Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Челябинский государственный университет»

На правах рукописи

КОСЕНКО Максим Юрьевич

МНОГОАГЕНТНАЯ СИСТЕМА ОБНАРУЖЕНИЯ И БЛОКИРОВАНИЯ БОТНЕТОВ ПУТЕМ ВЫЯВЛЕНИЯ УПРАВЛЯЮЩЕГО ТРАФИКА НА ОСНОВЕ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ДАННЫХ

Специальность 05.13.19
Методы и системы защиты информации, информационная безопасность

Диссертация на соискание ученой степени кандидата технических наук

Научный руководитель
доктор технических наук,
профессор Мельников А. В.

Челябинск – 2017

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы исследования

В течение последних нескольких десятилетий во всем мире можно наблюдать огромный рост Интернета и приложений на его основе. Использование Интернета становится неотъемлемой частью нашей жизни. Безусловно, он предоставляет высокий уровень удобства, но растущая зависимость от Интернета влечет ряд крупных проблем в области информационной безопасности. Таким образом, Интернет-безопасность становится все более и более актуальным аспектом для тех, кто использует Интернет для работы, бизнеса, образования или развлечения.

Большинство атак и мошеннических действий в Интернете осуществляется с помощью вредоносного программного обеспечения, которое включает в себя вирусы, трояны, черви, шпионские программы, ботнеты. Вредоносное программное обеспечение стало основным источником большинства зловредной активности в Интернете: целевые атаки [49], распределенные атаки типа «отказ в обслуживании» [11, 103], мошеннические действия [60, 111, 128], а также сканирование [92, 122]. Среди всех видов вредоносного программного обеспечения ботнеты являются основной платформой [128], которую злоумышленники используют как масштабный, согласованно действующий инструмент, используемый для поддержки постоянного роста преступной деятельности, такой, как DDoS, рассылка спама, фишинг и кража информации.

Данные крупнейших мировых компаний, специализирующихся в области защиты информации, таких как Prolexic/Akamai, Incapsula, показывают, что ботнеты являются основной угрозой безопасности в Интернете. В соответствии с рисунком 1, постоянно функционируют более 1200 ботнетов [149]. Традиционно обнаружение ботнетов осуществляется с помощью пассивного мониторинга и анализа сетевого трафика. Для обнаружения ботнетов выделяют подходы на основе поиска сигнатур либо аномалий в трафике. Менее популярны подходы,

использующие анализ DNS трафика или применение узлов-ловушек. Основным недостатком существующих решений обнаружения ботнетов является то, что они не учитывают взаимосвязь многоагентной природы ботнетов и этапов их жизненного цикла. В результате обнаружение получается частичным, и блокировать деятельность ботнета не представляется возможным. В связи с этим, решаемая в диссертационной работе задача, заключающаяся в разработке многоагентной системы обнаружения и блокирования ботнетов путем выявления управляющего трафика на основе методов интеллектуального анализа сетевого трафика, является актуальной. Актуальность темы подтверждают действия Правительства Российской Федерации, возложившие на Федеральную службу безопасности Российской Федерации полномочия по созданию государственной системы обнаружения, предупреждения и ликвидации последствий компьютерных атак на информационные ресурсы Российской Федерации – информационные системы и информационно-телекоммуникационные сети, находящиеся на территории Российской Федерации и в дипломатических представительствах и консульских учреждениях Российской Федерации за рубежом [35].

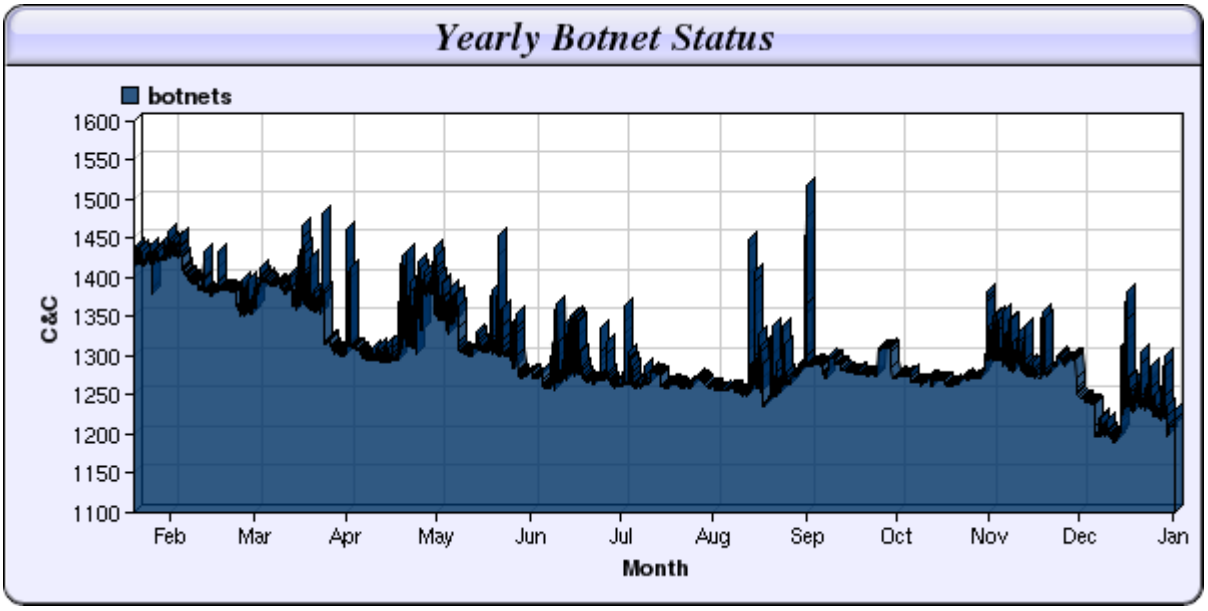

Рисунок 1 – Количество ботнетов за 2015 год

От решения вопроса качественного обнаружения и блокирования ботнетов зависит безопасность Интернета, инженерных и бизнес-систем, и поэтому данный

вопрос широко освещен в работах отечественных и зарубежных авторов, таких как А.Е. Архипов, В.И. Васильев, Дж. Бинкли, Т. Йен, В.А. Камаев, А. Карасаридис, А.В. Козачок, А. Г. Корченко, И.В. Котенко, Н.Н. Куссуль, А.В. Лукацкий, К. Ливадас, Дж. Митчелл, А.Н. Назаров, А. Рамачандран, М. Рейтер, В.А. Сердюк, Д.С. Сильнов, С. Сингх, Е. Стинсон, В. П. Фраленко, И.А. Ходашинский и др.

В существующих исследованиях предложено множество подходов к обнаружению ботнетов, но эти решения имеют различные ограничения:

– отсутствие механизма автоматической генерации сигнатур ботов;

– обнаружение ботнетов с конкретной организационной структурой (централизованной или децентрализованной);

– обнаружение ботнетов, работающих по специфичному протоколу (IRC или HTTP и др.);

– обнаружение ботнетов с определенной вредоносной активностью (сканирование или рассылка спама);

– множество ложных срабатываний.

Таким образом, задача разработки алгоритмического и программного обеспечения системы обнаружения и блокирования ботнетов с применением методов интеллектуального анализа данных является актуальной.

Объектом исследования в данной работе являются системы защиты от ботнетов в открытых компьютерных сетях, включая Интернет.

Предметом исследования являются методы и алгоритмы обнаружения ботнетов с использованием интеллектуального анализа данных в рамках многоагентного подхода.

Целью диссертационной работы является повышение защищенности информационных систем от атак, использующих ботнеты, на основе разработки и применения многоагентной системы обнаружения и блокирования ботнетов с использованием алгоритмов интеллектуального анализа данных.

Для достижения цели в работе были сформулированы и решены следующие **задачи**:

1. Исследовать распространенные атаки типа «отказ в обслуживании», процесс их реализации и механизмы защиты от них, проанализировать существующие подходы к выявлению ботнетов.

2. Разработать алгоритм обнаружения управляющего трафика ботнета в глобальных сетях с использованием технологий интеллектуального анализа данных.

3. Предложить архитектуру многоагентной системы обнаружения и блокирования ботнетов, проанализировать эффективность функционирования предложенных в диссертационном исследовании алгоритмов.

4. Разработать метод распределенного обнаружения управляющих компонент ботнета, позволяющий обнаруживать управляющие серверы и узлы сети, с которых осуществляется контроль атаки, основанный на сигнатуре управляющего трафика.

5. Разработать исследовательский прототип многоагентной системы обнаружения и блокирования ботнетов, дать рекомендации по практическому внедрению многоагентной системы обнаружения и блокирования ботнетов.

Методы исследования. При решении поставленных в работе задач использовались теоретико-множественные, агентно-ориентированные методы представления моделей, объектно-ориентированные методологии проектирования и разработки программных систем, а также методы интеллектуального анализа данных. Для оценки эффективности предлагаемых решений использовались методы функционального и информационного моделирования.

Положения, выносимые на защиту:

1. Результаты анализа состояния проблемы обнаружения ботнетов в открытых компьютерных сетях (включая Интернет), существующих методов защиты от распределенных атак типа «отказ в обслуживании» и методов обнаружения ботнетов.

2. Алгоритм обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition на основе интеллектуального анализа данных.

3. Архитектура интеллектуальной многоагентной системы обнаружения и блокирования ботнетов.

4. Метод распределенного обнаружения управляющих компонент ботнета, позволяющий обнаруживать управляющие серверы и узлы сети, с которых осуществляется контроль атаки, основанный на сигнатуре управляющего трафика.

5. Исследовательский прототип многоагентной системы обнаружения и блокирования ботнетов.

Научная новизна исследования заключается в следующем:

– предложен алгоритм обнаружения управляющего трафика ботнетов Botnet MultiAgent Recognition (далее – BNMAR), основанный на интеллектуальном анализе данных с возможностью автоматического формирования сигнатуры управляющего трафика, что, в отличие от существующих методов, позволяет решать задачу обнаружения ботнетов в автоматическом режиме независимо от протокола их управления. При этом алгоритм позволяет обнаруживать ботнеты централизованной и децентрализованной организационных структур, а также не зависит от типа вредоносной деятельности ботов;

– предложена архитектура многоагентной системы обнаружения ботнетов, соответствующая типовой архитектуре ботнета, что позволяет блокировать атаки на стороне её источника, тем самым разгрузив каналы передачи от вредоносного трафика, а также на основе разработанного алгоритма обнаружения управляющего трафика позволяет обнаруживать и блокировать пассивных участников ботнета;

– предложен метод распределенного обнаружения управляющих компонент ботнета, основанный на сигнатуре управляющего трафика, который, в отличие от существующих алгоритмов, за счет использования многоагентного

подхода позволяет обнаруживать управляющие серверы и узлы сети, с которых осуществляется контроль атаки.

Практическая значимость полученных результатов заключается в применении многоагентной системы для обнаружения и блокирования ботнетов путем обнаружения управляющего трафика ботнета на основе интеллектуального анализа данных, что обеспечивает повышение показателя F-меры обнаружения ботнета по сравнению с рядом известных систем от 9% до 26%, при этом доля ложных срабатываний не превышает 0,02. Разработанный прототип системы позволяет:

– повысить эффективность предотвращения распределенных атак, совершаемых ботнетами;

– обеспечить централизованный мониторинг кибер-угроз в сети Интернет;

– обеспечить процесс проведения кибер-расследований благодаря возможности обработки накопленных данных.

Достоверность и апробация результатов. Полученные в диссертационной работе результаты не противоречат известным теоретическим положениям и подтверждаются результатами апробации и внедрения прототипа многоагентной системы, реализующего представленные алгоритм и метод обнаружения компонент ботнетов.

По теме диссертации опубликовано 12 научных статей и тезисов докладов, из них 3 статьи в изданиях, рекомендованных ВАК для публикации основных результатов диссертаций на соискание учёной степени кандидата наук. Имеется свидетельство о государственной регистрации программы для ЭВМ.

Основные положения диссертационной работы докладывались и обсуждались на следующих научных конференциях:

– XIV Международной научной конференции «Компьютерные науки и информационные технологии», Уфа – Гамбург – Норвежские Фьорды, 2012;

– II Международной конференции "Интеллектуальные технологии обработки информации и управления", Уфа, Россия, 2014.

– I Международной научно-практической конференции «Технологии цифровой обработки и хранения информации» (DSPTech’2015), г. Уфа, 2015 г.

– Международный семинар «Ситуационное управление, интеллектуальные, агентные вычисления и кибербезопасность в критических инфраструктурах» (CM/IAC/CS/CI-2016), г. Иркутск, 2016 г.;

– XVII Байкальской Всероссийской конференции с международным участием «Информационные и математические технологии в науке и управлении», г. Иркутск, 2012.

Разработанный прототип многоагентной системы обнаружения и блокирования ботнетов используется в челябинском подразделении федерального провайдера-компании «Интерсвязь».

Результаты исследования используются в учебном процессе на кафедре «Информационные технологии и экономическая информатика» ФГБОУ ВО «Челябинский государственный университет» при проведении лекций и лабораторных работ по курсам «Система интеллектуального анализа данных» и «Защита информации» для студентов направления 02.04.02 «Фундаментальная информатика и информационные технологии» и 09.03.01 «Информатика и вычислительная техника».

Структура и объем диссертации.

Диссертация состоит из введения, четырех глав, заключения и списка литературы. Текст работы изложен на 149 страницах, содержит 43 рисунка и 33 таблицы. Список использованной литературы состоит из 149 наименований.

Глава 1 содержит обзор ботнетов и их жизненного цикла, анализ современных средств защиты от распределенных атак типа «отказ в обслуживании», методов обнаружения ботнетов. Описана концепция многоагентной системы обнаружения и блокирования ботнетов в разрезе семи аспектов, которые помогают сформировать представление об устройстве предлагаемой в диссертационной работе системы защиты. Формулируются цель и задачи исследования, решаемые в диссертации.

В главе 2 описывается архитектура многоагентной системы обнаружения ботнетов, кооперация агентов, разрабатывается алгоритм обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition, метод распределенного обнаружения управляющих компонент ботнета, с которых осуществляется контроль атаки ботнета.

В главе 3 содержится метод автоматического формирования базы ботов, проводится анализ эффективности разработанного алгоритма обнаружения управляющего трафика ботнета, основанного на применении технологий интеллектуального анализа данных, представлены результаты проведенных экспериментов.

Глава 4 посвящена разработке исследовательского прототипа многоагентной системы обнаружения и блокирования ботнетов путем выявления управляющего трафика на основе интеллектуального анализа данных, сформулированы требования к системе. Приведены результаты оценки эффективности разработанной системы, описана концепция практического внедрения многоагентной системы.

ГЛАВА 1 АНАЛИЗ СУЩЕСТВУЮЩИХ ПОДХОДОВ И МЕТОДОВ ОБНАРУЖЕНИЯ БОТНЕТОВ И РАСПРЕДЕЛЕННЫХ АТАК ТИПА «ОТКАЗ В ОБСЛУЖИВАНИИ»

В первой главе рассматриваются основные понятия и методы, связанные с вопросами организации и обнаружения ботнетов. Приводится обзор методов обнаружения ботнетов с заключением, отражающим их ограничение в рамках решаемой задачи. Приводится также таксономия средств обнаружения и описание жизненного цикла ботнетов. С учетом того, что предлагаемый метод опирается на этап обнаружения атак типа «отказ в обслуживании», в первой главе приводится классификация защитных механизмов от данного типа атак и обзор различных механизмов защиты.

1.1 Обзор ботнетов и их жизненного цикла

Одной из самых опасных угроз безопасности в сети Интернет являются ботнеты. Бот – это программное обеспечение робота, в контексте диссертационного исследования – экземпляр вредоносного программного обеспечения, работающий на зараженном компьютере автономно и автоматически без ведома пользователя. Код бота, как правило, профессионально написан финансируемыми преступными группами и содержит обширный набор функциональных возможностей [50], чтобы иметь возможность выполнять множество вредоносных действий. В некоторых случаях под термином «бот» подразумевается инфицированный ботом компьютер. Ботнет – это сеть ботов, которые находятся под удаленным контролем злоумышленника. Злоумышленника, контролирующего ботнет, называют бот-мастером. Бот-мастер управляет ботнетом посредством некоторых каналов команд и управления (Command and Control, C&C).

В настоящее время ботнеты являются одной из основных причин криминальной деятельности в Интернете [6, 60, 75, 111], включающей:

– *Распределенные атаки типа «отказ в обслуживании» (Distributed Denial of Service, DDOS).* Ботнету может быть отдана команда совершить целенаправленную, распределенную атаку типа «отказ в обслуживании» на любую систему в Интернете с целью поглотить ресурсы (например, пропускная способность) системы таким образом, что она не сможет должным образом обслуживать своих легитимных пользователей. В настоящее время практически все DDoS-атаки осуществляются с платформы ботнетов. Несмотря на простоту техники атаки DDoS, она является очень эффективной за счет размеров ботнета и общей пропускной способности ботов. К примеру, одна из самых известных атак за последнее время – это DDoS-атака против популярного веб-хостинга ИТ проектов GitHub в марте 2015 года [100].

– *Рассылка спама.* Примерно 55% электронной почты в сети Интернет является спамом [4], что составляет несколько миллиардов сообщений спама в интернет-трафике ежедневно. Большинство этих сообщений спама на самом деле отправлены из ботнетов. Точный процент спама, исходящий от ботнетов, может варьироваться в зависимости от различных статистических данных. Ряд известных ботнетов был использован для рассылки спама, в том числе Bobax [124], ранний спам-бот, использующий HTTP в качестве C&C, и Storm Worm (он же Peacomm) [77, 82], еще один печально известный P2P ботнет, агрессивно проводящий рассылку спама.

– *Фишинг*. Ботнеты широко используются для размещения вредоносных поддельных сайтов. Обычно преступники рассылают сообщения спама (например, с использованием ботнетов) с целью обманом заманить пользователя посетить поддельные сайты (как правило, связанные с финансовой деятельностью – интернет-банкингом). Таким образом, преступники могут получить доступ к конфиденциальной информации пользователей, такой, как имена пользователей, пароли и номера кредитных карт. Согласно отчету «Спам и фишинг» компании

«Лаборатория Касперского», в третьем квартале 2015 г. с помощью системы «Антифишинг» было предотвращено 36 300 537 попыток перехода пользователей на фишинговые сайты.

– *Кликфрод.* Бот-мастер может получать прибыль от управления кликами ботов на онлайн-объявления (то есть посылать HTTP-запросы на веб-страницы рекламодателя) с целью личной или коммерческой выгоды. Кликфрод может использоваться для повышения рейтинга веб-сайтов в поисковых системах. Например, ботнет Clickbot.A, использующийся для выполнения малозаметных атак мошеннического клика, симулирует поведение большого числа обычных пользователей [65].

– *Кража информации.* Боты активно используются для кражи конфиденциальной информации, такой, как номера кредитных карт, пароли или ключи авторизации на локальном компьютере пользователя. Бот может легко украсть пароль от аккаунта системы дистанционного банковского обслуживания, используя кейлоггеры и захват экрана.

– *Распространение других нежелательных программ*, например, рекламное/шпионское программное обеспечение. Ботнеты являются хорошей платформой для распространения множества других форм вредоносного программного обеспечения.

– *Абузоустойчивый хостинг*. Зараженные компьютеры могут использоваться для размещения на них различного запрещенного контента, например, детской порнографии или террористического материала.

Чтобы глубже понять природу ботнетов, нужно рассмотреть их жизненный цикл. Он, как правило, состоит из нескольких этапов, в соответствии с рисунком 1.1: концепция, распространение, взаимодействие, маркетинг, выполнение атаки, оценка результатов атаки [114, 115].

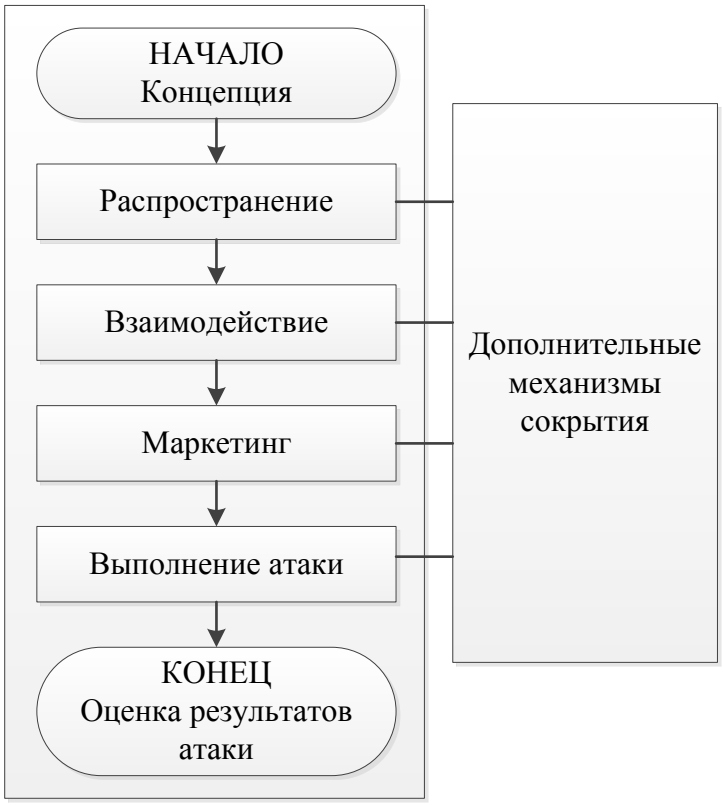

Рисунок 1.1 – Этапы жизненного цикла ботнета

На первом этапе жизненного цикла любого ботнета определяется его *концепция*. Для создания ботнета существенным элементом является мотивация. Именно от нее будут зависеть архитектура и разработка ботнета. На этой стадии устанавливаются конструктивные характеристики, которые определяются конкретной целью создания ботнета. Этап состоит из двух основных процессов: проектирования и разработки. На этапе проектирования конкретизируется организационная структура ботнета, которая может быть трех типов:

1. Централизованная. Все зараженные компьютеры находятся под централизованным управлением, т.е. координируются с помощью выделенного управляющего сервера. Такая структура является наиболее распространенной [45].

2. Децентрализованная. В этом случае все компьютеры передают команды управления между собой. Структура строится по технологии P2P [77, 82, 118, 131, 134].

3. Гибридная. Включает в себя несколько равноправных подсетей ботнета, каждая из которых управляется выделенным узлом [5].

Распространение. После того, как вредоносный код бота разработан, необходимо распространить его. Как правило, это достигается за счет эксплуатации уязвимостей узлов и внедрения на них ботов. Путей для распространения вредоносного программного обеспечения достаточно много. Например, можно заразить удаленные уязвимые узлы через прямую эксплуатацию уязвимостей либо, распространить через подходы социальной инженерии, такие, как электронная почта и мгновенные сообщения [83]. В последнее время бот-мастера используют скомпрометированные веб-серверы, чтобы заразить тех, кто посещает веб-сайты, применяя технику «попутной загрузки» [62, 69]. При реализации нескольких векторов распространения вредоносного кода бот-мастер может заразить много жертв. В настоящее время ботнет обычно содержит от десятков до сотен тысяч ботов, однако некоторые из них содержат несколько миллионов ботов.

Этап взаимодействия описывает взаимодействие между бот-мастером и ботами и включает в себя два различных процесса. Первый заключается в регистрации скомпрометированного узла в роли функционирующей части ботнета. Второй процесс состоит в обеспечении управляющей связи для ботнета. Бот-мастер должен держать связь с ботами через управляющий канал. Обмен информации состоит из передачи команд от бот-мастера к ботам и операций технического обслуживания (обновление кода, учет членства и т.д.).

Маркетинг. Наиболее распространенной мотивацией для разработчиков ботнета является получение денежной прибыли. С учетом того, что монетизация ботнета может происходить по-разному (к примеру, сдача в аренду, продажа услуг), разработчику понадобятся клиенты. Чтобы привлечь клиентов, разработчик должен опубликовывать информацию о ботнете на приватных форумах, описывать преимущества и возможности своей разработки.

Выполнение атаки. На этом этапе бот-мастер отдает команду ботам выполнить атаку. Ботнет осуществляет вредоносную деятельность.

Оценка результатов атаки. Конечной целью любого ботнета является возможность успешного выполнения атак. Для контроля реализации цели

необходимо проводить этап оценки результатов атаки. Если результат оценки показывает безуспешность функционирования, то ботнет становится бесполезным, и все ресурсы, задействованные на предыдущих этапах, становятся потраченными впустую.

В соответствии с рисунком 1.1, в рамках этапов жизненного цикла ботнета существуют различные дополнительные механизмы. Эти механизмы, как правило, сосредоточены на скрытии бота от средств защиты. В качестве примеров скрывающих механизмов можно привести шифрование, обфускацию кода, полиморфизм, IP-спуффинг, EMAIL-спуффинг.

В настоящей работе рассматривается проблема обнаружения ботнетов на этапе взаимодействия и выполнения атаки. Работа на этапе выполнения атаки может показаться малоэффективной в связи с уже случившимся фактом атаки, следовательно, ущерб в каком-то объеме узлам и сетям уже нанесен. С другой стороны, мы приобретаем возможность зафиксировать адреса скомпрометированных узлов, что дает нам преимущество в анализе трафика этих узлов и возможность выявить трафик этапа взаимодействия, что, в конечном счете, позволит формировать сигнатуру ботнета по управляющему трафику и обнаруживать ботов по всей глобальной сети.

1.2 Анализ методов обнаружения ботнетов

В этом разделе рассмотрим существующие методы обнаружения ботнетов; определим, почему эти методы недостаточны для обнаружения; опишем сходства и различия рассмотренных методов и предлагаемого в работе решения. В частности, для сравнения существующих решений описывается таксономия методов обнаружения ботнетов.

1.2.1 Обнаружение вторжений и вредоносного программного обеспечения

Существующие техники обнаружения вторжений и вредоносного программного обеспечения (далее – ВПО) можно классифицировать на сетевые решения и решения, функционирующие на узлах. Используемые на узлах техники обнаружения очень важны для распознавания исполняемых файлов ВПО и аномалии в поведении на уровне узла (например, вызывается определенный системный вызов и создаются определенные ключи реестра). Антивирусные инструменты полезны для традиционного обнаружения вирусов в течение длительного времени [129]. Другой типичный пример метода обнаружения вторжения на основе узла – это мониторинг системных вызовов [72].

Но когда встает проблема обнаружения ботнетов, эти методы обнаружения, основанные исключительно на анализе узла, имеют некоторые проблемы. Во-первых, традиционные антивирусные инструменты основаны на поиске сигнатур, таким образом, требуют объемной, точной и часто обновляемой базы сигнатур. Ботнеты могут легко избежать сигнатурного обнаружения, обновляя себя чаще, чем пользователи обновляют свои антивирусные базы. Во-вторых, системы обнаружения на основе узла находятся на том же уровне привилегий, что и боты на некотором узле. Таким образом, боты могут отключить антивирусные средства системы или использовать руткит-технологии, чтобы защитить себя от обнаружения на локальном узле. Частота обнаружения ботов относительно низка по сравнению с традиционными вредоносными программами. Например, вредоносное программное обеспечение Kraken было не замечено 80% коммерческих антивирусных средств. Проведенное в PandaLabs в 2007 году исследование установило, что даже при установленной актуальной защите (например, антивирусные средства) значительная часть персональных компьютеров (22,97%) заразились вредоносными программами. Таким образом, миллионы узлов Интернета связаны с деятельностью ботнетов [90], а фактический процент может быть еще выше. Кроме того, мониторинг узла в

реальном времени на основе поведения, как правило, сопровождается значительными накладными расходами системы, за счет чего такие решения могут стать менее привлекательными для конечных пользователей.

Таким образом, в рамках работы акцент делается на сетевые решения обнаружения. Оставшаяся часть этого раздела будет сфокусирована на работах, соответствующих исследованию, главным образом, основанных на сети.

Существующие исследования проблемы обнаружений вторжений, основанных на сети, предложили немало методов и систем обнаружения вторжений. Snort [116] и Bro [107] – пара представителей систем обнаружения вторжений (далее - СОВ), основанных на сигнатурах. Они полагаются на большую базу сигнатур для идентификации попыток вторжения в сетевом трафике. Основной недостаток сигнатурных СОВ похож на недостаток антивирусных средств – это невозможность определять новые атаки, потому что они ранее ни когда не встречались и, соответственно, не имеют сигнатур. СОВ, основанные на анализе аномалий, могут преодолеть это ограничение путем описания нормального, легитимного трафика. Соответственно, любое отклонение от этого описания будет считаться аномалией. Примерами таких систем являются PAYL [135] и Anagram [136]. Эти системы изучают полезную нагрузку входящих пакетов, проводят n-граммный анализ и выявляют эксплойт в полезной нагрузке. Основной недостаток решений на основе анализа аномалий – это большое количество ложных срабатываний.

До распространения ботнетов типичным вредоносным программным обеспечением являлись черви. Червь – это самораспространяющаяся вредоносная программа, которая копирует себя, используя сетевые технологии. Основным отличием ботнета от червей является наличие управляющего канала. Поэтому ботнеты являются более гибкими, чем черви. Гибкость заключается в возможности управления деятельностью ботов, в то время, как деятельность червей запрограммирована разработчиком заранее и отсутствует возможность выбирать функциональность червя после заражения.

Существует множество научных работ по обнаружению червей. Для обеспечения быстрого и автоматического распространения черви обычно используют сканирование [123], поэтому почти все механизмы обнаружения червей направлены на обнаружение сканирования или определенного поведения. Мур [102] предложил использовать распределенную «сеть телескопов» для своевременного предупреждения, которые наблюдали бы за событиями безопасности, такими, как прохождение трафика сканирования в сети Интернет. Провос [110], Дагон и др. [63] предложили использовать метод приманки, чтобы получить экземпляр червя и проанализировать его поведение. Цзоу и др. [148] предложили алгоритм обнаружения, основанный на фильтре Калмана, подход, который определяет злонамеренное сканирование больших пространств неиспользуемых IP адресов. Ву и др. [140] предложили алгоритм обнаружения, основанный на количестве пострадавших, который отслеживает возрастающее количество новых заражений за пределами сетей пострадавших. Джунг и др. [87, 88] и Уивер и др. [139] предложили подход обнаружения червей, основанный на наблюдении, что сканирование червей, как правило, вызывает большое количество неудачных соединений. Гу и др. [78] предложили алгоритм корреляции адресов назначения-источника для обнаружения червей с помощью подхода обнаружения аномалий.

Некоторые из указанных выше методов обнаружения вторжений и ВПО могут быть полезны в нахождении определенных аномалий ботнетов, но сами по себе не подходят для обнаружения ботнетов по следующим причинам:

1. Большинство СОВ фокусируются на изучении входящего трафика на наличие признаков попыток вторжения типа точка-точка. Как правило, они обнаруживают начальные входящие попытки вторжения, производя при этом огромное количество сигналов тревоги, что хорошо описано Соммером [121]. Тем не менее, отличить успешное заражение локального хоста от повседневного множества сканирования и попыток вторжения является сложной задачей, как и любой аспект сетевой обороны. Кроме того, из-за последних достижений в разработке ВПО, в частности, ботов, трудно определить, когда ВПО изначально

попадает в контролируемую сеть. Компьютеры могут быть заражены ботом не только путем традиционных удаленных эксплуатаций, но и множеством других способов. Например, пользователь может загрузить вредоносное вложение электронной почты, или пользователь может посетить веб-сайт и заразиться через атаку типа «Drive-by download», в последнее время очень актуальных [62, 95]. Уже зараженный ноутбук тоже может подключаться к контролируемой сети.

2. Ботнеты очень гибки. Жизненный цикл инфицирования может состоять из нескольких различных этапов. Однако существующие подходы рассматривают только определенный этап, такой, как сканирование, поэтому они имеют меньше шансов обнаружить ботнеты. Они могут вызывать ложные срабатывания, если узел, не являющийся ботом (нормальный хост либо зараженный другим видом ВПО), проявляет активность, схожую со сканированием. Они могут вызывать ложноотрицательные реакции, если бот не сканирует, или сканирует, но использует технику сканирования, отличную от определяемых системой обнаружения. Таким образом, по-прежнему существует необходимость в новых методах, которые больше подходят для обнаружения ботнетов.

Традиционные техники обнаружения вторжений и ВПО полезны для обнаружения определенных признаков ботнетов. Некоторые из этих существующих методов могут быть компонентами новой системы, которая будет сочетать их с новыми методами обнаружения.

1.2.2 Взаимосвязь оповещений и взаимодействие систем обнаружения вторжений

Во многих научных работах проводилось исследование взаимосвязи оповещений и методов сотрудничества СОВ, которые объединяют множество оповещений, событий и признаков обнаружения сетевого вторжения [24]. Эти методы позволяют аналитику получить интерпретацию потоков сигналов тревоги

от сетевых датчиков на более высоком уровне, тем самым уменьшить проблему шума в случае использования традиционных сетевых СОВ.

Основное назначение взаимосвязи оповещений – это сокращение журнала регистрации событий, обнаружение многоэтапных атак и распознавание намерения атаковать. В частности, методы, используемые для распознавания многоэтапных атак, имеют некоторое сходство с техникой вертикальной корреляции, используемой в системе BotHunter [79].

Один из подходов отслеживания сложных и многоэтапных атак заключается в явном указании этапов, отношений и порядка различных составляющих атаки. В качестве примера – системы USTAT [84] и NetSTAT [133], две СОВ, основанные на методах анализа изменения состояний. Они определяют компьютерную атаку как последовательность действий, вызывающих переход в состоянии безопасности системы. Кроме того, Вальёром и др. [70, 132] были определены взаимосвязи многоэтапных атак согласно сценариям атак, заданных с использованием STATL [68], языка для выражения атак в виде состояний и переходов. Другая подобная система – JIGSAW [130] для моделирования сложных атак использует понятия концепций и возможностей, а система, предложенная Нином и др. [105], обеспечивает формальную основу для корреляции оповещений; и наконец, CAML [57], язык для определения и выявления сценариев многоэтапных атак. В отличие от предлагаемого в работе метода, все эти методы основаны на строгих причинно-следственных связях, например, предусловия и постусловия или строгой временной последовательности атак. Одним из их очевидных ограничений является то, что зависимости и последовательности должны быть указаны вручную для всех атак, но такие зависимости/последовательности часто неизвестны либо очень неточны. Более того, отсутствие событий в зависимостях/последовательностях приведет к неудаче всей корреляции. Что же касается обнаружения ботнета, хоть узлы, инфицированные ботом, регулярно совершают последовательность общих действий (этапов), редко можно определить точно все этапы, и обнаружить их так же трудно, как и предсказать порядок и временной промежуток, в котором

произойдут эти этапы. Таким образом, приведенные выше техники корреляции оповещений не подходят для обнаружения ботнетов.

Основная цель взаимодействия СОВ – это сбор информации из различных источников, чтобы обнаружить распределенные и скоординированные атаки, такие, как распространение червя. Эти методы имеют некоторое сходство с методом обнаружения управляющего трафика ботнета, предлагаемого в работе.

Несколько работ, в том числе «EventMonitoring Enabling Responses to Anomalous Live Disturbances» [109], «Autonomous Agents For Intrusion Detection» [48], «Distributed Intrusion Detection System» [120] и «Coordinated Attack Response & Detection System» [142], предлагают использовать распределенную архитектуру, сочетающую несколько агентов для обнаружения вторжений и возможности реагирования. Эти методы обеспечивают многоуровневое распределенное решение для отслеживания распределенных и скоординированных атак через множество хостов. В другой работе Абад и др. [40] предлагают соотнести данные между различными источниками (например, регистрационные журналы syslog, firewall, netflow), чтобы улучшить точность обнаружения вторжений. Все эти методы отличаются от системы, предлагаемой в диссертационной работе, тем, что они просто обеспечивают абстрактное высокоуровневое архитектурное решение гибридных СОВ вместо предоставления конкретных алгоритмов и методов обнаружения для конкретных атак, таких, как ботнет [20].

Хсе и др. предложили систему Seurat [141], которая может обнаружить агрегированные аномальные события, такие, как распространение ВПО путем корреляции изменений файловой системы узла во времени. С целью обнаружения распространения ВПО по сети Малан [99] предложил использовать совместные группы машин для обмена сводками по последним выполненным системным вызовам. Эти два подхода аналогичны методу, предлагаемому в диссертационной работе. Принципиальное отличие заключается в том, что для мониторинга изменений системных файлов или последовательности системных вызовов они

требуют развертывания сенсоров на всех узлах в сети. Естественно, такое решение ведет к определенным проблемам:

– работа сенсоров может снижать производительность узлов;

– дороговизна осуществления внедрения на всех узлах;

– существует вероятность отключения сенсоров вредоносным ПО.

Поскольку предлагаемое в данной работе решение является сетевым решением, нам удается избежать вышеуказанных недостатков.

1.2.3 Отслеживание на основе Honeypot

Большая часть исследований ботнетов сосредоточена на понимании природы и всего потенциала угрозы ботнетов, например, на проблеме исследования размеров ботнета, проблеме сбора экземпляров ботов или отслеживания компонент ботнета.

Исследования размеров могут помочь понять угрозу ботнета. Кук и др. [60] провели несколько основных исследований динамики развития ботнетов. Дагон и др. [64] предложили использовать технику перехвата DNS для исследования ботнетов и отметили суточное поведение ботнетов. Барфорд и Ягнесваран [50] исследовали исходный код бота для формирования взгляда на ботнет изнутри. Они проанализировали структурные сходства, защитные механизмы, возможности управления крупными семействами ботов.

Для эффективного сбора информации о ботнете и его отслеживания исследователи часто используют методы приманки. Фрилинг и др. [73] использовали приманки для отслеживания ботнетов, отчеты которых помогали понять события ботнетов. Nepenthes [46] – приманка низкого уровня взаимодействия, которая имитирует несколько уязвимостей и автоматизирует сбор бинарных файлов ВПО. Раджаб и др. [42] провели комплексный подход для сбора экземпляров ботов и отслеживания ботнетов, тем самым обеспечив

углубленное изучение текущей деятельности ботнетов. Используя метод приманки, исследователи могут собирать экземпляры ботов. Дальнейший анализ бинарных файлов позволяет формировать сигнатуры для контентного ботнета либо получать информацию о C&C серверах (например, DNS sinkhole [64]).

Хотя приманки и являются эффективным инструментом для сбора информации о ботнетах, они имеют ряд ограничений. Во-первых, приманки низкого взаимодействия, такие, как Nepenthes [46], могут отлавливать атаки только ограниченного числа известных эксплойтов, для которых они специально имитируют уязвимую среду. Приманки высокого уровня взаимодействия могут не реализовывать всех сервисов, не решают проблему масштабирования. Во-вторых, приманки, как правило, предназначены для захвата вредоносных программ, распространяющихся с помощью сканирования удаленных уязвимостей [13], поэтому они не захватывают ВПО, использующее другие способы распространения, такие, как электронная почта или атака типа «Web drive-by download», которые являются двумя из самых популярных методов распространения [69, 95]. В третьих, нет никакой гарантии частоты или объема получения ВПО с помощью этого подхода, т.к. приманка может только ждать и надеяться, что ВПО сама свяжется с ней. В четвертых, ВПО может избежать сканирования сети с «хорошо известными» приманками [52], определить окружение виртуальных машин, часто используемых для разворачивания приманок [74, 147], и изменить свое поведение, чтобы избежать анализа. Наконец, приманки сообщают об инфекциях только на машинах-ловушках, они не могут сообщить о заражении машины, не являющейся ловушкой и функционирующей в корпоративной сети. Эти недостатки ограничивают использование приманок в качестве эффективных систем обнаружения ботнетов.

1.2.4 Существующие подходы обнаружения ботнетов

Вопрос обнаружения ботнетов не новая область. Уже было предложено множество различных подходов для обнаружения ботнетов.

А.Н. Назаров и И.К. Сачков [30, 33] предложили логико-вероятностный подход, позволяющий разрабатывать модели бот-атак, учитывающие лучшие практики противодействия атакам ботнетов. Предлагаемый вариант механизма принятия решений основывается на формализации априорного опыта экспертов по бот-атакам в нечёткой базе нечётких продукционных правил. Авторы предложили структуру интеллектуальной системы, состоящую из следующих уровней:

– система нечетких продукционных правил, описывающих работу идентификатора характеристик бот-атак с учетом экспертных оценок;

– нейро-нечеткая сеть, в структуре которой отражена система нечётких продукционных правил;

– четкая самообучаемая нейросеть для решения задачи кластеризации(классификации) входных данных бот-активности из веб-пространства.

Бинкли и Синх [54] для обнаружения IRC-ботнетов предложили объединить IRC-статистику и данные о рабочей нагрузке TCP (т.е. аномальная активность сканирования). Их подход позволяет определять только некоторые типы ботнетов, в частности, IRC-ботов, которые выполняют атаки сканирования.

Рамачандран и др. [112] предложили использовать контрразведку по DNSBL (черные списки DNS), чтобы найти членов ботнетов, генерирующих спам. Основное предположение данного подхода заключается в том, что бот-мастер может использовать DNSBL для определения статусов его ботов, т.е. подозрительно, когда машина запрашивает множество запросов о других, но сама редко запрашивается другими. Эта эвристика в некоторых случаях может быть полезной, но в целом не является действенной и может порождать множество

ложных срабатываний. В результате этот подход работает только в некоторых случаях выявления спам ботнетов.

Rishi [76] – это система обнаружения IRC-ботов на основе сигнатур, которая отслеживает совпадение с именами (никнеймами) известных IRC-ботов. Подобно всем сигнатурным средствам, антивирусам или СОВ, этот подход точен, но только если доступна обширная и точная база данных сигнатур. Данная система обладает и недостатками, присущими сигнатурным решениям – невозможно обнаружить ботов без использования шаблонов известных имен ботов.

Согласно И.В. Котенко и др., традиционно применяемые сигнатурные и эвристические методы выявления вредоносного программного обеспечения не способны обеспечить достаточный уровень обнаружения новых и ранее не известных вариантов [22, 23]. Это обусловливает применимость методов интеллектуального анализа данных в решении данной задачи. Так, Ливадас и др. [98, 127] предложили подход, основанный на машинном обучении. Для обнаружения ботнета используются различные характеристики трафика протоколов обмена сообщениями, такого, как IRC, на сетевом уровне (например, байт в секунду). Карасаридис и др. [89] изучали потоки сетевого уровня для контроля IRC-ботнетов в магистральных сетях. Эти два подхода аналогичны работе предлагаемого в диссертации решения в кластеризации, но отличаются во многих моментах [16]. Во-первых, они используются для обнаружения IRC-ботнетов (путем сопоставления известных шаблонов IRC трафика). Во-вторых, они могут обнаруживать ботнеты только с централизованной структурой. Предлагаемая в данной работе система не имеет предположений об использовании известных шаблонов управляющего трафика, а также может обнаруживать ботнеты с любой структурой, в том числе P2P.

Стинсон и Митчел предложили BotSwat [125], узловую систему отслеживания вредоносного программного обеспечения, идентифицирующую программы, которые используют сетевые данные (из ненадежного внешнего источника) как аргумент для системных вызовов. Эти данные не вводятся пользователем и не имеют явного разрешения на запуск в качестве параметров

системных вызовов. Таким образом, эта система пытается определить поведение возможного удаленного управления ботом. Этот подход может приводить к ложным срабатываниям (потому что многие легитимные программы используют данные трафика в аргументах своих системных вызовов) и снижению производительности (очень сложно анализировать распространение ВПО, поэтому этот подход, как правило, используется только для анализа, а не для обнаружения).

Йен и Рейтер предложили TAMD [144], систему определяющую потенциальное ВПО (в том числе и ботнет). Она агрегирует трафик, который разделяется одинаковым внешним адресом назначения, похожей полезной нагрузкой пакетов и включает в себя внутренние узлы с одинаковыми операционными системами. Концепция агрегирования трафика схожа с механизмами, используемыми в данной работе [15, 93]. Метод агрегирования TADM основан на наблюдении за объемами трафика в сети назначения, в частности, увеличении трафика по сравнению с некоторым историческим показателем. Кроме этого, метод агрегирования ограничен объединением ботов, использующих централизованную структуру C&C, значит, потерпит неудачу при использовании других структур (например, P2P). TADM стремится обнаружить широкий спектр потенциально подозрительных хостов, имеющих общие сети назначения, одинаковую полезную нагрузку, похожие платформы ОС. Следовательно, TADM может вызывать больший процент ложных срабатываний.

1.3 Обзор механизмов защиты от распределенных атак типа «отказ в обслуживании»

Серьезность проблемы DDOS-атак, их увеличивающаяся частота и продолжительность, изощренность [12, 94] привели к появлению многочисленных защитных механизмов [9, 25–27]. В то же время, хотя и разработаны многие решения, проблема все же остаётся актуальной. Есть несколько серьезных

факторов, которые препятствуют исследованиям противодействия DDOS-атакам. Существует множество возможных DDOS атак, очень немногие из которых могут быть обработаны на стороне жертвы. Поэтому необходимо иметь распределенную, возможно, скоординированную систему реагирования. При этом очень важно, чтобы реакция происходила во многих точках Интернета. Поскольку Интернет управляется распределенным образом, широкое развертывание любой защитной системы или сотрудничество между сетями достаточно сложно реализовать.

Распределенная система реагирования должна быть развернута сторонами, не подвергающимися прямому ущербу от атаки [14]. Из этого следует необычная экономическая модель; сторона, которая будет поддерживать стоимость развертывания и работоспособности, не является стороной, которая непосредственно извлекает выгоду от системы. Решение этого вопроса, скорее всего, возможно либо законодательным путем, либо внедрение системы защиты должно представлять обоюдный интерес.

Для того, чтобы разработать качественное защитное средство, необходимо иметь хорошее понимание DDOS-атак. Существуют общедоступные анализы популярных инструментов DDOS атак [53, 67], но не хватает информации о частоте различных типов атак (например UDP-наводнение, TCP SYN-наводнение) и параметрах распределенных атак: скорость, продолжительность, размер пакетов, количество атакующих ботов, попытки реагировать и их эффективность, принесенные убытки и т.д. Считается, что публичная отчетность по атакам принесет ущерб деловой репутации, поэтому, как правило, держится в секрете.

Рассматривая защитные механизмы от распределенных атак типа «отказ в обслуживании» можно предложить следующую их классификацию, представленную на рисунке 1.2 [101]: по уровню активности, по степени взаимодействия, по месту развертывания. В классификации по уровню активности выделяют предупреждающие и реагирующие методы защиты. Предупреждающие методы пытаются либо исключить возможность DDOS-атак

вообще, либо пытаются выдержать атаку таким образом, чтобы не отказывать в предоставляемой услуге легитимным пользователям [41].

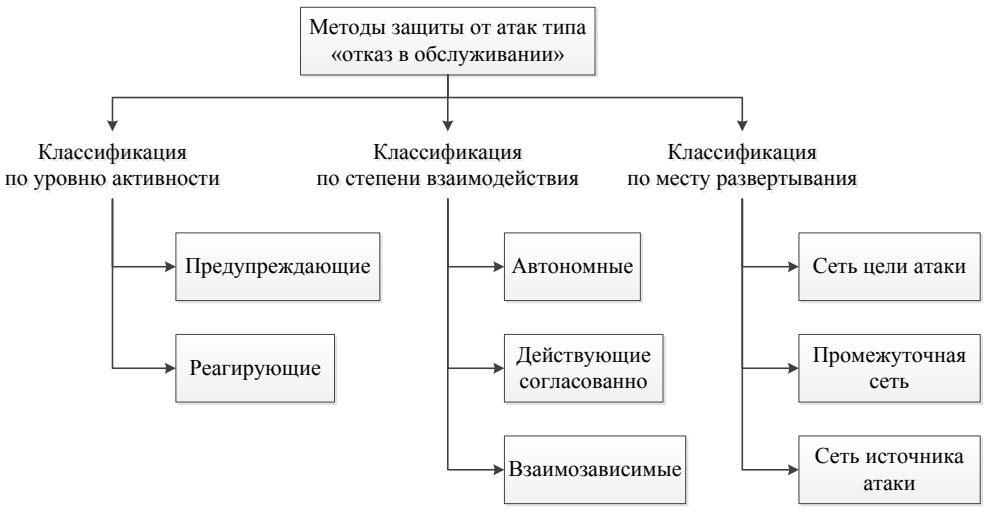

Рисунок 1.2 – Классификация методов защиты от атак типа «отказ в обслуживании»

Реагирующие методы стремятся смягчить последствия нападения на жертву. Для этого они должны обнаруживать атаки и реагировать на них. В данном случае важно как можно раньше обнаружить атаку и иметь низкую степень ложных срабатываний. Реакция может заключаться в изучении характеристик пакетов и передаче их конкретным реагирующим средствам. В классификации по степени взаимодействия можно выделить три типа: автономные, действующие согласованно (совместно), взаимозависимые. Автономные методы осуществляют независимую защиту в месте, где они развернуты (узел или сеть). В качестве простых примеров автономных методов можно привести межсетевые экраны или системы обнаружения вторжений. Если система выполняет свои функции распределенным образом, она все равно считается автономной, если может полностью развернуться в сети, которую защищает. Совместно действующие методы способны к автономному обнаружению и реагированию, но могут сотрудничать с другими механизмами. При этом производительность в совместной работе часто выше. Взаимозависимые механизмы не могут работать в одной точке развертывания автономно. Они либо

требуют развертывания в нескольких сетях, либо полагаются на другие объекты в задачах предотвращения, обнаружения атак и в эффективном реагировании. Т.е. зависимые механизмы, развернутые на одном маршрутизаторе, не дадут никакой выгоды.

От DDOS-атак можно защищаться в разных местах сети, таким образом, защитные механизмы также могут быть классифицированы в зависимости от места развертывания. Елена Мирковик и др. представили три возможных места в Интернете для развертывания DDOS защиты: сеть цели атаки, промежуточные сети, сеть источника атаки [101]. На рисунке 1.3 показана упрощенная схема сети Интернет, демонстрирующая различные места для развертывания защитных средств.

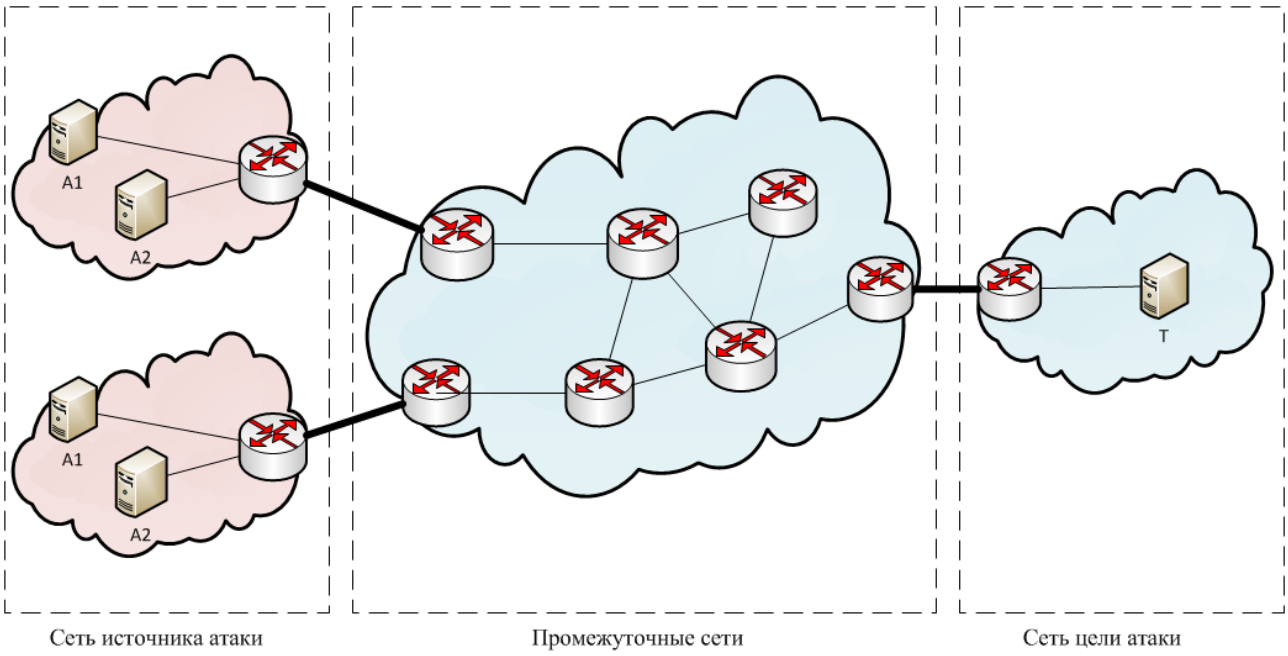

Рисунок 1.3 – Места развертывания защитных средств

Защитные механизмы также можно рассмотреть с точки зрения их стратегии [41]. Естественно, они будут перекликаться с рассмотренными классификациями, но в отличие от классификаций, стратегии позволяют отразить функциональный аспект защитного средства. Стратегии различных механизмов защиты от DDOS-атак можно разделить на четыре категории представленные на рисунке 1.4.: предупреждение, выявление, реагирование и толерантность.

Превентивные подходы, как было отмечено, пытаются исключить возможность DDOS-атак или предотвратить атаку до причинения значительного ущерба. Выявление может также быть классифицировано как обнаружение атаки либо идентификация источника атаки. Мониторы обнаружения атаки анализируют события в системе для выявления вредоносных попыток вызвать отказ в обслуживании. Это важный шаг, прежде чем осуществлять противодействие атаке. Целью идентификация источника атаки является поиск источника атаки независимо от адреса источника, содержащегося в поле вредоносного запроса. Механизмы реагирования, как правило, инициируются после обнаружения атаки, чтобы исключить или свести к минимуму последствия нападения на жертву. Целью стратегии толерантности является минимизация ущерба от атаки, в отсутствие способности дифференцировать вредоносные действия от легитимных действий. Чтобы инициировать механизмы толерантности, необходимо просто знать о том, что система находится под атакой.

Основываясь на сходствах различных решений, можно для каждой из четырех категорий защиты выделить несколько типов защитных механизмов. На рисунке 1.4 показана таксономия защитных механизмов для классификации существующих решений от DDOS атак.

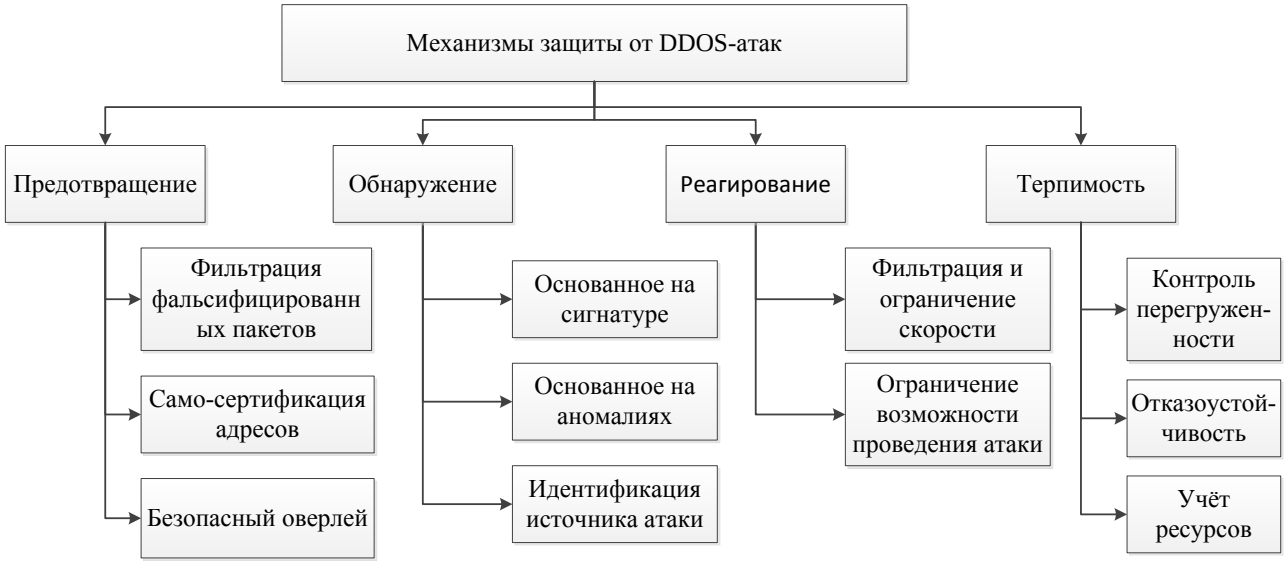

Рисунок 1.4 – Стратегии механизмов защиты от DDOS-атак

В рамках диссертационного исследования наибольший интерес представляют превентивные механизмы, направленные в большей степени на предотвращение атак и причинения значительного ущерба атакуемым ресурсам. Поэтому в рамках работы мы ограничимся рассмотрением только этой категорией защитных механизмов.

1.4 Превентивные механизмы защиты от DDOS-атак

Превентивные механизмы защиты от DDOS-атак стремятся прекратить атаку до того, как она приведет к повреждению системы. Механизмы предотвращения включают следующие: фильтрация фальсифицированных пакетов, самосертификация адресов, безопасный оверлей.

1.4.1 Фильтрация фальсифицированных пакетов

С целью скрыть происхождение DDOS-атаки многие злоумышленники фальсифицируют IP-адрес. Отраженные атаки и методы усиления атак опираются также на IP-спуфинг. Механизмы фильтрации предназначены для запрещения трафика DDOS-атаки с фальсифицированными адресами источников, путем отброса пакетов с ложными IP-адресами.

Фильтрация недопустимых адресов и валидация адресов источника.

Фильтрация недопустимых адресов и валидация адресов источников определены в требованиях к маршрутизаторам IPv4 IETF RFC 1812 [47]. Фильтрация недопустимых адресов указывает маршрутизатору не пересылать все недопустимые пакеты, где недопустимый пакет является пакетом, содержащим в заголовке отправителя или получателя IP-адрес, присвоенный Internet Assigned Numbers Authority (IANA) как IP-адрес специального назначения. Последние IPv4 адреса специального назначения определены в IETF RFC 6890 [61], другие

примеры недопустимых адресов включают в себя зарезервированное и нераспределенное пространство IP-адресов и адрес назначения 255.255.255.255/32.

Проверка адреса источника указывает, что маршрутизатор должен реализовать возможность фильтрации трафика на основе сравнения адреса отправителя в пакете и таблицы пересылки для логического интерфейса, через который пакет был получен [47]. Если такая фильтрация разрешена, маршрутизатор должен отбрасывать пакет без уведомления, если он был принят через интерфейс, отличающийся от того, через который будет пересылаться пакет, направленный по адресу отправителя. Иными словами, если маршрутизатор не будет пересылать пакет, содержащий данный адрес, через определенный интерфейс, ему не следует доверять этому адресу, указанному в поле отправителя, для пакета, поступившего через тот же интерфейс.

Фильтрация недопустимых адресов исключает возможность подмены для небольшого набора адресов. Злоумышленник может просто изменять подмены любых недопустимых адресов. Проверка адресов отправителя может устранить большинство подмен адресов. Тем не менее, с учетом количества ассиметричных маршрутов в Интернете вполне возможно, что путь возврата к адресу отправителя пакета может не следовать из того же интерфейса, на котором был принят пакет. Таким образом, использование подобной методики фильтрации пакетов может вызвать побочный ущерб для трафика легитимных пользователей.

Входящая/исходящая фильтрация.

Цель входящей/исходящей фильтрации в том, чтобы разрешить входящий или исходящий из сети трафик, только если адреса отправителя находятся в пределах ожидаемого диапазона IP-адресов. Входная фильтрация относится к фильтрации трафика, входящего в сеть, а выходная фильтрация относится к фильтрации трафика, покидающего сеть. Использование фильтрации на входе в сеть для защиты от DOS-атак описывается в RFC 2827 как Best Current Practice (BCP) [71].

Идея входящей/исходящей фильтрации показана на рисунке 1.5. В соответствии с рисунком первый атакующий находится в сети 198.51.100.0/24, подключение которой к Интернету обеспечивает провайдер А (ISP A). Фильтрация входящего трафика на интерфейсе маршрутизатора R2, который обеспечивает возможность подключения сети атакующего, ограничивает трафик, позволяя принимать лишь те пакеты, где адрес отправителя относится к блоку адресов 198.51.100.0/24 и запрещает злоумышленнику использовать недопустимый адрес отправителя, который не принадлежит другому адресному блоку.

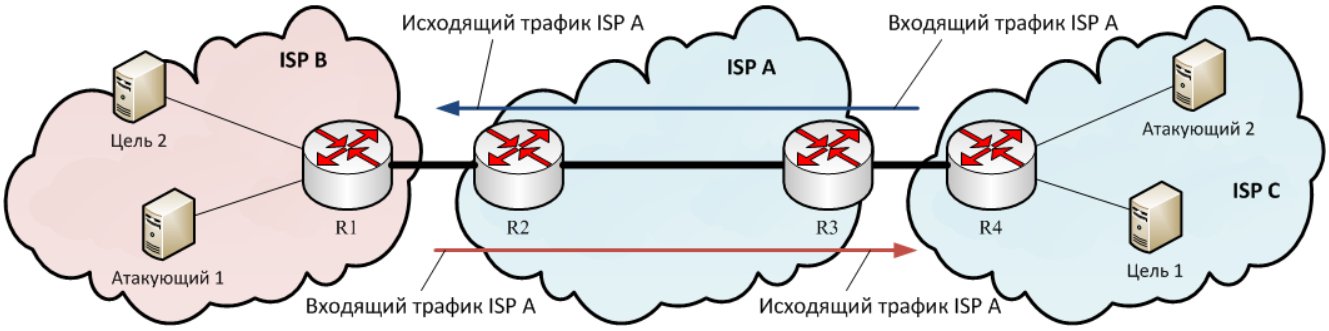

Рисунок 1.5 – Пример входящей/исходящей фильтрации

Фильтр входящих пакетов на маршрутизаторе 2 работает по следующему алгоритму:

– Если адрес отправителя в пакете относится к сети 198.51.100.0/24, то пакет пересылается в направлении получателя.

– Если адрес отправителя в пакете относится к любому другому блоку, то пакет отбрасывается.

Это и есть входящая фильтрация. Если граничный маршрутизатор R1 вместо R2 предоставляет ту же функцию фильтрации, тогда это будет называться исходящей фильтрацией.

Ключевым требованием фильтрации на входе/выходе является знание ожидаемых IP-адресов на определенный порт. В некоторых сетях со сложной топологией не просто получить эту информацию. Кроме того, стимула для интернет-провайдеров разворачивать входную/выходную фильтрацию нет,

поэтому в настоящий момент фильтрация используется не везде. Таким образом, у злоумышленников остается возможность выбора сети без фильтрации на входе/выходе для осуществления атак отказа в обслуживании с использованием фальсифицированных адресов отправителя.

Фильтрация на основе маршрута.

Парк и Ли предложили распределенную фильтрацию пакетов на основе маршрута (distributed packet filtering, DPF), подходящую для фильтрации потока поддельных пакетов [106]. DPF использует информацию о маршрутизации, чтобы определить, если пакет прибывает на маршрутизатор (например, пограничный маршрутизатор в автономной системе), действующий в отношении зарегистрированных адресов отправителя/получателя, учитывая ограничения доступности налагаемые маршрутизацией и топологией сети. DPF использует информацию топологии маршрутизации BGP для фильтрации трафика с поддельными адресами отправителя.

У DPF есть несколько ограничений. Если разрешается несколько путей при маршрутизации пакетов от отправителя к получателю, то становится достаточно легко атаковать, используя поддельные адреса отправителя, обходя фильтрацию на основе маршрута. Кроме того, DPF может удалить легитимные пакеты, если было отмечено изменение маршрута. Правила фильтрации в DPF имеют очень грубую детализацию уровня автономной системы (AC), и атакующий может обойти фильтры DPF, тщательно выбирая диапазон IP-адресов для подмены.

Протокол принудительной валидации адреса отправителя.

Чтобы преодолеть недостатки фильтрации на основе маршрута, Ли и др. предложили протокол «Source Address Validity Enforcement» (SAVE) [96]. SAVE постоянно распространяет сообщения, содержащие информацию о достоверных адресах источника из сетевого местоположения источника по всем направлениям. Таким образом, каждый маршрутизатор строит входящую таблицу, которая ассоциирует каждый интерфейс маршрутизатора с набором допустимых блоков адресов источников. Когда пакет приходит на интерфейс, маршрутизатор

проверяет по входящей таблице, поступает ли этот пакет с правильного направления.

SAVE преодолевает асимметрию маршрутизации в Интернете, периодически обновляя входящие таблицы на каждом маршрутизаторе. Он должен изменить протокол маршрутизации, что является непростой задачей, которая может занять много времени для выполнения. Несмотря на то, что SAVE фильтрует фальсифицированные пакеты для защиты других лиц, он не предоставляет прямые стимулы развертывания. Как при входящей/исходящей фильтрации и фильтрации на основе маршрута при частичной интеграции, злоумышленники всегда могут фальсифицировать IP адреса в сетях, которые не реализуют SAVE.

Фильтрация по количеству транзитных участков (хопов).

Джин и др. [86] представили фильтрацию пакетов с поддельными IP адресами, использующую метод называемый фильтрацией по количеству хопов (Hop-Count Filtering, HCF). Они утверждают, что хотя злоумышленник может подделать любое поле в заголовке IP пакета, он не сможет фальсифицировать количество транзитных участков в IP пакете, необходимых для достижения пункта назначения. Они предлагают метод, делающий заключение по информации о количестве транзитных участков из значения поля TTL в IP-заголовке. Используя метод расчета количества транзитных участков, основанный на TTL, жертва строит таблицу соответствия количества транзитных участков IP-адресу отправителя. Когда жертва получает пакет, она вычисляет количество транзитных участков для его IP-адреса и сравнивает его с количеством транзитных участков, хранящимся в таблице соответствия для идентификатора поддельного адреса пакета. HCF остается в бдительном состоянии по умолчанию, в котором он контролирует изменение количества транзитных участков без отбрасывания пакетов. При обнаружении потока фальсифицированных пакетов HCF переключается в активное состояние для изучения каждого пакета и отбрасывания пакетов с поддельными IP-адресами.

Преимущество HCF в том, что его можно разворачивать на стороне жертвы, что намного проще по сравнению с сетями, основанными на подходах фильтрации. Кроме того, потенциальная жертва имеет более сильный стимул для развертывания защитного механизма, чем провайдеры сетевых услуг. Тем не менее, HCF страдает от большого количества ложных срабатываний (ошибок первого и второго рода). Метод подсчета хопов в HCF опирается на изначальное значение поля TTL, а начальное значение TTL различно для различных операционных систем (ОС). Метод подсчета хопов HCF не работает, когда разница между начальными значениями TTL для различных ОС меньше, чем среднее количество хопов между конечными хостами Интернета. Начальное значение TTL для некоторых ОС действительно меньше, чем среднее количество хопов, например, начальные значения TTL 30, 32, 60 и 64. Легитимные же пакеты могут быть определены как поддельные из-за неточного соответствия IP-адреса количеству хопов или задержек в обновлении количества хопов. Даже если предположить, что вычисленное количество хопов точное, атакующие могут обманывать поддельными адресами с тем же количеством хопов, что и их хосты. Наконец, как и в любых других подходах, реализованных на стороне жертвы, HCF не может предотвратить трафик DDOS атаки, который поступает в огромном количестве на машину осуществляющую фильтрацию.

Защита источника IPv4 (IPv4 Source Guard).

Функция IP Source Guard обеспечивает фильтрацию IPv4-адресов отправителя на интерфейсах второго уровня, чтобы предотвратить вредоносный хост от подмены IP-адреса. IP Source Guard контролирует назначение адресов по протоколу DHCP и использует статические соответствия для автоматической настройки интерфейсов второго уровня для отбрасывания трафика, если IP-адрес отправителя отличается от IP-адреса, присвоенного этому интерфейсу.

Функция IP Source Guard реализуется на коммутаторах, которые используются в маленьких офисах (SOHO) или в сетях на уровне доступа. Данная функция опирается на то, что каждый узел имеет один сетевой интерфейс и каждый сетевой интерфейс имеет один адрес.

Метод «Passport».

«Passport» – это метод проверки подлинности адресов отправителя, разработанный Лиу и др. [97]. Он направлен на обеспечение того, что ни хост отправитель, ни АС не могут подменить адресное пространство, что и реализует Passport. На рисунке 1.6 представлена общая схема работы метода проверки подлинности адресов отправителя.

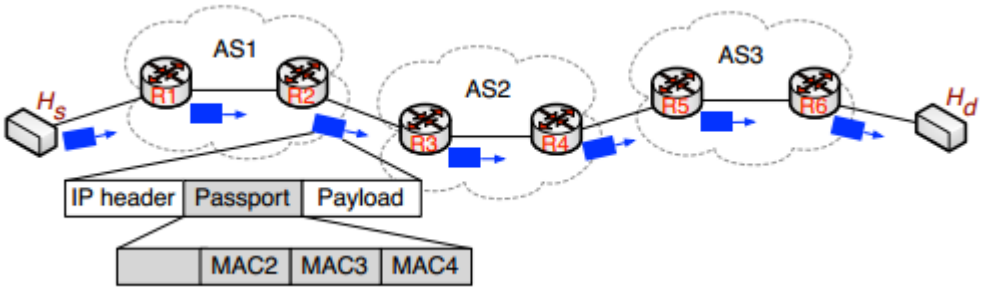

Рисунок 1.6 – Общая схема работы метода проверки подлинности адресов отправителя

Когда пакет покидает свою отправляющую АС, пограничный маршрутизатор отмечает в пакете код аутентификации сообщения (Message Authentication Code, MAC) для каждой АС по пути к сети назначеия. Каждый MAC вычисляется с использованием общего для отправляющей АС и следующей по пути АС секретного ключа. Когда пакет входит в следующую по пути АС пограничный маршрутизатор проверяет соответствующий MAC, используя секретный ключ с АС источником. Верификацию маршрутизатор выполняет, используя адрес отправителя пакета для поиска отправляющей АС, получает общий ключ и пересчитывает MAC. АС может получить соответствие между адресом источника и соответствующей исходящей АС из BGP, используя атрибут пути AS-PATH [80]. Если MAC-штампы маршрутизатора для адреса источника находятся за пределами его адресного пространства, MAC не будут проверяться на последующих АС. Маршрутизатор стирает значение MAC в пакете после верификации, чтобы предотвратить возможность проведения криптоанализа оффлайн. Пакет с неверным MAC в промежуточной АС отбрасывается в АС

назначения. Две АС получают парный секретный ключ, который используется для вычисления MAC, путем использования стантартного протокола обмена ключом Диффи-Хеллмана в их BGP анонсах.

Схема паспорта предотвращает хост одной АС от подмены адреса из других АС. В результате злоумышленники могут подделывать любой IP-адрес в пределах одной АС. Кроме того, каждый пограничный маршрутизатор, реализующий схему паспорта, нуждается в дополнительной памяти для хранения путей АС, общих ключей со всеми АС, что значительно увеличивает необходимый для работы объем памяти, учитывая общее количество сетевых диапазонов и АС в Интернете.

1.4.2 Само-сертификация адресов

Проблеме ответственности уделялось очень мало внимания на ранних стадиях проектирования сетей, и только в последние годы она начала рассматриваться сетевыми архитекторами. Одна из основных проблем, которая должна обсуждаться – это ответственность IP-адресов. Далее будут рассмотрены попытки решить эту проблему.

Протокол идентичности хоста.

Архитектура протокола идентичности хоста (Host Identity Protocol, HIP) [104] предлагает новое пространство имен, называемое идентичностью хоста, и новый протокол сетевой модели, называемый протоколом идентичности хоста [81], работающий между сетевым и транспортным уровнями. Пространство имен идентичности хоста состоит из идентификаторов хоста, где идентификатором хоста является открытый ключ из асимметричной пары ключей. Каждый хост может иметь более одного идентификатора хоста, но нет двух хостов, имеющих одинаковые идентификаторы. Существует важное различие между идентичностью хоста и идентификатором хоста. Идентичность хоста относится к абстрактной сущности, которая определяется в то время, как идентификатор хоста

является конкретной битовой последовательностью, которая используется в процессе идентификации.

Идентификатор хоста может быть опубликован, так как он считается открытым, либо не опубликован. Открытый идентификатор хоста должен храниться в системе DNS, либо в каталоге LDAP, либо в любой существующей инфраструктуре открытых ключей (PKI), в то время как неопубликованные идентификаторы хоста должны храниться только на хосте, который им владеет. Идентификаторы хоста используются только во встроенном протоколе согласования ключей, называемом HIP Base Exchange, другие протоколы используют тег идентичности хоста (Host Identity Tag, HIT), 128-битный хеш идентификатора хоста. HIT идентифицирует отправителя и получателя пакета в HIP пакете, который является IP пакетом, несущим сообщение протокола HIP.

IP-адрес в Интернете выполняет двойную роль: локация и идентификатор конечной точки. В архитектуре HIP имена конечной точки и локации отделены друг от друга. IP-адреса продолжают действовать в качестве локации. Идентификаторы хоста берут на себя роль идентификаторов конечных точек. Имена конечной точки на основе идентификаторов хоста немного отличаются от имен интерфейсов, идентичность хоста может быть одновременно доступна через несколько интерфейсов. В архитектуре HIP, TCP и UDP соединения более не привязаны к IP-адресу, идентификация хоста происходит с помощью HIT. На рисунке 1.7 отражены различия между связями логических объектов текущей архитектуры Интернета и HIP-архитектуры. Таким образом, HIP отделяет транспортный уровень от сетевого уровня, что позволяет каждому развиваться отдельно.

Тег идентичности хоста в HIP обладает таким важным свойством безопасности, как возможность удостоверять себя самостоятельно. Само-сертификация означает, что владелец идентификатора может доказать свое право собственности на идентификатор, не полагаясь на доверие третьей стороны. Так как HIT является хешем публичного ключа хоста и только этот хост имеет соответствующий закрытый ключ, хост может доказать своё право владения HIT

другому хосту, используя простой протокол запроса/ответа, когда другой хост запрашивает такое доказательство.

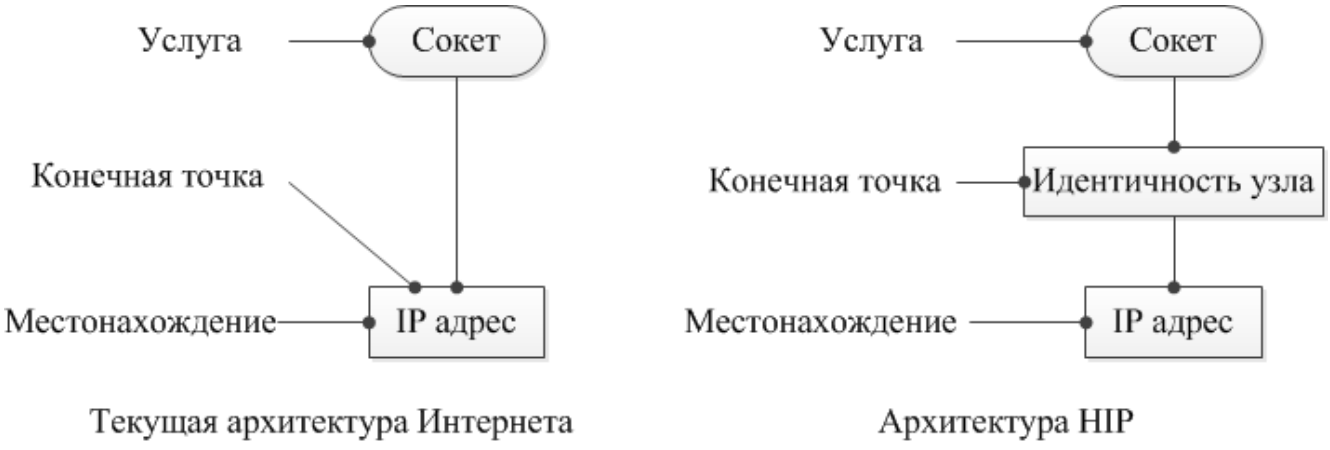

Рисунок 1.7 – Различия между связями логических объектов текущей архитектуры Интернета и HIP архитектуры

Архитектура HIP является эффективной в предотвращении фальсификации IP-адреса и в поддержании ответственности хоста за свои действия в замкнутом пространстве, где все знают открытые ключи друг друга. В этом случае должна быть своего рода инфраструктура открытых ключей или сервис соответствия открытых ключей адресу. Для очень масштабной распределенной системы, которая охватывает большое количество административных доменов, такой, как Интернет, очень трудно развернуть глобальную PKI. Не имея возможности надежно привязать идентификатор хоста к адресу, злоумышленник может фактически выдумать неограниченное количество идентификаторов хоста и использовать их для скрытия свое идентичности.

Подотчетность межсетевого протокола.

Подотчетность межсетевого протокола (Accountable Internet Protocol, AIP) была предложена Андерсеном и др. для того, чтобы обеспечить подотчетность сетевого уровня путем самоудостоверяющих адресов [44]. AIP предназначен для решения проблемы отсутствия связывания, с точки зрения безопасности, хоста со своими IP-адресами, а также отсутствия связывания номеров АС с диапазонами IP-адресов, которые принадлежат данной АС. Архитектура AIP предполагает, что

каждая АС располагает свою сеть в один или несколько подотчетных доменов (AD), каждый из которых имеет глобальный уникальный идентификатор (AID). Каждому узлу также присваивается уникальный идентификатор конечной точки (EID). По аналогии со структурой межсетевой адресации, AIP адресует хост, расположенный в некотором домене, используя адрес следующего формата: AID:EID. Оба идентификатора, AID и EID, разрядностью в 160 бит, их структура представлена на рисунке 1.8. 144 бита занимает хеш открытого ключа, для AID – это хеш открытого ключа подотчетного домена (AD), а для EID – это хеш открытого ключа соответсвующего хоста. Чтобы справиться с ситуацией, когда хост соединен с одним и тем же AD несколько раз, конечные 8 бит EID являются интерфейсными битами, которые однозначно идентифицируют интерфейс. Для AID интерфейсные биты устанавливаются в нули. В связи с тем, что AIP использует криптографические алгоритмы, которые со временем могут стать менее надежными, каждый AIP-адрес содержит 8-битный номер версии, указыавющий на схему подписи, использующейся для генерации адреса.

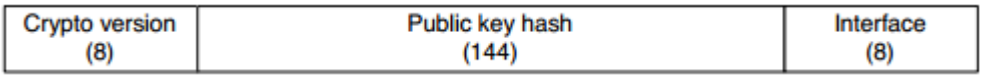

Рисунок 1.8 – Структура AIP адреса

AIP использует самоудостоверяющие адреса для обнаружения источника фальсифицированных адресов и применяет реализованный в сетевых картах «запорный» протокол, позволяющий получателю послать подписанное «запорное» сообщение тому отправителю, от которого получатель не хочет получать трафик. Таким образом, при распределенной флудинговой атаке жертва, используя «запорную» схему, может противостоять атаке.

Простота и эффективность против DDOS атак делают AIP очень привлекательным для разработок будущих межсетевых протоколов. Тем не менее, использование схемы адресации AIP в Интернете является большой проблемой с точки зрения маршрутизации и масштабируемости траффика. Кроме этого, так

как хосты сами генерируют свой EID, вредоносные хосты могут генерировать неограниченное количество EID и использовать их для проведения атаки. Большой проблемой может стать компрометация закрытого ключа узла либо подотчетного домена. Скомпрометировав ключ, злоумышленник может достаточно долго незаметно выдавать себя за жертву.

1.4.3 Безопасный оверлей

Безопасные оверлей – это один из подходов, целью которого является предотвращение DDOS-атак на ограниченные участки защищаемых сетей с помощью маршрутизации трафика, предназначенного для защищенной сети, через оверлейные сети, построенные поверх IP. Поскольку оверлейная сеть допускает только авторизованных пользователей и тщательно проектируется, обеспечивая свойство избыточности и сопротивления DOS, атакующим становится трудно вызвать отказ в обслуживании в защищаемых сетях или серверах. Безопасный оверлей предполагает, что оверлейная сеть – это единственный способ взаимодействия с защищаемой сетью для хостов, находящихся за пределами доверенного домена. Такая изоляция защищенной сети от остального Интернета осуществима либо путем скрытия IP адреса защищенной сети либо с помощью распределенных межсетевых экранов, фильтрующих весь входящий в защищенную сеть трафик, пропуская только трафик от доверенных в оверлейной сети узлов.

Secure Overlay Service (SOS).

Secure Overlay Service является архитектурой оверлейной сети, предложенной Керомайтис для проактивного предотвращения DOS атак [91]. Безопасный оверлей формируется путем выбора набора узлов, распределенных по всей глобальной сети и логически связанных безопасными туннелями. Целью SOS-архитектуры является обеспечение связи только между защищаемой

областью сети и пользователем, которому дается предварительное разрешение на взаимодействие с этой областью. Для достижения этой цели SOS сперва предполагает фильтрацию области вокруг защищаемого объекта, что достигается настройкой маршрутизаторов по периметру объекта пропускать только траффик от нескольких оверлейных узлов. Эти оверлейные узлы называются секретными сервлетами и выбираются защищаемым объектом. Секретные сервлеты вычисляют набор оверлейных узлов, которые будут выступать в качестве маяков, и информируют маяки о том, что они являются секретными сервлетами для защищаемого объекта. Источник, который хочет взаимодействовать с охраняемым объектом, в первую очередь контактирует с точкой доступа безопасного оверлея (Secure Overlay Access Point, SOAP). Только после аутентификации и авторизации запроса связи источника SOAP направляет траффик источника к одному из маяков. Маяк направляет пакеты источника к секретному сервлею, который уже перенаправляет пакеты через фильтрующий маршрутизатор к цели. Маршрутизация между оверлейными узлами SOS осуществляется с помощью услуги «струн» (Chord service) [126]. Рисунок 1.9 иллюстрирует взаимодействие между авторизованным источником и защищенной областью в архитектуре SOS.

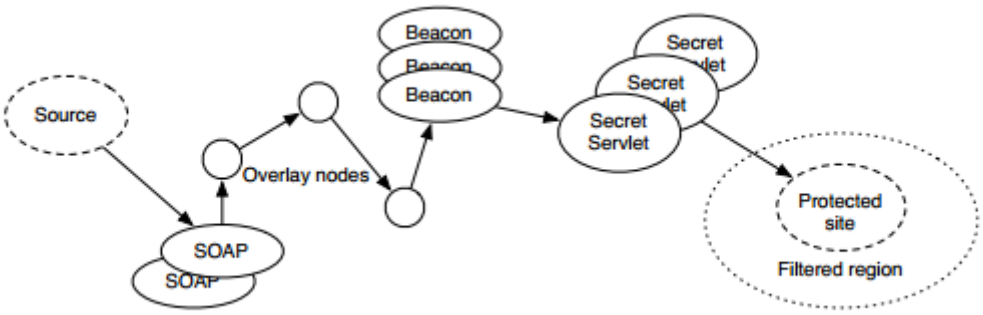

Рисунок 1.9 – Взаимодействие между авторизованным источником и защищенной областью в архитектуре SOS

SOS устойчива к DOS-атакам по следующим причинам:

1. Если атакована точка доступа, источник может выбрать альтернативную точку доступа.

2. Если атакован узел внутри оверлея, узел просто выходит из оверлея и сервис «струн» восстанавливает работоспособность.

3. Если секретные сервлеты стали известны и выступают в качестве точки атаки, то защищаемый объект может выбрать альтернативный набор секретных сервлетов.

С другой стороны, SOS не решает общую проблему DOS в Интернете. SOS предназначен только для защиты нескольких частных сервисов. Кроме того, использование в оверлейной сети создает более длинный и более медленный маршрут до пункта назначения. Результаты моделирования показали, что величина задержки в 10 раз больше, чем в случае прямой связи [91] . Наконец, строительство защищенной оверлейной сети требует установки и обслуживания дополнительных специальных узлов в сети, что подразумевает огромные дополнительные расходы.

1.5 Концепция многоагентной системы обнаружения и блокирования ботнетов

Ботнеты являются одной из самых опасных угроз в сети Интернет. При этом существующие средства защиты, такие, как системы обнаружения или предотвращения атак, системы обнаружения или отслеживания ботнетов, антивирусное программное обеспечение и honeypot-системы не обеспечивают защиту от вредоносной активности ботнетов в полной мере. Таким образом, по-прежнему существует необходимость в новых методах, которые больше подходят для обнаружения ботнетов. Традиционные техники обнаружения вторжений и ВПО полезны для обнаружения определенных признаков ботнетов. Некоторые из этих существующих методов могут быть компонентом новой системы, которая

будет сочетать их с новыми методами обнаружения. Как показано в проведенном обзоре, ботнет – это сложная многоагентная система с элементами интеллектуальной работы. На этапе распространения ботнетов проводится автоматическая процедура анализа программного обеспечения пользователя на предмет имеющихся уязвимостей, эксплуатацию которых можно использовать для заражения. После заражения также решаются сложные задачи: определение используемых пользователем защитных систем, их обход, скрытие своей работы, взаимодействие с управляющими серверами. Для защиты от ботнетов необходимо использовать систему защиты с уровнем сложности не меньше, чем у самих ботнетов. Значит, необходимо применять метод и систему, способную работать таким же распределенным способом, как и ботнет. Система должна обеспечивать возможность анализировать множество сетевых данных в разных сетях, обнаруживать сетевые атаки, влиять на фильтрацию трафика, выявлять сигнатуры вредоносного поведения и взаимодействовать между собой для эффективного выполнения перечисленных задач.

В рамках данной диссертационной работы предлагается многоагентная система обнаружения и блокирования ботнетов NET.BOTNET. Согласно проведенному обзору, данную систему можно рассмотреть в разрезе семи аспектов, которые помогут сформировать представление об устройстве этой системы защиты.

Первым аспектом является тип решения: узловое или сетевое. Ранее были рассмотрены преимущества и недостатки каждого решения. Среди систем обнаружения ботнетов было представлено BotSwat [125] – это узловое решение, и другие, в том числе NET.BOTNET, являющиеся сетевыми решениями.

Вторым аспектом является метод работы: сигнатуры или поведение/аномалии. В рамках этого аспекта Rishi [76] – единственное решение, основанное на сигнатурах. Остальные решения используют методы анализа поведения/аномалий. Преимущества и недостатки обоих решений уже обсуждались ранее. Стоит отметить, что предлагаемое решение объединяет различные методы работы, используя их в решении разных задач. Как пример для

обнаружения атак можно использовать оба метода, для выявления управляющего трафика ботнета используется метод анализа поведения, для обнаружения бота используется сигнатурный метод.

Режим работы системы – третий аспект. Можно выделить два режима: пассивный и активный. Все вышеперечисленные методы/системы пассивны. Т.е. они пассивно следят за сетевым трафиком и поведением систем. Преимущество пассивной стратегии в сравнении с активной заключается в безопасности, потому что не мешает существующей активности ботнета. Предлагаемое решение использует пассивную стратегию мониторинга. Отметим, что многим пассивным системам требуется больше количество времени, чтобы осуществить мониторинг, предлагаемое решение также имеет этот недочет.

Четвертым аспектом является фаза обнаружения. Жизненный цикл ботнета, описанный ранее, можно упростить до двух фаз в соответствии с рисунком 1.10: подготовка и эксплуатация. В фазе подготовки безвредный узел заражается ботом путем удаленного заражения или исполнением какого-нибудь вредоносного исполняемого файла и готовится к управлению через C&C. Узел начинает фазу подготовки, когда подвергся атаке, и заканчивает её, когда сформирована полная функциональность бота. Когда бот пытается подключиться к управляющему каналу, начинается фаза эксплуатации. Тогда он может быть направлен бот-мастером осуществлять любой вид вредоносной деятельности. Таким образом, можно классифицировать методы обнаружения ботнетов в соответствии со временем объявлении тревоги, либо на фазу подготовки, либо на фазе эксплуатации. Все представленные системы обнаружения, кроме BotHunter [79], в том числе и система, предлагаемая в диссертационной работе, работают только на фазе эксплуатации, обнаруживают уже имеющиеся боты, независимо от способа их проникновения в сеть.

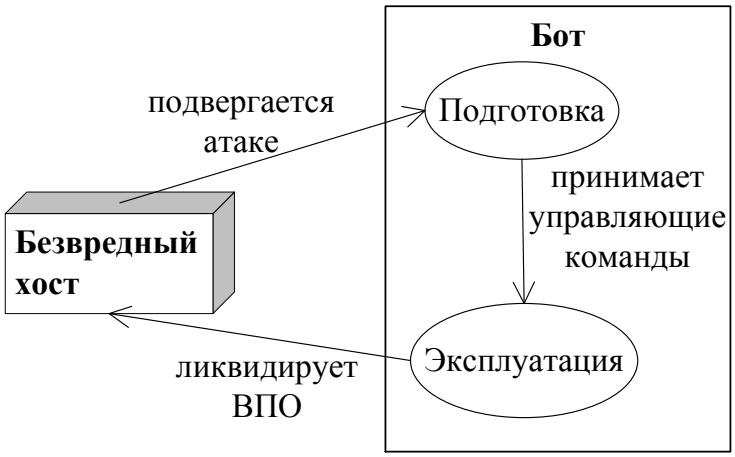

Рисунок 1.10 – Упрощённый жизненный цикл заражения ботом

В качестве пятого аспекта можно выделить цель обнаружения, будь то отдельный бот или сеть ботов. Ливадас и др. [98, 127], Карасаридис и др. [89], TAMD сосредотачиваются на выявлении группы ботов, в то время как другие системы сосредоточиваются на выявлении отдельных ботов. В принципе, это два вида решений, которые существенно дополняют друг друга. Подход на основе выявления групп требует наблюдения за множеством ботов (как минимум за двумя) для успешного обнаружения. Он позволяет обнаружить аномалию, которая не может быть заметна на уровне отдельного узла. Предлагаемое решение выявляет, в конечном счете, как отдельных ботов, так и группы, но для формирования сигнатуры анализирует трафик отдельных ботов в рамках распределенной группы.

Шестой аспект относится к предположению об обнаружении. Эти решения требуют дополнительной внешней информации (например, DNSBL). Среди рассмотренных решений некоторые (Рамачандран и др. [112]) требуют информации из других источников, таких, как DNSBL, а некоторые (Карасаридис и др. [89]) требуют предварительного кластерного анализа данных (например, результата сканирования), полученных от других систем. Предлагаемое решение требует информации из других источников, в частности, до выявления ботнета следует обнаружить совершаемую им атаку и получить адреса источников атаки для дальнейшего анализа их трафика.

Заключительный аспект – это зависимость системы от конкретной техники управления ботнетом (например, протокол или структура). Многие существующие решения работают только с определенным протоколом или структурой C&C. Rishi [76] и подход Бинкли и Синх [54] предназначены только для ботнетов на основе IRC. Точно так же, Ливадас и др. [98, 127] и Карасаридис и др. [89] в своих работах приводят подходы для обнаружения IRC-ботнетов. Но их методы могут быть применены для обнаружения ботнетов, использующих другие типы C&C, основанные на средствах мгновенного обмена сообщениями. Тем не менее, они по-прежнему ограничены централизованной структурой и зависят от знания профиля трафика. TAMD [124] не ограничена протоколом взаимодействия с C&C, но используемая агрегация по адресу назначения ограничивает подход для применения только для централизованной структуры ботнета. BotSwat [125], предложенный Рамачандран и др. [112], и решение, предлагаемое в работе, не зависят от техники управления ботнетом. Тем не менее, BotSwat [125] является узловым решением. Подход, предложенный Рамачандраном и др. [112] принципиально ограничен обнаружением только конкретных спам-ботов. NET.BOTNET является сетевым средством обнаружения ботнетов, не зависящим от протокола или структуры C&C.

Выводы по первой главе

Целью диссертационной работы является повышение защищенности информационных систем от атак с использованием ботнетов на основе разработки и применения многоагентной системы обнаружения и блокирования ботнетов с использованием алгоритмов интеллектуального анализа данных.

Для достижения цели в работе были сформулированы и решены следующие задачи:

1. Исследовать распространенные атаки типа «отказ в обслуживании», процесс их реализации и механизмы защиты от них, проанализировать существующие подходы к выявлению ботнетов.

2. Разработать алгоритм обнаружения управляющего трафика ботнета в глобальных сетях с использованием технологий интеллектуального анализа данных.

3. Предложить архитектуру многоагентной системы обнаружения и блокирования ботнетов, проанализировать эффективность функционирования предложенных в диссертационном исследовании алгоритмов.

4. Разработать метод распределенного обнаружения управляющих компонент ботнета, позволяющий обнаруживать управляющие серверы и узлы сети, с которых осуществляется контроль атаки, основанный на сигнатуре управляющего трафика.

5. Разработать исследовательский прототип многоагентной системы обнаружения и блокирования ботнетов, дать рекомендации по практическому внедрению многоагентной системы обнаружения и блокирования ботнетов.

ГЛАВА 2 РАЗРАБОТКА АРХИТЕКТУРЫ И АЛГОРИТМОВ СИСТЕМЫ ОБНАРУЖЕНИЯ БОТНЕТОВ

2.1 Архитектура многоагентной системы

Описанные в первой главе проблемы использования существующих методов обнаружения ботнетов позволяют предложить другой метод, основанный на многоагентном подходе [7, 21]. Использование многоагентного подхода предоставляет основное преимущество – динамическое решение задачи обнаружения ботнетов. Данный подход позволит создать гибкий и масштабируемый метод обнаружения.

Идея, вкладываемая в данный метод, состоит в следующем. Для того чтобы обнаружить ботнет, в первую очередь необходимо обнаружить распределенную атаку типа «отказ в обслуживании», для осуществления которой чаще всего прибегают к использованию ботнетов. После обнаружения атаки необходимо блокировать её на стороне источника атаки, а атакующее средство взять под наблюдение для выявления характерных признаков работы бота. Далее нужно попытаться идентифицировать других участников ботнета путем поиска в различных сетях ранее обнаруженных признаков работы бота.

За основу при построении метода обнаружения ботнетов была взята типовая структура сети Интернет,ы основанная на взаимодействии между автономными системами. Предлагаемый метод обнаружения ботов базируется на средстве защиты от распределенных атак типа «отказ в обслуживании» с возможностью обнаружения атаки в сети цели атаки, а предотвращения генерации атаки в сети источника. Для отображения структуры и функций системы обнаружения была создана функциональная модель. Модель создавалась с применением методологии функционального моделирования IDEF0. Данная методология основана на технологии анализа сложных систем SATD (Structured Analysis and

Design Technique), разработанной группой американских аналитиков во главе с Дугласом Россом в 1973 году [3]. Метод, предлагаемый стандартом IDEF0, предназначен для моделирования выполнения функций объекта, путем создания описательной графической модели, показывающей, что, как и кем делается в рамках функционирования системы. Функциональная модель представляет собой структурированное изображение функций системы, информации и объектов, связывающих эти функции. Модель строится методом декомпозиции: от крупных составных структур к более мелким. Элементы каждого уровня декомпозиции представляют собой действия по переработке информационных ресурсов при определенных условиях с использованием заданных механизмов. SADT-модель объединяет и организует диаграммы в иерархические структуры, в которых диаграммы наверху модели менее детализированы, чем диаграммы нижних уровней. Методология SADT создана специально для представления сложных искусственных систем путем построения моделей [32].

Контекстный блок функциональной модели в соответствии с рисунком 2.1 показывает, что на вход многоагентной системы обнаружения и блокирования ботнетов поступает сетевой трафик. Применяя различные методы и алгоритмы обеспечения защиты информации, в том числе разработанные в рамках диссертационного исследования, на выходе получается сигнатура бота, заблокированный ботнет, визуализация кибер-атак, информация об управляющих компонентах ботнета. Чтобы понять, какие функции должны быть включены в процесс защиты Интернета от атак ботнетов и как эти функции взаимосвязаны между собой, проведена декомпозиция этого процесса.

В результате получен ряд процессов, отражающих функциональные особенности системы, в соответствии с рисунком 2.2. Данные процессы решают достаточно известные задачи обеспечения защиты информации, более того, в большинстве своем эти задачи имеют определенное решение:

– задача обнаружения атаки типа «распределенный отказ в обслуживании»;
– задача блокирования атаки;

– задача выявления характерных признаков работы бота (выявление управляющего трафика и формирование сигнатуры);

– задача обнаружения бота;

– задача координации агентов системы;

– задача контроля и мониторинга работы агентов;

– задача накопления информации о кибер-угрозах;

– задача визуализации кибер-атак и ботнетов.

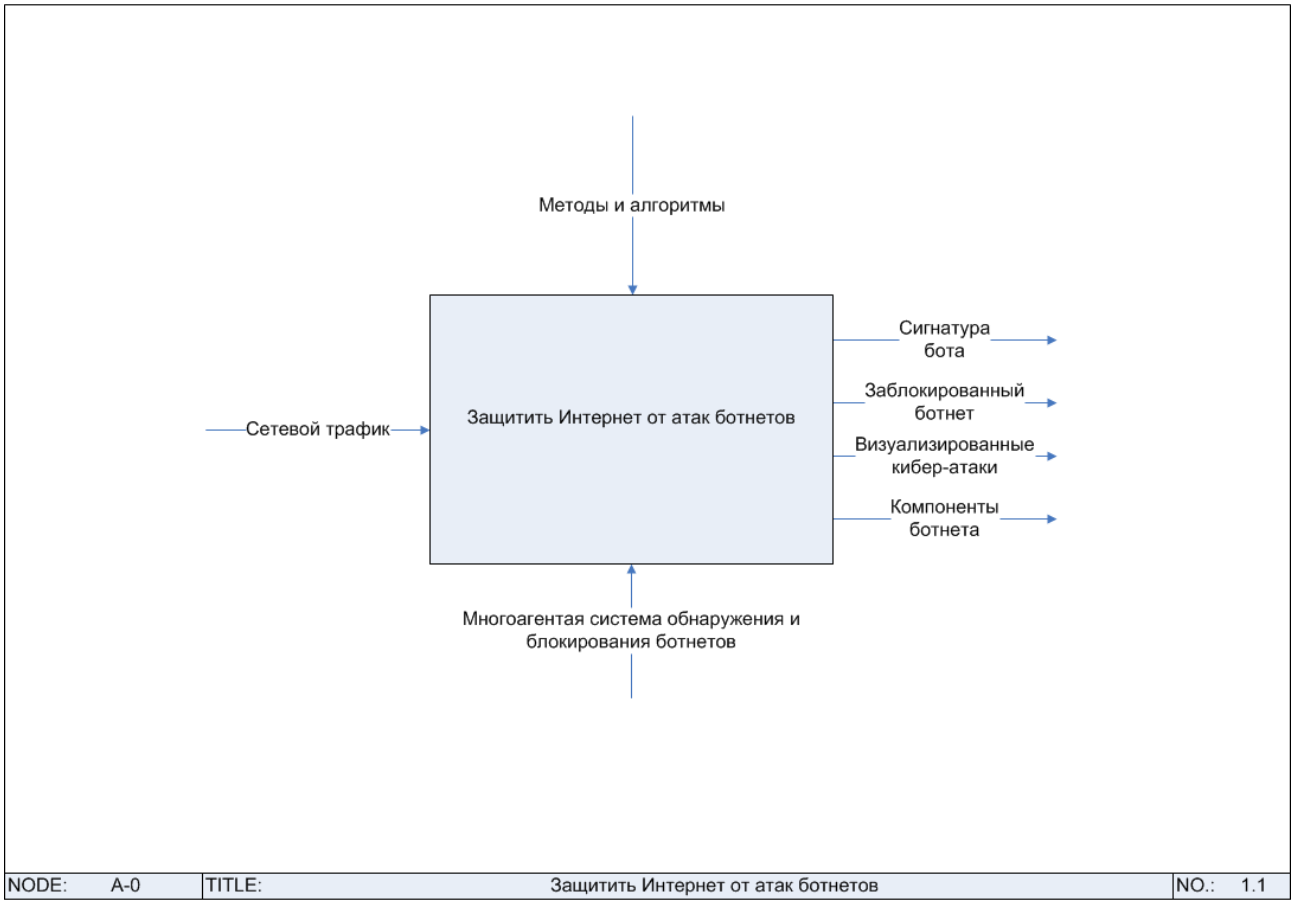

Рисунок 2.1 – Контекстная диаграмма функционирования многоагентной системы обнаружения и блокирования ботнетов

Для эффективной работы разрабатываемой системы процессы должны реализовываться в различных локациях относительно цели и источника атаки. Так, например, процессы «Обнаружить атаку», «Отследить управляющие узлы, с которых осуществляется контроль атаки» должны иметь реализацию в сетях нахождения атакуемого объекта. При этом для повышения эффективности работы

процесса отслеживания контролирующих атаку узлов данный процесс может иметь реализацию в сети отслеживаемых узлов. Процессы «Заблокировать бота», «Сформировать сигнатуру» должны реализовываться в сети источника атаки. Процесс «Осуществить мониторинг сети» должен реализовываться в нейтральных сетях по отношению к цели и источнику атаки, хотя может работать и в условиях попадания в ту или иную сеть. Процесс «Идентифицировать бота» должен реализовываться в максимальном количестве сетей для большего охвата процесса обнаружения ботов.

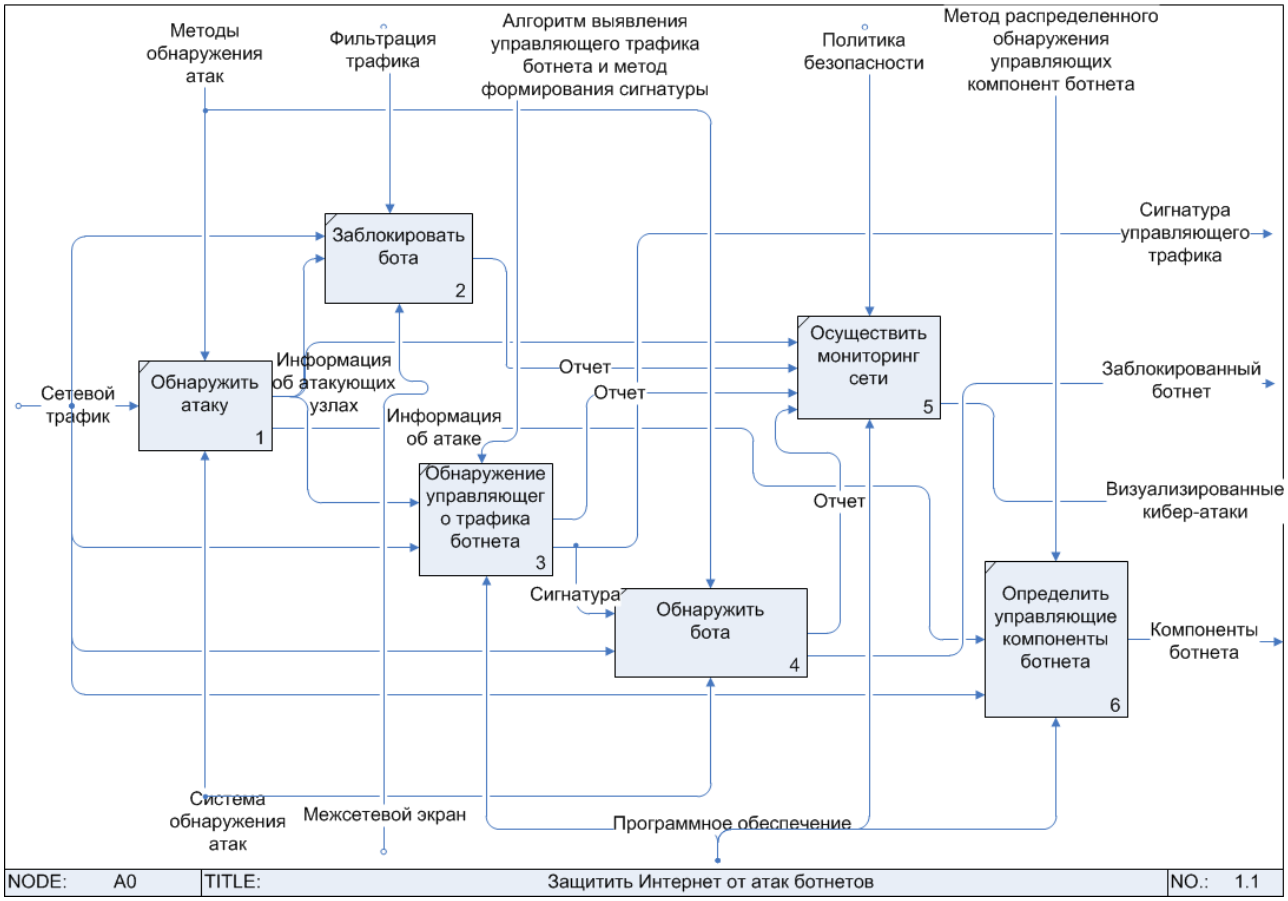

Рисунок 2.2 – Функциональная модель многоагентной системы обнаружения и блокирования ботнетов

Как можно увидеть на диаграмме, большинство процессов используют в качестве входных данных сетевой трафик. На выходе процессов формируются различные данные, приведенные в таблице 2.1.

Таблица 2.1 – Результаты работы процессов многоагентной системы

Процесс	Данные на выходе
«Обнаружить атаку»	информация об атакующих узлах, информация об атаке
«Заблокировать бота»	статус блокировки трафика атаки бота
«Обнаружение управляющего трафика»	статус формирования сигнатуры и по завершению процесса – сигнатура
«Обнаружить бота»	отчет об обнаружении бота по известной сигнатуре
«Осуществить мониторинг сети»	представление всех данных, полученных в ходе работы многоагентной системы
«Определить управляющие компоненты ботнета»	обширная информация об узлах, подозреваемых в управлении атакой

На основе анализа функциональной модели сделан вывод о том, что эффективным средством противодействия ботнетам может стать система, аналогичная по архитектуре ботнетам. По своей сути ботнет является многоагентной системой, совместная работа агентов которой приводит к эффективным кибер-атакам. Таким образом, система защиты тоже должна быть многоагентной, что приведет к высоким результатам защиты как от атак, производимых ботнетами, так и как общее средство противодействия зловредным сетям. Многоагентный подход фактически избавляет от проблем масштабирования при росте системы обнаружения. Выявленные характерные признаки взаимодействия ботов с контролерами ботнетов используются для динамического формирования сигнатур ботов. Сигнатуры позволяют обнаружить присутствие бота в других сетях. Такой подход помогает решить проблему автоматизации обнаружения ботов [17].

В качестве общих признаков ботов можно выделить:

– IP-адрес или доменное имя контролирующего центра ботнета;

– характеристики HTTP или IRC-пакетов с определенными командами управления;

– размерность сетевых пакетов;

– временные интервалы сетевых взаимодействий;

– трафик злонамеренной активности, к примеру, сканирование, рассылка спама, загрузка бинарных файлов [58, 146];

– информацию протоколов DNS, SMTP;

– используемый протокол обмена данными и порты транспортного уровня.

Процессы, выявленные в функциональной модели системы обнаружения ботнетов, можно отнести к различным классам функциональности: {Обнаружение, Блокирование, Исследование, Идентификация, Координация, Интерфейс}. Каждому классу может соответствовать свой тип агента, решающий задачи класса. С учетом необходимости анализа данных с распределенных компонент системы для реализации функций класса «Исследование» потребуется два типа агентов. Таким образом, многоагентная система обнаружения ботнета имеет вид:

$$MAS = \{A_{detection}, A_{blocking}, A_{discrovery}, A_{feature}, A_{identification}, A_{coordination}, A_{interface}\},$$

где

$A_{detection} = \{A^1_{detection}, \dots, A^n_{detection}\}$ – множество агентов обнаружения атаки типа «распределенный отказ в обслуживании». Агенты данного класса решают задачу обнаружения атак и реагируют на неё определенным в сценарии реагировании образом. В каждой автономной системе сети Интернет располагается как минимум один агент данного класса $A^i_{detection}$, где i=1..n – номер автономной системы сети Интернет.

$A_{blocking} = \{A^1_{blocking}, \dots, A^n_{blocking}\}$ – множество агентов, решающих задачу блокирования обнаруженной атаки. В каждой автономной системе сети Интернет располагается как минимум один агент данного класса $A^i_{blocking}$, где i=1..n – номер автономной системы сети Интернет.

$A_{discovery} = \{A^1_{discovery}, \dots, A^n_{discovery}\}$ – множество агентов, исследующих трафик скомпрометированных узлов. Класс агентов совместно с агентами $A_{feature}$ решает задачу выявления управляющего трафика ботнета. В каждой автономной системе сети Интернет располагается как минимум один агент данного класса $A^i_{discovery}$, где i=1..n – номер автономной системы сети Интернет.

$A_{feature}$ – множество агентов формирования сигнатуры. Участвуют в решении задачи выявления управляющего трафика ботнета. Агенты данного типа агрегируют в себе информацию, полученную в результате работы агентов $A_{discovery}$, получают шаблон управляющего трафика и формируют сигнатуру бота.

$A_{identification} = \{A^1_{identification}, \dots, A^n_{identification}\}$ – множество агентов обнаружения работы бота в рамках автономной системы. Агенты данного класса анализируют трафик сети на наличие признаков функционирования ботов. В каждой автономной системе сети Интернет располагается как минимум один агент данного класса $A^i_{identification}$, где i=1..n – номер автономной системы сети Интернет.

$A_{coordination}$ – множество агентов сети, решающих задачу распространения информации об активных агентах.

$A_{interface}$ – множество агентов сети, решающих следующие задачи: контроль и мониторинг работы сети агентов, визуализация атак, хранение информации.

Таким образом, структура многоагентной системы обнаружения ботов состоит из следующих элементов, представленных ниже в соответствии с рисунком 2.3:

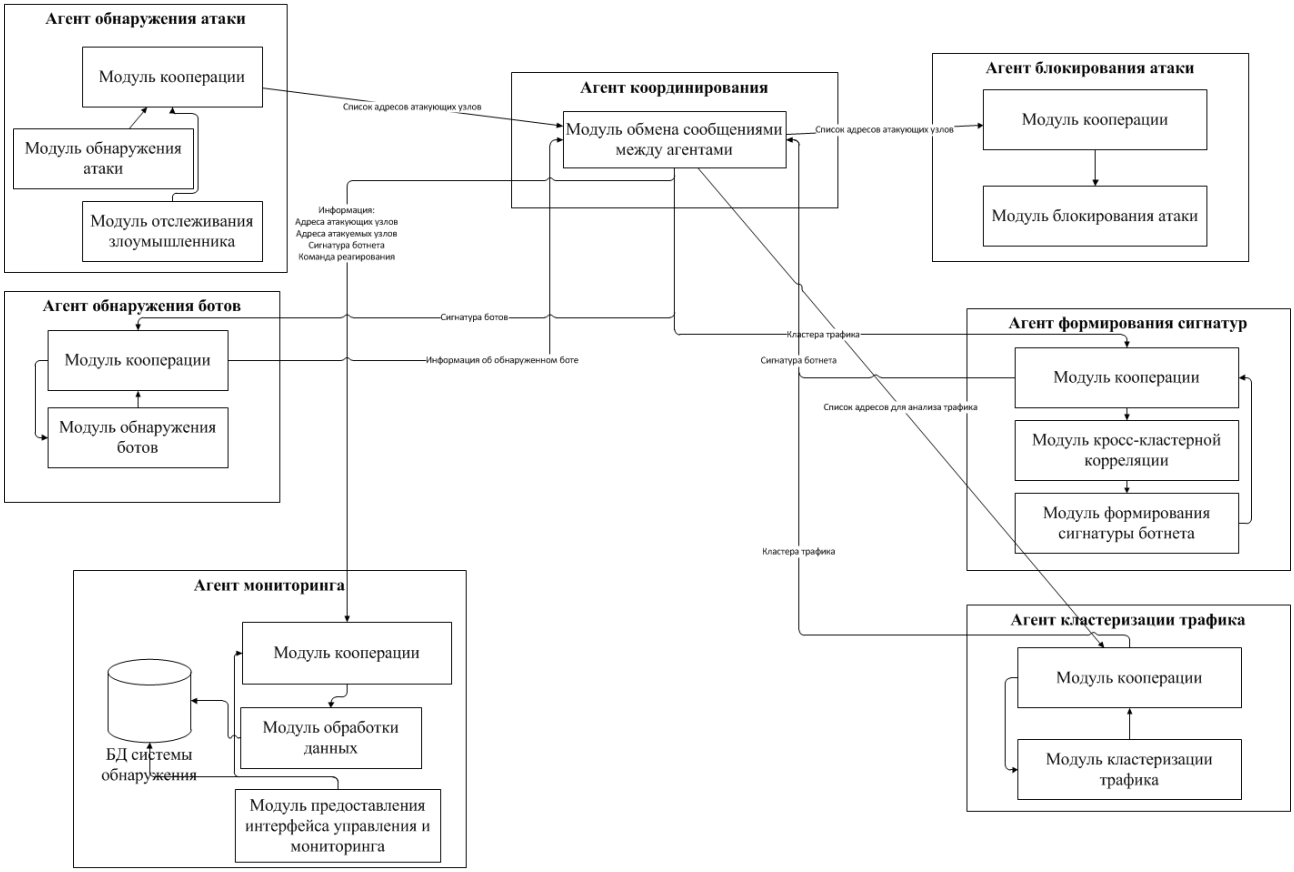

Рисунок 2.3 – Структура многоагентной системы обнаружения ботнетов

– агенты обнаружения атаки типа «распределенный отказ в обслуживании» $a_i \in A_{detection}$.

– агенты выявления управляющего трафика $a_i \in A_{discovery}$, $a_i \in A_{feature}$.

– агент обнаружения ботов $a_i \in A_{identification}$.

– агент блокирования атак $a_i \in A_{blocking}$. Функционирует, когда его расположение является сетью источника атаки. В частности, осуществляет реагирование на основе информации, полученной от агентов обнаружения атак, согласно профилю сетевой безопасности (блокирование систем, задействованных в реализации атаки, оповещение ответственных лиц по электронной почте, SMS).

– агент координации $a_i \in A_{coordination}$. Распространяет информацию о местонахождении различных агентов с целью осуществления взаимодействия между ними.

– интерфейсный агент $a_i \in A_{interface}$. Устанавливается в любой точке глобальной сети Интернет. Предназначен для контроля и мониторинга работы сети агентов, предоставления графического интерфейса визуализации обнаруженных атак, хранения и обеспечения доступа к истории обнаруженных атак.

Концептуальный алгоритм функционирования многоагентной системы заключается в следующем в соответствии с рисунком 2.4:

1. Агент обнаружения атаки типа «распределенный отказ в обслуживании» обнаруживает атаку на подконтрольную ему сеть.

2. Агент обнаружения атаки сообщает агенту координации информацию о сетях источников обнаруженной атаки.

3. Агент координации передает агентам блокирования, находящимся в соответствующих источникам атаки автономных системах, информацию об атакующем узле.

4. Агент координации передает агенту выявления признаков бота, контролирующего сеть источника атаки, информацию об атакующем узле.

5. Агент координации передает интерфейсному агенту информацию об атаке.

6. Агент блокирования прекращает злонамеренную активность узлов, находящихся в контролируемой им сети.

7. Агент исследования трафика анализирует активность узлов, замеченных в атаке. В результате чего выявляет множество кластеров коммуникационных потоков.

8. Агент исследования трафика скомпрометированного узла сообщает информацию о сетевых потоках узла агенту координации.

9. Агент координации передает информацию о сетевых потоках агенту формирования сигнатуры.

10. Агент формирования сигнатуры передает агенту координации сформированную сигнатуру бота, которую можно использовать для их обнаружения.

11. Агент координации рассылает информацию о работе ботов агентам идентификации ботов.

12. Агенты обнаружения ботов анализируют трафик в своей сети, пробуя обнаружить полученные признаки работы бота. В случае удачной идентификации передают информацию о боте агенту координации, который направляет её интерфейсному агенту для дальнейшего принятия решения.

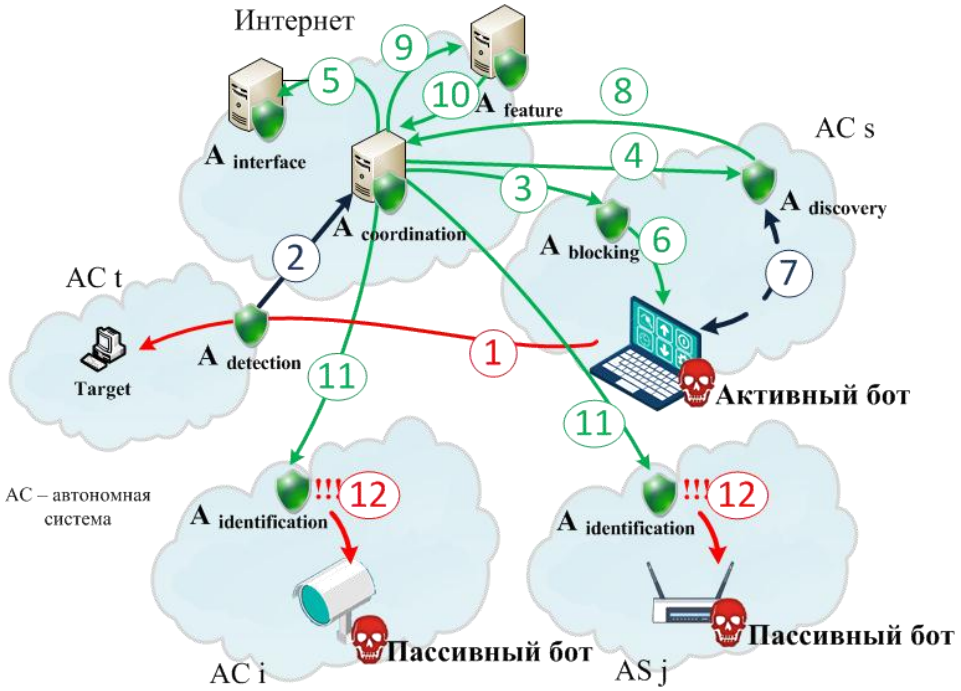

Рисунок 2.4 – Схема работы многоагентной системы обнаружения ботнетов

Описание модели агентов и их кооперации. В работе выделены следующие модели в соответствии с рисунком 2.5: модели всех упомянутых выше агентов, а также модель взаимодействия агентов. Модели агентов предназначены для представления процессов, решаемых агентами задач. Они включают базовые и специальные функции агентов, сценарии поведения агентов, протоколы взаимодействия.

Рисунок 2.5 – Представление основных моделей системы обнаружения ботнетов

Базовыми функциями агентов являются следующие функции: функции инициализации и окончания работы, мониторинг списка активных агентов, работа с модулями транспортного уровня (открытие/закрытие соединения, передача/приём сообщений). Для некоторых агентов предполагаются специализированные функции, основанные на базовых функциях. Для агентов обнаружения атаки их реализация будет зависеть от используемого метода обнаружения, для агентов выявления признаков бота – от используемых методов анализа деятельности бота, для агентов блокирования – от политики реагирования на атаку, для агентов обнаружения – от метода анализа сетевого трафика.

Протоколы взаимодействия агентов представляются в виде последовательности команд с определенными параметрами. В диссертационной работе для обмена сообщениями между агентами используются протоколы TCP и UDP. Выбор протокола основывается на уровне затрачиваемых ресурсов коммуникации.

Агенты могут реализовывать различные сценарии поведения. В некоторых случаях, сценарии конкретных агентов зависят от политики безопасности принятой в системе обнаружения.

Общая модель агента. Существует множество моделей многоагентных систем. В каждой из моделей обращается внимание на одну из сторон системы.

Согласно В.Б. Тарасову [34], выделяют следующие модели многоагентных систем: модели, являющиеся развитием понятия алгебраической системы, по А.И. Мальцеву, «Искусственный рой» [1], модель, предложенная К. Цетнаровичем, основанная на идее трехступенчатого определения основных понятий. Наиболее адекватной для поставленной задачи является модель, основанная на понятии алгебраической системы, по А.И. Мальцеву. Данная модель удачна в связи со следующими аспектами:

– Открытость [8]. Возможность агентов интегрироваться в системы, совместно решающие сложные задачи.

– Позволяет разделить уровни описания отдельных агентов и многоагентной системы как целого.

– Ориентирована на описание конечного множества действий.

– Модель ориентирована на искусственных агентов.

Таким образом, МАС можно выразить следующим образом [34]: MAS= (A, ENV, INT, ORG), где A – множество агентов; ENV – среда, в которой находится данная МАС; INT – множество типов взаимодействий между агентами; ORG – множество базовых организационных структур, соответствующих конкретным функциям (ролям) агентов и установившимся отношениям между ними.

С целью описать введенное множество INT взаимодействий агентов между собой, а также агентов и окружающей среды ENV, используются три языка разного уровня. Данные языки имеют различные коммуникационные функции: язык взаимодействия с другими агентами (L2), язык локального планирования (L1), язык исполнительного уровня (L0). Это позволит создать многоуровневую архитектуру агента, что приведет к разделению функциональных возможностей агента на несколько иерархических уровней. Каждый уровень сотрудничает с остальными в порядке иерархии. Архитектура InteRRaP (объединение реактивного поведения и рационального планирования, Integration of Reactive behavior and Rational Planning) является примером именно такой архитектуры. Акт взаимодействия с использованием некоторого языка Lx обозначим через int(Lx).

Тогда INT = ({int(L2)}, {int(L0)}). Язык L1 необходим для осуществления планирования работы агента в рамках множества базовых организационных структур ORG. Вопросы кооперации агентов будут рассмотрены позже.

Отдельный агент, в рамках выбранной модели может быть описан как четверка: $A_i = (E_i, I_i, ORG_i, C)$, где E_i – элементы коммуникационной среды, включая источники информации $(E_i \subseteq ENV)$; I_i – подмножество связей данного агента с другими $(I_i \subseteq INT)$; ORG_i – подмножество, содержащее функции агента, выполняемые в общей структуре многоагентной системы; C – внутренняя функциональная структура агента.

Внутреннюю функциональную структуру отдельного агента можно представить следующим образом $C = (K, F, I, G, B)$, где K – подсистема – ядро, отвечающее за динамическую реализацию ORG, F – подсистема, отвечающая за выполнение конкретных функций агента, I – подсистема, отвечающая за взаимодействия с источниками информации, G – подсистема, отвечающая за взаимодействие с другими агентами, B – база знаний агента. Метамодель агента представлена на рисунке 2.6.

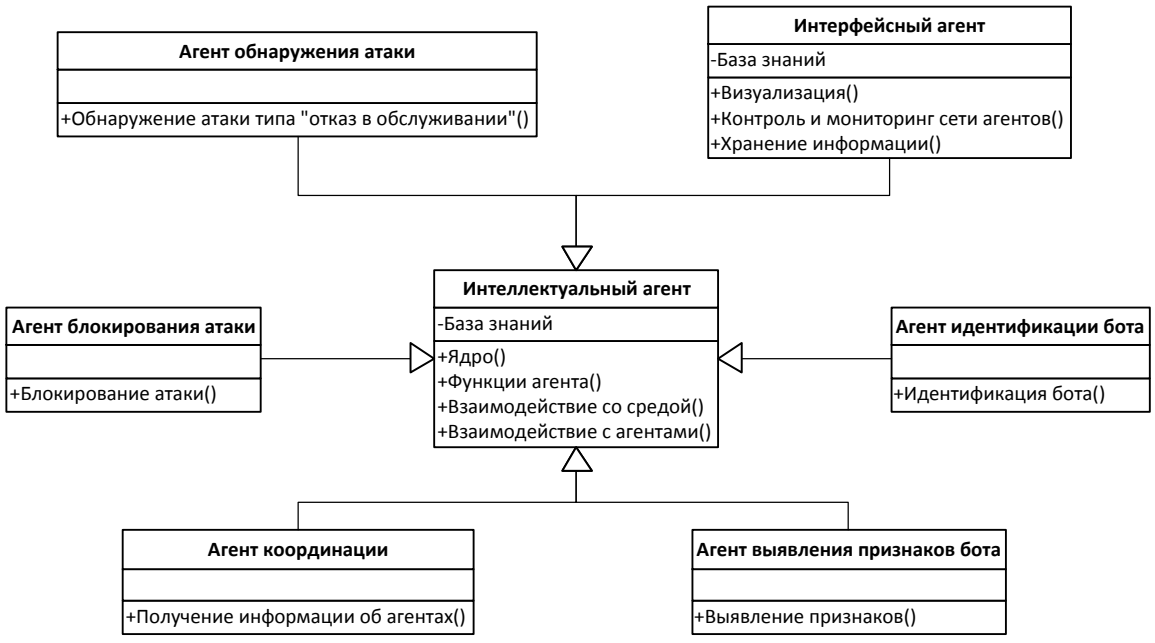

Рисунок 2.6 – Метамодель агента

Центральный блок метамодели описывает структуру базового агента, на основе которого будут строиться основные агенты системы. Дополнительные блоки описывают структуру основных агентов, отражая специальные функции, зависящие непосредственно от роли агента в системе.

2.2 Кооперация агентов

В основе описываемой архитектуры многоагентной системы лежит архитектура InterRRaP. Эта архитектура, в соответствии с концепцией смешанной архитектуры, заключающейся в реактивности и способности к планированию, состоит в представлении агента в виде многоуровневой сущности. Основной формой организации взаимодействия между агентами является кооперация. Смысл кооперации состоит в консолидации усилий агентов для достижения общей цели. При этом роли, функции и обязанности распределяются между агентами.

Описать кооперацию агентов между собой и с окружающей средой позволяет введение трех языков разного уровня. Данные языки имеют следующие коммуникационные функции [18]: кооперация с другими агентами (L2), локальное планирование (L1), исполнительные функции (L0). Базовая архитектура многоагентной системы представлена на рисунке 2.7.

Функциональное описание ролей. В многоагентной системе выделены следующие основные роли агентов:

- обнаружение распределенной атаки типа «отказ в обслуживании»;
- координатор;
- блокирование атаки;
- исследование трафика скомпрометированного узла;
- формирование сигнатуры;
- идентификация бота;

– интерфейс.

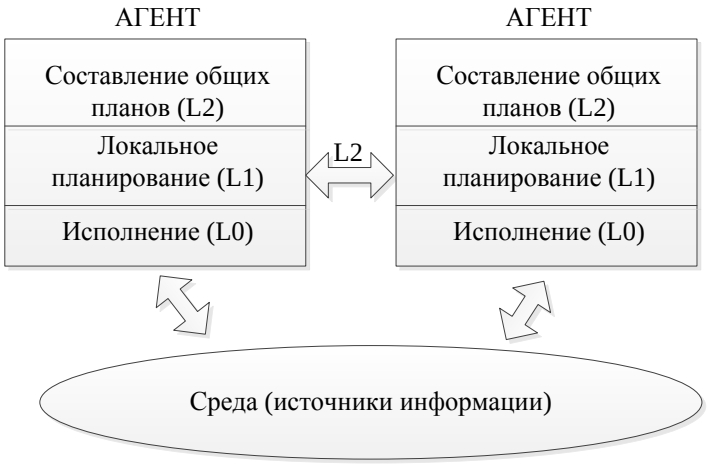

Рисунок 2.7 – Базовая архитектура многоагентной системы

Модель взаимодействий. Можно определить следующие взаимодействия между агентами:

– агент обнаружения распределенной атаки – агент-координатор;

– агент-координатор – агент блокирования атаки;

– агент-координатор – агент исследования трафика скомпрометированного узла;

– агент исследования трафика скомпрометированного узла – агент-координатор;

– агент-координатор – агент формирования сигнатуры;

– агент формирования сигнатуры – агент-координатор;

– агент-координатор – агент идентификации;

– агент идентификации – агент-координатор;

– агент-координатор – агент-интерфейс;

– агент-интерфейс – агент-координатор.

Диаграмма последовательностей взаимодействий при обнаружении ботнета представлена на рисунке 2.8.

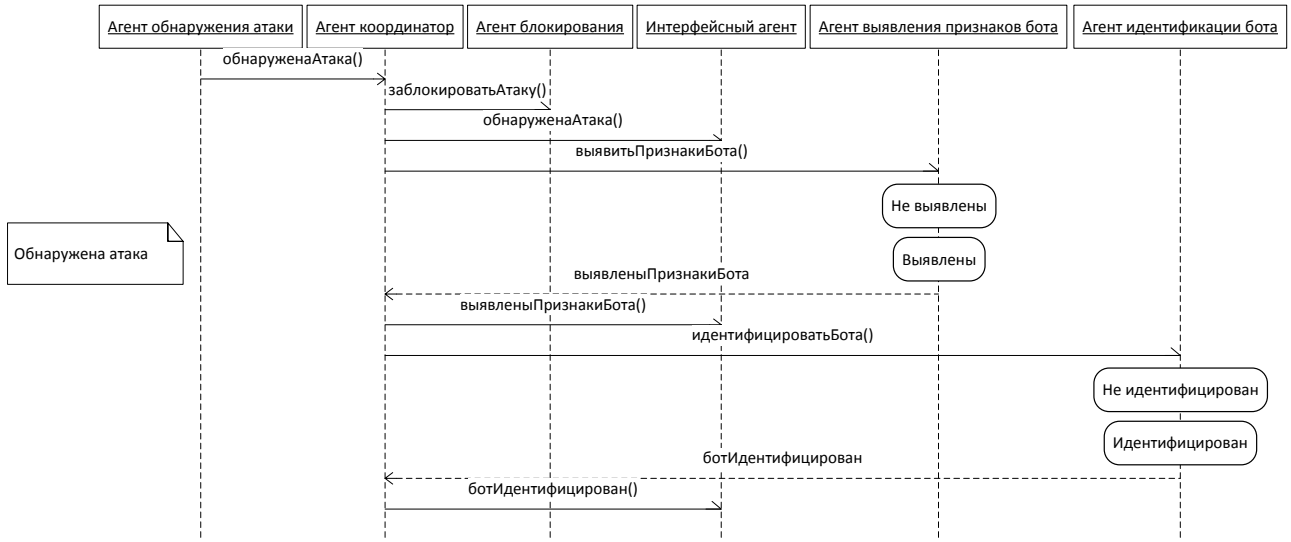

Рисунок 2.8 – Диаграмма последовательностей взаимодействий при обнаружении ботнета

Согласно В.Б. Тарасову [34], наиболее распространенными специальными моделями кооперации агентов в распределенном искусственном интеллекте являются:

1. модель аукционов;
2. протокол монотонных минимальных уступок [117];
3. модель договорных (контрактных) сетей Смита [119];
4. модель социальных зависимостей [56, 59].

Исходя из описания многоагентной системы обнаружения ботнетов, наиболее предпочтительной моделью кооперации агентов является модель договорных сетей Смита [119]. Данная модель обеспечивает координацию агентов в распределенных системах. Агент способен выполнять определенные задачи. В случае, если один агент затрудняется найти решение самостоятельно, в процессе решения определенной задачи, то он обращается за помощью к другим агентам. В этой ситуации агент, ищущий помощи, является заказчиком, а другие агенты — возможными исполнителями. Чаще всего заказчик ищет исполнителя не самостоятельно, а с помощью посредника, некоторого брокера. В качестве последнего выступает агент-менеджер. Агент-менеджер реализуется в качестве

мобильного агента, перемещаемого по сети. Остальные агенты, как правило, размещаются на своих сетевых узлах.

Стандартная модель Смита предполагает, что для выбора наиболее подходящего исполнителя заявки организуется конкурс между всеми откликнувшимися на запрос заказчика агентами. В работе предлагается, что выбор исполнителя будет осуществлять агент-менеджер по заданному критерию. Таким образом получится избежать основные недостатки данной модели: высокая загрузка каналов передачи, что особенно важно с учетом задачи применения модели, а также отсутствие продуманного метода выбора исполнителя.

В решаемой задаче агентом-менеджером является агент-координатор; агентом-заказчиком являются агент обнаружения распределенной атаки, агент выявления характеристик работы бота, агент-идентификатор, агент-интерфейс; агентом-подрядчиком могут являться агент блокирования атаки, агент выявления характеристик работы бота, агент-идентификатор, агент-интерфейс. Исходя из модели, процесс взаимодействия агентов можно описать набором INT= (A,RR,p,P), где A – множество агентов, RR – множество ролей агентов, p:A →RR – функция распределения ролей, P – общий протокол взаимодействия между агентами.

В данном случае RR = {«обнаружение распределенной атаки», «координатор», «блокирование атаки», «исследование трафика бота», «формирование сигнатуры», «обнаружение бота», «интерфейс»}. Функция распределения ролей определяется следующим образом:

$p\left(A^i_{detection}\right)$ = «обнаружение распределенной атаки»,

$p\left(A^i_{blocking}\right)$= «блокирование атаки»,

$p\left(A^i_{discovery}\right)$ = «исследование трафика бота»,

$p\left(A^i_{feature}\right)$ = «формирование сигнатуры»,

$p\left(A^i_{identification}\right)$ = «обнаружение бота»,

$p\left(A^i_{coordination}\right)$ = «координатор»,

$p\left(A^i_{interface}\right)$ = «интерфейс».

Протокол взаимодействия определяет набор правил и соглашений, управляющих взаимодействием. Данный протокол описывается специальным языком составления общих планов и взаимодействия агентов друг с другом – язык L2. Основные назначения языка L2 в рамках многоагентной системы – это:

– составление запросов агентов между собой;

– составление запросов к многоагентной системе;

– составление ответов на запросы.

Таким образом, язык L2 можно представить в виде совокупности компонент $\mathrm{L2} = (\mathrm{bnmar}, \mathrm{response}, \mathrm{shell})$.

Bnmar – базовые типы взаимодействий. Определяют конкретную задачу агента при работе в составе многоагентной системы. По сути, данные примитивы задают организационную структуру агента.

Response – результаты обработки запросов. Результат обработки должен формироваться в доступной форме, как для человека, так и для агента.

Shell – язык запросов оператора к многоагентной системе. Этот язык предназначен для осуществления оператором действий по мониторингу и контролю работы многоагентной системы.

Таким образом, можно выделить несколько режимов кооперативного взаимодействия агентов системы: режим независимого взаимодействия агентов, каждый агент работает независимо; режим управления для A_i , когда агент A_i отдает команду A_j ; режим подчинения для A_i , когда A_i исполняет поручения A_j .

2.3 Разработка алгоритма обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition

В данной работе разработан алгоритм обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition (BNMAR) на основе методов интеллектуального анализа данных. Разработанный алгоритм обнаруживает трафик ботнета независимо от используемого протокола или организационной структуры ботнета.

Чтобы создать общий подход обнаружения управляющего трафика, который сможет противостоять эволюции и изменениям методов управления ботнетами, необходимо изучить внутренние характеристики коммуникации ботнетов, которые получится использовать в алгоритмах и функциях обнаружения. В данном разделе определим понятие ботнета следующим образом: «скоординированная группа вредоносных объектов (ботов), которые контролируются с помощью бот-мастера посредством командно-контролирующего канала». Термин «вредоносные» означает, что эти боты используются для выполнения вредоносной деятельности. Например, в соответствии с данными собранными Дж. Чжуг и др. [146], около 53% вредоносной деятельности наблюдаемой в тысячах реальных ботнетов связаны со сканированием (распространением или DDOS), около 14% связано с загрузкой бинарных файлов (с целью обновления вредоносного ПО). Кроме того, большинство ботнетов, основанных на HTTP или P2P, используются для рассылки спама [77, 124]. Термин «контролируемые» означает, что боты для осуществления своей деятельности должны получить команды, связавшись с серверами C&C. Другими словами, должна быть связь между ботами и управляющими серверами, которые могут быть централизованными или распределенными. Термин «скоординированная группа» означает, что несколько (не менее двух) ботов в пределах одного ботнета выполняют аналогичные и связанные сетевые взаимодействия с управляющими серверами, а также

злонамеренные действия. Если же команды бот-мастера для каждого бота индивидуальны, то этот ботнет не функционирует в соответствии с нашим определением и выходит за рамки этой работы.

Алгоритм BNMAR в каждой точке функционирования контролирует коммуникации в разрезе «кто говорит кому». Агрегирует эту информацию с множества точек, анализирует путем сравнения кластеров сетевых потоков и, предполагая коммуникационную активность управляющего центра, находит шаблон скоординированной группы. Т.е. BNMAR проводит кластерный анализ похожих коммуникационных действия в управляющем трафике C&C и выполняет кросс-кластерную корреляцию для выявления кластеров, разделяющих модель общения. Благодаря этому можно динамически создавать сигнатуры для конкретного ботнета, используя его управляющий трафик.

Алгоритм реализуют два типа агентов: агент исследования трафика и агент формирования сигнатуры. Агенты исследования трафика наблюдают за трафиком скомпрометированных узлов и проводят его кластеризацию. Агент формирования сигнатур производит кросс-кластерную корреляцию между всеми кластерами, полученными от агентов исследования трафика всех скомпрометированных узлов. В результате определяется кластер трафика, который имеет схожий шаблон коммуникации, что позволяет генерировать сигнатуру. Структура работы агентов исследования трафика и формирования сигнатуры описана ниже в соответствии с рисунком 2.9.

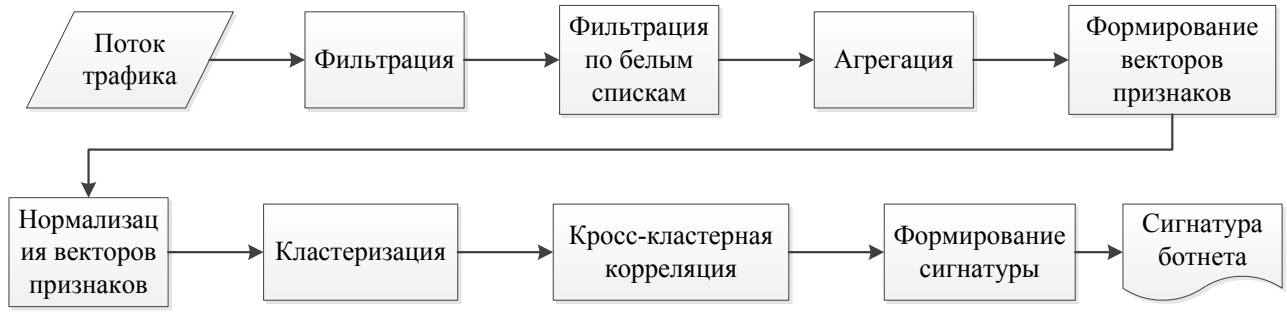

Рисунок 2.9 – Структура работы агентов исследования трафика и формирования сигнатуры

Прежде всего, необходимо отфильтровать ненужные потоки трафика. Это делается в два этапа: базовая фильтрация и фильтрация по белым спискам. Стоит отметить, что эти два этапа не являются критическими для нормального функционирования кластеризации. Эти этапы полезны для снижения нагрузки трафика, повышения эффективности процесса кластеризации и фильтрации легитимного пользовательского трафика. На этапе базовой фильтрации отбрасываются все потоки, которые не направлены от внутренних хостов к внешним хостам. Таким образом, алгоритм игнорирует потоки, связанные с сообщениями между внутренними узлами, а также потоки, инициированные внешними узлами по отношению к внутренним. Отфильтровываются также потоки, которые не находятся в состоянии «ESTABLISHED». В фильтрации по белым спискам отфильтровываются все потоки, направленные к хорошо известным легитимным серверам (т.к. Google, Yandex). Предполагается, что данные серверы вряд ли могут являться серверами C&C. Белый список основывается на рейтинге популярности сайтов от alexa.com и similarweb.com. При этом, помимо формирования общего белого списка, для каждой страны или региона используется в дополнение индивидуальный рейтинг ресурсов.

Следующим этапом после фильтрации является агрегация связных коммуникационных потоков с целью снижения нагрузки. В рамках временного интервала E (обычно несколько часов) все m TCP/UDP потоков, которые разделяют один и тот же протокол (TCP или UDP), адрес источника, адрес назначения и порт, объединяются в один коммуникационный поток $c_i = \{f_j\}_{j=1..m}$, где каждая f_j – это отдельный TCP/UDP поток. Множество $\{c_i\}_{i=1..n}$ объединяет все n коммуникационных потоков, наблюдаемых в интервале E, отражая коммуникацию наблюдаемого хоста.

Задача построения модели коммуникации состоит в выявлении коммуникационных потоков, которые являются общими для всех наблюдаемых узлов. Это может быть достигнуто путем кластеризации рассматриваемых потоков. Для того чтобы применить алгоритмы кластеризации для

коммутационных потоков, сначала необходимо представить потоки в подходящем векторном представлении. Для этого в алгоритме извлекается ряд статистических признаков из каждого коммутационного потока c_i и переводится в d-мерный вектор $p_i < R^d$. Можно описать эту задачу в качестве функции F : C → R^d. Функция F определяется следующим образом: учитывая коммуникационный поток c_i, вычисляется дискретное распределение четырех случайных величин:

– Количество потоков в час (flows per hour, fph). FPH вычисляется путем подсчета количества TCP/IP потоков в ci, которые присутствуют в каждом часе на интервале времени E;

– Количество пакетов в потоке (packets per flow, ppf). PPF вычисляется путем суммирования общего количества пакетов, пересылаемых в каждом TCP/UDP потоке c_i;

– Среднее число байт в пакетах (bytes per packets, bpp). Для каждого TCP/UDP потока f_j<c_i мы делим общее количество байт, переданных в f_j, на количество пакетов, отправленных в f_j;

– Среднее количество байт в секунду (bytes per second, bps). BPS вычисляется как общее количество байт, переданных в каждом потоке f_j<c_i, деленное на продолжительность f_j.

Пример результата этого процесса показан на рисунке 2.10, на котором отображены особенности посещений социальной сети «Вконтакте» пользователем, случайно выбранным из журнала реального сетевого потока.

С учетом дискретного распределения выборки каждой из этих четырех случайных величин вычисляется приблизительный вариант путем техники биннинга данных. Например, для того чтобы аппроксимировать распределение fph, разобьем ось x на 8 интервалов следующим образом: [0, k_1], (k_1, k_2], …, (k_7,∞). Значения k_1, …, k_7 вычисляются следующим образом. Сначала, вычисляется общее дискретное распределение выборки fph с учетом всех коммуникационных потоков трафика на интервале E. Затем вычисляются квантили $q_{10\%}$, $q_{20\%}$, $q_{30\%}$, $q_{40\%}$, $q_{50\%}$, $q_{70\%}$, $q_{90\%}$ полученного распределения и

устанавливаем k_1 = $q_{10\%}$, k_2 = $q_{20\%}$, k_3 = $q_{30\%}$, и т.д. Теперь для каждого коммуникационного потока можно описать его fph-распределение в качестве вектора из 8 элементов, где каждый элемент i представляет число раз fph принятых значений в пределах соответствующего интервала $(k_{i-1}, k_i]$. Этот алгоритм применяется также и для ppf, bpp, bps, и поэтому можно отрисовать каждый коммуникационный поток c_i в вектор-шаблон p_i из d = 32 элемента.

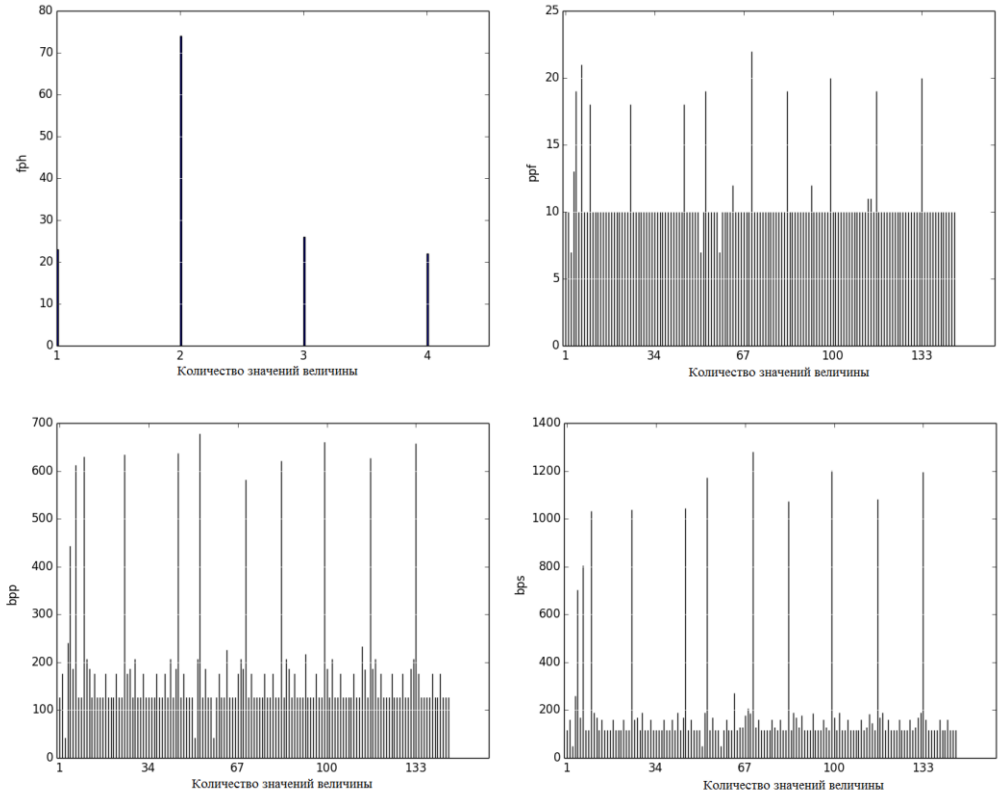

Рисунок 2.10 – Особенности посещений социальной сети «Вконтакте» пользователем, случайно выбранным из журнала реального сетевого потока

Масштабированный шаблон посещений, извлеченный из коммуникационного потока взаимодействий с социальной сетью «Вконтакте», показан на рисунке 2.11.

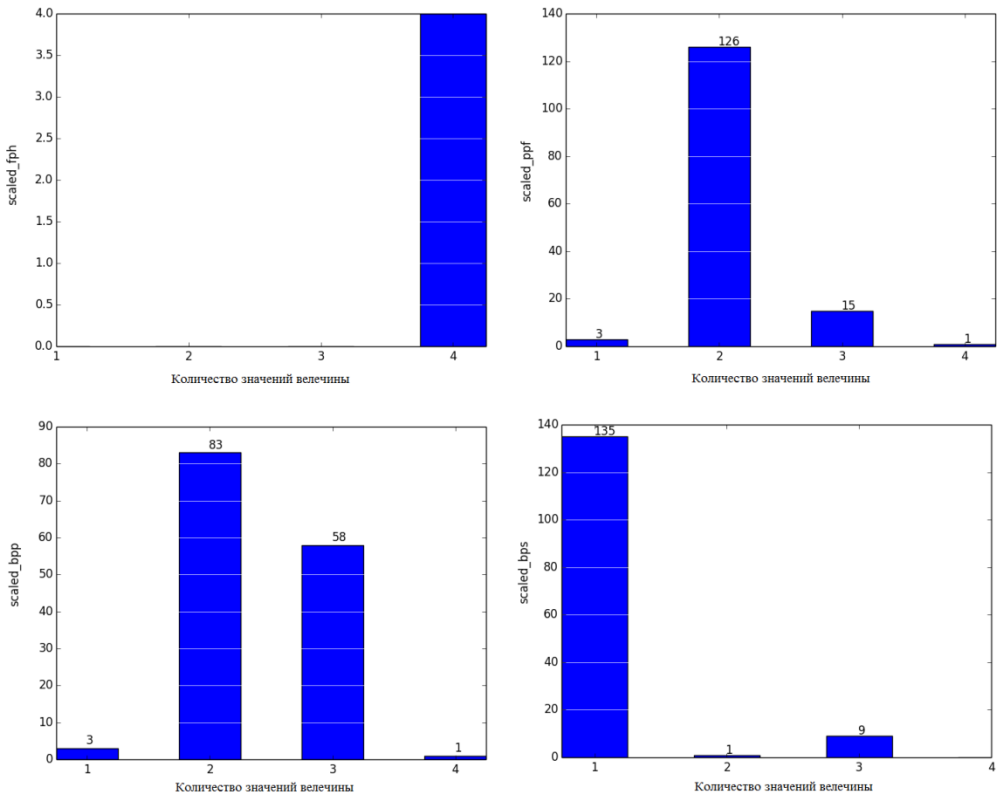

Рисунок 2.11 – Масштабированный шаблон посещений

Следующим этапом является кластеризация коммуникационных потоков. Цель заключается в поиске групп коммуникационных потоков, которые похожи друг на друга, тем самым выявляя потоки трафика к C&C серверам. Интуитивно, шаблоны векторов, которые находятся недалеко друг от друга в пространстве Rd, представляют коммуникационный поток с похожей коммуникационной структурой. Для примера предположим, что два бота, принадлежащих одному ботнету, подключены к двум различным C&C-серверам (некоторые ботнеты могут использовать множество серверов управления). Хотя соединения обоих ботов с серверами C&C будет находиться в разных коммуникационных потоках из-за различных пар источник/назначение, их характеристики трафика должны быть похожи. Т.е. в пространстве Rd эти коммуникационные потоки должны быть очень похожи. Для того чтобы найти управляющий трафик группы узлов, которые имеют похожие шаблоны связи, применяется техника кластеризации на наборе данных D векторного представления коммуникационных потоков. Кластеризация

выполняется техникой обучением без учителя. Как правило, эта техника нацелена на поиск значимых групп в данных относительно пространства признаков F. Определение «значимых кластеров» зависит от конкретной сферы применения. Цель заключается в группировании данных в кластеры, которые являются одновременно компактными и хорошо отделяемыми друг от друга, следовательно, подходящая метрика сходства определяется в пространстве признаков F [85].

Кластеризация коммуникационных потоков является сложной задачей, поскольку мощность D часто большая, даже для умеренно больших сетей, и размерность d-пространства признаков также большое. Кроме того, поскольку процент машин в сети, зараженных ботами, как правило, мал, то нам нужно отделить несколько ботнетов, связанных коммуникационным потоком, от большого количества доброкачественных коммуникационных потоков. Все это делает кластеризацию коммуникационных потоков достаточно накладной.

Кластеризация сетевых потоков осуществляется в два этапа, как показано на рисунке 2.12. Такой подход позволяет справиться со сложностью кластеризации.

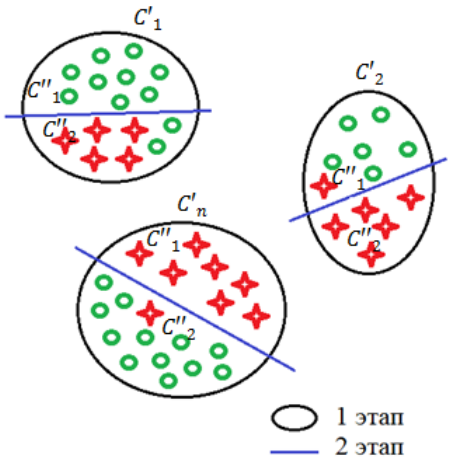

Рисунок 2.12 – Двухэтапная кластеризация

На первом этапе проводится первичная кластеризация в уменьшенном пространстве признаков $R^{d'}$, с d'<d, используя простой алгоритм кластеризации. Уменьшение размерности пространства признаков с d=32 признака до d'=8

признакам путем вычисления среднего и дисперсии распределений fph, ppf, bpp и bps для каждого коммуникационного потока. Затем применяется алгоритм кластеризации X-means на полученном представлении коммуникационного потока для нахождения крупных кластеров $\{C'_i\}_{i=1..y_1}$. Результатом первого шага кластеризации является множество $\{C'_i\}_{i=1..y_1}$, где y_1 – относительно большие кластеры. Таким образом, набор данных D разбивается на меньшие наборы данных (кластеры C'_i), содержащие набор точек, не очень отдаленных друг от друга. Поскольку кластеры $\{C'_i\}_{i=1..y_1}$ генерируются на первом шаге кластеризации, их количество не очень большое, теперь можно провести более трудоемкий анализ для каждого C'_i.

Для уточнения результата первого этапа кластеризации выполняется второй этап кластеризации на каждом отдельном наборе данных C'_i с помощью простого алгоритма кластеризации на полном описании коммуникационных потоков в R^d. Т.е. на втором шаге кластеризации используются все d=32 признака для представления коммуникационного потока. Второй шаг генерирует набор y_2 небольших и более точных кластеров $\{C''_i\}_{i=1..y_2}$.

Первый и второй этап кластеризации осуществлялись с использованием алгоритма XMeans [108]. XMeans – это эффективный алгоритм, основанный на очень популярном алгоритме кластеризации K-means [85]. В отличие от K-means, алгоритм XMeans не требует от пользователя заранее выбрать количество конечных кластеров. XMeans проходит несколько раундов K-means и выполняет эффективную валидацию кластеризации, используя информационный критерий Байеса [108], чтобы вычислить наилучшее значение количества кластеров. XMeans быстро и легко масштабируется по отношению к размеру набора данных [108].

Так как «обучение без учителя» является сложной задачей, результаты двухэтапного алгоритма кластеризации могут быть не идеальны. Как следствие, коммуникационные потоки, связанные с ботнетом, могут быть сгруппированы в несколько различных кластеров.

После того, как получены результаты кластеризации, алгоритм проводит кросс-кластерную корреляцию. Идея состоит в том, чтобы выделить кластеры с максимальным пересечением среди всех кластеров, полученных на скомпрометированных узлах. Т.е. эти кластеры должны быть наиболее близки друг к другу. В кластерном анализе для количественной оценки близости вводится понятие метрики. Сходство и различие между объектами устанавливаются в зависимости от значения пересечения двух кластеров. В качестве меры близости для решения этой задачи используется следующее выражение:

$$sim(C_i'', C_j'') = \max_{\substack{i,j=1,y_2 \\ i \neq j}} \left(\left| C_i'' \cap C_j'' \right| \right) \quad (2.1)$$

2.4 Метод распределенного обнаружения управляющих компонент ботнета, с которых осуществляется контроль атаки ботнета

В разделе «1.1 Обзор ботнетов и их жизненного цикла» говорилось, что существенным элементом создания ботнета является мотивация. Наиболее распространенной мотивацией для разработчиков ботнета является получение денежной прибыли. Монетизация ботнета может происходить по-разному, к примеру, сдача в аренду либо продажа услуг. В любом случае, в рамках совершения конкретной атаки можно выделить две основные роли злоумышленников: заказчик атаки и исполнитель атаки. Большинство существующих систем защиты от ботнетов фокусируются на обнаружении и блокировании управляющих серверов ботнетов и самих ботов, но это не решает основную проблему – идентификации злоумышленников, ведомых определенной мотивацией. Существующие защитные методы могут обнаружить и блокировать ботнет, но они не могут определить личность и местонахождение злоумышленников. Проблема ботнетов не будет окончательно разрешена, пока не будет разработан метод обнаружения злоумышленников через Интернет. В

результате, осознавая свою безнаказанность, злоумышленники могут спокойно создавать другие ботнеты и управлять ими. Если риск быть пойманными увеличится, то часть злоумышленников может воздержаться от создания и эксплуатации ботнетов. Таким образом, даже несовершенный метод отслеживания злоумышленников может эффективно сдерживать их активность.

Задача отслеживания злоумышленника, использующего ботнет, который был обнаружен, является достаточно сложной. Злоумышленник бывает онлайн непродолжительное время: дать команду ботнету или проверить статус ботов. Поэтому отслеживание должно происходить в режиме реального времени. Более того, злоумышленники часто не подключаются к управляющим серверам на прямую. В дополнение они могут обезопасить себя, шифруя управляющий трафик. Например, ботнет Agobot имеет встроенную поддержку протоколов SSL/TLS. Таким образом, успешный метод отслеживания злоумышленника должен эффективно работать, используя небольшие объемы зашифрованного трафика. Поэтому ни один из существующих методов отслеживания злоумышленников не может эффективно выполнять свою задачу в режиме реального времени. Например, методам [137, 138, 145] необходим большой объем трафика, чтобы иметь возможность отслеживать зашифрованный трафик, проходящий через цепочку прокси-серверов. В течение сессии боты обмениваются малым количеством пакетов с бот-мастером. Из-за малого объема трафика указанные методы с трудом подходят для обнаружения бот-мастера.

В диссертационном исследовании не ставится задача определения личности и местонахождения злоумышленников. Тем не менее, в работе предлагается метод, позволяющий предположить, что множество конкретных ip-адресов являются адресами исполнителя или заказчика атаки. Существующие методы обнаружения бот-мастера основаны на анализе трафика либо зараженных узлов, либо на анализе трафика управляющего центра. В диссертационной работе предлагается кардинально другой подход, основанный на анализе трафика атакуемой и атакующей машин.

Одним из основных этапов выполнения распределенных атак, в том числе и атаки типа «отказ в обслуживании», является осуществление контроля выполнения атаки (диаграмма прецедентов процесса организации атаки, совершаемой с использованием ботнета, представлена на рисунке 2.13). В реализации этой функции заинтересованы все участники организации атаки. При осуществлении атаки типа «отказ в обслуживании» организаторы атаки должны на протяжении всей атаки убеждаться, что атакуемый сервис находится в нерабочем состоянии. С этой целью они используют различное программное обеспечение. С учетом того, что большая часть подобных атак совершается на веб-ресурсы, проверить доступность можно либо с использованием стандартных браузеров, либо с использованием различных утилит, основанных на протоколе ICMP. Данные утилиты (например, ping) генерируют ICMP запросы типа «echo request» и обрабатывает поступающие в ответ пакеты типа «echo reply». Это позволяет определять задержки и частоту потери пакетов путем вычисления времени между отправкой запроса и получением ответа. Таким образом, получается косвенно определять загруженность каналов передачи данных и доступность удаленного узла. В случае успешно осуществленной атаки удаленный узел будет недоступным.

Со стороны атакуемого узла данная активность будет заметна и выражена множеством ICMP-запросов типа «echo request», что дает основание подозревать источники этих запросов в организации атаки и возможность составить список адресов источников. Поскольку одной из задач многоагентной системы является задача обеспечения процесса проведения кибер-расследований, данный список адресов и дополнительно собираемая информация по этим адресам может помочь в дальнейшем в процессе идентификации злоумышленников и привлечении их к ответственности.

Рисунок 2.13 – Диаграмма прецедентов процесса организации атаки, совершаемой с использованием ботнета

Описанные в данном разделе функции реализуются в модуле отслеживания узлов, контролирующих атаку. Данный модуль является частью агента обнаружения атаки. Архитектура модуля представлена на рисунке 2.14.

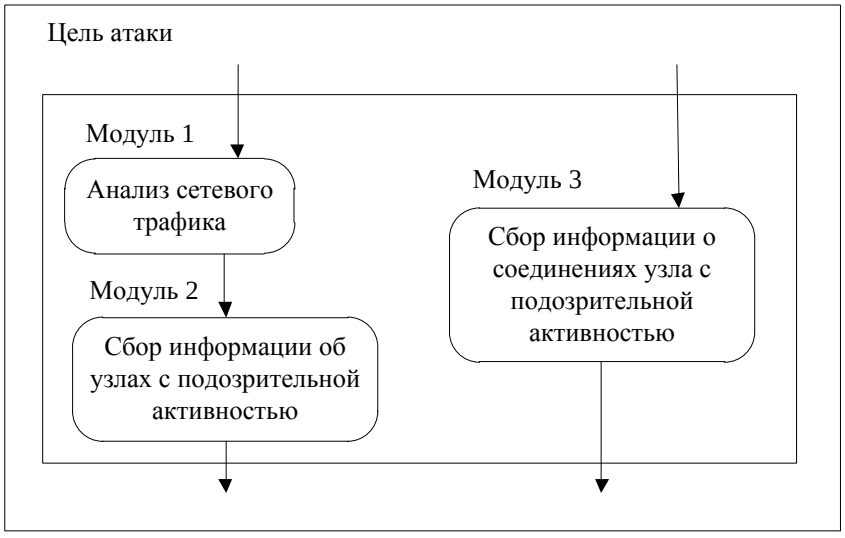

Рисунок 2.14 – Модуль отслеживания узлов, контролирующих атаку

Модуль имеет три подсистемы: анализ сетевого трафика, сбор информации об узлах с подозрительной активностью, сбор информации о соединениях узла с подозрительной активностью. Модуль активируется, когда происходит фиксация атаки агентом обнаружений атак. В процессе анализа сетевого трафика происходит подсчет всех входящих ICMP-запросов типа «echo-request» с уникальных ip-адресов, в результате получается таблица соответствия ip-адреса с количеством ICMP-запросов, пришедших с него. Данная таблица отсортирована в убывающем порядке количества ICMP-запросов. В соответствии с моделью организации атаки типа «отказ в обслуживании» в верхней части данной таблицы будут адреса узлов, с которых контролируют атаку. Эта таблица передается для обработки модулю сбора информации об узлах с подозрительной активностью. Задача этого модуля собрать разнообразную информацию о выявленных узлах, в том числе:

- тип и версия операционной системы;
- тип устройства;
- запущенные службы;
- порты;
- маршрут следования данных;
- DNS информация;
- регистрационные данные whois;
- географическое местоположение узла;
- уязвимости узла.

Подсистемы анализа сетевого трафика и сбора информации об узлах с подозрительной активностью работают на цели атаки. Подсистема сбора информации о соединениях узлов работает на агентах, расположенных максимально близко к обнаруженным узлам с подозрительной активностью. Она активируется на завершающем этапе работы всей многоагентной системы в рамках отдельной атаки, когда уже станут известны адреса управляющих серверов, что позволит отловить обращение бот-мастера к управляющему

серверу. В результате работы модуля отслеживания узлов, контролирующих атаку, вся информация собирается агентом мониторинга.

Заметим, предлагаемый метод имеет ограничение на использование только при распределенных атаках типа «отказ в обслуживании».

Выводы по второй главе

1. Приведена функциональная модель многоагентной системы обнаружения и блокирования ботнетов. Показано, что эффективным средством противодействия ботнетам может стать только система, аналогичная по сложности исполнения самим ботнетам. Описана структура и концептуальный алгоритм работы предлагаемой многоагентной системы.

2. Разработан алгоритм обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition на основе методов интеллектуального анализа данных. Разработанный алгоритм позволяет обнаружить трафик ботнета независимо от используемого протокола или организационной структуры ботнета.

3. Предложен метод распределенного обнаружения управляющих компонент ботнета, с которых осуществляется контроль атаки ботнета. Данный метод включает в себя сбор различной информации об адресах предполагаемых злоумышленников, такой, как тип и версия операционной системы, тип устройства, запущенные службы, порты, маршрут следования данных, DNS информация, регистрационные данные whois, географическое местоположение узла, уязвимости узла. Данная информация может иметь важное значение в процессе расследования инцидента.

ГЛАВА 3 ИССЛЕДОВАНИЕ ЭФФЕКТИВНОСТИ РАЗРАБОТАННЫХ АЛГОРИТМОВ

3.1 Метод автоматического формирования базы ботов

В рамках процесса разработки системы обнаружения ботнетов, основанном на многоагентном подходе, важным этапом является этап тестирования работоспособности и последующий анализ эффективности разработки. С учетом многоагентного подхода построения системы обнаружения процесс тестирования можно разбить на следующую последовательность действий:

- тестирование работоспособности каждого агента;
- тестирование разработанных алгоритмов;
- комплексное тестирование системы обнаружения.

Решение задач каждого этапа возможно на развернутом сетевом полигоне с работающим ботнетом. Таким образом, становится актуальной задача автоматического формирования базы ботов, действующих в составе существующих ботнетов.

Существуют различные способы получения экземпляров вредоносного программного обеспечения: проведение компьютерно-технической экспертизы, системы обнаружения атак, honeypot системы. Наиболее простым и популярным способом из перечисленных являются honeypot системы. Системы-приманки – это ресурсы информационной системы, предназначенные для несанкционированного или незаконного использования этого ресурса. Другими словами, это приманки, располагаемые в сети для привлечения злоумышленников. Honeypot чаще всего представляют собой виртуальные машины, предназначенные для имитации реальных машин, симулирующие или создающие видимость работы приложений и сервисов.

Приманка работает, обманывая нападающих, полагающих, что приманка – легитимный сервис. Нарушители нападают на систему, не зная, что за их

действиями тайно наблюдают и их регистрируют. Полная регистрация действий нарушителя дает возможность провести анализ атаки и воспользоваться им для организации защиты информационных систем. Анализ действий злоумышленников даёт возможность получить экземпляры вредоносного программного обеспечения.

С точки зрения назначения систем приманок, принято выделять производственные и исследовательские honeypot. Производственные honeypot-системы помогают обеспечивать состояние безопасности информационных систем организаций. Они используются для решения следующих основных задач:

– протоколирование;

– обнаружение;

– предупреждение.

Исследовательские honeypot-системы позволяют решить одну из самых важных проблем в области информационной безопасности – проблему нехватки информации. Исследовательские honeypot-системы предоставляют платформу для изучения различных угроз безопасности. Такие системы могут применяться:

– для сбора информации о средствах и техниках осуществления атак;

– для сбора информации об автоматизированных угрозах, в том числе об автоматически распространяемом вредоносном программном обеспечении;

– для предсказания новых вариантов атак, путем использования накопленных данных и моделирования;

– для улучшения понимания мотивации и организации злоумышленников.

По уровню взаимодействия honeypot-системы можно разделить на системы низкого уровня взаимодействия и системы высокого уровня взаимодействия.

Системы слабого взаимодействия работают, подражая определенной услуге, и ограничивают взаимодействие злоумышленника с системой. Действия злоумышленника ограничены уровнем эмуляции, предоставленной приманкой. Например, эмулируемый сервис SSH прослушивает определенный порт и эмулирует только процедуру авторизации. Злоумышленник никогда не пройдет

авторизацию такого сервиса, потому что другого функционала нет, при этом система будет регистрировать все используемые данные о логине и пароле. Примеры систем слабого уровня взаимодействия:

– Nepenthes – используется для автоматизации сбора вредоносных программ.

– Honeyd – используется для захвата действий атакующего.

– Honeytrap – инструмент для наблюдения за новыми атаками на сетевые сервисы.

Системы высокого уровня взаимодействия являются более сложными, так как они включают в себя реальные операционные системы и приложения. Например, сервис SSH будет уже полностью функционален, атакующий сможет зарегистрироваться в системе и развить атаку, проникая глубже в систему. В данном случае не накладывается никаких ограничений на взаимодействие злоумышленника с системой. Это позволяет администраторам захватывать полную информацию о методах атакующего в полном объеме. Примеры систем высокого уровня взаимодействия:

– Honeywall – используется для захвата, контроля и анализа атак. Создает архитектуру, которая позволяет разворачивать в ней системы как слабого, так и высокого уровня взаимодействия.

– High Interaction Honeypot Analysis Toolkit (HIHAT) – инструмент для преобразования произвольного PHP-приложения в веб-приманку высокого уровня взаимодействия.

– HoneyBow – инструмент сбора вредоносного программного обеспечения.

– Sebek – инструмент захвата действий атакующего.

В рамках данной работы было принято решение разработать исследовательскую honeypot-систему низкого взаимодействия, ориентированную на предоставление веб сервиса [19], тем самым получить систему, позволяющую автоматически формировать базу ботов, распространяемых через веб-приложения.

Чаще всего распространение ботов через веб-среду осуществляется автоматизированным способом. Запрограммированные боты автоматически сканируют пространство ip-адресов в поисках работающих серверов, предоставляющих веб-сервисы. После обнаружения веб-сервера боты начинают посылать запросы по заранее сгенерированной базе с надеждой, что на веб-сервере будет работать уязвимый для данных запросов веб-сервис. В случае, если веб-сервис оказывается уязвимым, на сервер загружается код бота, который в дальнейшем можно будет использовать для проведения атак.

Идея реализуемой honeypot-системы заключается в перенаправлении всех входящих запросов на веб-сервер к определенному скрипту, задача которого заключается в протоколировании всех поступающих запросов.

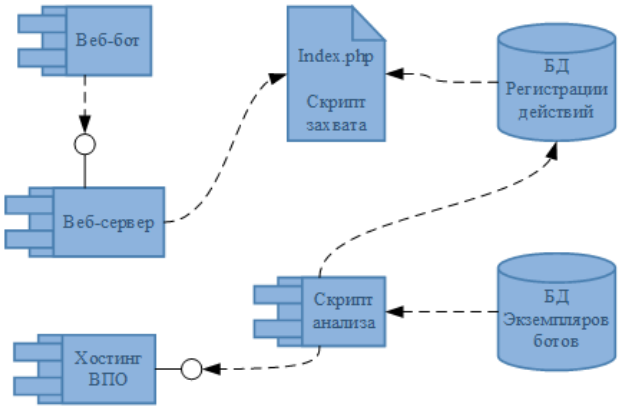

Рисунок 3.1 – Диаграмма компонент системы формирования базы ботов, распространяемых через веб-приложения

Далее скрипт анализа запротоколированных запросов выявляет попытки закачки на сервер посторонних скриптов по протоколу HTTP или FTP. При обнаружении такой попытки скрипт анализа самостоятельно закачивает экземпляры ботов и сохраняет их в базу данных. Таким образом, можно построить диаграмму компонент описанного процесса формирования базы ботов, распространяемых через веб-приложения, представленную на рисунке 3.1, а также диаграмму взаимодействий в рамках процесса, представленную на рисунке 3.2.

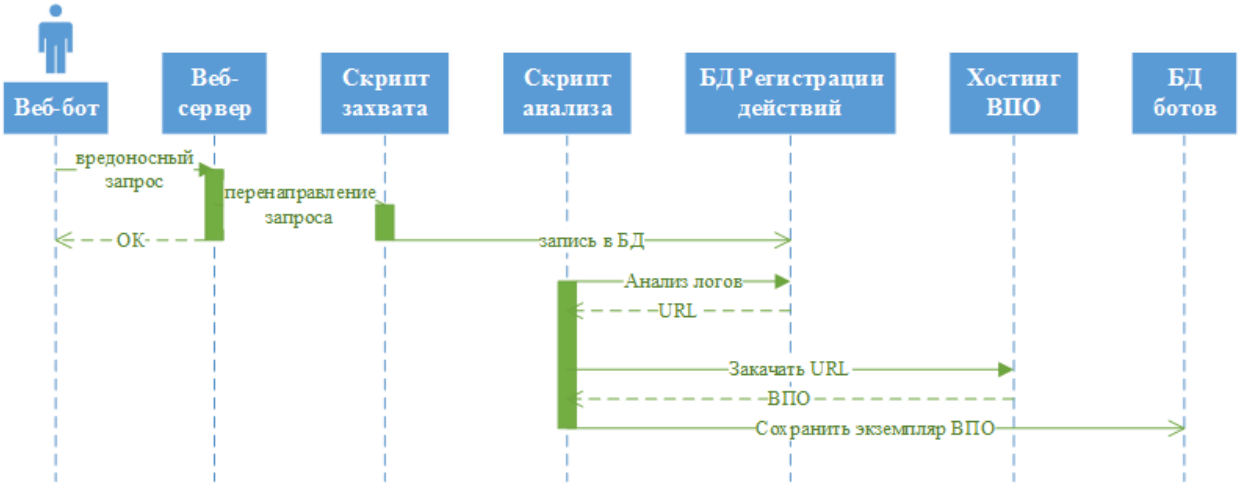

Рисунок 3.2 – Диаграмма взаимодействий системы формирования базы ботов, распространяемых через веб-приложения

Для проведения исследования веб-сервер был настроен на перенаправление всех входящих HTTP запросов на сценарий sniff.php (рисунок 3.3).

```php
<?php
    $loginfo['date']=date('c');
    $loginfo['env']=var_export($_ENV,true);
    $loginfo['get']=var_export($_GET,true);
    $loginfo['post']=var_export($_POST,true);
    file_put_contents('/tmp/log.txt',var_export($loginfo,true),FILE_APPEND);
?>
```

Рисунок 3.3 – Скрипт регистрации входящих запросов

Данный сценарий принимает HTTP запросы и регистрирует их в файле «/tmp/log.txt». Далее скрипт анализа выявляет попытки загрузки внешних скриптов. Пример попытки загрузки вредоносного программного обеспечения представлен на рисунке 3.4.

```
)array (
  'date' => '2015-03-16T09:13:54+04:00',
  'env' => 'array (
)',
  'get' => 'array (
)',
  'post' => 'array (
  \'action\' => \'lay_navigation\',
  \'eoltype\' => \'unix\',
  \'token\' => \'\',
  \'configuration\' => \'a:1:{i:0;O:10:"PMA_Config":1:{s:6:"source";s:29:"ftp://178.32.7.44/pub/124.php";}}\',
)',
```

Рисунок 3.4 – Пример ссылки на загрузку кода бота

По результатам работы системы было выявлено множество злонамеренных обращений к веб-серверу. Большинство проводимых атак было направлено на использование уязвимости CVE-2009-1151: phpMyAdmin '/scripts/setup.php' PHP Code Injection RCE PoC v0.11. Данная уязвимость позволяет атакующим ботам загрузить на веб-сервер PHP-код. В нашем случае атакующие пытались, используя эту уязвимость, загрузить на сервер код бота. Данные о количестве попыток загрузки вредоносного кода за сутки в период с 15 марта 2015 г. по 25 марта 2015 г представлены на рисунке 3.5. Всего же за время работы honeypot-системы было загружено 18 ботов со следующими именами: ball.php, weekend.php, 31.php, bot.php, Dervint.php, ok.php, kok.php, a.php, mala.php, 124.php, 43.php, 44.php, 45.php, newbot.php, pma.php, 0.php, 1.php, d.php. Данные о количестве загрузок каждого бота за период работы системы представлены на рисунке 3.6.

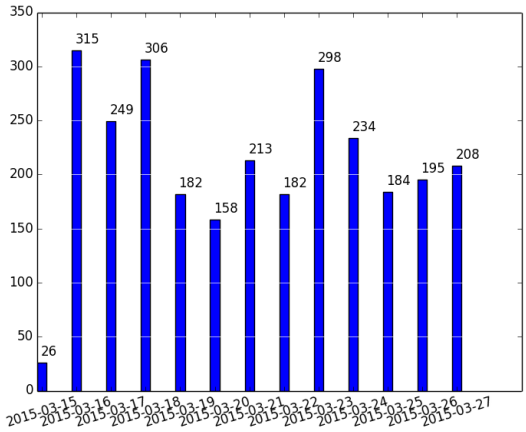

Рисунок 3.5 – Количество попыток загрузки вредоносного кода

Все полученные в рамках исследования боты ориентированы на взаимодействие посредством протокола IRC. Боты подключаются к заданным в коде IRC-серверам, регистрируются на приватных каналах и путем обмена сообщениями на этих каналах получают команды на исполнение и оповещают о результате своей работы. Практически все полученные боты обладают функционалом, перечисленным в таблице 3.1.

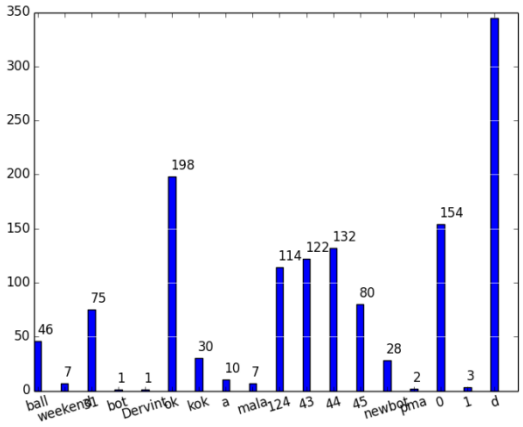

Рисунок 3.6 – Количество загрузок определенных ботов

Таблица 3.1 – Описание функциональности бота

Функция в коде бота	Описание
mail	отправка почтового сообщения
dns	получает доменное имя либо адрес хоста
uname	возвращает описание операционной системы, в которой работает
eval	исполняет переданный php код
exec	запускает переданную команду
download	осуществляет загрузку переданного файла
udpflood	осуществляет атаку типа наводнение UDP
tcpconn	осуществляет атаку типа наводнение TCP Connection
slowread	осуществляет атаку типа SlowRead
slowloris	осуществляет атаку типа SlowLoris
syn	осуществляет атаку типа наводнение SYN
httpflood	осуществляет атаку типа наводнение HTTP

3.2 Описание схемы проведения экспериментального исследования

Целью проведения экспериментального исследования являлась проверка работоспособности разработанного алгоритма обнаружения управляющего трафика ботнета BNMAR в реальной сети. Эксперимент проводился в кампусной сети Челябинского государственного университета. Это позволило создать условия, максимально близкие к реальности. В частности, боты функционировали

на реальных рабочих станциях, что обеспечивает наличие разнородного пользовательского трафика. Используемые в эксперименте агенты располагались на подконтрольных виртуальных серверах. Применялись два основных агента, участвующих в определении управляющего трафика бонета: агент исследования трафика скомпрометированных узлов и агент формирования сигнатуры. Информацию для обработки агенты исследования трафика получали по протоколу Netflow. Для этого коммутаторы в сети скомпрометированных узлов реализовывали функцию сенсора Netflow и передавали данные коллектору. Коллектор Netflow работает в качестве модуля агента исследования трафика. Для каждого скомпрометированного узла данные коллектором собираются отдельно друг от друга. Структура сети эксперимента представлена на рисунке 3.7.

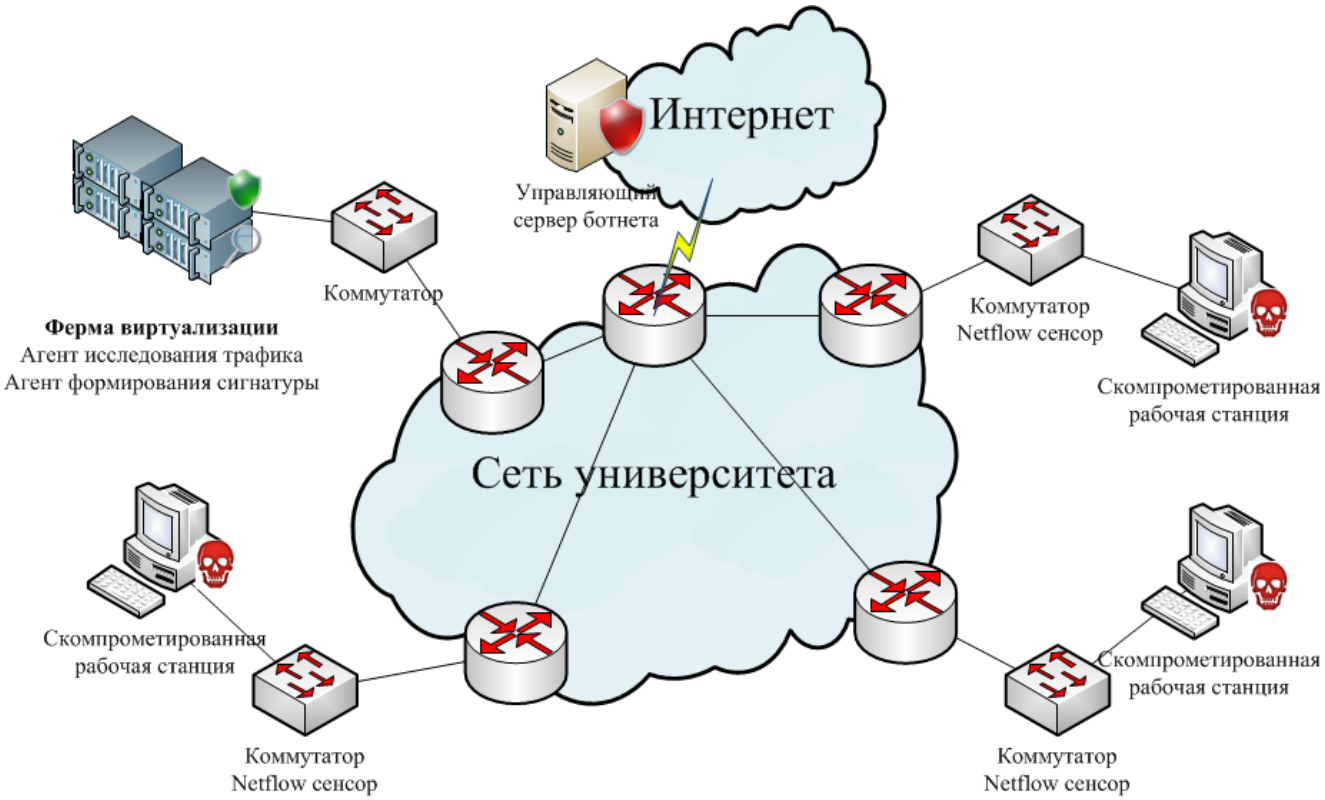

Рисунок 3.7 – Топология экспериментального стенда

В качестве бота использовался модифицированный код одного из бота, полученного с помощью системы, описанной в разделе 3.1. Модификация кода заключалась в ограничении злонамеренного функционала и настройке параметров

взаимодействия с управляющим сервером. В связи с отсутствием программного обеспечения управляющего сервера, оно было разработано с минимальным функционалом, позволяющим отслеживать статус ботов:

– отображение списка активных ботов;

– отображение по всем ботам информации о последнем времени в онлайн, сетевом адресе, операционной системы.

Методология проведения эксперимента состоит из нескольких шагов, представленных на рисунке 3.8. Подготовка физической и виртуальной сетевой инфраструктуры заключалась в настройке технологии Netflow на всех коммутаторах, к которым подключены скомпрометированные рабочие станции. Далее производились загрузка и запуск программного обеспечения бота на рабочих станциях. Скомпрометированные рабочие станции работали под управлением операционной системы Windows 7, 8 и имели следующие сетевые адреса: 10.1.18.91/24, 10.1.9.239/24, 10.1.15.25/24. На следующем этапе осуществлялся сбор данных сенсорами netflow в течение недели. Данные собирались агентом исследования трафика, который развернут в серверной ферме Института информационных технологий ЧелГУ. Данный агент работал на виртуальном сервере с характеристиками, приведенными в таблице 3.2.

Таблица 3.2 – Характеристики виртуального сервера

Характеристика	Значение
операционная система	CentOS 7
сетевой адрес	10.253.2.27/24
количество процессоров	2
модель процессоров	QEMU Virtual CPU version 1.5.3
частота процессоров, МГц	1995.192
объем оперативной памяти, Гбайт	15,5

Агент исследования данных, согласно алгоритму BNMAR, поэтапно выполнял свои функции. В результате его работы для каждого

скомпрометированного узла сформировались наборы кластеров, соответствующие трафику узла. Наборы этих кластеров передавались агенту формирования сигнатуры, одной из основных функций которого является сравнение кластеров из наборов с целью получить наиболее совпадающие кластеры. Совпадающие кластеры должны содержать данные об управляющем трафике ботнета.

Рисунок 3.8 – Методология проведения эксперимента

С целью выяснить эффективность разработанного алгоритма на заключительном этапе проводилась оценка результатов работы алгоритма. Задача оценки заключается в определении того, насколько точно и полно найденные кластеры соответствуют управляющему трафику ботнета. С учетом того, что в рамках экспериментального исследования имеется информация об управляющем сервере ботнета (сетевой адрес, порт, протокол взаимодействия), можно вычислить классические метрики оценки алгоритмов извлечения информации

(точность и полнота) и использовать их в качестве базиса для производной метрики – F-меры (сбалансированная F-мера, мера Ван Ризбергена) [113]. Метрика F-мера объединяет в себе информацию о точности и полноте алгоритма, т.е. приводит к балансу метрики точности и полноты, одновременно максимальные значения которых в реальной ситуации труднодостижимы.

Точность алгоритма в пределах кластера – это доля трафика, действительно принадлежащая управляющему трафику ботнета, относительно всего трафика, отнесенного к этому кластеру. Полнота алгоритма – это доля управляющего трафика ботнета в кластере относительно всего управляющего трафика ботнета, содержавшегося в собранных данных. Эти значения рассчитываются для каждого кластера на основании таблицы сопряженности, представленной в таблице 3.3 [143].

Таблица 3.3 – Таблица сопряжённости

		Экспертная оценка	
		Положительная	Отрицательная
Оценка системы	Положительная	TP	FP
	Отрицательная	FN	TN

В таблице сопряженности содержится информация о том, сколько раз система приняла верное и сколько раз неверное решение, где:

– TP (true positives) – истинно-положительные решения, т.е. решения, отнесенные к кластеру трафика ботнета и действительно им являющиеся;

– TN (true negatives) – истинно-отрицательные решения, т.е. решения, отнесенные к трафику отличному от трафика ботнета и действительно им являющиеся;

– FP (false positives) – ложно-положительные решения, т.е. решения, отнесенные к кластеру трафика ботнета, но им не являющиеся;

– FN (false negatives) – ложно-отрицательные решения, т.е. решения, отнесенные к трафику отличному от трафика ботнета, но являющиеся трафиком ботнета.

Точность кластеризации рассчитывается по следующей формуле:

$$PRECISION = \frac{TP}{TP+FP} \quad (3.1)$$

Полнота кластеризации рассчитывается по следующей формуле:

$$RECALL = \frac{TP}{TP+FN} \quad (3.2)$$

Метрика F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю. Рассчитывается данная метрика следующим образом:

$$F = 2\frac{PRECISION \times RECALL}{PRECISION+RECALL} \quad (3.3)$$

Метрика F-мера сводит к одному числу две других основополагающих метрики: точность и полноту. Что, в свою очередь, упрощает процесс принятия решения при оценке изменения алгоритма в лучшую или худшую сторону. Результаты экспериментов представлены в следующем разделе.

3.3 Описание результатов эксперимента

В данном разделе проводится оценка качества предложенного в работе алгоритма BNMAR. В таблицах 3.4 – 3.6 приведены статистические данные анализируемого трафика за 11 дней для всех трех наблюдаемых скомпрометированных узлов. В течение каждого дня первым, вторым и третьим узлом было передано порядка 900 тысяч, 700 тысяч и 400 тысяч пакетов (TCP, UDP) и 14000, 2400, 11000 потоков соответственно. Показано, что фильтрация эффективна с точки зрения уменьшения объема данных. Агрегирование также меняет конечные данные, в итоге мы получаем примерно 3900, 1500, 5400 агрегированных потоков в сутки. Естественно, количество анализируемых потоков прямо зависит от активности пользователей скомпрометированных узлов.

Таблица 3.4 – Статистические данные анализируемого трафика узла 10.1.18.91/24

День	До фильтрации		После фильтрации	
	потоки	пакеты	потоки	пакеты
1	12	114	3	44
2	11250	834213	2180	700298
3	13703	1209107	3511	1031395
4	17881	708615	4661	513133
5	15739	248548	3602	92848
6	13116	1000724	3507	854401
7	16162	515229	3726	289143
8	27166	957639	7863	682809
9	19562	994261	5717	791468
10	24994	3440192	7716	3016088
11	3874	70446	593	22366

Таблица 3.5 – Статистические данные дампа собранного трафика 10.1.9.239/24

День	До фильтрации		После фильтрации	
	потоки	пакеты	потоки	пакеты
1	1441	89234	843	49017
2	1378	16005	1254	13758
3	1385	15627	1241	13584
4	1404	18088	1250	13752
5	1483	21215	1296	16231
6	4337	170688	2395	115383
7	3853	64635	2215	39758
8	3085	51634	1539	29063
9	3281	104845	1336	58888
10	2550	151909	1310	61754
11	3274	60905	1907	44865

Таблица 3.6 – Статистические данные дампа собранного трафика 10.1.15.25/24

День	До фильтрации		После фильтрации	
	потоки	пакеты	потоки	пакеты
1	0	0	0	0
2	2586	244999	1353	218298
3	17180	1082525	8396	722751
4	10699	248616	4636	166777
5	11252	322304	5579	237188
6	13667	384838	5384	273958
7	16711	453499	9255	368590
8	8398	156354	4887	112934
9	13663	416410	6984	293183
10	15721	422758	6842	297342
11	11813	770181	6108	675914

Далее проводилась кластеризация анализируемого трафика. Одна из решаемых задач – это выбор наиболее подходящего алгоритма кластеризации потоков трафика. Для этого проводилось сравнение трёх алгоритмов кластеризации: алгоритм кластеризации плотностным методом (DBSCAN), алгоритм, дополняющий метод кластеризации K-means эффективной оценкой числа кластеров (XMeans), алгоритм максимального правдоподобия Expectation-maximization (EM). Данные алгоритмы самостоятельно определяют количество кластеров. Алгоритмы сравнивались с точки зрения способности генерировать кластеры таким образом, чтобы управляющий трафик ботнета распределялся в один или несколько кластеров, при этом данные кластеры содержали преимущественно управляющий трафик. Для оценки качества работы алгоритма обнаружения управляющего трафика использовалась метрика F-мера (F), являющаяся единой метрикой, объединяющей метрики точности (P) и полноты

(R). В качестве меры близости векторов признаков x и y размерности n использовалось Евклидово расстояние:

$$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}. \quad (3.4)$$

В результате эксперимента для каждого скомпрометированного узла был выделен набор кластеров. При этом эксперимент проводился для разных временных интервалов, используемых на этапе агрегирования потоков трафика. Результаты эксперимента представлены в таблице 3.7.

Таблица 3.7 – Показатели эффективности выделения управляющего трафика ботнета

Алгоритм	E	Узел 1			Узел 2			Узел 3		
		P	R	F	P	R	F	P	R	F
DBSCAN	2	0,0333	0,6154	0,0632	1,0000	0,7523	0,8586	0,0307	0,8511	0,0593
	4	0,0257	0,8400	0,0498	1,0000	0,7091	0,8298	0,0163	0,8571	0,0320
	6	0,0144	1,0000	0,0285	1,0000	0,6053	0,7541	0,0091	1,0000	0,0180
	8	0,0113	1,0000	0,0224	1,0000	0,6000	0,7500	0,0073	1,0000	0,0146
	10	0,0084	1,0000	0,0167	1,0000	0,5417	0,7027	0,0060	0,9444	0,0118
	24	0,0057	1,0000	0,0112	0,0194	0,6923	0,0378	0,0035	1,0000	0,0070
EM	2	0,0077	1,0000	0,0153	0,0635	1,0000	0,1196	0,0068	1,0000	0,0136
	4	0,0056	1,0000	0,0111	0,0380	1,0000	0,0731	0,0048	1,0000	0,0096
	6	0,1724	0,8824	0,2885	1,0000	0,7105	0,8308	0,1284	1,0000	0,2275
	8	0,1493	0,6250	0,2410	1,0000	0,6667	0,8000	0,1520	1,0000	0,2639
	10	0,1268	0,6429	0,2118	1,0000	0,6667	0,8000	0,1196	0,6111	0,2000
	24	0,1525	0,8182	0,2571	1,0000	0,4615	0,6316	0,0828	1,0000	0,1529
XMEANS	2	0,8788	0,7436	0,8056	0,9811	0,9541	0,9674	0,0476	0,7234	0,0892
	4	1,0000	0,4000	0,5714	1,0000	0,9455	0,9720	0,1875	0,4286	0,2609
	6	0,1282	0,5882	0,2105	1,0000	0,8947	0,9444	0,6667	0,9474	0,7826
	8	1,0000	0,3750	0,5455	1,0000	0,6667	0,8000	0,3830	0,9474	0,5455
	10	0,4167	0,3571	0,3846	1,0000	0,7083	0,8293	0,2414	0,3889	0,2979
	24	0,0408	0,5455	0,0759	1,0000	0,6154	0,7619	0,2917	0,5833	0,3889

Результаты показывают, что наиболее эффективным является использование алгоритма XMeans. В отличие от алгоритма X-means, алгоритм DBSCAN определяет большую часть трафика бота, о чем говорят высокие показатели полноты, но не выделяет данный трафик в отдельный кластер, а совмещает его в один кластер с пользовательским трафиком. С точки зрения быстродействия алгоритмов, среднее время, затраченное на кластеризацию

алгоритмом DBSCAN, сравнимо со средним временем, затрачиваемым при работе XMeans. При кластеризации с использованием алгоритма EM затрачивается существенно большее время. Таким образом, дополнительным преимуществом алгоритма XMeans является его быстродействие.

На рисунках 3.9. – 3.11. наглядно показано разбиение алгоритмом XMeans трафика трех скомпрометированных узлов. Для узла 10.1.18.91/24 выделено 15 кластеров, управляющий трафик ботнета в основной массе попал в кластер с номером 15. Для узла 10.1.9.239/24 выделено 13 кластеров, четвертый содержит управляющий трафик ботнета. Для узла 10.1.15.25/24 выделено 16 кластеров, большая часть искомого трафика оказалась в кластере 13.

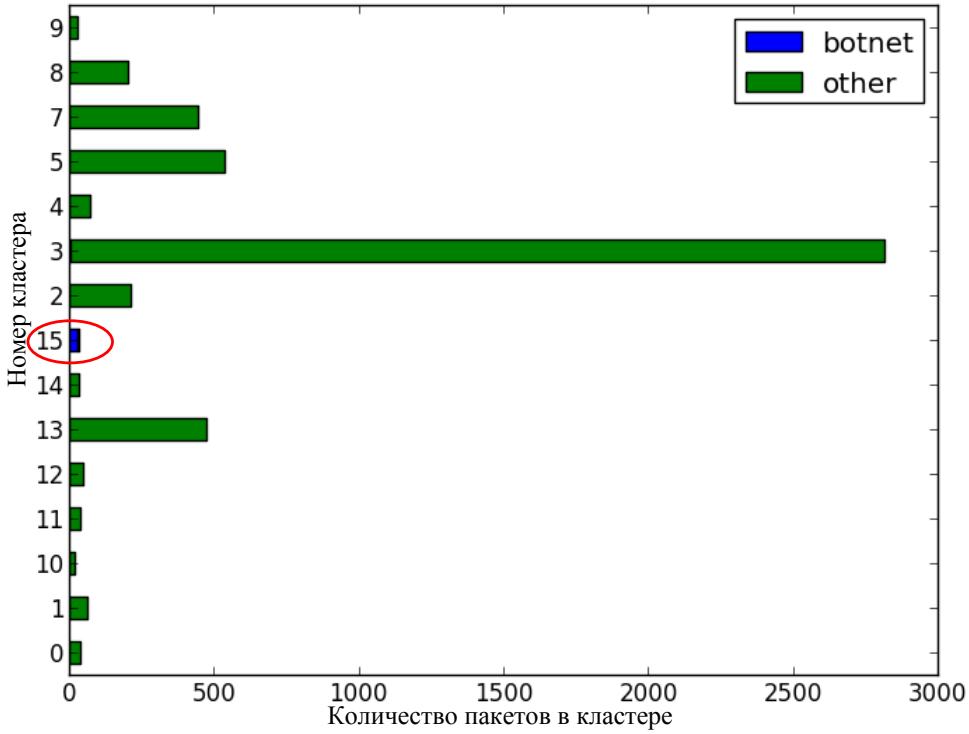

Рисунок 3.9 – Представление разбиения алгоритмом XMEANS трафика первого скомпрометированного узла

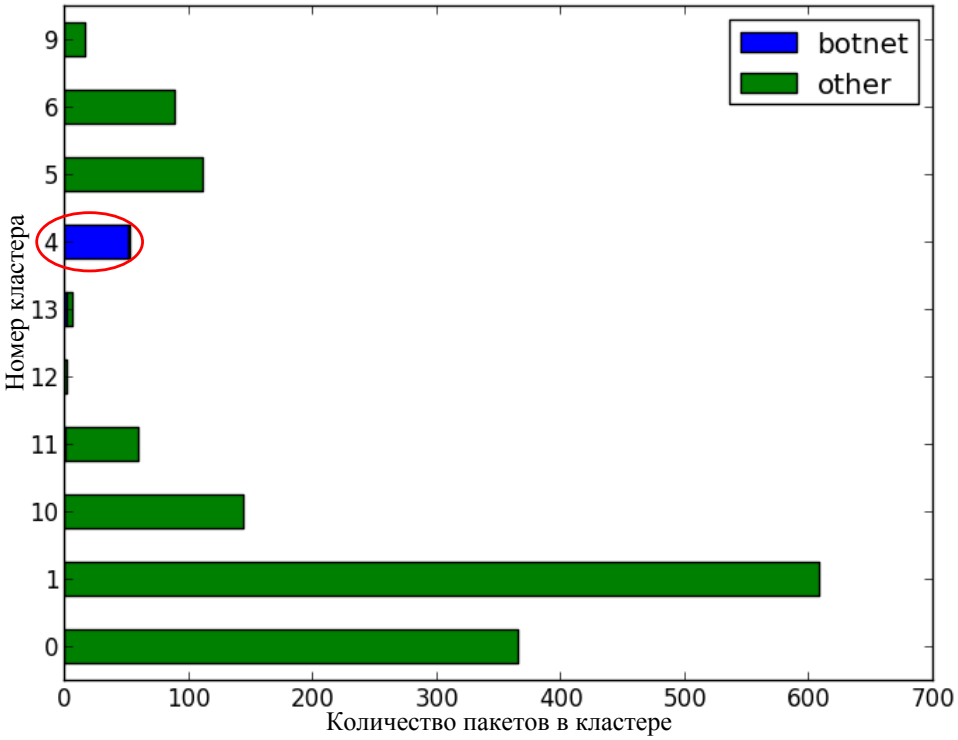

Рисунок 3.10 – Представление разбиения алгоритмом XMEANS трафика второго скомпрометированного узла

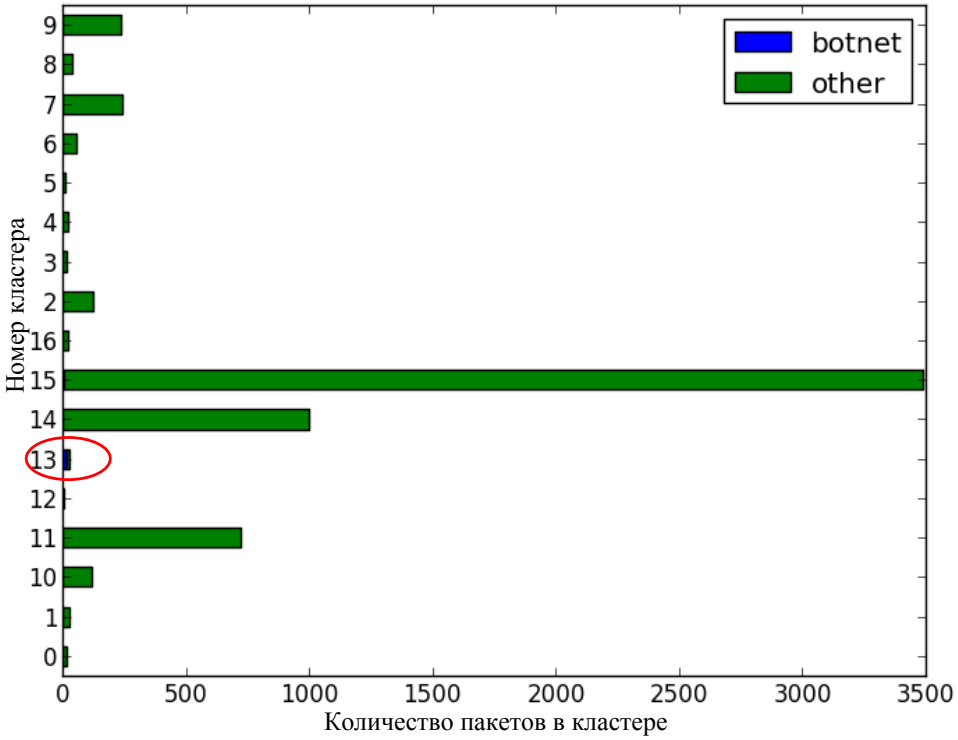

Рисунок 3.11 – Представление разбиения алгоритмом XMEANS трафика третьего скомпрометированного узла

Для улучшения показателей обнаружения управляющего трафика ботнета далее проводился второй этап кластеризации. Второй этап в качестве входных данных использует кластеры, образованные на первом этапе кластеризации, как было описано в главе 2. Полученные результаты отражены в таблицах 3.8 – 3.10.

Таблица 3.8 – Показатели эффективности для узла 10.1.18.91/24

Алгоритм	Интервал	Precision	Recall	F-measure
XMEANS	2	0,87	0,65	0,74
	4	0,63	1,00	0,77
	6	0,25	0,75	0,38
	8	0,50	0,75	0,60
	10	0,83	1,00	0,91
	24	0,67	0,67	0,67

Таблица 3.9 – Показатели эффективности для узла 10.1.9.239/24

Алгоритм	Интервал	Precision	Recall	F-measure
XMEANS	2	0,85	0,96	0,90
	4	0,25	0,50	0,33
	6	0,20	0,50	0,29
	8	0,17	1,00	0,29
	10	0,89	0,89	0,89
	24	0,75	1,00	0,86

Таблица 3.10 – Показатели эффективности для узла 10.1.15.25/24

Алгоритм	Интервал	Precision	Recall	F-measure
XMEANS	2	0,79	0,79	0,79
	4	0,93	0,93	0,93
	6	0,75	0,75	0,75
	8	0,88	0,58	0,70
	10	1,00	1,00	1,00
	24	0,33	0,50	0,40

Таблица 3.11 – Общие показатели эффективности

F-мера 10.1.18.91	F-мера 10.1.9.239	F-мера 10.1.15.25	Средняя F-мера
0,74	0,90	0,79	0,81
0,77	0,33	0,93	0,68
0,38	0,29	0,75	0,47
0,60	0,29	0,70	0,53
0,91	0,89	1,00	0,93
0,67	0,86	0,40	0,64

Были рассчитаны показатели чувствительности и специфичности, которые представлены в таблице 3.12, а также доли ложно-положительных и ложно-отрицательных решений, представленных в таблицах 3.13, 3.14.

Таблица 3.12 – Показатели чувствительности и специфичности

		2	4	6	8	10	24
10.1.18.91	SEN	0,65	1,00	0,75	0,75	1	0,67
	SPC	0,97	0,99	0,9	0,99	0,96	0,95
	SEN+SPC	1,62	1,99	1,65	1,74	1,96	1,62
10.1.9.239	SEN	0,96	0,5	0,5	1	0,89	1
	SPC	0	0,57	0,97	0,95	0,99	0,98
	SEN+SPC	0,96	1,07	1,47	1,95	1,88	1,98
10.1.15.25	SEN	0,79	0,93	0,75	0,58	1	0,5
	SPC	0,99	0,97	0,99	0,95	1	0,75
	SEN+SPC	1,79	1,9	1,74	1,53	2	1,25
Среднее SEN+SPC		1,45	1,65	1,62	1,74	1,95	1,62

Таблица 3.13 – Доля ложно-положительных решений

Интервал	FPR			
	10.1.18.91	10.1.9.239	10.1.15.25	Среднее
2	0,03	1,00	0,01	0,35
4	0,01	0,43	0,03	0,16
6	0,10	0,03	0,01	0,05
8	0,01	0,05	0,05	0,03
10	0,04	0,01	0,00	0,02
24	0,05	0,02	0,25	0,11

Таблица 3.14 – Доля ложно-отрицательных решений

Интервал	FNR			
	10.1.18.91	10.1.9.239	10.1.15.25	Среднее
2	0,35	0,04	0,21	0,20
4	0,00	0,50	0,07	0,19
6	0,25	0,50	0,25	0,33
8	0,25	0,00	0,42	0,22
10	0,00	0,11	0,00	0,04
24	0,33	0,00	0,50	0,28

По результатам анализа второго этапа кластеризации можно сделать следующие выводы:

– подтверждается решение о выборе алгоритма кластеризации;

– в качестве временного интервала агрегирования алгоритма BNMAR можно выбрать 10-часовой интервал. В связи с этим, данный временной интервал и будет использован для реализации исследовательского прототипа;

– частота ошибки первого рода, характеризующая частоту классификации легитимного трафика как трафик ботнета, равняется 0,02.

Выводы по третьей главе

1. Разработан метод автоматического формирования базы ботов. В результате реализации метода была разработана honeypot-система низкого взаимодействия, ориентированная на предоставление веб-сервиса. Построена база экземпляров вредоносного программного обеспечения, необходимых для проведения экспериментов. Объем полученной базы составляет 18 ботов.

2. Описана методология проведения эксперимента. Собран экспериментальный стенд. Для проведения вычислительных экспериментов была проведена модификация кода бота, полученного посредством разработанной honeypot-системы. Бот был модифицирован в следующих функциях:

- использование протокола HTTP в качестве управляющего канала;
- ограничен злонамеренный функционал;
- функции взаимодействия с управляющим сервером.

3. Проведены вычислительные эксперименты с целью оценки эффективности предложенного алгоритма обнаружения управляющего трафика ботнета.

4. В результате проведенных экспериментов для предлагаемого в работе алгоритма обнаружения управляющего трафика ботнета был выбран алгоритм кластеризации – XMEANS. Результаты данного алгоритма для всех скомпрометированных узлов оказались наивысшими. В сравнении с другими рассмотренными алгоритмами результат в среднем получился на 30% лучше. Кроме этого, данный алгоритм имеет наименьшее время выполнения. Выбран параметр алгоритма – временной интервал, равный 10 часам.

5. Показатели эффективности разработанного алгоритма получились следующими:

- среднее значение F-меры 0,93;
- доля ложноположительных решений 0,02;
- доля ложноотрицательных решений 0,04.

ГЛАВА 4 РАЗРАБОТКА ИССЛЕДОВАТЕЛЬСКОГО ПРОТОТИПА МНОГОАГЕНТНОЙ СИСТЕМЫ ОБНАРУЖЕНИЯ БОТНЕТОВ

В этой главе рассматривается фаза разработки исследовательского прототипа многоагентной системы обнаружения ботнетов. Согласно модели жизненного цикла программного обеспечения, в рамках фазы разработки можно выделить следующие этапы [36]:

- определение требований;
- спецификация;
- проектирование;
- тестирование.

В дальнейшем будем придерживаться перечисленных этапов для описания разработки исследовательского прототипа.

4.1 Формирование требований к многоагентной системе обнаружения ботнетов

Представленная в исследовании многоагентная система обнаружения ботнетов является комплексным решением, состоящим из программного обеспечения различного предназначения. Используемое программное обеспечение можно классифицировать следующим образом:

- Система предотвращения вторжений.
- Межсетевой экран.
- Система мониторинга сетевой безопасности.

Предлагаемая система по сути своей работы является системой обнаружения, таким образом, к данной системе можно предъявить требования, которым должны удовлетворять системы обнаружения атак. С этой целью были рассмотрены следующие нормативные документы:

– ГОСТ Р ИСО/МЭК 15408 Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий. Стандарт, устанавливающий основные понятия и принципы оценки безопасности ИТ [39], а также определяющий общую модель оценки, которой посвящены различные части стандарта, предназначенного для использования в качестве основы при оценке характеристик безопасности продуктов ИТ. ИСО/МЭК 15408 полезен в качестве руководства при разработке, оценке и/или приобретении продуктов ИТ с функциональными возможностями безопасности.

– Приказ ФСТЭК России №638 от 6 декабря 2011 г. «Об утверждении требований к системам обнаружения вторжений». Требования предназначены для разработчиков средств защиты, заявителей на сертификацию, а также испытательных лабораторий и органов по сертификации. Требования включают общие требования и требования к функциям безопасности систем обнаружения вторжений.

– Требования ФСБ России к средствам обнаружения компьютерных атак. 2012 г.

Руководящий документ ФСТЭК учитывает, что системы обнаружения вторжений бывают активными (системы предотвращения вторжений) и пассивными (системы обнаружения вторжения). В документе приведена также классификация систем обнаружения атак. В соответствии с общепринятой практикой [2], выделяются три типа систем обнаружения вторжений:

– по уровню обнаружения – это сетевые и узловые системы обнаружения вторжений;

– по используемым методам обнаружения – сигнатуры, поведение, профили;

– по месту расположения.

Как было отмечено ранее, многоагентная система обнаружения ботнетов является активной системой, а значит, является именно системой предотвращения вторжений и относится к системам обнаружения вторжений уровня сети.

Требования к системам обнаружения вторжений включают общие требования и требования к функциям безопасности. Некоторые функциональные требования к системе обнаружения и противодействия ботнет-атакам приводятся в работе Назарова А.Н. и др. [10]. Структура требований к системам обнаружения вторжений приведена на рисунке 4.1. С целью дифференцировать требования к функциям безопасности систем обнаружения вторжений установлено шесть классов защиты данных систем.

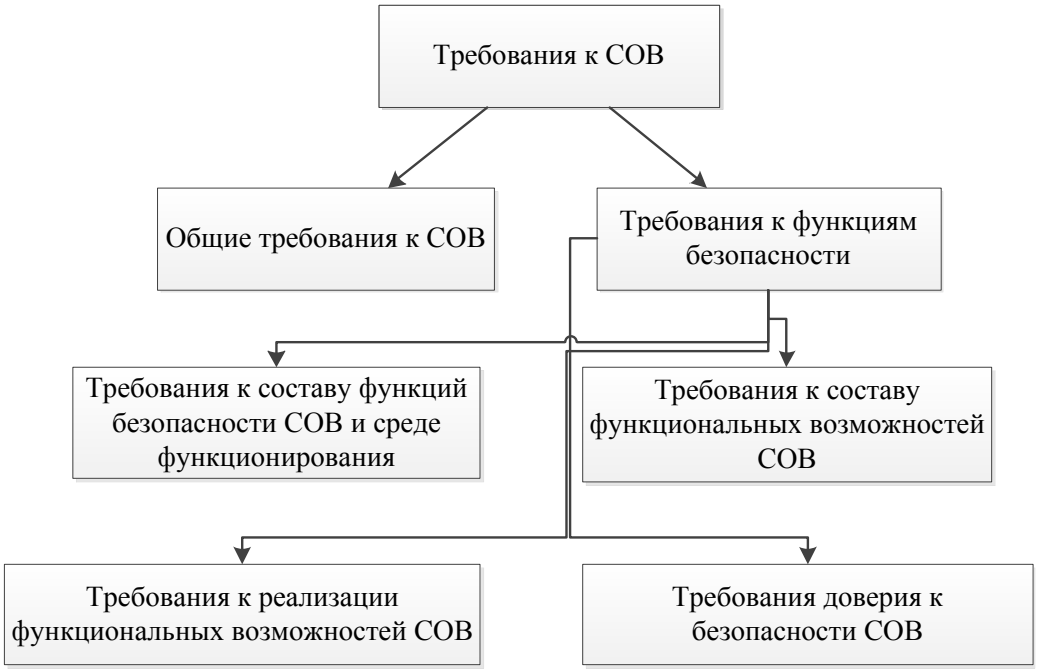

Рисунок 4.1 – Структура требований к СОВ

Первый класс является самым высоким, шестой класс – самым низким:

– Системы обнаружения вторжений, соответствующие 6 классу защиты, применяются в информационных системах персональных данных 3 и 4 классов.

– Системы обнаружения вторжений, соответствующие 5 классу защиты, применяются в информационных системах персональных данных 2 класса.

– Системы обнаружения вторжений, соответствующие 4 классу защиты, применяются в государственных информационных системах, в которых обрабатывается информация ограниченного доступа, не содержащая сведений, составляющих государственную тайну, в информационных системах персональных данных 1 класса, а также в информационных системах общего пользования II класса.

– Системы обнаружения вторжений, соответствующие 3, 2 и 1 классам защиты, применяются в информационных системах, в которых обрабатывается информация, содержащая сведения, составляющие государственную тайну.

Тип системы обнаружения вторжений и класс защиты определяет совокупность требований к функциям безопасности СОВ. Эти требования, установленные в соответствии с ГОСТ Р ИСО/МЭК 15408, делятся на 4 набора:

– Требования к составу функций безопасности СОВ и сред, в которых эти СОВ функционируют.

– Требования к составу функциональных возможностей СОВ, обеспечивающих реализацию функций СОВ.

– Требования к реализации функциональных возможностей СОВ.

– Требования доверия к безопасности СОВ.

Для СОВ уровня сети 4 класса выделяют десять функций безопасности, которые должны быть реализованы:

– Разграничение доступа к управлению СОВ.

– Управление работой СОВ.

– Управление параметрами СОВ.

– Управление установкой обновлений базы правил СОВ.

– Анализ данных СОВ.

– Аудит безопасности СОВ.

– Контроль целостности СОВ.

– Сбор данных о событиях и активности в контролируемой ИС.

– Реагирование СОВ.

– Маскирование СОВ.

В качестве функций безопасности среды функционирования выделяются:

– Обеспечение доверенного маршрута с администраторами СОВ (аутентификация и защищенный канал управления).

– Обеспечение доверенного канала обновлений базы правил.

– Обеспечение условий безопасного функционирования (в т.ч. регистрация событий внутри СОВ).

– Управление атрибутами безопасности.

Функциональные требования безопасности СОВ уровня сети 4 класса включают:

– требования по осуществлению сбора данных СОВ;

– требования к анализу данных СОВ;

– требования к реагированию СОВ;

– требования к средствам обновления базы решающих правил СОВ;

– требования по защите СОВ;

– требования по управлению режимами выполнения функций безопасности (работой СОВ);

– требования по управлению данными функций безопасности (данными СОВ);

– требования по управлению ролями субъектов;

– требования к средствам администрирования СОВ;

– требования к аудиту функционирования СОВ.

Функциональные требования безопасности для СОВ выражены на основе компонентов требований из ГОСТ Р ИСО/МЭК 15408-2, при этом часть требований сформулирована в явном виде в стиле компонентов из ГОСТ Р ИСО/МЭК 15408-2.

Состав функциональных требований безопасности (ФТБ) обеспечивает следующие функциональные возможности СОВ уровня сети 4 класса:

– возможность сбора информации о сетевом трафике;

– возможность выполнения анализа собранных данных СОВ о сетевом трафике в режиме, близком к реальному масштабу времени, и по результатам анализа фиксировать информацию о дате и времени, результате анализа, идентификаторе источника данных, протоколе, используемом для проведения вторжения;

– возможность выполнения анализа собранных данных с целью обнаружения вторжений с использованием сигнатурного и эвристических методов;

– возможность выполнения анализа собранных данных с целью обнаружения вторжений с использованием эвристических методов, основанных на методах выявления аномалий сетевого трафика на заданном уровне эвристического анализа;

– возможность обнаружения вторжений на основе анализа служебной информации протоколов сетевого уровня базовой эталонной модели взаимосвязи открытых систем;

– возможность фиксации факта обнаружения вторжений или нарушений безопасности в журналах аудита;

– уведомление администратора СОВ об обнаруженных вторжениях по отношению к контролируемым узлам ИС и нарушениях безопасности с помощью отображения соответствующего сообщения на консоли управления;

– возможность автоматизированного обновления базы решающих правил;

– возможность тестирования (самотестирования) функций безопасности СОВ;

– возможность со стороны уполномоченных администраторов (ролей) управлять режимом выполнения функций безопасности СОВ;

– возможность со стороны уполномоченных администраторов (ролей) управлять данными СОВ;

– поддержка определенных ролей для СОВ и их ассоциации с конкретными администраторами СОВ и пользователями ИС;

– возможность администрирования СОВ;

– возможность генерации записей аудита для событий, потенциально подвергаемых аудиту;

– возможность ассоциации каждого события аудита с идентификатором субъекта, его инициировавшего;

– возможность предоставлять возможность читать информацию из записей аудита;

– ограничение доступа к чтению записей аудита;

– поиск, сортировка, упорядочение данных аудита.

Требования доверия к безопасности СОВ охватывают вопросы управления конфигурацией, поставки и эксплуатации системы, разработки, руководства, поддержки жизненного цикла, тестирования, оценки уязвимостей и обновления базы решающих правил. Требования доверия к безопасности СОВ сформированы на основе компонентов требований из ГОСТ Р ИСО/МЭК 15408–3; при этом часть требований сформулирована в явном виде в стиле компонентов из ГОСТ Р ИСО/МЭК 15408–3 [29, 31, 38].

РД определяет состав стандартных и специальных функциональных компонентов СОВ, устанавливаемых в соответствии с ГОСТ Р ИСО/МЭК 15408-2, зависящих от типа СОВ и класса защиты. В данной части можно выделить следующие моменты. Функция автоматического блокирования атак устанавливается только для СОВ 2-го класса защиты и выше. На этом же уровне устанавливается требование удаленного управления. Графический интерфейс управления требуется, начиная с 3-го класса защиты. Для 1-го класса СОВ появляется требование звуковой сигнализации об обнаруженных вторжениях. Данное требование неуместно с учетом использования в глобальной распределенной сети. В этом случае на консоль СОВ может поступать до нескольких миллионов сигналов тревоги [28].

Согласно нормативным документам ФСТЭК России, основными компонентами системы обнаружения вторжений являются сенсоры и

анализаторы. Сенсоры собирают информацию о пакетах данных, передаваемых в пределах информационной системы (ИС), в которой установлены эти сенсоры. Сенсоры СОВ-уровня сети могут быть реализованы в виде программного обеспечения, устанавливаемого на стандартные программно-технические платформы, а также в виде программно-технических устройств, подключаемых к ИС. Анализаторы выполняют анализ собранной датчиками информации, генерируют отчеты по результатам анализа и управляют процессами реагирования на выявленные вторжения. Решение об обнаружении вторжения СОВ принимают в соответствии с результатами анализа информации, собираемой датчиками СОВ, с применением базы решающих правил СОВ [38].

4.2 Проектирование структуры исследовательского прототипа

Согласно архитектуре многоагентной системы обнаружения ботнетов, разрабатываемый исследовательский прототип должен состоять из семи функциональных подсистем, каждая их которых соответствует определенному агенту. Структура исследовательского прототипа многоагентной системы обнаружения ботнетов приведена на рисунке 2.3.

Каждый из агентов имеет модуль кооперации. Этот модуль необходим агентам для реализации взаимодействия друг с другом. Посредством этого модуля агенты обмениваются данными между собой и передают команды управления в случае поддержки таковых. Агент обнаружения атаки содержит модуль обнаружения атак, помогающий обнаружить атаку, сформировать список атакующих узлов и передать его для обработки другим агентам. Агент блокирования атаки, принимая список узлов, замеченных в проведении атаки, формирует запрещающее правило фильтрации трафика атаки, и применяет его в модуле блокирования атаки. Агент исследования трафика атакующей машины, используя модуль кластеризации, агрегирует весь трафик за определенный период и кластеризует его, после чего передаёт получившиеся кластеры агенту

формирования сигнатур. Агент формирования сигнатур с помощью модуля кросс-кластерной корреляции проводит кросс-кластерную корреляцию кластеров, полученных в результате анализа трафика всех узлов, замеченных в одной атаке, и генерирует сигнатуру для распознавания ботов посредством модуля формирования сигнатур. Агент обнаружения ботов состоит из одноименного модуля. Агент мониторинга состоит из следующих модулей: модуль обработки данных, модуль предоставления интерфейса управления и мониторинга и базы данных. И последний агент координирования, отвечающий за обеспечение кооперации агентов, состоит из модуля обмена сообщениями.

Разработка исследовательского прототипа с нуля не имеет смысла, так как задачи многих агентов решаются различными зарекомендовавшими себя средствами с открытым исходным кодом. Соответствия модулей агентов и классов систем приведены в таблице 4.1.

Таблица 4.1 – Соответствия модулей агентов и классов систем

Модуль агентов	Класс системы
Модуль обнаружения атаки	Система обнаружения атак
Модуль блокирования атаки	Межсетевой экран
Модуль обнаружения ботов	Система обнаружения ботов
Модуль обмена сообщениями	Связующее программное обеспечение
Модуль обработки данных	Диспетчер бинарных файлов событий СОВ
Модуль предоставления интерфейса управления и мониторинга	Система мониторинга безопасности сети

Таким образом, для большинства функций агентов можно использовать готовые системы с открытым исходным кодом. Разработки требуют модуль кооперации, модуль отслеживания злоумышленника, модуль кластеризации трафика, модуль кросс-кластерной корреляции и модуль формирования сигнатуры ботнета.

В качестве системы обнаружения атак и системы обнаружения ботов была выбрана система Suricata – это система обнаружения/предотвращения атак на основе правил, используемая для мониторинга сетевого трафика и уведомления системного администратора при возникновении подозрительных событий. Данная система распространяется под лицензией GPLv2 [43, 66].

Базовой операционной системой при разработке исследовательского прототипа являлась CentOS – дистрибутив Linux, основанный на коммерческом Red Hat Enterprise Linux компании Red Hat, лицензия GPL. Поэтому в качестве межсетевого экрана использовался Netfilter – межсетевой экран, встроенный в ядро Linux с версии 2.4. Netfilter управляется утилитой iptables.

В качестве связующего программного обеспечения использовалась платформа, реализующая систему обмена сообщениями между компонентами программной системы на основе стандарта AMQP (Advanced Message Queuing Protocol). RabbitMQ выпускается под лицензией Mozilla Public License [55].

Для приема файлов событий от агентов обнаружения атак и ботов использовался диспетчер бинарных файлов событий Barnyard2, распространяемый по лицензии GPLv2. Задача мониторинга и визуализации работы системы реализовывалась с помощью веб-системы мониторинга сетевой безопасности Snorby, распространяемой по лицензии GPLv3 [51]. Диаграмма развертывания системы приведена на рисунке 4.2.

В процессе разработки исследовательского прототипа использовались разные языки программирования. Для создания модулей агентов использовались языки программирования Perl и Python. Эти языки эффективны с точки зрения реализации алгоритмов, являются языками общего назначения, что позволяет создавать приложения смешанной архитектуры. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных. Используемые языки поддерживают наиболее известные и признанные парадигмы программирования. Для реализации алгоритмов интеллектуального анализа

данных использовался язык программирования R – язык программирования для статистической обработки данных и работы с графикой, а также свободная программная среда вычислений с открытым исходным кодом в рамках проекта GNU. R поддерживает широкий спектр статистических и численных методов и обладает хорошей расширяемостью с помощью пакетов. Пакеты представляют собой библиотеки для работы специфических функций или специальных областей применения. В базовую поставку R включен основной набор пакетов, а всего по состоянию на 2016 год доступно более 8131 пакетов [37]. Помимо прочего, для решения различных системных задач разрабатывались скрипты, обрабатываемые командным процессором unix систем – Bash.

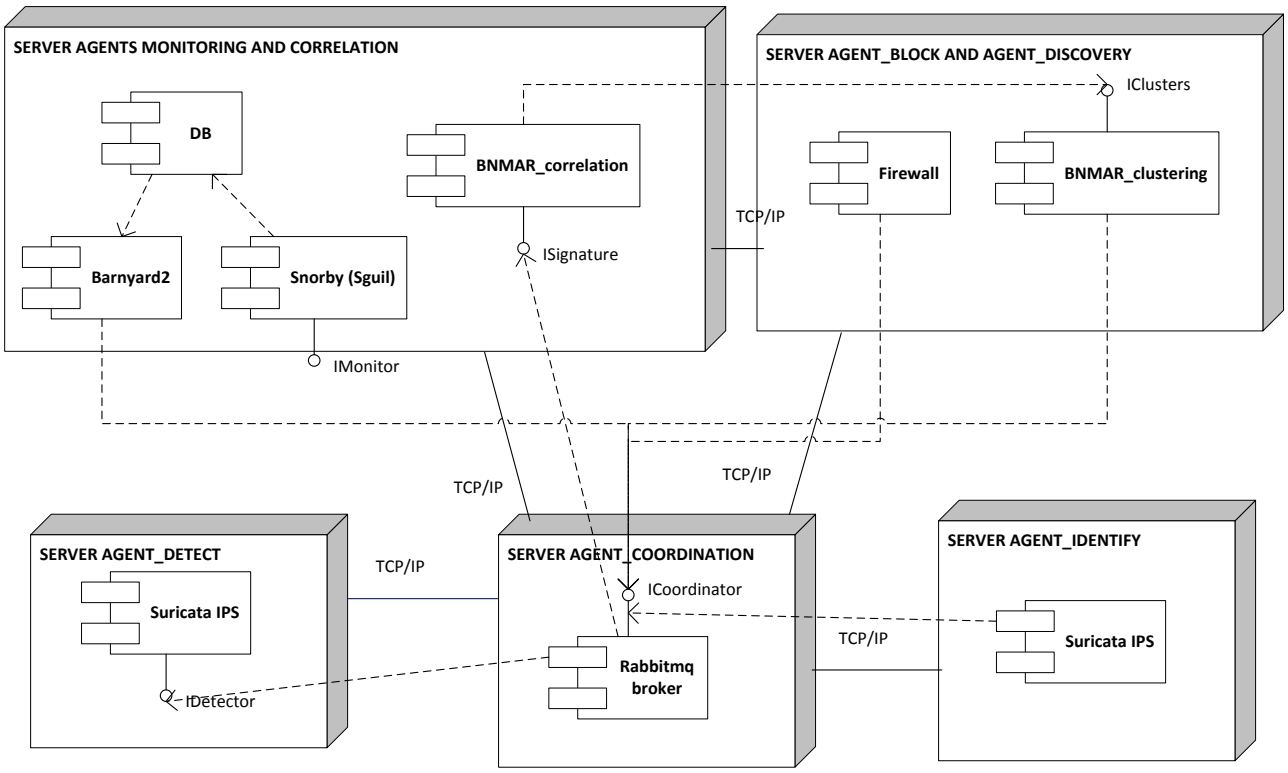

Рисунок 4.2 – Диаграмма развертывания системы обнаружения ботнетов

С целью агрегировать данные работы различных модулей системы была разработана база данных. Структура базы данных отражена в информационной модели данных, построенной по методике IDEF1X. IDEF1X является методом разработки реляционных баз данных и использует условный синтаксис для построения универсального представления структуры данных. Получаемое

представление не зависит от конечной реализации базы данных, программной или аппаратной платформы. Структура базы данных показана на рисунке 4.3.

Спроектированная база данных позволяет накапливать ключевую информацию о происходящих атаках в сети Интернет, включая информацию о ботнетах, их управляющих компонентах, таких как, управляющие серверы и узлы сети, с которых осуществляется контроль атаки, сигнатурах управляющего трафика. Накопленная информация имеет большое значение для обеспечения централизованного мониторинга кибер-угроз и процесса проведения расследований кибер-преступлений.

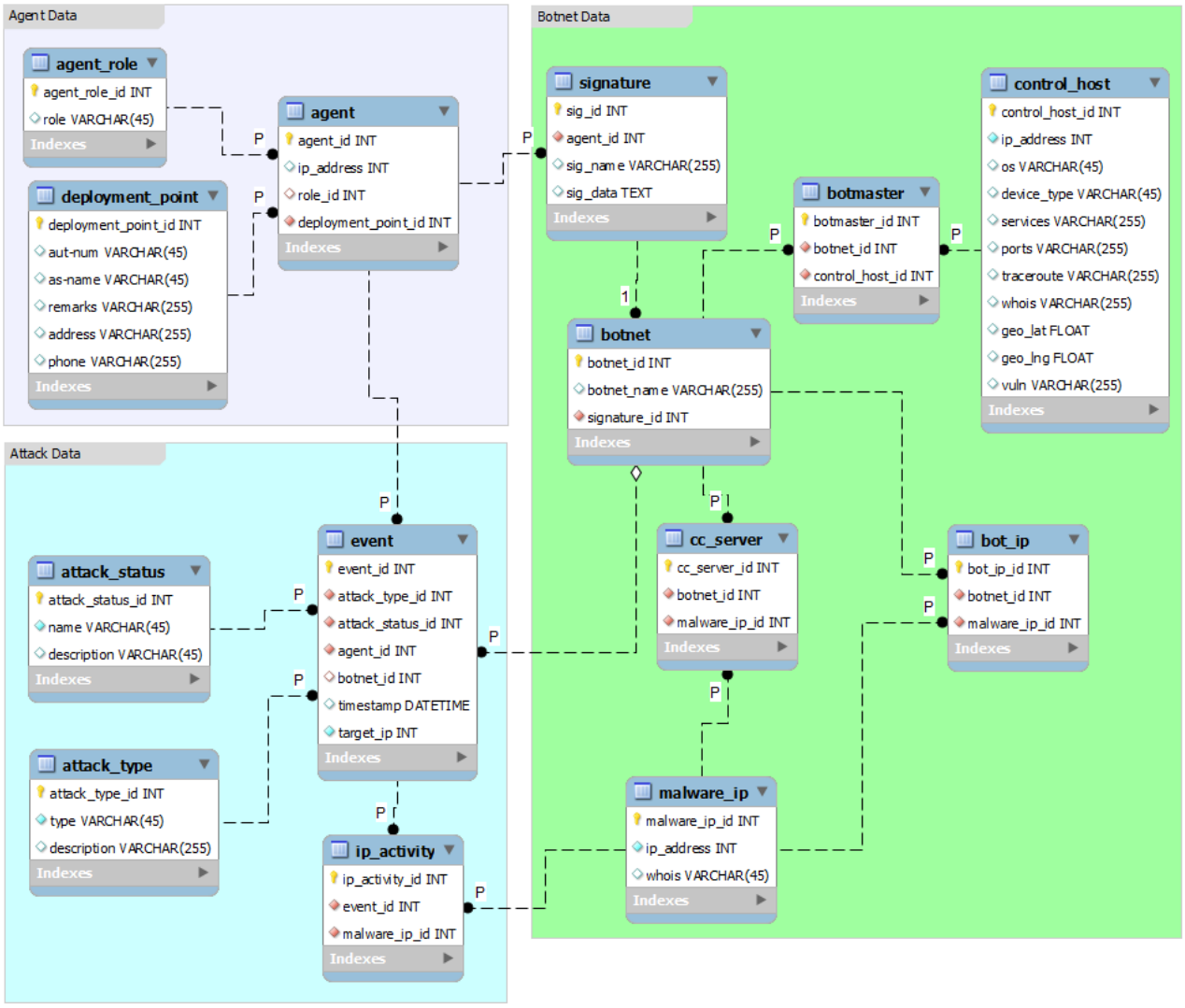

Рисунок 4.3 – Информационная модель данных системы

Информация хранится для трех основных объектов многоагентной системы обнаружения ботнетов: агент, событие безопасности, ботнет. Агенты системы обнаружения взаимосвязаны с событиями безопасностями, т.к. обнаруживают происходящие атаки, а также взаимосвязаны с ботнетами – формируют сигнатуру ботнета. Ботнеты путем своей зловредной деятельности инициируют события безопасности.

Для хранения информации об агентах используются соответствующие таблицы, структура которых представлена в таблицах 4.2, 4.3 и 4.4.

Таблица 4.2 – Структура таблицы агентов «agent»

Имя поля	Тип	Содержание поля
agent_id	INT (PRIMARY KEY)	Уникальный идентификатор агента
ip_address	INT	Сетевой адрес агента
role_id	INT (FOREIGN KEY)	Идентификатор роли агента
deployment_point_id	INT (FOREIGN KEY)	Идентификатор точки развертывания агента

Таблица 4.3 – Структура таблицы точек развертывания агентов «deployment_point»

Имя поля	Тип	Содержание поля
deployment_point_id	INT (PRIMARY KEY)	Уникальный идентификатор точки развертывания
aut-num	VARCHAR(45)	Номер AS
as-name	VARCHAR(45)	Имя AS
Remarks	VARCHAR(255)	Замечания
Address	VARCHAR(255)	Адрес точки развертывания
Phone	VARCHAR(255)	Телефон точки развертывания

Таблица 4.4 – Структура таблицы ролей агентов «agent_role»

Имя поля	Тип	Содержание поля
agent_role_id	INT (PRIMARY KEY)	Уникальный идентификатор роли агента
Role	VARCHAR(45)	Роль агента

В процессе работы системы агенты обнаружения атаки формируют событие безопасности, информация о котором в дальнейшем формируется путем работы остальных агентов, выявляющих управляющий трафик ботнетов и зараженных узлов. Для хранения информации о событиях безопасности используются соответствующие таблицы, структура которых представлена в таблицах 4.5, 4.6, 4.7 и 4.8.

Таблица 4.5 – Структура таблицы событий безопасности «event»

Имя поля	Тип	Содержание поля
event_id	INT (PRIMARY KEY)	Уникальный идентификатор события безопасности
attack_type_id	INT (FOREIGN KEY)	Идентификатор типа атаки
attack_status_id	INT (FOREIGN KEY)	Идентификатор статуса атаки
agent_id	INT (FOREIGN KEY)	Идентификатор агента, обнаружившего атаку
botnet_id	INT (FOREIGN KEY)	Идентификатор ботнета, реализующего атаку
Timestamp	DATETIME	Временная метка фиксации события
target_ip	INT	Сетевой адрес цели атаки

Таблица 4.6 – Структура таблицы типов атак «attack_type»

Имя поля	Тип	Содержание поля
attack_type_id	INT (PRIMARY KEY)	Уникальный идентификатор

		типа атаки
Type	VARCHAR(45)	Тип атаки
Description	VARCHAR(255)	Описание атаки

Таблица 4.7 – Структура таблицы статуса атаки «attack_status»

Имя поля	Тип	Содержание поля
attack_status_id	INT (PRIMARY KEY)	Уникальный идентификатор статуса события безопасности
Name	VARCHAR(45)	Статус события безопасности
Description	VARCHAR(45)	Описание статуса события безопасности

Таблица 4.8 – Структура таблицы сетевых адресов узлов, участвующих в определенном событии безопасности «ip_activity»

Имя поля	Тип	Содержание поля
ip_activity_id	INT (PRIMARY KEY)	Уникальный идентификатор сетевого адреса, участвующего в определенной атаке
event_id	INT (FOREIGN KEY)	Идентификатор события безопасности - атаки
malware_ip_id	INT (FOREIGN KEY)	Идентификатор сетевого адреса

В процессе работы агентов выявления управляющего трафика, агентов обнаружения зараженных узлов, в том числе модуля отслеживания злоумышленника, формируется информация об определенном ботнете, сигнатуре управляющего трафика, участниках и узлах управления. Для хранения

информации о ботнетах используются соответствующие таблицы, структура которых представлена в таблицах 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 и 4.15.

Таблица 4.9 – Структура таблицы сигнатур ботнетов «signature»

Имя поля	Тип	Содержание поля
sig_id	INT (PRIMARY KEY)	Уникальный идентификатор сигнатуры управляющего трафика ботнета
agent_id	INT (FOREIGN KEY)	Идентификатор агента, сформировавшего сигнатуру
sig_name	VARCHAR(255)	Наименование сигнатуры
sig_data	TEXT	Сигнатура

Таблица 4.10 – Структура таблицы ботнетов «botnet»

Имя поля	Тип	Содержание поля
botnet_id	INT (PRIMARY KEY)	Уникальный идентификатор ботнета
botnet_name	VARCHAR(255)	Наименование ботнета
signature_id	INT (FOREIGN KEY)	Идентификатор сигнатуры ботнета

Таблица 4.11 – Структура таблицы сетевых адресов злонамеренных узлов «malware_ip»

Имя поля	Тип	Содержание поля
malware_ip_id	INT (PRIMARY KEY)	Уникальный идентификатор сетевого адреса
ip_address	INT	Сетевой адрес
Whois	VARCHAR(45)	Информация whois о сетевом адресе

Таблица 4.12 – Структура таблицы управляющих серверов «cc_server»

Имя поля	Тип	Содержание поля
cc_server_id	INT (PRIMARY KEY)	Уникальный идентификатор управляющего сервера
botnet_id	INT (FOREIGN KEY)	Идентификатор ботнета
malware_ip_id	INT (FOREIGN KEY)	Идентификатор сетевого адреса

Таблица 4.13 – Структура таблицы сетевых адресов скомпрометированных узлов «bot_ip» принадлежащих определенному ботнету

Имя поля	Тип	Содержание поля
bot_ip_id	INT (PRIMARY KEY)	Уникальный идентификатор бота
botnet_id	INT (FOREIGN KEY)	Идентификатор ботнета, к которому принадлежит бот
malware_ip_id	INT (FOREIGN KEY)	Идентификатор сетевого адреса бота

Таблица 4.14 – Структура таблицы управляющих узлов «control_host»

Имя поля	Тип	Содержание поля
control_host_id	INT (PRIMARY KEY)	Уникальный идентификатор узла, управляющего ботнетом
ip_address	INT	Сетевой адрес узла, управляющего ботнетом
Os	VARCHAR(45)	Информация об операционной системе узла, управляющего ботнетом
device_type	VARCHAR(45)	Тип устройства узла, управляющего ботнетом

Продолжение таблицы 4.14

Имя поля	Тип	Содержание поля
Services	VARCHAR(255)	Запущенные службы узла, управляющего ботнетом
Ports	VARCHAR(255)	Порты узла управляющего ботнетом
Traceroute	VARCHAR(255)	Маршрут следования данных
Whois	VARCHAR(255)	Информация whois сетевого адреса узла, управляющего ботнетом
geo_lat	FLOAT	Географическое местоположение узла, управляющего ботнетом
geo_lng	FLOAT	Географическое местоположение узла, управляющего ботнетом
Vuln	VARCHAR(255)	Уязвимости узла, управляющего ботнетом

Таблица 4.15 – Структура таблицы сетевых адресов управления определенным ботнетом «botmaster»

Имя поля	Тип	Содержание поля
botmaster_id	INT (PRIMARY KEY)	Уникальный идентификатор ботмастера
botnet_id	INT (FOREIGN KEY)	Идентификатор ботнета
control_host_id	INT (FOREIGN KEY)	Идентификатор узла, управляющего ботнетом

Для моделирования физического развертывания агента исследования трафика использовалась диаграмма развертывания, приведенная на рисунке 4.4.

Были разработаны программные модули обеспечивающие функциональность агентов исследования трафика и формирования сигнатур.

Описание разработанных программных средств:

– Основной модуль. Отвечает за инициализацию алгоритма обнаружения управляющего трафика, согласование и координацию функциональных модулей. Выполняет функции кросс-кластерной корреляции и последующее формирование сигнатуры управляющего трафика (Bnmar.py);

– Модуль фильтрации трафика. Реализует описанную процедуру фильтрации данных о трафике. Входными данными является информация о потоках трафика, такая как время соединения, сетевые адреса, порты источника и назначения, протокол взаимодействия (Filtration.pl).

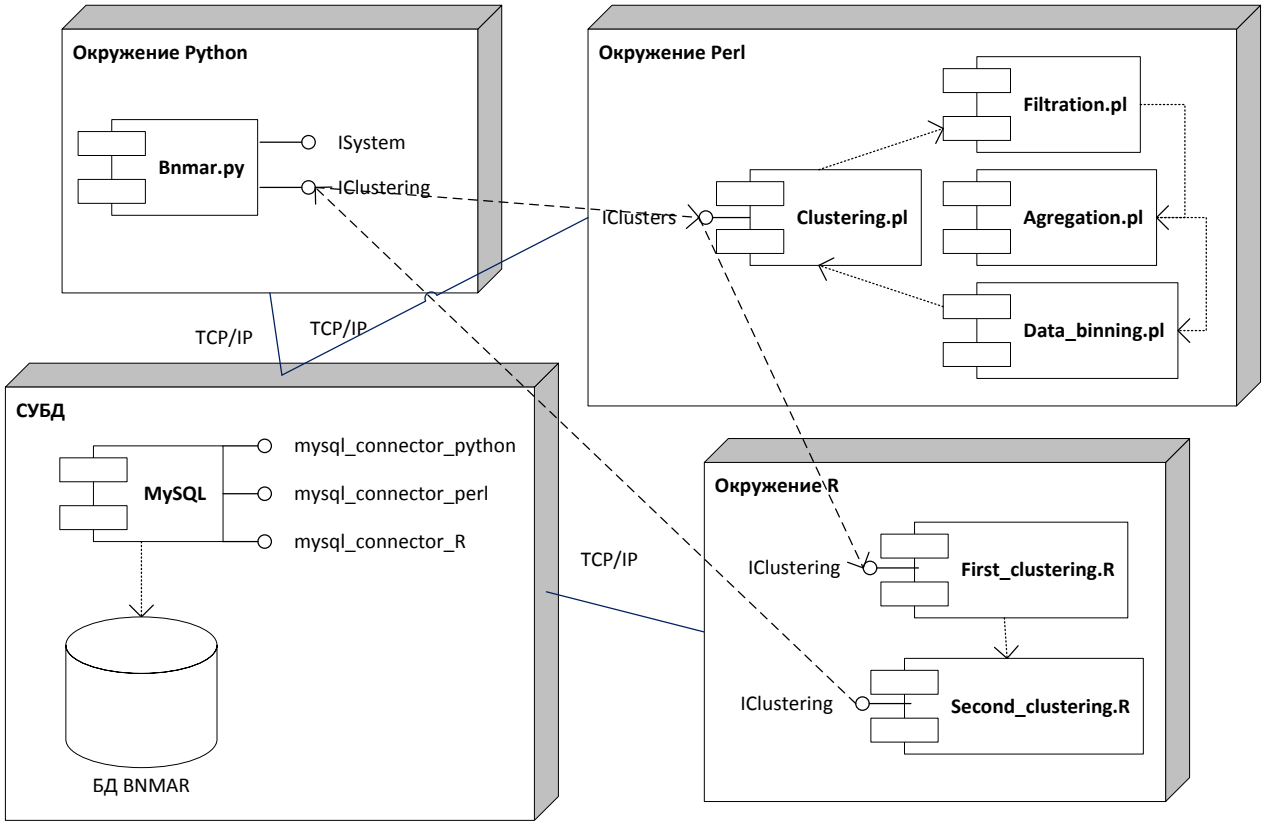

Рисунок 4.4 – Диаграмма развёртывания агента исследования трафика

– Модуль агрегации данных. Входными данными являются результат работы утилиты фильтрации и временной интервал, в рамках которого будет

проводиться агрегирование потоков. Результатом работы является файл с агрегированными схожими потоками (Aggregation.pl).

– Модуль векторизации агрегированных данных. Входными данными является файл агрегированных потоков и временной интервал, с которым проводилась агрегация потоков. Если временной интервал не задан, утилита будет ожидать ручной ввод интервала для векторизации критериев. Результатом работы является файл с векторизованными критериями (Data_binning.pl).

– Модуль контроля выполнения алгоритма BNMAR (Clustering.pl). Основными функциями являются: подготовка полученных данных к кластеризации, запуск процедуры двухэтапной кластеризации. Первый этап кластеризации выполняется с помощью скрипта First_clustering.R. Второй этап выполняется с помощью скрипта Second_clustering.R. Результатом работы является информация о распределении потоков трафика по кластерам.

– Модули, выполняющие кластеризацию трафика (First_clustering.R и Second_clusternig.R).

Настройка системы осуществляется путем настройки конфигурационных файлов агентов. Мониторинг событий безопасности и ботнетов осуществляется через веб-интерфейс и файлы журналов.

4.3 Тестирование работы системы

С целью проверки эффективности работы разработанного прототипа проведено тестирование в реальных условиях. Тестирование проводилось на базе инфраструктуры челябинского провайдера «Интерсвязь». В нескольких сегментах сети провайдера были развернуты агенты многоагентной системы обнаружения и блокирования ботнетов. Развертывание проводилось согласно описанной ранее диаграмме развертывания системы обнаружения ботнетов.

Структура сети при проведении тестирования многоагентной системы приведена на рисунке 4.4. Используемый ботнет состоял из

скомпрометированных компьютеров в полностью контролируемом сетевом сегменте. Создавалось два контролируемых сетевых сегмента. В одном контролируемом сегменте размещались тестируемые средства защиты, во втором сегменте средства защиты не устанавливались. Управляющий сервер развернут на хостинге в сети Интернет. Атакуемый компьютер находился в одном из сетевых сегментов провайдера и осуществлял функцию веб-сервера. В этом же сегменте была развернута точка многоагентной системы в составе следующих агентов: агент обнаружения атаки, агент блокирования атаки, агент исследования трафика, агент координации. Точка многоагентной системы была развернута в полностью контролируемом сетевом сегменте с полным составом агентов.

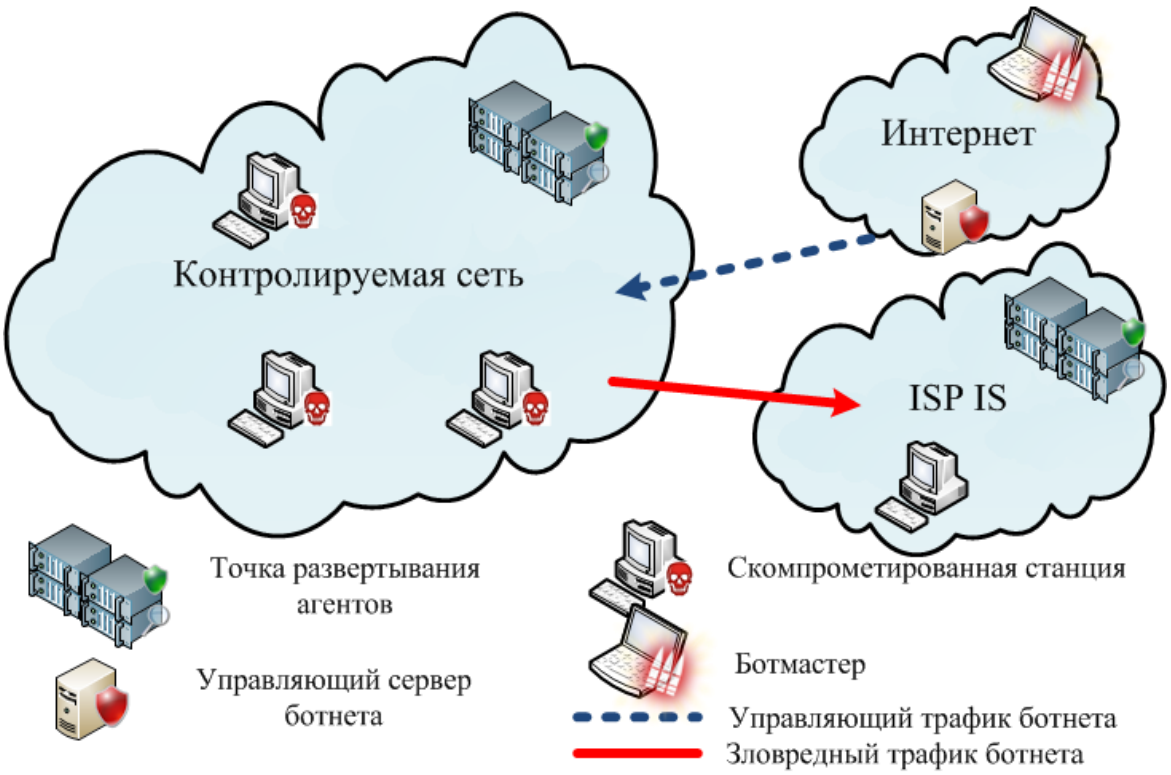

Рисунок 4.4 – Структура сети при проведении тестирования многоагентной системы обнаружения и блокирования ботнетов

В таблице 4.16 представлено количество активных и пассивных ботов с указанием точки их развертывания. Активным считается бот, который участвует в атаке, пассивным – не участвующий.

Таблица 4.16 – Объем ботнета

	Количество ботов в сегменте с развернутыми средствами защиты	Количество ботов в сегменте без средств защиты	Всего ботов
	18	2	20
Активных	15	1	16
Пассивных	3	1	4

Созданный ботнет управлялся с помощью панели управления, позволяющей отслеживать текущий статус ботов и отдавать команды выборочным скомпрометированным машинам. Панель управления представлена на рисунке 4.5.

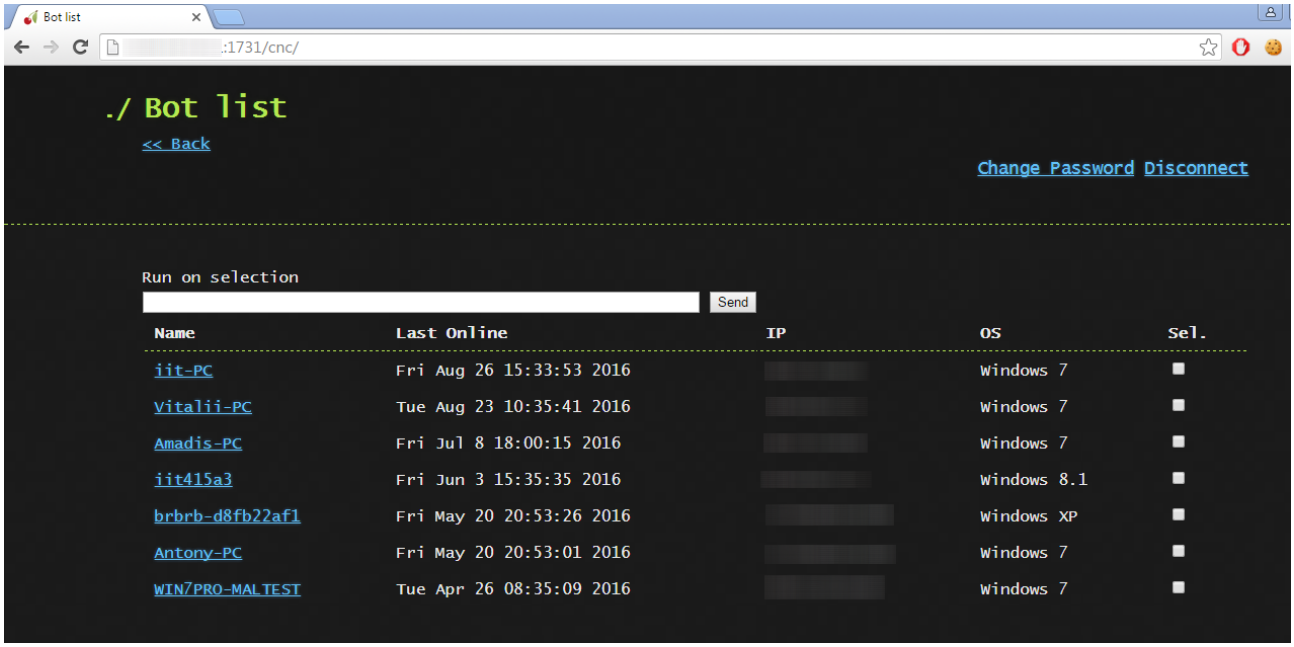

Рисунок 4.5 – Панель управления ботнета

Согласно плану тестирования исследовательского прототипа, группа ботов атаковала целевой компьютер, используя атаку «HTTP POST flood» [41]. При этом с целью проверки возможности обнаружения пассивных ботов в атаке были задействованы не все боты. Для осуществления контроля выполнения атаки с

компьютера ботмастера была запущенна утилита ping с сетевым адресом целевого узла в качестве параметра.

Сравнение эффективности разработанного прототипа проводилось со свободно распространяемыми защитными системами, близкими по предназначению: сетевая система предотвращения вторжений (СПВ) Snort [116] и сетевая система обнаружения активности ботов BotHunter [79]. СПВ Snort использует сигнатурный подход. Без конкретно заданной сигнатуры управляющего трафика данная система не сможет обнаружить пассивного бота. При условии, что база сигнатур СПВ является актуальной, обнаружение вредоносной активности бота является более вероятным событием. Таким образом, будем считать выявление СПВ сетевого адреса атакующего хоста, как положительное событие выявления бота. Система BotHunter использует подход анализа аномалий трафика. В течение работы она проводит мониторинг трафика и пытается выявить признаки различных этапов жизненного цикла бота. В случае обнаружения достаточного количества признаков система сообщает об обнаружении бота. Таким образом, BotHunter с большей вероятностью обнаружит бота, который попал в сеть после внедрения этой системы.

В результате тестирования систем были получены следующие результаты. Система предотвращения вторжений Snort обнаружила 15 скомпрометированных узлов верно. Система BotHunter обнаружила 11 скомпрометированных узлов верно и один – нет. Разработанный прототип обнаружил 18 скомпрометированных узлов. Результаты тестирования приведены в таблице 4.17. Отметим, что первые две системы обнаружили только активных ботов, в то время как 3 из 18 ботов, обнаруженных прототипом, находились в пассивном состоянии.

Таблица 4.17 – Результаты тестирования многоагентной системы

	Полнота	Точность	F-мера
NET.BOTNET	0,90	1,00	0,95
Snort	0,75	1,00	0,86
BotHunter	0,55	0,92	0,69

Показатель точности высокий у всех систем. У первых двух систем данный показатель максимальный в связи с использованием сигнатуры. Результирующим для общей оценки становится показатель полноты обнаружения, имеющий разброс от 15 до 35 %. Это можно объяснить следующими факторами. Во-первых, разработанный прототип, в отличие от остальных систем, обнаружил часть пассивных ботов, находящихся в зоне действия многоагентной системы. Во-вторых, система BotHunter устанавливалась в сегмент с уже существующими ботами, что повлияло на результат работы. В целом показатель F-меры разработанного прототипа превышает аналогичные показатели Snort и BotHunter на 9 и 26 %.

Разработанный прототип выявил сетевой адрес узла, с которого осуществлялся контроль атаки. Сравнение функциональности систем представлено в таблице 4.18.

Таблица 4.18 – Результаты тестирования функциональности многоагентной системы

	Обнаружение атаки	Обнаружение ботов	Формирование сигнатуры	Обнаружение ботмастера
NET.BOTNET	да	да	да	да
Snort	да	да	нет	нет
BotHunter	нет	да	нет	нет

Концепция практического внедрения многоагентной системы.

Разработанная многоагентная система будет максимально эффективна при условии повсеместного распространения, т.е. чем больше будет точек развертывания, тем эффективней будет результат от её работы. С этой целью предлагается концепция практического внедрения многоагентной системы, представленная на рисунке 4.7.

Систему можно внедрять по двум направлениям: в виде SDK-платформы и открытого программного обеспечения. Распространение в качестве открытого

программного обеспечения может обеспечить внедрение системы в секторе малого и среднего бизнеса, владельцы которого обеспокоены вопросами защиты информации. По данному направлению систему возможно распространять либо в виде образа жесткого диска, либо в скомпонованном виде установочного файла программного обеспечения агентов.

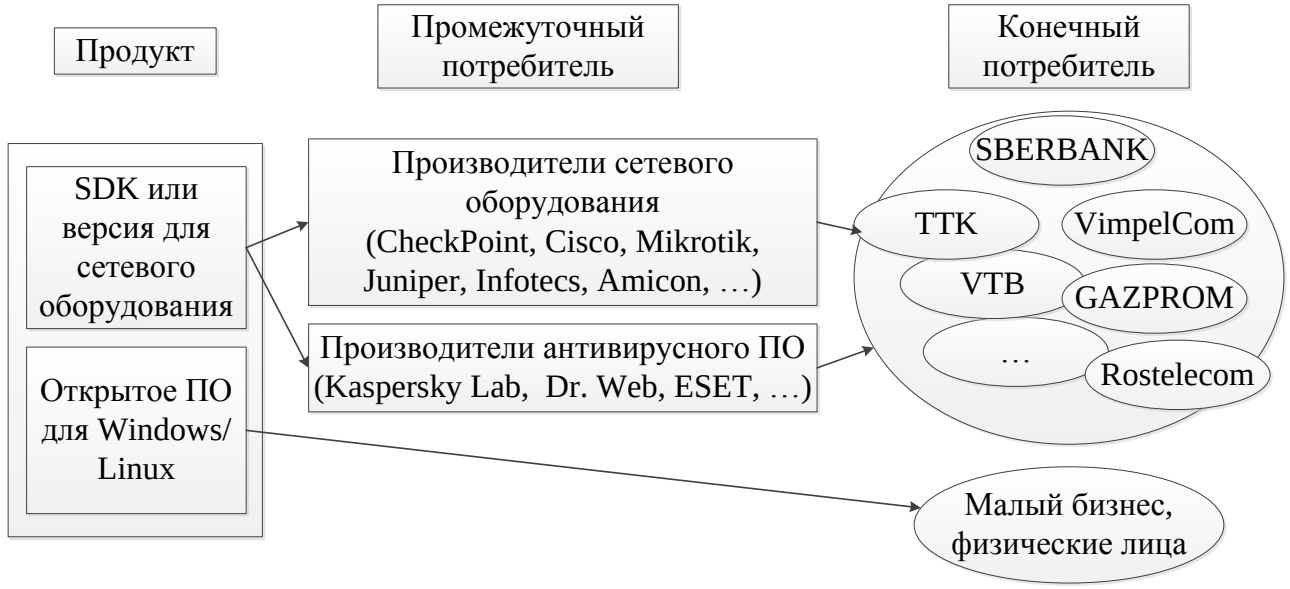

Рисунок 4.7 – Концепция практического внедрения многоагентной системы

Это повлияет на масштабы распространения системы, а значит, приведет к более качественной её работе в целом. Распространение в виде SDK-платформы позволит антивирусным компаниям и разработчикам сетевого оборудования интегрировать агентов системы в свои продукты, используемые уже не только физическими лицами и малым бизнесом, но и крупными телекоммуникационными, финансовыми, промышленными компаниями.

Выводы по четвертой главе

1. Сформулированы требования к многоагентной системе обнаружения и блокирования ботнетов.

2. Спроектирована структура исследовательского прототипа системы. Разработано программное обеспечение, выполняющее функции агента исследования трафика, агента формирования сигнатуры и агента координатора. С целью реализации функций остальных типов агентов проведен выбор различного свободно распространяемого программного обеспечения.

3. Подготовлена инфраструктура тестирования исследовательского прототипа с использованием платформы провайдера челябинской области. Проведено тестирование исследовательского прототипа, построенного на базе предложенного алгоритма обнаружения управляющего трафика ботнетов. Прототип обнаружил 90% ботов используемого ботнета, в том числе ботов, не используемых для проведения атаки. Был также обнаружен сетевой адрес, с которого проводился контроль выполнения распределенной атаки типа «отказ в обслуживании».

4. Проведено сравнение эффективности разработанного прототипа системы обнаружения и блокирования ботнетов со свободно распространяемыми защитными системами, близкими по предназначению: сетевой системой предотвращения вторжений Snort и сетевой системой обнаружения активности ботов BotHunter. Показатель F-меры разработанного прототипа превышает аналогичные показатели Snort и BotHunter на 9 и 26 % соответственно. Сравнение показало преимущества использования разработанного прототипа в функциях формирования сигнатуры и обнаружения компонент ботнета.

5. Предложена концепция практического внедрения многоагентной системы обнаружения и блокирования ботнетов. Концепция описывает возможность повсеместного распространения многоагентной системы для повышения эффективности её использования.

ЗАКЛЮЧЕНИЕ

В диссертационной работе решена задача повышения защищенности информационных систем от атак осуществляемых ботнетами на основе разработки и применения многоагентной системы обнаружения и блокирования ботнетов с использованием алгоритмов интеллектуального анализа данных.

Основные результаты выполненной работы заключаются в следующем:

1. Проведен анализ состояния проблемы обнаружения ботнетов в открытых компьютерных сетях (включая Интернет), существующих методов защиты от распределенных атак типа «отказ в обслуживании» и методов обнаружения ботнетов. Анализ выявил отсутствие полноценной защиты от вредоносной активности ботнетов.

2. Предложен алгоритм обнаружения управляющего трафика ботнета Botnet MultiAgent Recognition на основе интеллектуального анализа данных с возможностью автоматического формирования сигнатуры управляющего трафика. В отличие от существующих методов, использование предложенного алгоритма позволяет решать задачу обнаружения ботнетов в автоматическом режиме, независимо от протокола их управления. При этом алгоритм позволяет обнаруживать ботнеты централизованной и децентрализованной организационных структур, а также не зависит от типа вредоносной деятельности ботов.

3. Предложена архитектура интеллектуальной многоагентной системы обнаружения и блокирования ботнетов, соответствующая типовой архитектуре ботнета, что позволяет блокировать атаки на стороне её источника, тем самым разгрузив каналы передачи от вредоносного трафика, а также на основе разработанного алгоритма обнаружения управляющего трафика позволяет обнаруживать и блокировать пассивных участников ботнета.

4. Предложен метод распределенного обнаружения управляющих компонент ботнета, основанный на сигнатуре управляющего трафика, который, в

отличие от существующих алгоритмов, за счет использования многоагентного подхода позволяет обнаруживать управляющие серверы и узлы сети, с которых осуществляется контроль атаки.

5. Разработан исследовательский прототип многоагентной системы обнаружения и блокирования ботнетов. Эффективность разработанного прототипа подтверждена методом компьютерного моделирования ботнета и его вредоносной деятельности. Прототип обеспечивает повышение показателя F-меры обнаружения ботнета по сравнению с рядом известных систем от 9% до 26%, при этом доля ложных срабатываний не превышает 0,02. Прототип обеспечивает обнаружение ботов, не используемых для проведения атаки, а также обнаружение сетевых адресов, с которых проводится контроль выполнения распределенной атаки типа «отказ в обслуживании».

6. Предложена концепция практического внедрения многоагентной системы обнаружения и блокирования ботнетов, описывающая возможность повсеместного распространения многоагентной системы обнаружения и блокирования ботнетов для повышения эффективности её использования.

СПИСОК ЛИТЕРАТУРЫ

1. Адамацкий А. И., Холланд О. Роящийся интеллект: представления и алгоритмы // Информационные технологии и вычислительные системы. – 1998. – № 1. – С. 45–53.

2. Барабанов А. В., Марков А. С., Цирлов В. Л. Сертификация систем обнаружения вторжений / А. В. Барабанов, А. С. Марков, В. Л. Цирлов // Открытые системы. СУБД. – 2012. – № 3. – С. 31–33.

3. Бондаренко М. Ф., Маторин С. И., Соловьева Е. А. Моделирование и проектирование бизнес-систем: методы, стандарты, технологии / М. Ф. Бондаренко, С. И. Маторин, Е. А. Соловьева. – Харьков : Компания СМИТ, 2004. – 272 с.

4. Вергелис М., Щербакова Т., Демидова Н. Г. Kaspersky Security Bulletin. Спам в 2015 году. [Электронный ресурс] – Режим доступа https://securelist.ru/analysis/ksb/27926/kaspersky-security-bulletin-spam-v-2015-godu.

5. Гайворонская, С. А. Исследование методов обнаружения шеллкодов в высокоскоростных каналах передачи данных [Текст] : дис. … канд. физико-математических наук : 05.13.11/ С.А. Гайворонская. – Москва, 2014.

6. Гончаров Н. О., Горчаков Д. С. Расследование инцидентов, связанных с мобильными бот-сетями и вредоносным программным обеспечением // Проблемы информационной безопасности. Компьютерные системы. – 2015. – № 4. – С. 28–34.

7. Городецкий В. И. Модель многоагентной системы защиты информации // Известия ЮФУ. Технические науки. – 2000. – № 2. – С. 322.

8. Городецкий В. И., Грушинский М. С., Хабалов А. В. Многоагентные системы (обзор) // Новости Искусственного Интеллекта. – 1998. – № 2.

9. Киселев А. Взгляд изнутри: как работает механизм защиты от DDoS в решении Kaspersky DDoS Prevention [Электронный ресурс] // Журнал «Системный администратор». – Режим доступа: http://samag.ru/archive/article/3122.

10. Комаров А. А., Назаров А. Н. Функциональные требования к системе обнаружения и противодействия ботнет-атакам на корпоративные сети // Техника средств связи. Серия: техника телевидения. – 2013. – № 1. – С. 140–151.

11. Косенко М. Ю. Злонамеренное использование облачных технологий // Информационные технологии и системы: материалы Первой междунар. конф., Банное, Россия. – 2012. – С. 67–70.

12. Косенко М. Ю. Реализация распределенной атаки типа «отказ в обслуживании» на основе облачных технологий [Текст] // Информационные и математические технологии в науке и управлении. – Т. 2, ч. 2: Тр. XVII Байкал. Всерос. конф., Иркутск: ИСЭМ СО РАН, 2012. – С. 174–180.

13. Косенко М. Ю. Сбор информации при проведении тестирования на проникновение // Вестник УрФО. Безопасность в информационной сфере – № 3(9). – 2013. – С. 11–15.

14. Косенко М. Ю. Модель выявления и блокирования распределенной атаки типа «отказ обслуживании» с источником атаки из облачной инфраструктуры // Информационные технологии и системы: труды Второй междунар. конф., Банное, 2013. – С. 134–136.

15. Косенко М. Ю. Метод обнаружения ботнетов на основе многоагентного подхода // Труды второй международной конференции «Интеллектуальные технологии обработки информации и управления», 10-12 ноября, Уфа, Россия, 2014. – С. 20–22.

16. Косенко М. Ю. Интеллектуальный анализ данных в задаче обнаружения ботнетов // Вестник УрФО. Безопасность в информационной сфере – № 1(19). – 2016. – С. 22–30.

17. Косенко М. Ю., Мельников А. В. Метод идентификации ботнетов на основе многоагентного подхода // Вестник Воронежского государственного университета. Серия: Системный анализ и информационные технологии, Воронеж: Воронежского государственного университета. – 2015. – С. 89–96.

18. Косенко М. Ю., Мельников А. В. Кооперация агентов многоагентной системы идентификации ботнетов // Труды Четвертой Международной научной

конференции «Информационные технологии и системы», Челябинск: Челябинского государственного университета. – 2015. – С. 128–130.

19. Косенко М. Ю., Мельников А. В. Метод автоматического формирования базы ботов для классификации типов взаимодействия в ботнетах // Труды третьей международной конференции «Информационные технологии интеллектуальной поддержки принятия решений». – 2015. – С. 74–77.

20. Косенко М. Ю., Мельников А. В. Вопросы обеспечения защиты информационных систем от ботнет атак // Вопросы кибербезопасности. – № 4(17). – 2016. – С. 20–28.

21. Котенко И. В., Городецкий В. И. Архитектура базовых агентов многоагентной системы защиты информации в компьютерных сетях // Известия ЮФУ. Технические науки. – 2000. – № 2. – С. 38–51.

22. Котенко И. В., Комашинский Д. В. Концептуальные основы использования методов data mining для обнаружения вредоносного программного обеспечения // Защита информации. Инсайд. – 2010. – № 2. – С. 74–82.

23. Котенко И. В., Комашинский Д. В. Методы интеллектуального анализа данных для выявления вредоносных программных объектов: обзор современных исследований // Вопросы защиты информации. – 2013. – № 4. – С. 21–33.

24. Котенко И. В., Саенко И. Б. Построение системы интеллектуальных сервисов для защиты информации в условиях кибернетического противоборства // Труды СПИИРАН. – 2012. – № 22 (3). – С. 84–100.

25. Котенко И. В., Уланов А. В. Защита от ddos-атак: механизмы предупреждения, обнаружения, отслеживания источника и противодействия // Защита информации. Инсайд. – 2007. – № 1. – С. 60–67.

26. Котенко И. В., Уланов А. В. Защита от ddos-атак: механизмы предупреждения, обнаружения, отслеживания источника и противодействия // Защита информации. Инсайд. – 2007. – № 2. – С. 70–77.

27. Котенко И. В., Уланов А. В. Защита от ddos-атак: механизмы предупреждения, обнаружения, отслеживания источника и противодействия // Защита информации. Инсайд. – 2007. – № 3. – С. 62–69.

28. Лукацкий А. В. Новый РД ФСТЭК по IPS [Электронный ресурс] // Бизнес без опасности. – Режим доступа: http://lukatsky.blogspot.ru/2012/03/ips.html.

29. Марков А. С., Цирлов В. Л., Барабанов А. В. Методы оценки несоответствия средств защиты информации // Радио и связь. – 2012. – С. 192.

30. Назаров А. Н. О подходах к созданию интеллектуальной системы анализа атак из интернета // XII Всероссийское совещание по проблемам управления ВСПУ-2014 Институт проблем управления им. В.А. Трапезникова РАН. – 2014. – С. 9208-9215.

31. Половко И. Ю., Пескова О. Ю. Анализ функциональных требований к системам обнаружения вторжений // Известия Южного федерального университета. Технические науки. – 2014. – № 2. – С. 86–92.

32. Рубцов С.В. Методология структурного анализа и проектирования / С.В. Рубцов, ComputerWorld Россия, 2000.

33. Сачков И.К., Назаров А.Н. Автоматизация противодействия бот-атакам // T-Comm: телекоммуникации и транспорт. – 2014. – № 6 (8). – С. 5–9.

34. Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика. / В.Б. Тарасов, Эдиториал УРСС, 2002. – 352 с.

35. Указ Президента Российской Федерации от 15 января 2013 г. №31с. О создании государственной системы обнаружения, предупреждения и ликвидации последствий компьютерных атак на информационные ресурсы Российской Федерации. – 2013.

36. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств [Текст]. – М.: Стандартинформ, 2011 – 100 с.

37. Российское зеркало CRAN [Электронный ресурс] // R-project. – Режим доступа: https://cran.gis-lab.info.

38. ИТ.СОВ.С4.ПЗ. Методический документ ФСТЭК России. Профиль защиты систем обнаружения вторжений уровня сети четвертого класса защиты [утв. ФСТЭК России. 2012 г.].

39. ГОСТ Р ИСО/МЭК 15408-2-2013. Информационная технология. Методы и средства обеспечения безопасности. Критерии оценки безопасности информационных технологий [Текст]. – М.: Стандартинформ, 2013.

40. Abad C. Log Correlation for Intrusion Detection: A Proof of Concept / Abad, C., Taylor, J., Sengul, C., Yurcik, W., Zhou, Y., Rowe, K. // In Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC'03). – 2003. – P. 255–264.

41. Abliz M. Internet Denial of Service Attacks and Defense Mechanisms // University of Pittsburg. – 2011. – 50 p.

42. Abu Rajab M. A multifaceted approach to understanding the botnet phenomenon / Rajab, M., Zarfoss, J., Monrose, F., Terzis, A. // Proceedings of the 6th ACM SIGCOMM on Internet measurement - IMC '06. – 2006. – P. 41-52.

43. Albin E., Rowe N.C. A realistic experimental comparison of the Suricata and SNORT intrusion-detection systems // 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA). – 2012. – P. 122–127.

44. Andersen D.G. Accountable internet protocol (aip) / Andersen, D.G., Feamster, N., Koponen, T., Moon, D., Shenker, S. // ACM SIGCOMM Computer Communication Review. 2008. – № 4 (38). – P. 339–350.

45. AsSadhan B. Detecting Botnets Using Command and Control Traffic / B. AsSadhan, J. Moura, D. Lapsley, C. Jones, W. Strayer // Eighth IEEE International Symposium on Network Computing and Applications. – 2009. – P. 156–162.

46. Baecher P. The Nepenthes Platform: An Efficient Approach to Collect Malware / P. Baecher, M. Koetter, T. Holz, M. Dornseif, F. Freiling // Springer Berlin Heidelberg. – 2006. – P. 165–184.

47. Baker F. Requirements for IP Version 4 Routers [Electronic resource] // Internet Engineering Task Force. – URL: https://tools.ietf.org/html/rfc1812.

48. Balasubramanian, J., Garcia-Fernandez, J., Isacoff, D., Spafford, E., Zamboni, D. An Architecture for Intrusion Detection Using Autonomous Agents // Technical Report 98/05, COAST Laboratory, Purdue University. – 1998.

49. Balduzzi M., Ciangaglini V., McArdle R. Targeted attacks detection with SPuNge // 11th Annual Conference on Privacy, Security and Trust, PST 2013. –2013. – P. 185–194.

50. Barford P., Yegneswaran V. An Inside Look at Botnets / Boston, MA: Springer US, 2007. – P. 171–191.

51. Bejtlich R. The Practice of Network Security Monitoring: Understanding Incident Detection and Response / R. Bejtlich. – 2013. – 376 p.

52. Bethencourt J., Franklin J., Vernon M. Mapping internet sensors with probe response attacks // 14th USENIX Security Symposium. – 2005. – P. 193–208.

53. Bhuyan, M. H., Kashyap, H. J., Bhattacharyya, D. K., & Kalita, J. K. Detecting distributed denial of service attacks: Methods, tools and future directions // The Computer Journal. – 2014. – № 4 (57). – P. 537–556.

54. Binkley J.R., Singh S. An Algorithm for Anomaly-based Botnet Detection // Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet SRUTI'06. – 2006. – P. 7.

55. Boschi S., Santomaggio G. RabbitMQ Cookbook / S. Boschi, G. Santomaggio, Packt Publishing, 2013. – 288 p.

56. Castelfranchi C. Social power - A point missed in multi-agent / Y. Demazeau, J.P. Muller, Elsevier, DAI adn HCI. – 1990. – P. 49–62.

57. Cheung S., Lindqvist U., Fong M.W. Modeling multistep cyber attacks for scenario recognition // Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX. – 2003. – P. 284–292.

58. Collins, M.P., Shimeall, T.J., Faber, S., Janies, J., Weaver, R., De Shon, M., Kadane, J. Using uncleanliness to predict future botnet addresses // Proceedings of the 7th ACM SIGCOMM conference on Internet measurement: ACM Press, 2007. – P.93-104.

59. Conte R., Castelfranchi C., Miceli M. Limits and Levels of Cooperation: Disentangling Various Types of Prosocial Interaction / Y. Demazeau, J.-P. Muller // Decentralized A.I. 2: Proc. of the 2nd European Workshop on Modelling Autonomous Agents in a Multi-Agent World. – 1991. – P. 147–157.

60. Cooke E., Jahanian F., McPherson D. The Zombie roundup: understanding, detecting, and disrupting botnets // Proceedings of the Steps to Reducing Unwanted Traffic on the Internet on Steps to Reducing Unwanted Traffic on the Internet Workshop. – 2005. – P. 6.

61. Cotton M., Vegoda L. Special-Purpose IP Address Registries // Internet Engineering Task Force. URL: https://tools.ietf.org/html/rfc6890.

62. Cova M., Kruegel C., Vigna G. Detection and analysis of drive-by-download attacks and malicious JavaScript code / New York, New York, USA: ACM Press, 2010. – 281 p.

63. Dagon, D., Qin, X., Gu, G., Lee, W., J.B., Levine, J.G., Owen, H.L. HoneyStat: Local Worm Detection Using Honeypots // Recent Advances in Intrusion Detection. – 2004. – P. 39–58.

64. Dagon D., Zou C.C., Lee W. Modeling Botnet Propagation Using Time Zones. // Proc. 13th Annual Network and Distributed System Security Symp. (NDSS'06). – 2006. – № 6. – P. 2–13.

65. Daswani N., Stoppelman M. The anatomy of Clickbot.A // Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. – 2007. – P. 11.

66. Day D.J., Burns B.M. A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines // The Fifth International Conference on Digital Society, Gosier, Guadeloupe. – 2011. – P. 187–192.

67. Dietrich S., Long N., Dittrich D. Analyzing distributed denial of service tools: The shaft case / USENIX Association : NASA Goddard Space Flight Center, 2000. – P. 329–339.

68. Eckmann S., Vigna G., Kemmerer R. STATL: An attack language for state-based intrusion detection // Journal of Computer Security. – 2002. – № 1–2 (10). – P. 71–103.

69. Egele M., Kirda E., Kruegel C. Mitigating drive-by download attacks: Challenges and open problems // Springer Berlin Heidelberg, 2009. – P. 52–62.

70. Elshoush H.T., Osman I.M. An Improved Framework for Intrusion Alert Correlation // Lecture Notes in Engineering and Computer Science. – 2012. – № 1. – P. 518–523.

71. Ferguson P., Senie D. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. // Internet Engineering Task Force. URL: https://tools.ietf.org/html/rfc2827.

72. Forrest S., Hofmeyr S.A. A sense of self for unix processes // Security and Privacy, IEEE Symposium. – 1996. – P. 120–128.

73. Freiling F.C., Holz T., Wicherski G. Botnet Tracking: Exploring a Root-Cause Methodology to Prevent Distributed Denial-of-Service Attacks // Springer-Verlag, 2005. – P. 319–335.

74. Garfinkel, T., Adams, K., Warfield, A., Franklin, J. Compatibility is not transparency: VMM detection myths and realities // Proceedings of USENIX. Hot Topics in Operating Systems. – 2007. – P. 1–6.

75. Gibbs P.M. Botnet Tracking Tools // SANS Institute. InfoSec Reading Room. – 2014. – 34 p.

76. Goebel J., Holz T. Rishi: identify bot contaminated hosts by IRC nickname evaluation // HotBots'07 Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. – 2007. – P. 8-20.

77. Grizzard, J.B., Sharma, V., Nunnery, C., Kang, B.B.H. Peer-to-peer botnets: overview and case study // Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets. – 2007. – 25p.

78. Gu, G., Sharif, M., Qin, X., Dagon, D., Lee, W., Riley, G. Worm Detection, Early Warning and Response Based on Local Victim Information. // Proceedings of 20 th Annual Computer Security Applications Conference. – 2004. – P. 136–145.

79. Gu, G., Perdisci, R., Zhang, J., Lee, W. BotHunter: detecting malware infection through IDS-driven dialog correlation // USENIX Security '07 Proceedings of the 16th USENIX Security Symposium. – 2007. – P. 12-28.

80. Hares S., Rekhter Y., Li T. A Border Gateway Protocol 4 (BGP-4) [Electronic resource] // Internet Engineering Task Force. – URL: https://tools.ietf.org/html/rfc4271.

81. Henderson, T.R., Ahrenholz, J.M., Kim, J.H. Host Identity Protocol // Internet Engineering Task Force URL: https://tools.ietf.org/html/rfc5201.

82. Holz, T., Steiner, M., Dahl, F., Biersack, E., Freiling, F. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm // Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats. – 2008. – P. 1-9.

83. Ianelli N., Hackworth A. Botnets as a Vehicle for Online Crime // The International Journal of Forensic Computer Science. – 2007. – P. 19–39.

84. Ilgun K., Kemmerer R.A., Porras P.A. State transition analysis: A rule-based intrusion detection approach // Software Engineering, IEEE Transactions. – 1995. – № 3. – P. 181–199.

85. Jain A.K., Murty M.N., Flynn P.J. Data clustering: a review // ACM Computing Surveys. – 1999. – № 3 (31). – P. 264–323.

86. Jin C., Wang H., Shin K.G. Hop-Count Filtering: An Effective Defense Against Spoofed DDoS Traffic // Proceedings of the 10th ACM conference on Computer and communication security - CCS ’03. – 2003. – P. 30-41.

87. Jung J., Schechter S.E., Berger A.W. Fast Detection of Scanning Worm Infections // Recent Advances in Intrusion Detection. 7th International Symposium, RAID 2004. – 2004. – P. 59–81.

88. Jung J., Paxson V., Berger A. W. , Balakrishnan H. Fast portscan detection using sequential hypothesis testing // Proceedings of the IEEE Symposium on Security and Privacy. Los Alamitos, CA, USA: IEEE Computer Society, 2004. – P. 211-225.

89. Karasaridis A., Rexroad B., Hoeflin D. Wide-scale botnet detection and characterization // Proceedings of the first USENIX workshop on hot topics in Botnets, HotBots. – 2007. – 7 p.

90. Karim A., Salleh R.B., Shiraz M., Shah S.A.A, Awan I. Botnet detection techniques: review, future trends, and issues // Computers & Electronics. – 2014. – № 11 (15). – P. 943–983.

91. Keromytis A., Misra V., Rubenstein D. SOS: Secure Overlay Services // New York, USA: ACM Press, 2002. – P. 61–72.

92. Kosenko M.U. A distributed network scanning during penetration testing // Computer Science and Information Technologies (CSIT 2013), Ufa: Ufa State Aviation Technical University USATU Editorial-Publishing Office. – 2013. – P.54-56.

93. Kosenko M.U. Method «Botnet MultiAgent Recognition» (BNMAR) for Botnet Traffic Detection Based on Data Mining Technologies // Proceedings of the Workshop on Tehnologies of Digital Signal Processing and Storing (DSPTech'2015). – 2015. – P. 153–157.

94. Kosenko M.U., Melnikov A.V. Implementing a distributed attack such as «Denial of service» based on cloud computing // CSIT'2012: proceedings of the 114th International workshop on computer science and information technologies, Ufa-Hamburg-Norwegirg Fjords. – 2012. – P. 130–133.

95. Le, V.L., Welch, I., Gao, X., Komisarczuk, P. Anatomy of drive-by download attack // Proceedings of the Eleventh Australasian Information Security Conference - Volume 138. – 2013. – P. 49–58.

96. Li, J., Mirkovic, J., Wang, M., Reiher, P., and Zhang, L. SAVE: Source Address Validity Enforcement Protocol // In Proceedings of the IEEE INFOCOM. 2002. C. 1557–1566.

97. Liu, X., Li, A., Yang, X., Wetherall, D. Passport: secure and adoptable source authentication // Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation. 2008. C. 365–378.

98. Livadas C., Walsh R., Lapsley D., Strayer W.T. Using machine learning techniques to identify botnet traffic // 31st IEEE conference on local computer networks. – 2006. – P. 967–974.

99. Malan D.J., Science C. Rapid Detection of Botnets through Collaborative Networks of Peers // School of Engineering and Applied Sciences. Harvard. – 2007. – 104 p.

100. Marczak, B., Weaver, N., Dalek, J., Ensafi, R., Fifield, D., McKune, S. China's Great Cannon [Electronic resource] / 2015. – URL: https://citizenlab.org/2015/04/chinas-great-cannon/.

101. Mirkovic J., Reiher P. A taxonomy of DDoS attack and DDoS defense mechanisms // SIGCOMM Comput. Commun. – 2004. – № 2 (34). – P. 39–53.

102. Moore D., D. Shannon, C. Voelker, Savage, S. Network Telescopes: Technical Report // Technical Report, Cooperative Association for Internet Data Analysis (CAISA). – 2002.

103. Moore, D., Shannon, C., Brown, D. J., Voelker, G. M., Savage, S. Inferring Internet denial-of-service activity // ACM Transactions on Computer Systems. – 2006. – № 2 (24). – P. 115–139.

104. Nikander P., Moskowitz R. Host Identity Protocol (HIP) Architecture [Electronic resource] // Internet Engineering Task Force. – URL: https://tools.ietf.org/html/rfc4423.

105. Ning P., Cui Y., Reeves D.S. Constructing Attack Scenarios through Correlation of Intrusion Alerts // Proceedings of the 9th ACM conference on Computer and communications security. – 2002. – P. 10.

106. Park K., Lee H. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets // ACM SIGCOMM Computer Communication Review. – 2001. – № 4 (31). – P. 15–26.

107. Paxson V. Bro: a system for detecting network intruders in real-time // Computer Networks. – 1999. – № 23–24 (31). – P. 2435–2463.

108. Pelleg, D., Moore, A. X-means: Extending K-means with efficient estimation of the number of clusters // Proceedings of the Seventeenth International Conference on Machine Learning table of contents. – 2000. – P. 727–734.

109. Porras P.A., Neumann P.G. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances // Proc. 20th National Information Systems Security Conference. – 1997. – P. 353–365.

110. Provos N. A virtual honeypot framework // Proceedings of the 13th USENIX Security Symposium. – 2004. – P. 1–14.

111. Ramachandran A., Feamster N. Understanding the network-level behavior of spammers // ACM SIGCOMM Computer Communication Review. – 2006. – № 4 (36). – P. 291-302.

112. Ramachandran A., Feamster N., Dagon D. Revealing botnet membership using DNSBL counter-intelligence // Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet. – 2006. – Volume 2, №36. – P. 49-54.

113. Rijsbergen C.J. Van Evaluation // Information Retrieval. – 1979.

114. Rodriguez-Gomez R. Analysis of Botnets Through Life-Cycle // Proceedings of the International Conference Security and Cryptography (SECRYPT). – 2011. – P. 257–262.

115. Rodriguez-Gomez R.A., Macia-Fernandez G., Garcia-Teodoro P. Survey and taxonomy of botnet research through life-cycle // ACM Computing Surveys. – 2013. – № 4 (45). – P. 1–33.

116. Roesch M. Snort: Lightweight Intrusion Detection for Networks. // LISA '99: 13th Systems Administration Conference. – 1999. – P. 229–238.

117. Rosenschein J.S., Zlotkin G. Rules of Encounter: Designing Conventions for Automated Negotiation among Computers / J.S. Rosenschein, G. Zlotkin // MIT Press. – 1994. – P. 83-84.

118. Rossow C. SoK: P2PWNED - Modeling and Evaluating the Resilience of Peer-to-Peer Botnets // IEEE Symposium on Security and Privacy. – 2013. – P. 97-111.

119. Smith R.G. The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver // IEEE Transactions on Computers. – 1980. – № 12 (C-29). – P. 1104–1113.

120. Snapp S.R., Dias, J.B.G.V., Goan, T., Heberlein, L.T., Ho, C., Levitt, K.N., Mukherjee, B., Smaha, S.E., Grance, T., Teal, D.M., Mansur, D. DIDS (Distributed intrusion detection system) - Motivation, architecture, and an early prototype // Proceedings of the 14th national computer security conference. – 1996. – P. 361–370.

121. Sommer R., Paxson V. Enhancing byte-level network intrusion detection signatures with context // Proceedings of the 10th ACM conference on Computer and communication security - CCS '03. – 2003. – P. 262.

122. Staniford S., Hoagland J.A., McAlerney J.M. Practical automated detection of stealthy portscans // Journal of Computer Security. – 2002. – № 1–2 (10). – P. 105–136.

123. Staniford S., Paxson V., Weaver N. How to 0wn the Internet in Your Spare Time // USENIX Security Symposium. – 2002. – P. 149-167.

124. Stewart Joe Bobax trojan analysis [Electronic resource]. – URL: https://www.secureworks.com/research/topbotnets?threat=bobax.

125. Stinson, E., Mitchell, J.C. Characterizing Bots' Remote Control Behavior // DIMVA '07 Proceedings of the 4th international conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg. – 2007. – P. 89–108.

126. Stoica I. Chord: A Scalable Peer-to-Peer Lookup Protocol for Internet Applications // ACM SIGCOMM Computer Communication Review. – 2001. – № 4 (31). – P. 149–160.

127. Strayer, W. T., Walsh, R., Livadas, C., Lapsley, D. Detecting botnets with tight command and control // Proceedings of the 31st IEEE Conference on Local Computer Networks. – 2006. – P. 195–202.

128. Strayer W.T., Lapsley, D.E. , Walsh, R., Livadas, C. Botnet Detection: Countering the Largest Security Threat / Springer Publishing Company, Incorporated. – 2008. – P. 1-24.

129. Szor P. The art of computer virus research and defense // Choice Reviews Online. – 2005. – № 3 (43). – P. 43-1613-43–1613.

130. Templeton S.J., Levitt K. A requires/provides model for computer attacks // Proceedings of the 2000 workshop on New security paradigms - NSPW '00. – 2000. – P. 31–38.

131. Lu T.T., Liao H.Y., Chen M.S. An Advanced Hybrid P2p Botnet 2.0 // World Academy of Science,. Engineering and Technology. – 2011. – № 57. – P. 595-597.

132. Valeur, F., Vigna, G., Kruegel, C., Kemmerer, R.A. A comprehensive approach to intrusion detection alert correlation // IEEE Transactions on Dependable and Secure Computing. – 2004. – № 3 (1). – P. 146–168.

133. Vigna G., Kemmerer R.A. NetSTAT: A Network-Based Intrusion Detection Approach // Journal of computer security. – 1999. – №7 – P. 37–71.

134. Vogt R., Aycock J., Jacobson M.J. Army of Botnets // Proc. 14th Annual Network and Distributed System Security Symposium. – 2007. – P. 111–123.

135. Wang K., Cretu G., Stolfo S.J. Anomalous Payload-Based Worm Detection and Signature Generation // Springer-Verlag. – 2006. – P. 227–246.

136. Wang K., Parekh J.J., Stolfo S.J. Anagram: A content anomaly detector resistant to mimicry attack // Recent Advances in Intrusion Detection. – 2006. – № 4219. – P. 226–248.

137. Wang X., Chen S., Jajodia S. Tracking Anonymous Peer-to-Peer VoIP Calls on the Internet // ACM Conference on Computer and Communications Security. –2005. – P. 81–91.

138. Wang X., Chen S., Jajodia S. Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems // In Proc. IEEE Symposium on Security and Privacy. – 2007. – P. 116–130.

139. Weaver N., Staniford S. Very fast containment of scanning worms, revisited // Malware Detection. – 2007. – P. 113–145.

140. Wu, J., Vangala, S., Gao, L., Kwiat, K. An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques // Proceedings of the Symposium on Network and Distributed Systems Security. – 2004. – P. 143-156.

141. Xie, Y., Kim, H., O'Hallaron, D., Reiter, M., Zhang, H. Seurat: A pointillist approach to anomaly detection // Recent Advances in Intrusion Detection. – 2004. – P. 238–257.

142. Yang, J., Ning, P., Wang, X.S., Jajodia, S. CARDS: A Distributed System for Detecting Coordinated Attacks // IFIP TC11 16th Annual Working Conference on information security. – 2000. – P. 171–180.

143. Yang Y. An Evaluation of Statistical Approaches to Text Categorization // Information Retrieval. – 1999. – № 1 (1). – P. 69–90.

144. Yen T.F., Reiter M.K. Traffic aggregation for malware detection // Springer Berlin Heidelberg. – 2008. – P. 207–227.

145. Yoda K., Etoh H. Finding a connection chain for tracing intruders // Computer Security - ESORICS. – 2000. – P. 191–205.

146. Zhuge J., Holz T., Han X. Characterizing the IRC-based Botnet Phenomenon // Science And Technology. – 2007. – P. 1–16.

147. Zou C.C., Cunningham R. Honeypot-aware advanced botnet construction and maintenance // International Conference on Dependable Systems and Networks, IEEE. – 2006. – P. 199–208.

148. Zou C.C., Towsley D., Gong W. The monitoring and Early Detection for Internet Worms // Proceegings of the 10th ACM Conference on Computer and Communication Security. – 2004. – № 13. – P. 961-974.

149. The Shadowserver Foundation [Electronic resource]. – URL: https://www.shadowserver.org.