | + | - | - |
| 7. | Привилегированный доступ к сетевому оборудованию, коммутатор sw144 | Высокий | + | + | + | + |
| 8. | Привилегированный доступ к сетевому оборудованию, коммутатор sw144 | Высокий | + | + | + | + |
| 9. | Привилегированный доступ к сетевому оборудованию, коммутатор r182 | Высокий | + | + | + | + |
| 10. | Привилегированный доступ к контроллеру домена s181 | Высокий | - | - | + | + |
| 11. | Привилегированный доступ к контроллеру домена s181 | Высокий | + | + | + | + |
| 12. | Привилегированный доступ к контроллеру домена s181 | Высокий | + | + | + | + |
| 13-21. | Привилегированный доступ к хостам h120, h123, h124, h125, h176, h177, h178, h179, h180 | Средний | - | - | + | + |
| 22-30. | Привилегированный доступ к хостам h120, h123, h124, h125, h176, h177, h178, h179, h180 | Средний | + | + | + | + |
| 31. | Привилегированный доступ к хостам h179 | Средний | + | + | + | + |
| 32. | Получение доступа к информации на виртуализируемых web-серверах vh184 | Средний | - | - | + | + |
| 33. | Получение доступа к информации на виртуализируемых web-серверах vh184 | Средний | + | + | - | - |
| 34. | Получение доступа к информации на виртуализируемых web-серверах vh186 | Средний | + | + | + | + |
| 35. | Получение доступа к информации на виртуализируемых web-серверах vh184 | Средний | + | + | + | + |
| 36. | Получение доступа к информации на виртуализируемых web-серверах vh186 | Средний | + | + | + | + |
| 37. | Получение доступа к информации на виртуализируемых web-серверах vh184 | Высокий | + | + | + | + |
| 38. | Получение доступа к информации на виртуализируемых web-серверах vh186 | Высокий | + | + | + | + |

В процессе тестирования системы анализа защищенности было показано, что использование полунатурного моделирования при анализе защищенности позволило понизить количество отказов на 66% для первой библиотеки сценариев; 71% для второй и третьей библиотеки сценариев, 75% для четвертой библиотеки сценариев.

В результате анализа полноты выявлено, что для каждой библиотеки сценариев количество выявленных уязвимостей, относительно исходных данных, увеличено на 92%, 86%, 29%, 28% соответственно.

ЗАКЛЮЧЕНИЕ

Главным результатом диссертационного исследования является повышение эффективности анализа защищенности ИС, достигнутого за счет применения новых подходов к моделированию ИС и злоумышленных воздействий. Основные результаты работы состоят в следующем:

1) Проанализирован процесс управления ИБ в части проведения анализа защищенности ИС. Выявлены недостатки существующих методов, реализующих процесс анализа защищенности ИС, и обоснована необходимость разработки нового метода анализа защищенности ИС с использованием полунатурного моделирования.

2) Проанализированы подходы к построению модели ИС и смены ее состояний. Для моделирования ИС было предложено использовать графологический подход, рассматривающий ИС как сложную систему, состоящую из множества допустимых и недопустимых состояний. Последовательность смены состояний, завершающаяся недопустимым состоянием, может являться результатом злоумышленного воздействия и соответствует сценарию злоумышленника. Предложено использовать метод верификации для построения библиотеки сценариев злоумышленника.

3) Разработана модель ИС, позволяющая описать хосты, компоненты, ресурсы и уязвимости ИС в терминах модели Крипке и представить цели злоумышленника в терминах темпоральных логик.

4) Разработано алгоритмическое обеспечение для моделирования состояний ИС в процессе анализа защищенности, включающее алгоритм построения ведущих агентов-злоумышленников, алгоритм построения ведомых агентов-злоумышленников.

5) Разработан метод анализа защищенности ИС в процессе управления ИБ на основе полунатурного моделирования, который, не нарушая корректность функционирования ИС, позволяет повысить количество рассмотренных сценариев

атак и обеспечить поиск уязвимостей, информация о которых не была доступна на момент проведения аудита.

6) На основе предложенных моделей и методов разработана система поддержки принятия решений для решения задач управления информационной безопасностью на этапе анализа защищенности, в которой реализован принцип полунатурности экспериментов и мультиагентный подход для описания сценариев действий злоумышленника и оценки состояний ИС в процессе моделирования возможных воздействий.

7) Разработанная система была апробирована при решении задачи анализа защищенности ИС в одной организации. В результате проведенных испытаний были сокращено количество отказов при анализе защищенности в среднем на 71%, полнота анализа уязвимостей увеличена в среднем на 59%.

8) Получены свидетельства о регистрации программ для ЭВМ № 2018617086 от 18.06.20 г.18 и № 2017616255 от 05.06.2017 г.).

Практическая значимость полученных результатов состоит в том, что они могут быть успешно реализованы в системах аудита защищенности и принятия решений по улучшению процессов управления информационной безопасностью в любых организациях и предприятиях, независимо от сложности эксплуатируемых в них ИС.

Дальнейшие научные исследования в данном направлении связаны с упрощением процесса построения модели ИС и агентов, а именно разработкой типовых компонентов информационной системы с учетом их версий и известных уязвимостей; разработкой типовых агентов под известные способы эксплуатации уязвимостей, в том числе на основе интеграции с инструментами анализа защищенности.

СПИСОК ЛИТЕРАТУРЫ

1. Об информации, информационных технологиях и о защите информации [Текст]: федер. закон РФ от 27.07.2006 №149-ФЗ: в редакции от 18.03.2019. – 26 с.
2. Липатников, В.А. Модель процесса управления информационной безопасностью распределенной информационной системы на основе выявления и оценки уязвимостей [Текст] / В.А. Липатников, А.А. Шевченко // Информационные системы и технологии. – 2018. – №1 (105). – С. 114-123.
3. Оладько, В.С. Модель оценки защищенности автоматизированного рабочего места пользователя [Текст] / В.С. Оладько // Информационные системы и технологии. – 2016. – № 1 (93). – С. 92-99.
4. Баранова, Е.К. Анализ защищенности информационной системы на основе открытых источников информации [Текст] / Е.К. Баранова, Д.В. Смирнов // Проблемы информационной безопасности: сборник материалов V Всероссийской с международным участием научно-практической конференции. – 2019. – С. 79-80.
5. Романов, А.А. Подход к идентификации компьютерных атак на автоматизированные информационные системы в условиях высокой неопределенности [Текст] / А.А. Романов // Прикладная физика и математика. – 2019. – № 1. – С. 25-30.
6. Кульмамиров, С.А. Эффективное раннее обнаружение атак на ресурсы информационных систем [Текст] / С.А. Кульмамиров, Ж.М. Алимжанова, К.Н. Алибекова. // Актуальные научные исследования в современном мире. – 2019. – № 2-1 (46). – С. 9-13.
7. Волкова, В.Н. Классификация методов и моделей в системном анализе [Текст] / В.Н. Волкова, В.Н. Козлов, В.Е. Магер, Л.В. Черненькая // Международная конференция по мягким вычислениям и измерениям. – 2017. – № 1. – С. 223-226.
8. Левина, Т.М. Имитационная модель информационной системы выбора технических и метрологических характеристик предприятия [Текст] / Т.М. Левина, А.А. Макунева // Стратегия развития и инноваций: материалы научно-

практической конференции, посвященной 70-летию ООО «Газпром нефтехим Салават». – 2018. – С. 126-129.

9. Путнин, В.И. Структурная модель системы информационного поиска [Текст] / В.И. Путнин // Перспективы науки. – 2019. – № 2 (113). – С. 10-13.

10. Зеленков, Ю.А. Гибкая корпоративная информационная система: концептуальная модель, принципы проектирования и количественные метрики [Текст] / Ю.А. Зеленков // Бизнес-информатика. – 2018. – № 2 (44). – С. 30-44.

11. Иванов, А.К. Математические модели информационного пространства иерархических систем [Текст] / А.К. Иванов // Автоматизация процессов управления. – 2018. – № 1 (51). – С. 48-57.

12. Мачуева, Д.А. Моделирование процесса информационного взаимодействия в социальных системах [Текст] / Д.А. Мачуева, И.М. Ажмухамедов // Системы управления, связи и безопасности. – 2018 – № 2. – С. 18-39.

13. Ферко, К.П. Оценка результативности автоматического анализа защищенности информационных систем программными средствами [Текст] / К.П. Ферко // Информатика: проблемы, методология, технологии: сборник материалов XVIII международной научно-методической конференции: в 7 томах. – 2018. – С. 225-230.

14. Пестерев, П.В. Структура и алгебраическая модель информационной поисковой системы предприятия на основе мультиагентной системы [Текст] / П.В. Пестерев // Информационные технологии в науке и производстве: материалы V Всероссийской молодежной научно-технической конференции. – 2018. – С. 193-198.

15. Бежитская, Е.А. Многоагентные технологии в задачах управления [Текст] / Е.А. Бежитская, П.И. Казанцева. // Актуальные проблемы авиации и космонавтики. – 2018. – № 4 (14). – С. 289-291.

16. Бабич, М.Ю. Понятие рационального агента и многоагентные системы [Текст] / М.Ю. Бабич // Проблемы информатики в образовании, управлении, экономике и технике: сборник статей XVII Международной научно-технической конференции. – 2017. – С. 11-16.

17. Тищенко, Е.Н. Защищенность, как один из основных показателей потребительского качества распределенных информационных систем [Текст] / Е.Н. Тищенко // Проблемы достижения хозяйственной устойчивости и социальной сбалансированности: императивы экономической политики: глава в книге. – 2017. – С. 95-114.

18. Чипчагов, М.С. Защищенность информации в распределённых информационных системах [Текст] / М.С. Чипчагов, А.С. Вербицкий, В.А. Титов // Вестник института мировых цивилизаций. – 2018. – № 2 (19). – С. 135-138.

19. Махутов, Н.А. Защищенность сложных технических систем: способы обеспечения в условиях наличия широкого спектра неопределенностей [Текст] / Н.А. Махутов, Д.О. Резников // Проблемы безопасности и чрезвычайных ситуаций. – 2017. – № 6. – С. 26-47.

20. Хисаева, Г.Ф. Тестирование информационной системы на защищенность [Текст] / Г.Ф. Хисаева, И.М. Гарипов, В.В. Герасимов // Студенческий. – 2018. – № 16-1 (36). – С. 34-36.

21. Леонова, Н.Л. Имитационное моделирование: конспект лекций [Текст] / Н.Л. Леонова – СПб.: СПбГТУРП, 2015. – 94 с.

22. ГОСТ Р 50922-2006. Защита информации. Основные термины и определения [Текст]. – Взамен ГОСТ Р 50922-96; введ. 2008-02-01. – Москва: Стандартинформ, 2008. – IV, 12 с.; 29 см.

23. Горохов, А. В. Основы системного анализа: учеб. пособие для вузов [Текст] / А.В. Горохов. – Москва: Издательство Юрайт, 2018. – 140 с.

24. Ashby, W.R. Introduction to Cybernetics [Электронный ресурс] / Principia Cybernetica Web. URL: <http://pespmc1.vub.ac.be/ASHBBOOK.html> (дата обращения: 15.09.2018).

25. Вьюненко, Л.Ф. Имитационное моделирование: учебник и практикум для академического бакалавриата [Текст] / Л.Ф. Вьюненко, М.В. Михайлов, Т.Н. Первозванская; под ред. Л.Ф. Вьюненко. — Москва: Издательство Юрайт, 2019. — 283 с.

26. Intelligent Agents: Multi-Agent Systems [Текст] / Alfredo Garro [и др.]. – Elsevier Inc., 2018. – 6 с.

27. Thompson, B. An agent-based modeling framework for cybersecurity in mobile tactical networks [Текст] / B. Thompson, J. Morris-King // Journal of Defense Modeling and Simulation. – 2018. – № 2 (15). – С. 205-218.

28. Dharmalingam, J.M. An agent based intelligent dynamic vulnerability analysis framework for critical SQLIA attacks: Intelligent SQLIA vulnerability analyzer agent [Текст] / J.M. Dharmalingam, M. Eswaran // International Journal of Intelligent Information Technologies. – 2018. – № 3 (14). – С. 56-82.

29. Савельева, Т.С. Этапы осуществления кибератак на ресурсы компании и меры защиты, принимаемые операторами связи для их предотвращения [Текст] / Т.С. Савельева, Ф.Т. Байрушин // Современные научные исследования и разработки. – 2018. – № 6 (23). – С. 589-590.

30. Franklin, M.D. Hierarchical modeling for strategy-based multi-agent multi-team systems [Текст] / M.D. Franklin // Lecture Notes in Networks and Systems. – 2020. – № 70. – С. 312-325.

31. Yannakakis, G.N. Artificial Intelligence and Games [Текст] / G.N. Yannakakis, J. Togelius – Springer, 2018. – 337 с.

32. Donalds, C. Toward a cybercrime classification ontology: A knowledge-based approach [Текст] / C. Donalds, K.-M. Osei-Bryson // Computers in Human Behavior. – 2019. – № 92. – С. 403-418.

33. Satzinger, J.W. Systems Analysis and Design in a Changing World, 6th ed [Текст] / J.W. Satzinger– 2017. – 514 С.

34. Kotenko, I. Experiments with simulation of botnets and defense agent teams [Текст] / I. Kotenko // Proceedings of 27th European Conference on Modelling and Simulation. – 2013. – с. 61-67.

35. Compromised user credentials detection in a digital enterprise using behavioral analytics [Текст] / S. Shah [и др.] // Future Generation Computer Systems. – 2019. – № 93. – С. 407-417.

36. Provoking the adversary by dual detection techniques: An extended stochastic game theoretical framework [Текст] / A. Salem [и др.] // Proceedings of 2018 International Conference on Networking and Network Applications. – 2018. – С. 47-51.

37. Машкина, И.В. Вопросы обеспечения защищенности информации при её обработке в автоматизированных системах управления [Текст] / И.В. Машкина, И.Р. Гарипов // Информационные технологии интеллектуальной поддержки принятия решений: труды VI Всероссийской конференции (с приглашением зарубежных ученых). – 2018. – С. 245-249.

38. Garipov, I.R. The analysis of the problems of supporting information security in industrial control systems [Текст] / I.R. Garipov, I.V. Mashkina. // Computer science and information technologies: proceedings of the 19th International Workshop. – 2017. – С. 28-32.

39. Арьков, П.А. Полумарковская модель реализации угрозы в информационной системе [Текст] / П.А. Арьков // Материалы X Международной научно-практической конференции «Информационная безопасность». – 2008. – № 1. – С. 30-32.

40. Загинайлов, Ю.Н. Теория информационной безопасности и методология защиты информации: учебное пособие [Текст] / Ю.Н. Загинайлов. – Москва, Берлин: Директ-Медиа, 2015. – 253 с.

41. Cao, F. Vulnerability Model and Evaluation of the UEFI Platform Firmware Based on Improved Attack Graphs [Текст] / F. Cao, Q. Li, Z. Chen // Proceedings of the IEEE International Conference on Software Engineering and Service Sciences. – 2018. – С. 225-231.

42. Modeling and clustering attacker activities in IoT through machine learning techniques [Текст] / P. Sun [и др.] // Information Sciences. – 2019. – № 479. – С. 456-471.

43. A model-driven approach for vulnerability evaluation of modern physical protection systems [Текст] / A. Drago [и др.] // Software and Systems Modeling. – 2019. – № 1 (18). – С. 523-556.

44. Щеглов, К.А. Моделирование угроз целевых атак [Текст] / К.А. Щеглов, А.Ю. Щеглов // Информационные технологии. – 2018. – № 5 (24). – С. 339-344.

45. Мияйлович, Ж. Монадические исчисления и проблема разрешимости [Электронный ресурс] / Официальный сайт кафедры Математической теории интеллектуальных систем и лаборатории Проблем теоретической кибернетики механико-математического факультета. URL: [http://www.intsys.msu.ru/magazine/archive/v3\(1-2\)/mijajlovic.pdf](http://www.intsys.msu.ru/magazine/archive/v3(1-2)/mijajlovic.pdf) (дата обращения: 10.06.2018).

46. Потапов, В.И. Численно-аналитическое решение игровой задачи противоборства аппаратно-избыточной динамической системы с противником, действующих в условиях неполной информации о поведении участников игры [Текст] / В.И. Потапов // Омский научный вестник. – 2017. – № 4 (154). – С. 107-110.

47. Щеглов, К.А. Эксплуатационная безопасность. Моделирование реализации угроз атак потенциальным нарушителем [Текст] / К.А. Щеглов, А.Ю. Щеглов // Информационные технологии. – 2017. – № 1 (23). – С. 34-41.

48. MulVAL [Электронный ресурс] / MulVAL Project at Kansas State University. URL: <http://people.cis.ksu.edu/~xou/mulval/index.html> (дата обращения: 18.06.2018).

49. Мартынова, Л.Е. Анализ моделей представления сценариев злоумышленников [Текст] / Л.Е. Мартынова, М.Ю. Умницын // Материалы научной сессии ВолГУ. – 2017. – С. 577-582.

50. Roby, M. Finite State Machine. Reactive System [Электронный ресурс] / Персональная страница М. Roby. URL: <http://www.cs.waikato.ac.nz/~robi/> (дата обращения: 13.06.2018).

51. Handbook of Model Checking [Текст] / Edmund M. Clarke [и др.]. – Springer International Publishing, 2018. – 1210 с.

52. Лифшиц, Ю. Верификация программ и темпоральные логики [Электронный ресурс] / Laboratory of Mathematical Logic at PDMI. URL:

<http://download.yandex.ru/class/lifshits/lecture-note03.pdf> (дата обращения: 20.06.2018).

53. Model Checking@CMU [Электронный ресурс] / The SMV System. Carnegie Mellon University. URL: <http://www.cs.cmu.edu/~modelcheck/smv.html> (дата обращения: 26.05.2018).

54. Bier, V. Risk analysis beyond vulnerability and resilience – characterizing the defensibility of critical systems [Текст] / V. Bier V, A. Gutfraind, // European Journal of Operational Research. – 2019. – № 276. – С. 626-636.

55. New approach to categorical semantics for procedural languages [Текст] / W. Steingartner [и др.] // Computing and Informatics. – 2017. – № 36. – С. 1385-1414.

56. Умницын, М.Ю. Отслеживание состояния информационной системы на основе анализа данных о событиях [Текст] / М.Ю. Умницын, С.В. Михальченко // Прикаспийский журнал: управление и высокие технологии. – 2017. – № 4 (40). – С. 165–173.

57. Approaches to information systems modeling for the study of the reliability and safety of their functioning [Текст] / Т. Omelchenko [и др.] // Proceedings of the 6th International Conference on System Modeling and Advancement in Research Trends, SMART. – 2017. – С. 128-133.

58. Omelchenko, T. Model of enterprise's information security management [Текст] / Т. Omelchenko, М. Umnitsyn М., А. Nikishova, N. Sadovnikova // Information Technologies in Science, Management, Social Sphere and Medicine' (ITSMSSM 2016): proceedings of III International Scientific Conference, Tomsk. – 2017. – 6 с..

59. Naveiro, R. Large-scale automated forecasting for network safety and security monitoring [Текст] / R. Naveiro, S, Rodríguez, Insua Ríos // Applied Stochastic Models in Business and Industry. – 2019. – 10 с.

60. Muñoz, F.R. Analyzing the traffic of penetration testing tools with an IDS [Текст] / F.R. Muñoz, E.A. Armas Vega, L.J.G. Villalba // Journal of Supercomputing. – 2018. – № 12 (74). – С. 6454-6469.

61. Гирчева, Е.А. Обнаружение и анализ уязвимостей [Текст] / Е.А. Гирчева Е.А., М.Ю. Умницын // Актуальные вопросы информационной безопасности

регионов в условиях глобализации информационного пространства: материалы VI Всерос. науч.-практ. конф., Волгоград. – 2017. – С. 147-151.

62. CVSS [Электронный ресурс] / Forum of Incident Response and Security Teams. URL: <http://www.first.org/cvss> (дата обращения: 10.09.2018).

63. CVE: Common Vulnerabilities and Exposures [Электронный ресурс] / MITRE. URL: cve.mitre.org (дата обращения: 10.09.2018).

64. CWE: Common Weakness Enumeration [Электронный ресурс] / MITRE. URL: <http://cwe.mitre.org/data> (дата обращения: 10.09.2018).

65. NVD [Электронный ресурс] / National Vulnerability Database. URL: <http://nvd.nist.gov> (дата обращения: 10.09.2018).

66. Microsoft Security Bulletin [Электронный ресурс] / Microsoft. URL: <https://docs.microsoft.com/en-us/security-updates/securitybulletins/securitybulletins> (дата обращения 10.09.2018).

67. Умницын, М.Ю. Формализация построения сценариев злоумышленных воздействий на информационную систему [Текст] / М.Ю. Умницын, Н.П. Садовникова // Известия Волгоградского государственного технического университета. – 2016. – № 11 (190). – С. 72-75.

68. McMillan, K.L. The SMV version 2.5. User Manual [Электронный ресурс] / The SMV System. URL: <http://cs.nyu.edu/courses/spring06/G22.3033-005/smvmanual.pdf> (дата обращения: 10.06.2018).

69. NuSMV 2.6 User Manual [Электронный ресурс] / Roberto Cavada [и др.] // ITC-irst Trento University. URL: <http://nusmv.irst.itc.it/NuSMV/userman/v24/nusmv.pdf> (дата обращения: 06.06.2018).

70. NuSMV: a new symbolic model checker [Электронный ресурс] / NuSMV Home Page. URL: <http://nusmv.fbk.eu/> (дата обращения: 04.06.2018).

71. Умницын, М.Ю. Система анализа рисков при топологическом описании уязвимостей информационной системы [Текст] / М.Ю. Умницын // Проблемы обеспечения информационной безопасности в регионе: материалы II Регион. науч.-практ. конф., Волгоград. – 2009. – С. 39-43.

72. Hermanowski, D. Proactive Risk Assessment Based on Attack Graphs: An Element of the Risk Management Process on System, Enterprise and National Level [Текст] / D. Hermanowski, R. Piotrowski // Proceedings of the 20th International Conference on High Performance Computing and Communications, 16th International Conference on Smart City and 4th International Conference on Data Science and Systems. – 2018. – С. 1435-1441.

73. Extensible Markup Language (XML) [Электронный ресурс] / W3C. URL: <https://www.w3.org/TR/xml/> (дата обращения: 17.06.2018).

74. Хантер, Д. XML: Базовый курс [Текст] / Д. Хантер, Д. Рафтер, Д. Фаусетт. – М.: Издательский дом «Вильямс», 2009. – 1344 с.

75. Свидетельство о государственной регистрации программы для ЭВМ № 2017616255. Программный комплекс «Встроенный предметно-ориентированный язык для генерации программного кода» [Текст] / Г.А. Попов, Т.А. Тихомирова, М.Ю. Умницын, К.П. Кривошеев. – дата государственной регистрации в Реестре программ для ЭВМ 05.06.2017.

76. Model Checking@CMU [Электронный ресурс] / The SMV System. Carnegie Mellon University. URL: <http://www.cs.cmu.edu/~modelcheck/smv.html> (дата обращения: 10.06.2018).

77. Cadence SMV [Электронный ресурс] / Cadence Berkeley Labs. URL: <http://www.kenmcml.com/smv.html> (дата обращения: 10.06.2018).

78. FSAP/NUSMV-SA [Электронный ресурс] / FSAP/NUSMV-SA. URL: <https://es-static.fbk.eu/tools/FSAP> (дата обращения: 10.06.2018).

79. SPIN [Электронный ресурс] / Bell Labs. URL: <http://spinroot.com> (дата обращения: 10.06.2018).

80. McMillan K.L. The SMV version 2.5. User Manual [Электронный ресурс] / The SMV System. URL: <http://cs.nyu.edu/courses/spring06/G22.3033-005/smvmanual.pdf> (дата обращения: 10.06.2018).

81. Дровникова, И.Г. Основные виды уязвимостей и взаимосвязь компонентов безопасности при обосновании показателей надёжности системы защиты информации от несанкционированного доступа в автоматизированных

системах [Текст] / И.Г. Дровникова, А.С. Етепнев, Е.А. Рогозин // Приборы и системы. Управление, контроль, диагностика. – 2019. – № 3. – С. 59-64.

82. Шилкина, А.Т. Тенденции развития риск ориентированного подхода в контексте индустрии 4.0 [Текст] / А.Т. Шилкина, О.Е. Варакина // Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Экономические науки. – 2019. – № 1 (12). – С. 9-20.

83. Ziemke, D. Bicycle traffic and its interaction with motorized traffic in an agent-based transport simulation framework [Текст] / D. Ziemke, S. Metzler, K. Nagel // Future Generation Computer Systems. – 2019. – № 97. – С. 30-40.

84. Умницын, М.Ю. Применение многоагентных систем для задачи мониторинга информационной системы [Текст] / М.Ю. Умницын // Актуальные вопросы информационной безопасности регионов в условиях глобализации информационного пространства: материалы Всерос. науч.-практ. конф., Волгоград. – 2012. – С. 143-148.

85. Dimensions in programming multi-agent systems [Текст] / Boissier O. [и др.] // Knowledge Engineering Review. – 2019. – № 34. – 10 с.

86. Multi-agent modeling and simulation of graph-based predator-prey dynamic systems: A BDI approach [Текст] / A. Bădică [и др.] // Expert Systems. – 2018. – № 35. – 9 с.

87. Soar Home [Электронный ресурс] / Soar Cognitive Architecture. URL: <https://soar.eecs.umich.edu/> (дата обращения: 10.06.2018).

88. Ferguson, I.A. TouringMachines: Autonomous Agents with Attitudes [Электронный ресурс] / Computer and Information Science Department's at the University of Massachusetts Dartmouth. – 19 с. URL: http://www.cis.umassd.edu/~hxu/courses/cis563/s05/ferg92_2.pdf (дата обращения: 10.06.2018).

89. Аксенов, К.А. Применение гибридной мультиагентной архитектуры в системе поддержки принятия решений снабжения сети автозаправочных станций

[Текст] / К.А. Аксенов, А.Л. Неволлина // Современные наукоемкие технологии. – 2016. – № 12-1. – С. 14-18.

90. Helsinger, A. Cougaar: A Scalable, Distributed Multi-Agent Architecture [Электронный ресурс] / A. Helsinger, M. Thome, T. Wright // Cougaar website. URL: <http://cougaar.org> (дата обращения: 10.06.2018).

91. Urban, C. PECS: A Reference Model for the Simulation of Multi-Agent Systems [Текст] / C. Urban // Tools and Techniques for Social Science Simulation. – 2000. – С. 83-114.

92. Cohen, P.R. Rational interaction as the basis for communication in Intentions in Communication [Текст] / P.R. Cohen, H.J. Levesque. – Cambridge: The MIT Press, 1990. – С. 221-256.

93. Vieira, R. On the formal semantics of speech-act based communication in an agent-oriented programming language [Текст] / R. Vieira, A. Moreira, M. Wooldridge // Journal of Artificial Intelligence Research. – 2007. – №29. – С. 221-267.

94. Ancona, D. Coo-AgentSpeak: cooperation in AgentSpeak through plan exchange [Текст] / D. Ancona, V. Mascardi, J.F. Hübner // Proceedings of the Third International Joint Conference on autonomous agents and multi-agent systems. – 2004. – С. 19-23.

95. Schurr, N., Coordination advice: a preliminary investigation of human advice to multi-agent teams [Текст] / N. Schurr, P. Scerri, M. Tambe // Spring symposium on interaction between humans and autonomous systems over extended operation. – 2004. – С. 220-253.

96. Home [Электронный ресурс] / I.R.S. Intelligent Reasoning Systems. URL: <http://www.marcush.net> (дата обращения: 16.10.2018).

97. Home [Электронный ресурс] / Spark. URL: <http://www.ai.sri.com/~spark> (дата обращения: 16.10.2018).

98. 3APL Homepage [Электронный ресурс] / An Abstract Agent Programming. URL: <http://www.cs.uu.nl/3apl> (дата обращения: 16.10.2018).

99. Pnueli, A. Specification and development of reactive systems [Текст] / A. Pnueli // In information processing. – 2006. – С. 845–858.

100. Schurr, N. Evolution of a teamwork model [Текст] / N. Schurr, S. Okamoto, R.T. Maheswaran // Materials of Cognitive Modeling and Multi-Agent Interactions Conference. – 2005. – С. 256-270.
101. Bordini, R.H. Programming multi-agent systems in AgentSpeak using Jason [Текст] / R.H. Bordini, F.J. Hubner, M. Wooldridge. – Wiley, 2007. – 294 с.
102. Wooldridge, M. An Introduction to multi-agent systems [Текст] / M Wooldridge. – John Wiley & Sons Ltd, 2002. – 366 с.
103. Люгер, Д.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем [Текст] / Д.Ф. Люгер – М.: Вильямс, 2005. – 864 с.
104. Bratko, I. Prolog programming for artificial intelligence [Текст] / I. Bratko. – Addison-Wesley, 1990. – 120 с.
105. Аграновский, В.А. Анализ поведения злоумышленника при организации вторжения в компьютерную сеть, оснащенную системой обнаружения вторжений [Текст] / В.А. Аграновский, Д.Н. Сергеев // Материалы X Международной научно-практической конференции «Информационная безопасность». – 2008. – ч. 1. – С. 300-301.
106. Rao, A.S. AgentSpeak(L): BDI agents speak out in a logical computable language [Текст] / A.S. Rao // In Proceedings of the seventh workshop on modeling autonomous agents in a multi-agent world. – 1996. – С. 42-55.
107. Shoham, Y. Agent-oriented programming [Текст] / Y. Shoham // Artificial Intelligence. – 1993. – № 60. – С. 51-92.
108. Overview [Электронный ресурс] / Jadex software projects. URL: <http://jadex.informatik.uni-hamburg.de> (дата обращения: 15.10.2018).
109. Description [Электронный ресурс] / Jason home. URL: <http://jason.sf.net> (дата обращения: 15.10.2018).
110. Sun, R. A concise introduction to multi-agent systems and distributed artificial intelligence [Текст] / R. Sun. – Morgan & Claypooy Publishers, 2007. – 71 с.
111. Shoham, Y. Multiagent systems: algorithmic, game-theoretic, and logical foundations [Текст] / Y. Shoham. – Cambridge University Press, 2008. – 504 с.

112. Innes, A.F. *TouringMachines: autonomous agents with attitudes* [Текст] / A.F. Innes. – Computer Laboratory, University of Cambridge, 1992. – 250 с.
113. Muller, J.P. *The design intelligence agents: a layered approach* [Текст] / J.P. Muller // *Lectures notes in computer science*. – 1996. – №117 – 30 с.
114. Касьянов, В.Н. *Лекции по теории формальных языков, автоматов и сложности вычислений* [Текст] / В.Н. Касьянов. – Новосибирск: НГУ, 1995. – 112 с.
115. ISO/IEC 14977-1996. *Extended BNF: a standard syntactic metalanguage based on BNF*. – 1996.
116. Jack [Электронный ресурс] // AOS group URL: <http://aosgrp.com/products/jack/> (дата обращения: 14.04.2018).
117. Pokahr, A, *The Jadex Project: Programming Model* [Текст] / A. Pokahr, L. Braubach, K. Jander // *Multiagent Systems and Applications*. – 2009. – С. 21-53.
118. *Tuning Java to Run Interactive Multiagent Simulations over Jason* [Текст] / V. Fernández-Bauset [и др.] // *Advances in Artificial Intelligence*. – 2010. – № 6464 – С. 354-363.
119. Winikoff, M. *Jack Intelligent Agents: An Industrial Strength Platform* [Текст] / M. Winikoff // *Multi-Agent Programming*. – 2005. – С. 175-193.
120. Vidal, J.M. *Fundamentals of Multiagent Systems* [Текст] / J.M. Vidal. – 2010. – 155 с.
121. Sycara, K.P. *Multiagent Systems* [Текст] / K.P. Sycara // *AI magazine*. – 1998. – № 2 (19) – 14 с.
122. Nigam, V., Leite, J. *Adding Knowledge Updates to 3APL* [Текст] / V. Nigam, J. Leite // *International Workshop on Programming Multi-Agent Systems*. – 2006. – с. 165-181.
123. Albrecht, S., *Multiagent Learning. Foundations and Recent Trends* [Текст] / S. Albrecht, P. Stone // *IJCAI 2017 conference*. – 2017. – 186 с.
124. Bellifemine, F. *Developing Multi-agent Systems with JADE* [Текст] / F. Bellifemine, A. Poggi, G. Rimassa // *International Workshop on Agent Theories, Architectures, and Languages*. – 2010. – С. 89-103.

125. Cobot in LambdaMOO: An Adaptive Social Statistics Agent [Текст] / Charles Lee Isbell Jr. [и др.] // Autonomous Agents and Multi-Agent Systems. – 2006. – № 3 (13) – С. 327–354.

126. Хопкрофт, Дж., Введение в теорию автоматов, языков и вычислений [Текст] / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман – М.: Издательский дом «Вильямс», 2002. – 528 с.

127. Брукшир, Д.Г. Введение в компьютерные науки [Текст]: пер. с англ. / Д.Г. Брукшир. – М.: Издательский дом «Вильямс», 2001. – 688 с.

128. Ответственность за нарушение авторских и смежных прав при использовании нелицензионного программного обеспечения [Электронный ресурс] // Юридическая помощь URL: <http://www.vist.spb.ru/soft/ur-help/146-soft-otvetstvennost.html> (дата обращения: 10.01.2019).

129. Юридические аспекты приобретения права использования программного обеспечения на основании неисключительной лицензии (неисключительного права) [Электронный ресурс] // Юридическая помощь. URL: <http://www.vist.spb.ru/soft/ur-help/145-soft-ur-aspekt.html> (дата обращения: 10.01.2019).

130. Таненбаум, Э. Современные операционные системы [Текст] : пер. с англ. / Э. Таненбаум. – СПб.: «Питер», 2007. – 1040 с.

131. Таненбаум, Э. Архитектура компьютера [Текст] : пер. с англ. / Э. Таненбаум. – СПб.: «Питер», 2007. – 848 с.

132. Магьюир, С. Создание надёжного кода [Текст] : пер. с англ. / С. Магьюир. – Microsoft Press, 2004. – 220 с.

133. Мак-Коннел, С. Совершенный код [Текст] / С. Мак-Коннел. – Microsoft Press, 1993. – 320 с.

134. Шмуллер, Дж. Освой самостоятельно UML 2 за 24 часа. Практическое руководство. [Текст] / Дж. Шмуллер – М.: «Вильямс», 2005. – 416 с.

135. Буч, Г. UML. [Текст] / Г. Буч, А. Якобсон, Дж. Рамбо. – СПб.: «Питер», 2006. – 736 с.

136. Цыбулин, А.М. Исследование противоборства службы безопасности и злоумышленников на многоагентной модели [Текст] / А.М. Цыбулин, А.В. Никишова, М.Ю. Умницын // Известия ЮФУ. Технические науки. – 2008. – № 8 (85) – С. 94-99.

137. Свидетельство о государственной регистрации программы для ЭВМ № 2018617086. Программный комплекс для полунатурного анализа защищенности информационной системы [Текст] / Умницын М.Ю. – дата государственной регистрации в Реестре программ для ЭВМ 18.06.2018.

138. Simulation of Malicious Scenarios using Multi-Agent Systems [Текст] // M. Umnitsyn [и др.] / Proceedings of the 7th International Conference on System Modeling and Advancement in Research Trends, SMART. – 2018. – С. 3-9.

139. Adall, S. The DARPA advanced logistics project [Текст] / S. Adall, L. Pigaty // Annals of Mathematics and Artificial Intelligence. – 2003. – №37 – 108 с.

140. Urban, C. PECS — A reference model for the simulation of multi-agent systems [Текст] / C. Urban // Tools and Techniques for Social Science Simulation. – 2000. – С. 83-114.

141. Bratman, M.E. Intention, plans and practical reasoning [Текст] / M.E. Bratman. – CSLI Publications, 1999. – 310 с.

142. Якубайтис, Э.А. Теория автоматов [Текст] / Э.А. Якубайтис, В.О. Васюкевич, А.Ю. Гобземис // Теория вероятностей. Математическая статистика. Теоретическая кибернетика. – 1976. – №. 13 – С. 109-188.

143. Руководящий документ. Концепция защиты средств вычислительной техники и автоматизированных систем от несанкционированного доступа к информации [Текст]: утв. Гостехкомиссией России: введ в действие с 30.03.1992. – 7 с.

144. API Specification [Электронный ресурс] // Java 2 Platform, Standard Edition, v 1.4.2. Sun Microsystems URL: <http://java.sun.com/j2se/1.4.2/docs/api> (дата обращения: 06.03.2018).

145. Умницын, М.Ю. Применение многоагентного подхода для моделирования злоумышленных воздействий на информационную систему [Текст]

/ М.Ю. Умницын, А.А. Евдокименко // Проблемы обеспечения информационной безопасности в регионе: материалы IV Регион. науч.-практ. конф., Волгоград. – 2011. – С. 83-90.

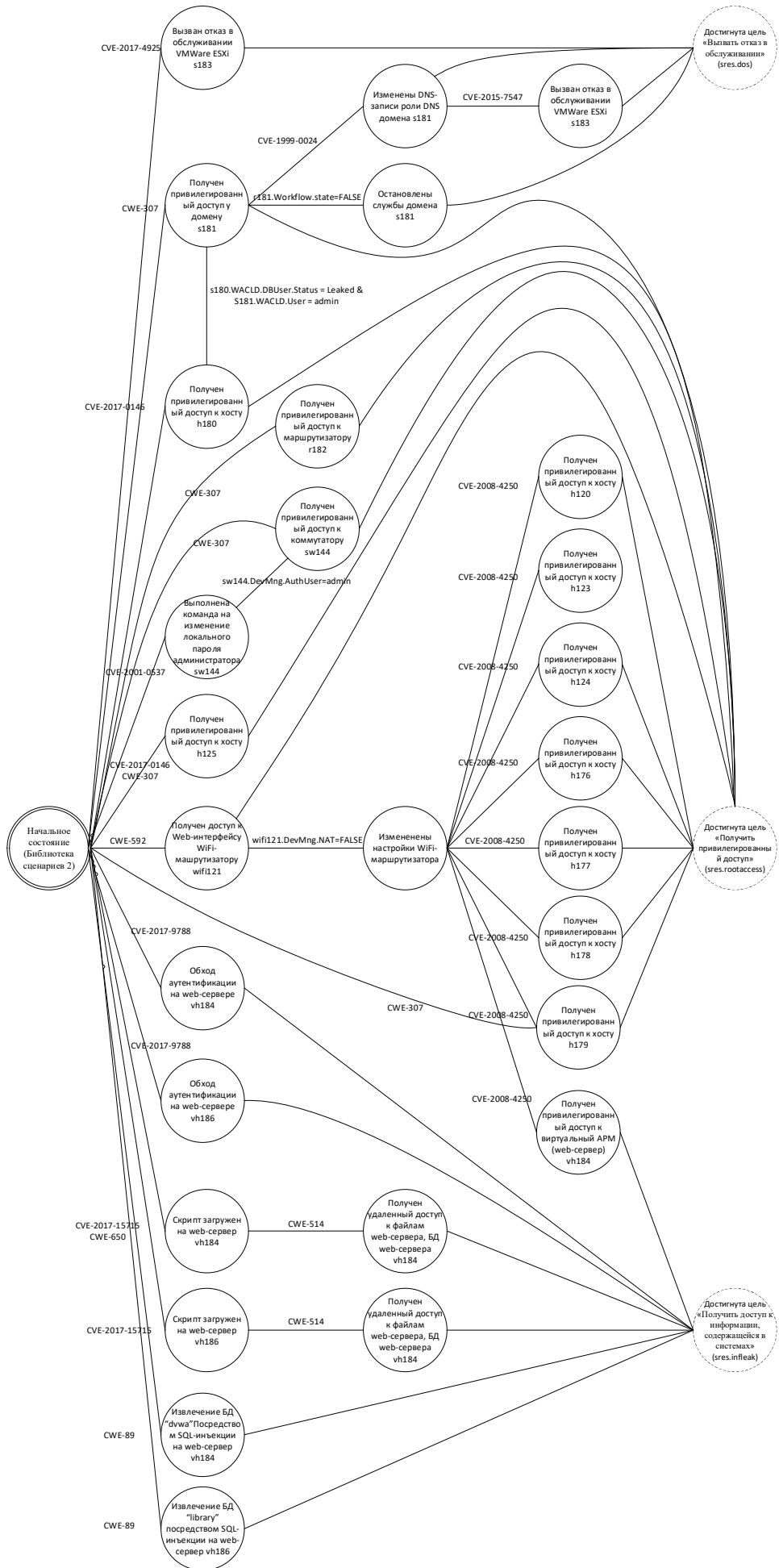
146. Умницын, М.Ю. Подход к полунатурному анализу защищенности информационной системы [Текст] / М.Ю. Умницын // Известия Волгоградского государственного технического университета. – 2018. – № 8 (218) – С. 112-116.

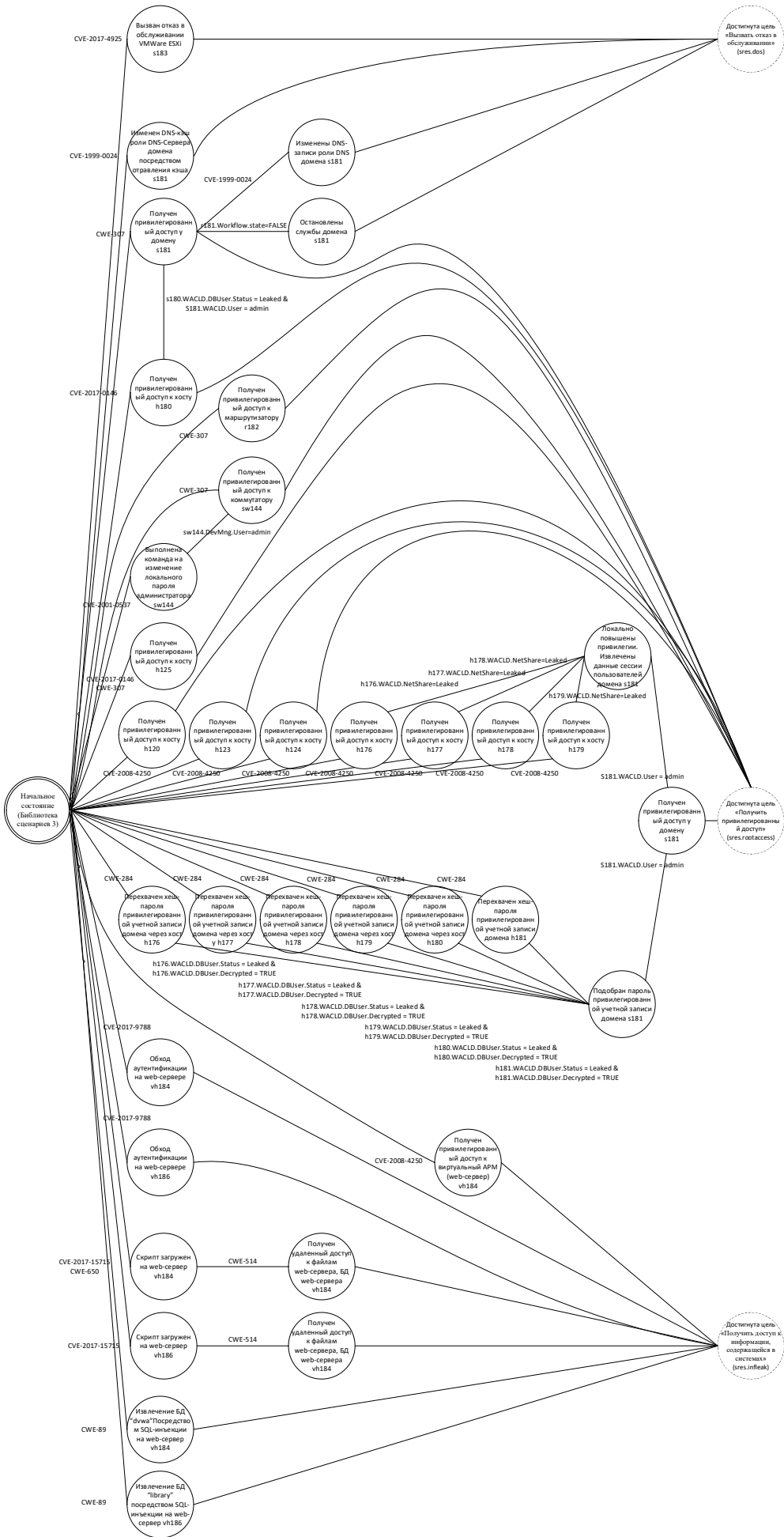
147. Умницын, М.Ю. Применение многоагентного подхода для полунатурного моделирования злоумышленных воздействий [Текст] / М.Ю. Умницын // Вестник Волгоградского государственного университета. Серия 10: Инновационная деятельность. – 2011. – № 5 – С. 38-41.

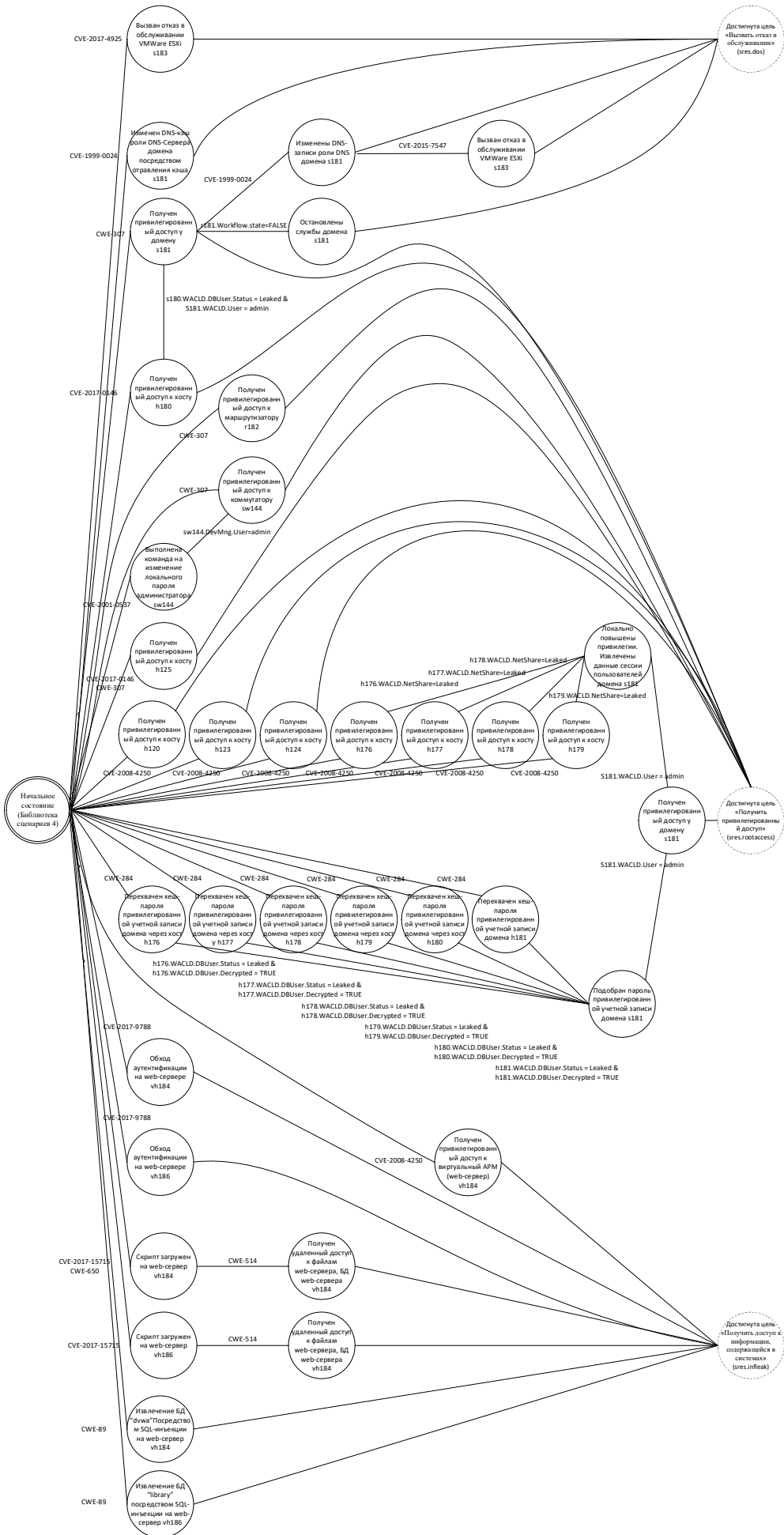
148. Умницын, М.Ю. Оценка защищенности информационной системы с помощью метода полунатурного моделирования злоумышленных воздействий [Текст] / М.Ю. Умницын, А.А. Евдокименко // Проблемы модернизации региона в исследованиях молодых ученых: материалы VI межрегиональной научно-практической конференции, Волгоград. – 2010. – С. 371-372.

149. Bridging Basics [Электронный ресурс] // Internetworking Technology Handbook. Cisco Systems URL: <http://www.cisco.com/en/US/docs/internetworking/technology/handbook/Bridging-Basics.html> (дата обращения: 03.02.2018).

150. Рожкова, Е.О. Обзор и сравнение сканеров уязвимостей [Текст] / Е.О. Рожкова, И.В. Ильин // Научное сообщество студентов XXI столетия. ТЕХНИЧЕСКИЕ НАУКИ: сб. ст. по мат. XXX междунар. студ. науч.-практ. конф. – 2015. – № 3(29) – С. 77-87.







Приложение Б

Свидетельства о государственной регистрации программы для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО
о государственной регистрации программы для ЭВМ
№ 2018617086

**Программный комплекс для полунатурного анализа
защищенности информационной системы**

Правообладатель: *Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет» (RU)*

Автор: *Умницын Михаил Юрьевич (RU)*

Заявка № **2018614181**
Дата поступления **26 апреля 2018 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ **18 июня 2018 г.**

Руководитель Федеральной службы
по интеллектуальной собственности



Г.П. Ивлиев



РОССИЙСКАЯ ФЕДЕРАЦИЯ



СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2017616255

Программный комплекс "Встроенный
предметно-ориентированный язык для генерации
программного кода"

Правообладатель: *Федеральное государственное автономное
образовательное учреждение высшего образования
«Волгоградский государственный университет» (RU)*

Авторы: *Попов Глеб Александрович (RU), Тихомирова Татьяна
Александровна (RU), Умницын Михаил Юрьевич (RU), Кривошеев
Михаил Павлович (RU)*

Заявка № 2017613197

Дата поступления 10 апреля 2017 г.

Дата государственной регистрации
в Реестре программ для ЭВМ 05 июня 2017 г.

Руководитель Федеральной службы
по интеллектуальной собственности

Г.П. Ивлиев Г.П. Ивлиев



Приложение В

Акт о внедрении результатов диссертационного исследования

УТВЕРЖДАЮ
Заместитель генерального
директора по производству
ООО «Инжиниринговый центр
РЕГИОНАЛЬНЫЕ СИСТЕМЫ»
Кирпо М.А.



2019 г.

АКТ

о внедрении результатов
кандидатской диссертационной работы Умницына М.Ю., на тему
«Метод анализа защищенности информационной системы с использованием
полунатурного моделирования»

Комиссия в составе:

Председатель комиссии:

1. Заместитель директора департамента системной интеграции – А.С.
Рябухин;

Члены комиссии:

2. Начальник отдела комплексных систем защиты
информации – Д.О. Руденко;

3. Руководитель группы проектирования систем информационной
безопасности – А.А. Бешта

составила настоящий акт о том, что результаты диссертационной работы в
виде:

– метод анализа защищенности информационной системы с
использованием полунатурного моделирования;

– метод построения модели ИС для исследования ее защищенности
внедрены в Обществе с ограниченной ответственностью «Инжиниринговый
центр РЕГИОНАЛЬНЫЕ СИСТЕМЫ».

Применение результатов диссертационной работы позволило повысить
эффективность при анализе защищенности и аудите ИС за счет:

– комбинирования уязвимостей при анализе защищенности
(уточнения списка актуальных угроз);

– упрощения процесса описания компонентов (применения унифицированной структуры);

– повышения полноты анализа уязвимостей.

Количество отказов при анализе защищенности было сокращено на 22%.

Разработчик

 М.Ю. Умницын

Заместитель директора
департамента системной интеграции

 А.С. Рябухин

Начальник отдела
комплексных систем защиты информации

 Д.О. Руденко

Руководитель группы проектирования
систем информационной безопасности

 А.А. Бешта