МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра Прикладная математика и информатика
(наименование)

09.04.03 Прикладная информатика
(код и наименование направления подготовки)

Управление корпоративными информационными процессами
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ)

на тему «Исследование нейросетевого метода обнаружения сетевых атак»

Обучающийся В.А. Оглов
(Инициалы Фамилия) (личная подпись)

Научный руководитель канд. техн. наук., Д.Г. Токарев
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Оглавление

Используемые сокращения и обозначения

АНС – Автоассоциативная Нейронная Сеть;

БД – База Данных;

ИАД – Интеллектуальный анализ данных;

ИНС – Искусственная Нейронная Сеть;

КФ – Корреляционная функция;

ОС – Операционная Система;

ПО – Программное Обеспечение;

СМИ – Средства Массовой Информации;

СОВ (IDS) – Система Обнаружения Вторжений (Intrusion Detection System);

СУБД – Система Управления Базами Данных;

ЦП – Центральный процессор;

APT – Advanced Persistent Threat – целевая кибератака;

CMS – Content Management System – система управления контентом сайта;

DoS – Denial of Service – отказ в обслуживании;

DDoS – Distributed Denial of Service – распределённая атака «отказ в обслуживании»;

DPI – Deep Packet Inspection – технология накопления статистических данных, проверки и фильтрации сетевых пакетов по их содержимому;

HTTP – HyperText Transfer Protocol – протокол прикладного уровня передачи данных;

ICMP – Internet Control Message Protocol – протокол межсетевых управляющих сообщений;

IP – Internet Protocol – межсетевой протокол;

IPS (СПВ) – Intrusion Prevention System (Система Предотвращения Вторжений);

IoT – Internet of Things – Интернет вещей – концепция вычислительной сети физических предметов («вещей»), оснащённых встроенными технологиями для взаимодействия друг с другом или с внешней средой;

LDAP – Lightweight Directory Access Protocol – облегчённый протокол прикладного уровня для доступа к службе каталогов X.500;

MLP – Multilayer Perceptron – многослойный персептрон;

NIDS (ССОВ) – Network IDS (Сетевая СОВ);

PHP – Hypertext Preprocessor – скриптовый язык общего назначения для создания веб-страниц;

SQL – Structured Query Language – язык структурированных запросов;

SQLIA – SQL-Injection Attack – атака SQL-инъекция;

SSI – Server Side Includes – язык для динамической «сборки» веб-страниц на сервере из отдельных составных частей и выдачи клиенту полученного HTML-документа;

SSI Injection – вид уязвимости, использующий вставку серверных команд в HTML-код или запуск их напрямую с сервера;

TCP – Transmission Control Protocol – протокол управления передачей;

UDP – User Datagram Protocol – протокол пользовательских датаграмм;

WAF – Web Application Firewall – брандмауэр или фаервол для веб-приложений, который устанавливает правила обмена данными по протоколу HTTP;

WASC – Web Application Security Consortium – международная некоммерческая организация, объединяющая экспертов-профессионалов в области безопасности веб-приложений;

WSTC – Web Security Threat Classification – классификация угроз веб-безопасности;

XML – eXtensible Markup Language – расширяемый язык разметки;

XSS – Cross-site Scripting – межсайтовое выполнение сценариев.

Введение

Стремительно развивающиеся информационные технологии открывают больше возможностей, как для легитимной, так и для несанкционированной деятельности. Интернет – удобный и эффективный канал коммуникаций, обеспечивающий новые возможности для оказания различного рода полезных услуг и перспективы для развития бизнесов. Не смотря на все достоинства, у интернета есть один существенный недостаток – он небезопасен. Все чаще в СМИ встречаются новости о киберпреступлениях, взломе сайтов, мошенничестве, утечке пользовательской информации.

Одним из проявлений процесса информатизации общества является масштабное развитие сетевых сервисов со всеми присущими проблемами безопасности, что ставит перед сетевыми администраторами и специалистами по информационной безопасности задачу обеспечения полного и единого контроля над системами, целостности, доступности и конфиденциальности данных.

К сожалению, даже в условиях активно развивающегося веб-хакинга – взлома веб-сайтов и веб-серверов, на которых эти сайты размещены, универсального и абсолютно эффективного противодействия не разработано до сих пор.

А между тем на сегодняшний день веб-приложения самая простая мишень для атак, потому что они общедоступны и используются повсеместно: и в банковской сфере, и в туризме, и в бытовой, и в социальной жизни и т.д.

В свете того, что изощренность и новизна способов сетевых вторжений, а также механизмы обхода существующих систем безопасности опережают в развитии средства обнаружения вторжений, крайне необходимы новые разработки, позволяющие оперативно обнаруживать не только известные, но и ранее неизвестные нарушения безопасности, что в свою очередь позволит предотвращать сетевые атаки, избегая последствия и ущерба, которые они наносят.

Глава 1 Проблемы и средства защиты веб-ресурсов

Защита веб-сервисов представляет собой одну из самых актуальных проблем в сфере обеспечения безопасности данных в сети. Интерактивная природа веб-приложений порождает новые виды угроз, которые не могут быть эффективно нейтрализованы с помощью классических методов защиты. К примеру, ещё в 2014 году более половины всех атак на корпоративные системы проводились через уязвимости веб-приложений, несмотря на использование стандартных средств безопасности [78].

Большая часть сайтов, представленных в сети, обладает разнообразными слабыми местами и регулярно становится объектом изучения злоумышленников, которые стремятся найти и использовать эти недостатки для своих целей. Совершая всевозможные атаки на сетевой ресурс, злоумышленники могут преследовать различные цели, и выведение сервиса из рабочего состояния, и хищение персональных и конфиденциальных данных пользователей.

Уязвимости веб-сайтов делятся на несколько больших классов. Большинство уязвимостей - следствие ошибочного или небезопасного программного обеспечения. А некоторые уязвимости являются неотъемлемым свойством самого языка программирования, например, языка PHP или SQL.

Веб-сайты зачастую работают с базами данных, в которых в зависимости от специфики ресурса хранится различного рода личная и конфиденциальная информация пользователей и клиентов. Например, база данных электронного магазина может содержать огромное количество номеров кредитных карт, база данных сайта визового центра или авиакомпании - паспортные данные и т.д. Целью целого множества атак в Интернете является именно кража таких данных, например, для дальнейшей продажи [77]. Поскольку веб-сайты размещаются на серверах, успешная атака может позволить злоумышленнику проникнуть в командную строку сервера. Доступ к командной строке часто открывает путь к правам суперпользователя, что означает полный контроль

над сервером. Это предоставляет возможность выполнять любые операции с атакуемой системой и является одной из наиболее распространённых уязвимостей.

Рассмотрим основные угрозы информационной безопасности веб-сайтов.

1.1 Основные угрозы безопасности веб-сайтов

Классификация основных угроз безопасности веб-сайтов была проведена международной организацией Web Application Security Consortium (WASC) и в результате был создан документ Web Security Threat Classification (WSTC) [98].

Согласно данному документу угрозы безопасности веб-сервисов делятся на атаки и уязвимости. При этом в классификации выделены 34 класса атак и 15 классов недостатков, которые структурированы в такие разделы как «Аутентификация», «Авторизация», «Логические атаки», «Атаки на клиентов», «Разглашение информации», «Выполнение команд» и «Другие» [80].

Раздел «Аутентификация» («Authentication») охватывает атаки, которые нацелены на методы проверки подлинности пользователей, служб или приложений, применяемые в веб-приложениях. Основные типы атак связаны с обходом защитных механизмов либо использованием уязвимостей в системах аутентификации, реализованных на стороне веб-серверов:

– Подбор (Brute Force). Это метод взлома, при котором злоумышленник пытается подобрать пароль или ключ доступа путем перебора всех возможных комбинаций символов до тех пор, пока не будет найдена правильная;

– Недостаточная аутентификация (Insufficient Authentication). Это уязвимость, возникающая, когда механизмы проверки подлинности пользователя недостаточно надежны или отсутствуют, позволяя

злоумышленникам получать доступ к системе или данным без должной идентификации;

– Небезопасное восстановление паролей (Weak Password Recovery Validation). Это уязвимость, возникающая, когда процесс восстановления пароля не имеет достаточной защиты, например, использует легкодоступные или непроверенные данные (вопросы безопасности, коды подтверждения) для идентификации пользователя, что позволяет злоумышленникам легко обойти защиту и получить доступ к учетной записи.

Раздел «Авторизация» («Authorization») рассматривает атаки, нацеленные на системы, которые определяют права пользователей, служб или приложений на выполнение тех или иных действий. На многих веб-ресурсах доступ к определенному контенту или функционалу предоставляется только ограниченному кругу лиц, тогда как для остальных он должен быть закрыт. Однако злоумышленники могут применять различные методы для расширения своих полномочий и получения несанкционированного доступа к защищенным материалам или возможностям системы. Например:

– Предсказуемое значение идентификатора сессии (Credential/Session Prediction). Позволяет определить или предугадать значения идентификаторов сессий или учетных данных пользователей, основываясь на их закономерностях или слабой генерации;

– Недостаточная авторизация (Insufficient Authorization). Недостаточная авторизация возникает, когда система не проверяет должным образом права пользователя на выполнение определенных действий или доступ к ресурсам;

– Отсутствие таймаута сессии (Insufficient Session Expiration). Это уязвимость, при которой сессия пользователя не завершается вовремя после периода бездействия или после выхода из системы. Это создает возможность для злоумышленника использовать активную сессию и получить несанкционированный доступ к учетной записи;

– Фиксация сессии (Session Fixation). Используя данный класс атак, злоумышленник заставляет жертву использовать заранее известный идентификатор сессии, который был подделан или назначен злоумышленником.

Раздел «Атаки на клиентов» («Client-side Attacks») посвящен описанию способов атак, направленных на пользователей веб-сервера. При взаимодействии с сайтом между пользователем и сервером возникает доверительная связь, как на техническом, так и на психологическом уровне. Пользователь рассчитывает на получение легального контента и не предполагает, что сайт может быть источником угроз. Воспользовавшись этим доверием, злоумышленник может применять различные методы для атак на клиентов, такие как:

– Подмена содержимого (Content Spoofing). Это атака, при которой злоумышленник изменяет или подделывает контент на веб-странице, чтобы ввести пользователя в заблуждение;

– Межсайтовое выполнение сценариев (Cross-site Scripting, XSS). Это уязвимость, при которой злоумышленник внедряет вредоносный код (обычно JavaScript) на веб-страницу, чтобы он выполнялся в браузере жертвы;

– Расщепление HTTP-запроса (HTTP Response Splitting). Эта уязвимость возникает при некорректной обработке веб-сервером входных данных, что позволяет злоумышленнику внедрить дополнительные HTTP-заголовки или содержимое в ответ сервера.

Раздел «Выполнение кода» («Command Execution») рассматривает атаки, нацеленные на запуск произвольного кода на веб-сервере. Большинство серверов обрабатывают запросы, используя данные, предоставленные пользователем, которые иногда применяются для формирования команд, генерирующих динамический контент. Если при разработке системы безопасность не была должным образом учтена, злоумышленник может изменять исполняемые команды, эксплуатируя уязвимости следующих типов:

– Переполнение буфера (Buffer Overflow). Уязвимость позволяет злоумышленнику внедрить и выполнить произвольный код на целевой системе или вызвать отказ в обслуживании (DoS). Это происходит за счет записи данных, превышающих допустимый размер буфера, что приводит к повреждению памяти;

– Атака на функции форматирования строк (Format String Attack). Уязвимость позволяет злоумышленнику извлечь конфиденциальные данные из памяти, изменить содержимое памяти или вызвать сбой программы. Это достигается за счет передачи вредоносных строк форматирования, которые нарушают работу уязвимых функций обработки данных;

– Внедрение операторов LDAP (LDAP Injection). Уязвимость позволяет внедрить вредоносные запросы в поля ввода, используемые для взаимодействия с LDAP-сервером;

– Выполнение команд ОС (OS Commanding). Это атака, при которой злоумышленник внедряет и выполняет произвольные команды операционной системы через уязвимости веб-приложения. Это может привести к полному контролю над сервером или доступу к конфиденциальным данным;

– Внедрение операторов SQL (SQL Injection). Это атака, при которой злоумышленник вводит вредоносные SQL-запросы в поля ввода данных, чтобы манипулировать базой данных. Это может позволить получить несанкционированный доступ, изменить или удалить данные, а также нарушить работу приложения;

– Внедрение серверных сценариев (SSI Injection). Атака, позволяет внедрить команды SSI (Server-Side Includes) в веб-приложение через уязвимые поля ввода для их дальнейшего выполнения на сервере;

– Внедрение операторов XPath (XPath Injection). Атака позволяет ввести вредоносные выражения XPath в запросы для манипулирования XML-данными.

Раздел «Разглашение информации» («Information Disclosure») рассматривает атаки, нацеленные на сбор дополнительных сведений о веб-

приложении. Эксплуатируя подобные уязвимости, злоумышленник может выяснить используемые программные продукты, версии клиентского и серверного ПО, а также установленные патчи. В некоторых случаях утекающая информация может включать пути к временным файлам или резервным копиям. В большинстве случаев эти данные не нужны для работы пользователей. Многие серверы предоставляют избыточный объем служебной информации, хотя её следует максимально ограничивать. Чем больше данных о системе станет доступно атакующему, тем выше вероятность успешного взлома. Для этих целей используются такие виды атак, как:

– Индексирование директорий (Directory Indexing). Это уязвимость, при которой веб-сервер предоставляет доступ к списку файлов и папок в директории из-за отсутствия файла index или неправильной настройки;

– Идентификация приложений (Web Server/Application Fingerprinting). Атака используется для сбора информации о веб-сервере или приложении, включая тип ПО, версии и установленные компоненты, путем анализа ответов сервера, заголовков и других доступных данных;

– Утечка информации (Information Leakage). Это непреднамеренное раскрытие данных через ошибки в конфигурации, сообщения об ошибках, комментарии в коде или другие источники;

– Обратный путь в директориях (Path Traversal). Это атака, при которой злоумышленник получает доступ к файлам за пределами корневой директории веб-сервера, используя специальные символы для перемещения по файловой системе;

– Предсказуемое расположение ресурсов (Predictable Resource Location). Уязвимость позволяет найти или получить доступ к скрытым или незащищенным файлам, папкам или функциям веб-приложения, используя логику именования или стандартные пути размещения.

Раздел «Логические атаки» («Logical Attacks») рассматривает атаки, которые нацелены на использование уязвимостей в работе функций или логики приложения. Логика работы программы определяет ожидаемый

порядок действий для выполнения конкретных операций, таких как сброс паролей, создание учетных записей, участие в аукционах или проведение транзакций в интернет-магазинах. Для завершения задачи пользователь обычно должен последовательно выполнить несколько шагов. Однако злоумышленник может найти способ обойти эти этапы или манипулировать ими для достижения своих целей. К данному типу относятся следующие виды атак:

– Злоупотребление функциональными возможностями (Abuse of Functionality). В данной атаке используются легальные функции приложения в непредусмотренных целях, нарушая его нормальную работу или получая несанкционированные преимущества.

– Отказ в обслуживании (Denial of Service). Атака направлена на перегрузку системы или приложения, чтобы сделать его недоступным для пользователей. Это достигается путем создания большого количества запросов или эксплуатации уязвимостей, что приводит к сбоям в работе сервиса.

– Недостаточное противодействие автоматизации (Insufficient Anti-automation). Это уязвимость, при которой приложение не защищено от злоупотребления автоматизированными скриптами или ботами, что позволяет злоумышленникам проводить массовые атаки, такие как подбор паролей или спам.

– Недостаточная проверка процесса (Insufficient Process Validation). Это уязвимость, при которой приложение не проверяет корректность выполнения промежуточных этапов процесса. Это позволяет злоумышленнику пропускать шаги или изменять их порядок для достижения несанкционированных результатов.

Чуть подробнее остановимся на самых распространенных атаках типа «Выполнение кода».

1.2 Атаки Buffer Overflow и Code Injection

Атаки типа переполнения буфера (Buffer Overflow) и внедрения кода (Code Injection) – это основные угрозы для веб-ресурсов, и именно они чаще всего используются злоумышленниками для взлома сайтов. Одна из самых распространенных и известных атак внедрения кода – SQL-injection. Остановимся на них более подробно.

1.2.1 Buffer Overflow

Переполнение буфера считается одним из самых известных дефектов в программном обеспечении и часто используемых методов атак, поскольку многие высокоуровневые языки программирования применяют технологию стекового кадра. Эта технология предполагает хранение данных в стеке процесса, где пользовательские данные смешиваются с управляющей информацией, такой как адрес начала стекового кадра и точка возврата из функции.

Переполнение буфера возникает в случае, когда объем данных превышает отведённое для них пространство, и программа пытается записать информацию за пределами зарезервированной области памяти. Это часто происходит из-за отсутствия должной проверки границ буфера [72]. При переполнении данные затирают соседние участки памяти, что может спровоцировать сбои в работе программы. Если атакующий получает контроль над процессом переполнения, это открывает возможность для серьёзных нарушений безопасности.

Эксплуатация переполнения буфера дает злоумышленнику возможность изменять критические данные процесса в оперативной памяти системы. Некоторые разновидности этой уязвимости, например, переполнение стекового кадра, позволяют загружать и запускать произвольный машинный код под видом программы, используя права учетной записи, от имени которой она работает [26]. Кроме того, переполнение буфера может стать причиной

отказа в обслуживании, так как повреждение памяти вызывает сбои и ошибки в работе программ.

В более опасных сценариях злоумышленник может изменить последовательность выполнения программы, заставив её выполнять произвольные действия в рамках её контекста. Такая возможность возникает в нескольких ситуациях. Например, через переполнение буфера можно модифицировать служебные участки памяти, включая адрес возврата из функций, хранящийся в стеке. Кроме того, при переполнении могут быть изменены значения переменных, используемых в программе.

Переполнение буфера представляет собой одну из самых частых проблем в области безопасности и иногда затрагивает веб-серверы. Тем не менее, атаки, направленные на использование этой уязвимости, применяются к веб-приложениям сравнительно редко. Это связано с тем, что злоумышленнику обычно требуется тщательно изучить исходный код или исполняемый файл программы. Атака на программу, работающую на удаленном сервере, осложняется тем, что приходится действовать "вслепую", что значительно снижает вероятность успешного результата. Переполнение буфера обычно возникает при создании программ на языках C и C++, в частности, при использовании встроенных функций strcpy (), strcat () и streadd (), поскольку они подвержены переполнениям буфера [80].

Рекомендуется использовать Java в качестве языка программирования, поскольку Java не подвержен переполнениям буфера [64].

Кроме того, уязвимости переполнения буфера в популярных сетевых сервисах были основным средством, используемым для распространения всех известных интернет-червей [10], [44], [91].

Известны различные виды, механизмы, методы эксплуатации переполнения буфера [72], [69], [46], [68], [41].

Также, необходимо отметить, что атаки переполнения буфера совершенствуются и учатся обходить системы обнаружения вторжений (СОВ), что существенно затрудняет их обнаружение.

1.2.2 SQL Injection

Данные обычно хранятся в специализированных базах, к которым обращаются с помощью запросов, чаще всего составленных на языке SQL (Structured Query Language – структурированный язык запросов). SQL – это особый язык программирования, предназначенный для взаимодействия с серверами баз данных. Он позволяет извлекать данные (команда SELECT), добавлять новые записи (команда INSERT), изменять существующие (команда UPDATE) и удалять информацию (команда DELETE). Также существует оператор UNION, который используется для объединения результатов двух запросов. Большинство веб-серверов поддерживают этот язык, а веб-приложения применяют SQL-запросы для работы с данными, например, при обновлении личной информации пользователя или заполнении форм на сайте.

Отсутствие должной проверки данных, вводимых пользователем, дает возможность злоумышленнику встроить в форму веб-интерфейса специальный код, содержащий фрагмент SQL-запроса. Это приводит к изменению логики выполнения итоговых SQL-запросов. Такая атака называется инъекцией, причем наиболее распространенной ее разновидностью является SQL-инъекция. Эта уязвимость считается крайне опасной, так как позволяет атакующему получить доступ к базе данных и осуществлять чтение, модификацию или удаление данных, которые должны быть для него недоступны.

Внедрение SQL-операторов (SQL Injection) представляет собой один из наиболее часто встречающихся методов атак на веб-серверы, которые формируют SQL-запросы к базам данных на основе пользовательского ввода.

Уязвимость для подобных атак возникает, когда данные, предоставленные пользователями, встраиваются в SQL-запросы без надлежащей проверки или очистки. Это позволяет злоумышленнику внедрить вредоносный код и манипулировать запросами, направленными к базе данных.

Современные веб-сайты часто применяют скрипты и SQL для динамической генерации контента страниц. При этом в SQL-запросы нередко

включаются данные, предоставленные пользователями, что злоумышленники могут использовать для внедрения вредоносного SQL-кода через формы ввода. Если защитные механизмы отсутствуют или недостаточны, такой код выполняется на сервере с теми же правами, которыми обладает компонент приложения, ответственный за выполнение запросов (например, сервер базы данных или веб-сервер). Это может привести к тому, что атакующий получит полный контроль над СУБД и операционной системой сервера. Более того, некоторые системы управления базами данных поддерживают выполнение системных команд через SQL [90]. В целом существует несколько разновидностей атак методом инъекций [35].

Можно выделить основные виды SQL- инъекций [38]:

– Классическая SQL Injection – предполагает прямое внедрение вредоносного SQL-кода в запросы, которые отправляются на сервер базы данных. Злоумышленник использует уязвимости ввода для изменения логики запросов и получения доступа к данным или выполнения несанкционированных действий;

– Error-based SQL Injection – использует сообщения об ошибках СУБД для получения информации о структуре базы данных. Ошибки могут содержать детали, такие как имена таблиц или столбцов, что помогает атакующему составить более точные запросы для дальнейшего взлома;

– Boolean-based SQL Injection – основана на анализе ответов сервера (истина/ложь) в зависимости от внедренного кода. Злоумышленник посылает специально сконструированные запросы, чтобы определить, какие части базы данных существуют или какие данные содержатся в ней;

– Time-based SQL Injection – основана на замерах времени отклика сервера. Злоумышленник внедряет запросы, которые искусственно задерживают выполнение ответа. По времени реакции сервера можно судить о корректности внедренного запроса или содержимого базы данных;

– Out-of-band SQL Injection – используется, когда атакующий не может получить данные напрямую через HTTP-ответы, но может заставить

сервер передавать информацию через другие каналы (например, DNS-запросы или HTTP-запросы к внешнему серверу). Этот метод применяется реже и зависит от возможностей СУБД и конфигурации системы.

Пример:

Допустим, веб-приложение использует форму для аутентификации, которая обрабатывается следующим образом:

SQLQuery = “SELECT Username FROM Users WHERE Username = '" & strUsername & "' AND Password = '" & strPassword & "'" strAuthCheck = GetQueryResult(SQLQuery)”

Здесь разработчики напрямую используют значения, введенные пользователем (strUsername и strPassword), для формирования SQL-запроса. Представим, что злоумышленник вводит такие данные:

Login: ' OR "='

Password: ' OR "='

В результате серверу будет передан следующий SQL-запрос:

SELECT Username FROM Users WHERE Username = "OR"="AND Password = " OR "="

Вместо того чтобы сопоставлять введенные логин и пароль с данными из таблицы Users, этот запрос проверяет, равна ли пустая строка самой себе. Поскольку такое сравнение всегда возвращает истину (True), злоумышленник получает возможность войти в систему под именем первого пользователя, находящегося в таблице.

Как правило, выделяют два ключевых способа использования уязвимостей SQL-инъекций: стандартную атаку и слепую атаку (Blind SQL Injection). В первом случае атакующий подбирает параметры запроса, ориентируясь на сообщения об ошибках, которые возвращает веб-приложение.

Пример:

Добавляя оператор union к запросу, злоумышленник проверяет доступность базы данных:

http://example/article.asp?ID=2+union+all+select+name+from+sysobjects. Сервер генерирует сообщение, аналогичное следующему:

“Microsoft OLE DB Provider for ODBC Drivers error '80040e14' [Microsoft][ODBC SQL Server Driver][SQL Server]All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists”

Отсюда можно заключить, что оператор union был успешно отправлен на сервер, и теперь злоумышленнику нужно определить количество параметров, используемых в исходном запросе select.

При проведении слепой SQL-инъекции стандартные сообщения об ошибках заменяются на общие уведомления для пользователя о некорректном вводе. Хотя выполнение SQL-инъекции возможно и в таких условиях, выявление уязвимости становится значительно сложнее.

Наиболее часто используемый способ проверки наличия уязвимости заключается в добавлении условий, которые возвращают истинное или ложное значение. Например, запрос вида http://example/article.asp?ID=2+and+1=1 должен вернуть ту же страницу, что и http://example/article.asp?ID=2 , так как условие 'and 1=1' всегда истинно. Однако, если добавить выражение, которое возвращает ложь, например, http://example/article.asp?ID=2+and+1=0 , сервер либо выдаст сообщение об ошибке, либо не сформирует страницу. Если наличие уязвимости подтверждается, дальнейшая эксплуатация ничем не отличается от стандартного случая.

Известно множество источников информации с описанием и методами эксплуатации уязвимостей SQL Injection [51], [21], [23], [92].

При этом существует несколько достаточно простых программных способов защит от SQL-инъекции.

Во-первых, защититься от атак SQL-инъекций можно за счет тщательной проверки данных, вводимых пользователем. Например, в PHP для этого предусмотрена функция mysql_real_escape_string, которая удаляет из

строки потенциально опасные элементы, используемые в SQL-инъекциях. Рекомендуется применять эту функцию для фильтрации всех данных перед их включением в SQL-запросы.

Во-вторых, можно анализировать запросы на наличие ключевых слов языка SQL, таких как SELECT, UNION, ORDER, CHAR, WHERE или FROM, чтобы выявить подозрительные паттерны.

Кроме того, значительно усложнить задачу злоумышленнику можно, отключив отображение ошибок на экране при некорректных запросах.

Для обеспечения безопасности учетных данных доступа к базе данных рекомендуется вынести функции подключения в отдельный файл и подключать его ко всем страницам сайта.

Защиту от SQL-инъекций также можно усилить, проверяя результат каждого выполненного запроса, включая количество найденных записей. Если запрос не возвращает ни одной записи, пользователя можно перенаправить, например, на главную страницу. Такой подход станет эффективной мерой против SQL-инъекций.

Если сайт не содержит разделов, где выполняется запись или изменение данных в базе, то для пользователя базы данных следует настроить права только на чтение. Для разделов, требующих расширенных прав, рекомендуется создать отдельного пользователя с доступом исключительно для редактирования определенной таблицы. Что касается системы управления контентом (CMS), то для неё стоит завести отдельного пользователя базы данных, обладающего полными правами.

Для защиты сайта от атак, связанных с внедрением кода, необходимо отказаться от хранения паролей для доступа к базе данных в незащищенном виде. Даже если злоумышленник сможет получить эту информацию через SQL-инъекцию, шифрование данных значительно усложнит их использование. Рекомендуется применять хэширование, например, с использованием алгоритмов shal и md5, чтобы повысить безопасность хранения паролей [40].

Поскольку атаки SQL-инъекций используют уязвимое веб-приложение и код базы данных, единственный способ предотвратить их – это устранить уязвимости кода. Любое место, в котором код динамически генерирует SQL-запрос с использованием данных из внешнего источника, должен быть тщательно проверен.

1.3 Обзор и анализ существующих средств обнаружения вторжений типа Buffer Overflow и Code Injection

Главной целью защиты сайтов является создание ресурса, максимально соответствующего стандартам безопасности, либо адаптация существующего сайта путем выявления текущих и потенциальных уязвимостей и проведения необходимых мероприятий для их устранения и предотвращения в будущем. Важно понимать, что обеспечение безопасности сайта включает не только защиту кода и программного обеспечения, но также безопасность процессов управления, надежное хранение учетных данных, защиту от перегрузок, а также решение различных организационных и технических задач. Сайт представляет собой хранилище данных, от которых зависит успешная работа с клиентами, сотрудничество с работниками компании и сторонними организациями, а также рейтинг самой компании. По этой причине очень важно обеспечить безопасность сайта, причем не только веб-страниц, но и всего того, с чем он непосредственно связан. Вся информация, распространяемая и получаемая с помощью сайта, должна быть защищена.

Крупные организации могут себе позволить использовать автоматические средства сканирования исходного кода и сканеры уязвимостей сервисов в Интернете, которые выявляют общие технические уязвимости, такие как угрозы SQL-инъекций, межсайтового скриптинга, фальсификации параметров, переполнения буфера и прочих. Также существуют брандмауэры веб-приложений (Web application firewall, WAF), обеспечивающие защиту за пределами традиционных сетевых брандмауэров

и систем обнаружения/предотвращения вторжений. Многие, например, созданные Imperva Inc. и Barracuda Networks Inc., могут помочь предотвратить распространенные уязвимости в программном обеспечении. Лучшие в своем классе WAF могут распознавать методы обхода СОВ на основе сигнатур (например, обфускация атаки путем кодирования частей введенной команды) и определять является ли трафик нормальным для конкретной сети или нет, и соответствующим образом адаптировать его поведение. При этом в случае обнаружения отклонений, WAF может закрыть потенциальные атаки [67]. Однако такие средства являются достаточно дорогостоящими и используются далеко не повсеместно.

Именно поэтому, в настоящее время задача разработки доступного и универсального средства обеспечения безопасности в Интернете решается многими специалистами и исследователями в связи с чем, существует достаточно много решений, разработок и публикаций в области средств обнаружения распространенных атак на веб-ресурсы.

В настоящем обзоре разработки последних лет систематизированы согласно используемым подходам и технологиям для обнаружения атак. А именно, выделены подходы, основанные на сигнатурах, обнаружении аномалий, комбинированные подходы, технологии, основанные на разборе запроса, моделировании, обнаружении на уровне ОС, а также технологии, основанные на применении методов машинного обучения и нейронных сетей.

1.3.1 Подходы, основанные на сигнатурах

Во многих современных средствах обнаружения вторжений активно используется сигнатурный подход, поэтому существует достаточно много публикаций с разработанными правилами, выражениями и сигнатурами для обнаружения SQL-инъекций [19].

В статье [72] предлагается метод обнаружения атак внедрения кода, основанный на динамическом анализе кода с использованием эмуляции на сетевом уровне. Nemu, прототип системы обнаружения атак, использует эмулятор процессора для динамического анализа правильных

последовательностей команд в проверенном трафике. Основываясь на эвристике поведения во время их выполнения, система идентифицирует шаблоны, характерные для выполнения shell-кода (исполняемого двоичного кода), и, таким образом, может обнаруживать наличие вредоносного кода в произвольных входных данных. Авторы разработали эвристику, которая охватывает наиболее широко используемые типы shell-кода.

В статье [42] обсуждаются различные методы обнаружения и предотвращения атак инъекции кода и предлагается инструмент под названием Code Injection Detection Tool (CIDT), который обнаруживает и предотвращает выполнение таких атак на веб-приложение как SQL инъекции и межсайтовый скриптинг (Cross-Site Scripting, XSS). Предлагаемая система имеет два модуля: детектор скриптов и детектор запросов. HTTP-запрос, поступающий с клиентской стороны, вместо веб-сервера передается на CIDT. CIDT представляет собой прокси-агент на основе политики, который классифицирует полученный запрос как обычный запрос или запрос, содержащий подзапрос, а затем определяет соответствующий тип атаки, если она обнаружена. В случае, когда вредоносное содержимое в запросе найдено любым из модулей, запрос считается недействительным и его выполнение на веб-сервере запрещено.

В статье [43] представлен метод автоматического обнаружения атак переполнения буфера, основанный на классификации сетевого трафика с использованием предопределенного набора сетевых показателей. Предлагаемый набор показателей, характеризующих атаки переполнения буфера, позволяет обходиться без глубокой проверки пакетов (Deep Packet Inspection, DPI). В этой статье описаны два экспериментальных исследования технологии автоматического обнаружения атак переполнения буфера на уязвимые сетевые службы и представлены принципы выбора сетевых показателей и их применения.

1.3.2 Технологии обнаружения на основе разбора и анализа запросов или кода

Большинство исследователей для обнаружения SQL-injection атак предлагают использовать разбор запросов, так, например, в статье [88] описан инструмент DIGLOSSIA, способный эффективно и точно обнаруживать SQL и NoSQL инъекции на PHP-приложения на стороне сервера во время выполнения без каких-либо изменений в приложениях или серверных базах данных. Для определения инъекций кода авторы используют разбор запроса и строгий контроль доступа к операциям с базами данных, чтобы устранить возможность выполнения произвольного кода. DIGLOSSIA точно определяет «код» и «не код» благодаря отслеживанию запроса на уровне символов, и, таким образом, избегает ложных срабатываний. В работе явно определено, что «не кодом» являются только значения (числовые и строковые литералы) и зарезервированные значения (NULL, TRUE и т.д.). «Код» представляет собой все зарезервированные ключевые слова, операторы и вызовы методов, а также идентификаторы (переменные, типы и названия методов). DIGLOSSIA определяет, сгенерирован код приложением или пользователем, посредством двойного парсинга (синтаксического анализа). DIGLOSSIA накладывает незначительные издержки производительности и не требует каких-либо изменений в существующих приложениях, базах данных, веб-серверах или веб-браузерах и может быть легко внедрён.

В работе [89] используется многоуровневый подход, который объединяет синтаксический и семантический анализ запроса, в механизме для обнаружения SQL-injection атак. Этот механизм разделяет основной код сценария и код SQL-инъекций, запрос анализируется на предмет недействительных ключевых слов SQL (например, комментариев, если в SQL-запросе найдены какие-либо комментарии, то он отклоняется и возвращается ошибка). Все уровни архитектуры системы служат для анализа пользовательского ввода и защиты веб-приложений от SQL-инъекций. Когда пользователь предоставляет необходимые данные в веб-формы, на уровне

синтаксической проверки модуль оценки структуры запроса проверяет структуру входного SQL-запроса с помощью XML-кода. Он принимает созданный XML-файл и схему в качестве входных данных и выполняет проверку XML-схемы. Если проверка проходит успешно, структура запроса считается правильной. Далее следует уровень семантической проверки и защиты от логически некорректных, неразрешенных запросов, с помощью которых злоумышленник может получить информацию о вводимых параметрах, типах данных столбцов в таблице, названии таблиц и т.д. На уровне настройки создания ошибок имеется четыре модуля проверки. Если строка SQL проходит проверку, то строка передается модулю обработки ошибок, где выполняется запрос. После выполнения строки SQL модуль обработки запросов возвращает результат в виде набора данных веб-серверу. Когда злоумышленник пытается получить информацию о структуре таблицы или другие данные с помощью некорректного запроса, модуль настройки ошибок обрабатывает это как SQL-ошибку и отправляет соответствующее сообщение на веб-сервер для сбора данных о клиенте. Авторы утверждают, что данный метод универсален и совместим с любыми типами баз данных, независимо от платформы.

В статье [58] предлагается метод обнаружения атак с внедрением SQL-кода на основе комбинации статического и динамического анализа. Суть метода заключается в удалении значений атрибутов в SQL запросах во время выполнения (динамический подход) для анализа и сравнении статических запросов SQL с динамически генерируемыми запросами после удаления значений атрибутов (статический подход). Предложенный метод просто удаляет значения атрибутов, что делает его независимым от СУБД. В предлагаемом методе не используются сложные операции, такие как разбор деревьев или организация отдельных библиотек.

Для обнаружения SQL-инъекций в [87] используют подход, основанный на динамическом анализе SQL-запроса и семантическом сравнении проверяемого запроса пользователя и соответствующего штатного запроса.

Сравнение выполняется путем разбора каждого из операторов и сравнения структуры синтаксического дерева. Если синтаксические деревья обоих запросов эквивалентны, то заключается, что проверяемый запрос должен вызвать эквивалентные семантические действия на сервере базы данных и не содержит SQL-инъекции. Отличие синтаксических деревьев является признаком атаки SQL-инъекция и запрос не выполняется.

В статье [83] предлагают метод обнаружения атак переполнения буфера путем анализа полезной нагрузки сетевых пакетов в поисках shell-кода, который является удаленно исполняемым компонентом атаки переполнения буфера. Анализируя shell-код, можно определить, какие системные вызовы использует эксплойт (фрагмент программного кода, использующий уязвимости в программном обеспечении для проведения атаки), и, следовательно, работу эксплойта. Данный подход в состоянии обнаружить не только существующие атаки переполнения буфера, но и ранее не обнаруживаемые сигнатурными методами без потребности в определении сигнатур для каждой новой атаки. Метод был реализован и протестирован для атак переполнения буфера на Linux на архитектуре Intel x86 с использованием Snort NIDS. Авторами предлагается препроцессор, который позволяет обнаруживать новые атаки переполнения буфера с небольшим количеством ложных срабатываний. Решение предоставляет подробную информацию об атаках и для поиска уязвимой службы. Кроме того, плагин вместе со Snort обеспечивает возможность захвата атак, что может быть использовано для составления сигнатур. Препроцессор в настоящее время ориентирован только на обнаружение атак переполнения буфера на архитектуре x86 под управлением Linux, однако предварительные тесты показали, что этот подход может быть применим к FreeBSD, а также к вариантам UNIX. В будущем авторы планируют разработать решение и для других операционных систем и архитектур, чтобы детектор мог быть сконфигурирован в соответствии с серверами, работающими в сетях. Для тестирования детектора будет использоваться более обширный набор атак переполнения буфера. В будущем

также планируются исследования возможных способов оптимизации этого подхода. Последующие версии решения могут также представлять собой уже виртуальный процессор для анализа shell-кода для идентификации атак, использующих обфускацию.

В статье [99] предлагается инструмент уровня приложений для обнаружения и блокировки атак переполнения буфера - SigFree, который не требует каких-либо предварительно известных шаблонов, поэтому он может блокировать новые, неизвестные атаки в режиме реального времени. Идея заключается в том, что атаки переполнения буфера обычно содержат исполняемые файлы, тогда как законные клиентские запросы никогда не содержат исполняемых файлов. Алгоритм обнаружения кода в SigFree основан на методе анализа потока данных для анализа полезной нагрузки пакета и определения действительно ли пакет содержит код. SigFree блокирует атаки, обнаруживая наличие кода. SigFree свободен от сигнатур, поэтому он может блокировать новые и неизвестные атаки переполнения буфера. SigFree характеризуется экономичным развертыванием в Интернете, низкой стоимостью развертывания и обслуживания. Тестирование SigFree показало, что он может блокировать все пакетные типы атак с внедрением кода (выше 750) с небольшим количеством ложных срабатываний.

1.3.3 Технологии обнаружения, основанные на моделировании

В статье [62] предлагается система обнаружения вторжений Double Guard с облегченной виртуализацией, которая моделирует поведение сети для многоуровневых веб-приложений пользовательских сессий и способна идентифицировать SQL-атаки и XSS-атаки. Использующаяся в системе модель сопоставления (Mapping Model) разработана для сопоставления запросов веб-сервера и последующих запросов БД, выявления различных типов атак и минимизации ложных срабатываний, как для статического, так и для динамического веб-приложения. В модели Double Guard только законные пользователи могут получить доступ к веб-приложению.

1.3.4 Технологии обнаружения на уровне ОС

В статье [63] предлагается платформа, которая использует аппаратную виртуализацию Kernel-based Virtual Machine (KVM) для быстрого и точного обнаружения атак внедрения кода. Преимущества аппаратной виртуализации заключаются в обеспечении эффективной и точной проверке буферов путем непосредственного выполнения последовательностей инструкций на ЦП. Благодаря такому подходу не требуется отслеживать каждую инструкцию машинного уровня или выполнять небезопасную оптимизацию. Авторы планируют разработку ядра – основу для построения средств защиты от атак, называемую ShellOS, которая работает как гостевая ОС с использованием KVM, и чья единственная задача состоит в обнаружении и анализе атак с внедрением кода.

В статье [82] описан способ обнаружения известных атак внедрения кода, использующих технологии обхода современных систем обнаружения вторжений (obfuscated code injection attacks), и ранее неизвестных атак. Способ заключается в выполнении тех сегментов сетевого трафика, которые потенциально могут содержать исполняемый код, и мониторинге его выполнения для обнаружения вызовов операционной системы. Авторы внедрили прототип детектора на хосте Linux, модифицировав Snort, и показали, что выполнение кода в песочнице, отслеживание его и запись системных вызовов позволяет обнаруживать атаки инъекции кода независимо от используемых методов обфускации.

В статье [94] представлен новый подход для обнаружения атак внедрения кода - Bee Master. Bee Master применяет парадигму honeypot к процессам ОС. Основная идея состоит в том, чтобы выставить регулярные процессы ОС как приманку для вредоносных программ.

Несмотря на значительные исследовательские и инженерные усилия [48], [52], [54], [66], [85], [96], атаки переполнения буфера остаются серьезной угрозой безопасности.

Практически все приведенные в обзоре способы и другие: [93], [75], [84], [47], предполагают либо инспекцию содержимого пакетов, либо отслеживание состава запросов, либо формирование базы шаблонов атак и поиск соответствующих признаков, что неизменно влечет за собой затраты на обслуживание в виду активно появляющихся модификаций атак и новых способов обхода СОВ на основе сигнатур. Наиболее распространенными являются передовые методы обфускации кода, самомодифицирующийся код, полиморфизмы. Кроме того, атаки внедрения кода зачастую достаточно индивидуальны для каждого веб-сервиса и планируются целенаправленно. Все это существенно осложняет успешное обнаружение атак существующими способами.

В связи с этим наиболее перспективным и соответствующим современным реалиям направлением для создания эффективной системы обнаружения атак с внедрением кода является обнаружение аномалий сетевой активности, технология, нейтрализующая основной недостаток сигнатурных методов – обнаружение лишь известных атак.

1.3.5 Подходы, основанные на обнаружении аномалий

Системы обнаружения аномалий оповещают о событиях, отличающихся от установленных профилей нормального поведения, которые могут быть потенциальными вредоносными действиями. Основным преимуществом систем обнаружения аномалий является их способность обнаруживать ранее неизвестные атаки, так как они не полагаются на какие-либо знания о специфике известных атак или вторжений.

Например, для обнаружения аномалий в SQL-запросах в научной статье [27] предложен подход, состоящий из трех этапов: создание вектора характеристик на основе разрешенных (корректных) SQL-запросов из обучающего набора; группировка этих запросов с использованием кластеризации на основе векторов признаков; обучение рекуррентной нейронной сети долгой краткосрочной памяти (LSTM) для классификации на основе результатов кластеризации. Основная идея заключается в том, что

SQL-запросы из обучающего набора можно разделить на несколько категорий, и если анализируемый запрос не соответствует ни одной из них, он считается аномальным. Вектор характеристик формируется на основе SQL-запросов обучающего набора, специфичного для конкретного приложения. Каждый элемент вектора представляет собой значение от 0 до 1, которое указывает частоту появления определенной лексемы языка SQL в запросе. Если лексема отсутствует в новом запросе, но присутствует в обучающем наборе, ей присваивается значение 0. Таким образом, для обучающего набора SQL-запросов строится матрица размером m×n, где m – количество запросов, а n – количество уникальных лексем SQL в выборке. На этапе классификации LSTM-сеть получает вектор признаков нового запроса и возвращает два значения: номер категории, с которой запрос имеет наибольшее сходство, и число от 0 до 1, представляющее вероятность принадлежности запроса к этой категории.

В статье [86] предлагается платформа для выявления SQL-инъекций, которая основывается на методах выявления некорректного использования и отклонений от нормы. Главная концепция платформы состоит в формировании профиля допустимого поведения базы данных. Для обнаружения атак, связанных с SQL-инъекциями, применяется метод анализа данных, который фиксирует определенные «следы» SQL-операторов, используемых в процессе работы системы. Эти данные помогают идентифицировать подозрительные запросы. Любой входящий запрос, «след» которого не совпадает ни с одним из тех, что есть в разрешенном наборе, с большой вероятностью считается вторжением.

В статье [56] предлагается модель обнаружения SQL инъекций на основе аномалий, которая наблюдает за трафиком базы данных на стороне самого сервера базы данных. Предложенная автором модель обнаружения аномалий может использоваться в сочетании с существующими методами, чтобы снизить риск внедрения SQL инъекций.

Однако известно, что подходам, основанным на обнаружении аномалий, свойственна высокая частота ложных срабатываний. Поэтому зачастую с целью объединения достоинств и взаимной нейтрализации недостатков используют комбинированные подходы.

1.3.6 Комбинированные технологии

В проекте [14] описана система, которая анализирует данные, вводимые пользователями через HTML-формы, и выявляет потенциальные паттерны атак. При обнаружении подобного шаблона система блокирует атаку и фиксирует инцидент в логах. Если злоумышленник продолжает отправлять аналогичные паттерны, система полностью ограничивает доступ для этого пользователя. Механизм блокирует пользователей, которые пытаются войти в систему с аномально высокой скоростью. Система использует как сигнатурные, так и аномальные модели обнаружения для выявления угроз и атак SQL-Injection (White Space Manipulation attack, Comments attack, String Concatenation attack, UNION Injection attack) на систему. В базе данных системы собраны образцы шаблонов таких атак, например, система ищет символы комментариев, символы операции конкатенации или функции concat, ключевое слово “UNION” ищется системой для обнаружения атаки UNION Injection. Система также должна искать двоичные, шестнадцатеричные и десятичные символы в представленном тексте, чтобы обнаружить соответствующие вариации атаки SQL-Injection. Обнаружение аномалий основано на подсчете количества попыток пользователя получить доступ к системе, независимо от того, были ли обнаружены какие-либо шаблоны атаки SQL-Injection. Когда количество посещений превышает заданный порог, система автоматически блокирует пользователя на время. Кроме того, система ограничивает передачу ненужной информации, то есть пользователю не выдаются сообщения об ошибках, что также уменьшает вероятность атаки.

Согласно исследованию в [61] применение технологий машинного обучения является одним из лучших методов среди других методов обнаружения атак типа SQL-injection.

1.3.7 Подходы, основанные на применении методов машинного обучения

Авторами статьи [65] предложена система обнаружения SQL-атак с применением методов машинного обучения. На основе SQL-запросов, сгенерированных веб-приложением, были созданы параметры для модели обнаружения. Система, основанная на аномалиях, использует ряд различных моделей для обучения на профилях штатного доступа к базе данных. Затем, SQL-запросы сравниваются со сгенерированной моделью для проверки на несоответствие. Эти модели позволяют обнаруживать неизвестные атаки с небольшим числом ложных срабатываний и ограниченными накладными расходами. Однако если модель будет не эффективно обучена, то может возникнуть множество ложных срабатываний. Данное решение лишь уменьшает возможность выполнения SQL-атак.

В статье [100] для обнаружения атак инъекции кода на приложения на основе HTML5 предлагается использовать алгоритмы классификации машинного обучения. Экспериментальный результат показал, что точность данного метода обнаружения достигает 95,3 процента. Также предлагается усовершенствованная модель контроля доступа для уменьшения ущерба от атаки. Кроме того, применяются фильтры для удаления кода JavaScript из данных, чтобы предотвратить атаки. Эффективность и рациональность были подтверждены с помощью моделирования.

1.3.8 Технологии обнаружения на основе применения искусственных нейронных сетей

В статье [74] предлагается нейросетевая модель распознавания образов для обнаружения и классификации SQL-injection атак. Предлагаемая модель состоит из трех основных элементов: генератора URL-адресов, чтобы генерировать тысячи вредоносных и доброкачественных URL-адресов, классификатора URL-адресов для того, чтобы: 1) классифицировать каждый сгенерированный URL-адрес (доброкачественный или вредоносный URL-адрес и 2) классифицировать вредоносные URL-адреса в соответствии с

разными категориями SQL-injection атаки, и нейросетевой модели, чтобы: 1) обнаруживать в данном URL вредоносный или доброкачественный URL-адрес и 2) определять тип атаки SQL-injection для каждого вредоносного URL-адреса. Модель сначала обучалась в три этапа, а затем оценивалась с помощью тысяч доброкачественных и вредоносных URL-адресов. С помощью данной модели можно выявлять семь типов SQL-injection атак на основе сигнатур этих семи популярных типов SQL-injection атак. По результатам экспериментов предложенная нейросетевая модель для обнаружения и классификации атаки SQL-injection показывает хорошую производительность с точки зрения точности, ошибок первого и второго рода.

В статье [101] предлагается гибридная сеть глубокого обучения (Hybrid Deep Learning Network, HDLN) для обнаружения атак с внедрением кода на гибридные приложения на основе HTML5. Сначала авторы извлекают новые возможности из абстрактного синтаксического дерева (Abstract Syntax Tree, AST) JavaScript и используют три метода для выбора ключевых функций и характеристик, среди которых тест Хи-квадрат. После получения векторов функций и HDLN-сеть обучается отличать уязвимые приложения от нормальных. Результаты экспериментов для проверки метода показали, что подход к обнаружению с HDLN достигает 97,55 процента в точности и 97,60 процента в AUC, что превосходит другие традиционные классификаторы и обладает более высокой среднюю точность, чем другие методы обнаружения.

В статье [81] предложена система обнаружения и блокировки SQL-инъекций с помощью мультиагентной системы с использованием искусственной нейронной сети (MLP).

Стоит отметить, что системы обнаружения вторжений, зарегистрированные в реестре российского программного обеспечения, в большинстве своем используют сигнатурные методы [5]. А в так называемых системах обнаружения аномалий аналитика зачастую ограничивается данными не детальнее типа протокола. Ранее, когда индустрия мошенничества и хакинга в Интернете не была столь распространена, а злоумышленники

пытались эксплуатировать только уже известные уязвимости ПО, использования поверхностного анализа и сигнатурных методов хватало для защиты сетевых ресурсов. Однако учитывая особенности современных атак, которые могут быть растянуты во времени, так называемые APT (Advanced Persistent Threat), трафик которых маскируется путем шифрования и обфускации (запутывания), сигнатурные методы малоэффективны. Кроме того, современные атаки используют различные способы обхода IDS. При этом трудозатраты на конфигурирование и поддержку традиционных систем обнаружения вторжений уже сейчас превышают разумные пределы, и вскоре они могут стать нецелесообразными.

В данных условиях растет перспективность исследований в области применения технологий обнаружения аномалий для обнаружения атак, поэтому остановимся подробнее на обзоре существующих моделей обнаружения аномалий.

1.4 Модели обнаружения сетевых атак на основе аномалий

В настоящее время большинство работ, связанных с СОВ, оценивающих адресованные к БД SQL-запросы, посвящено методам обнаружения аномалий. Технологию обнаружения вторжений, основанную на обнаружении аномалий, удобнее охарактеризовать в сравнении с сигнатурным подходом. Например, СОВ, основанные на сигнатурах, не способны распознать аномалии, угрозы или признаки, которые явно не описаны в их базе данных, даже если они незначительно отличаются от известных образцов. В то же время СОВ, работающие на основе анализа аномалий (anomaly-based IDS), используют заученное нормальное поведение сети как эталон для выявления отклонений. Однако такие системы склонны к частым ложным срабатываниям, тогда как у сигнатурных СОВ такие ошибки практически не встречаются. Это связано с тем, что сформировать всеобъемлющий профиль нормального поведения достаточно трудоемкая задача, а обнаруженные сетевые аномалии могут быть

связаны как с деятельностью злоумышленников, некомпетентных пользователей, так и с программно-аппаратными неисправностями (неисправностью аппаратуры и дефектами программного обеспечения).

Другими словами, можно сказать, что сигнатурные СОВ ищут в потоке трафика известные признаки поведения атак, а СОВ на основе аномалий пытаются отслеживать отклонения от нормальных признаков.

Поскольку число векторов несанкционированных проникновений в Интернете за последние годы значительно увеличилось, осложняя эффективное и универсальное применение сигнатурных методов обнаружения, возросла и потребность в разработке новых технологий обнаружения вторжений, устраняющих существующие недостатки. Сейчас исследователи уделяют больше внимания разработке систем обнаружения, основанных на анализе аномалий, поскольку они позволяют выявлять как уже известные, так и новые, ранее неизученные отклонения.

По технике обнаружения аномалий можно выделить три основные модели:

– Модель обнаружения без учителя (Unsupervised anomaly detection) [60]. Системы, работающие с использованием подобных техник, не требуют априорных знаний о данных, а значит, наименее эффективны. Алгоритм распознавания в режиме без учителя базируется на предположении о том, что аномальные выбросы встречаются гораздо реже нормальных данных. После обработки всех данных, наиболее отдаленные определяются как аномалии. Для применения такой модели необходимо иметь весь набор данных, т.е. она не может применяться в режиме реального времени. Кроме того, в случаях если предположение неверно, то такая система обнаружения будет генерировать слишком много ложных срабатываний;

– Модель обнаружения частично с учителем (Semi-supervised anomaly detection) [8]. Такая модель основана на исходных данных, представляющих собой исключительно нормальный класс. Алгоритмы, работающие в режиме распознавания частично с учителем, не требуют

информации о классе аномальных экземпляров. Поскольку создание модели исключительно для нормального поведения является более простой задачей (из-за невозможности предугадать все возможные аномалии), этот подход применяется наиболее часто. Обучившись на одном классе, система может определять принадлежность новых данных к нему, что и позволяют обнаруживать данные, не принадлежащие к нормальному классу, то есть аномалии. Таким образом такая модель способна распознавать отклонения и потенциальные вторжения в отсутствие заранее определенной информации о них;

– Модель распознавания с учителем (Supervised anomaly detection). В модели такого типа предполагается, что есть два класса сущностей: нормальное и аномальное поведение. Данная методика требует обучения как на основе свободного от атак трафика, так и на основе трафика, содержащего помеченные атаки. То есть обучающая выборка должна полноценно представлять систему и включать экземпляры данных нормального и аномального классов. Работа алгоритма происходит в два этапа: обучение и распознавание. На первом этапе по обучающей выборке строится модель, содержащая нормальный и аномальный класс, после чего новые, не изученные данные, не имеющие метки, сравниваются с каждым классом для принятия решения, к какому из них данные принадлежат. В большинстве случаев предполагается, что данные не меняют свои статистические характеристики, иначе возникает необходимость изменять классификатор [8]. Применение такой модели осложняется необходимостью формирования данных для обучения, что не всегда возможно. Зачастую аномальный класс представлен значительно меньшим числом данных, чем нормальный, что может приводить к неточностям в полученной модели.

Методы обнаружения аномального поведения, чаще всего использующиеся в системах обнаружения вторжений, основаны на предположении о том, что на базе совокупности неких параметров сетевого трафика можно сформировать «образ» нормального функционирования и

любые значительные отклонения от него увеличивают вероятность, что это атака. Для построения нормального профиля используется набор данных свободный от аномалий.

В зависимости от способа формирования профиля нормального поведения для выявления SQL-injection можно выделить следующие подходы:

– формирование профиля путём синтаксического анализа текста SQL-запроса;

– формирование профиля путем семантического анализа SQL-запроса;

– формирование профиля на основе различных характеристик запроса (лексических, темпоральных, ресурсных и др.) [33];

– формирование профиля на основе наиболее характерной статистической информации;

– формирование модели профиля, например, путем обучения нейронной сети значениям признаков, характеризирующих штатное функционирование.

Рассмотрим подробно лишь основные методы обнаружения аномалий, а именно:

– Поведенческие методы;

– Методы обнаружения на основе анализа содержимого [72];

– Методы интеллектуального анализа данных.

Каждый класс включает в себя достаточно много различных методов, некоторые из них будут подробнее изложены ниже.

1.4.1 Поведенческие методы

Наиболее часто встречающимся представителем поведенческих методов определения нарушений в сети является статистических анализ, в основе которого лежит сопоставление текущего состояния сетевой инфраструктуры с неким заранее описанным посредством признаков состоянием, считающимся штатным для конкретной сети. Методы статического анализа имеют

различные интерпретации, но все они основаны на получении и накоплении каких-либо динамических характеристик сетевого трафика для формирования профиля, соответствующего типичному поведению. Сложность подхода заключается только в том, что нужно найти определенные структурированные характеристики, однозначно определяющие корректную конфигурацию и функционирование объекта защиты, сформулировать адекватные критерии определения потенциальной угрозы и контролировать изменения выбранных статистических характеристик объекта, например, трафика.

Так, применительно к статистическому анализу сетевого TCP/IP трафика необходимо выделить основные показатели, характеризующие штатное состояние сети и по которым можно интерпретировать и анализировать историю сетевого взаимодействия. В ранних подходах к обнаружению сетевых вторжений на основе аномалий моделировали статистические свойства трафика с использованием информации из заголовка пакета или об уровне соединения. Для создания профиля могут использоваться поля заголовков TCP, UDP, IP, ICMP протоколов и содержимое полей данных. Статистические модели для TCP-подключений строились на основе информации о номерах портов, IP-адресах и сведений о состоянии TCP-соединения.

Для формирования профиля могут применяться различные методы, например, методы математической статистики [24]. При этом нетривиальной оказывается задача определения пороговых значений, её решение требует глубоких знаний о защищаемом объекте. После формирования профиля осуществляется динамический контроль состояния по выделенным признакам и при обнаружении существенных отклонений должен подаваться сигнал об обнаружении аномалии, которая может свидетельствовать о начале атаки. Для обнаружения отклонений обычно используют классификатор для классификации трафика. Другими словами, в таких системах главным сигналом для выявления аномалии служит существенное расхождение между локальными статистическими параметрами и их глобальными аналогами. Это

отклонение может быть зафиксировано классификатором во время этапа анализа на предмет аномалий.

Основным достоинством использования таких методов является отсутствие необходимости в обновлении базы сигнатур, что значительно экономит человеческие и вычислительные ресурсы, упрощая задачу сопровождения системы обнаружений и позволяя обнаруживать ранее неизвестные, сложные и распределенные во времени атаки. Системы, использующие статистические алгоритмы, способны адаптироваться к изменениям, однако это достоинство превращается в недостаток, которым может воспользоваться злоумышленник для постепенного «приручения» системы к новому поведению, включающему несанкционированную деятельность. Кроме того, неотъемлемым недостатком использования статистического анализа является принятие предположения о том, что наблюдения независимы и не меняют свои статистические характеристики, являются одинаково распределенными. Это затрудняет обнаружение аномалий в любых видах нестационарных данных, ведь, как известно, интернет-трафик зависит от периодических изменений и серийной корреляций [39]. Системы обнаружения на основе статистических методов достаточно чувствительны, что повышает вероятность ложных срабатываний в сравнении с другими методами. Также, известно, что существуют подходы для обхода защиты, организованной с помощью таких систем.

Для реализации поведенческих методов обнаружения аномалий также известны применения вейвлет-анализа, фрактального и мультифрактального анализа [34], анализа энтропии, спектрального анализа и многих других [39].

Основным недостатком поведенческих методов, базирующихся на описании профиля с помощью статистических характеристик данных, содержащихся в заголовках пакетов, является то, что такого описания недостаточно для обнаружения вторжений на сетевые ресурсы, предлагающие общественные услуги, сайты и связанные с ними сервера. Дело в том, что такие сервера получают множество запросов в час и значительных изменений

во время атаки по сравнению с нормальным профилем, описанном на основе поверхностных данных, может не быть. А именно публичные ресурсы являются основной мишенью для вторжений и атак. В связи с этим исследования в области обнаружения аномалий на сетевом уровне были направлены на улучшение моделирования доброкачественного трафика, за счет использования других атрибутов сетевых пакетов.

1.4.2 Обнаружение аномалий на основе содержимого

В методике обнаружения аномалий на основе контента вместо моделирования атрибутов заголовков пакетов или сетевых соединений получают статистические модели, которые пытаются захватить семантическую информацию протоколов прикладного уровня, учитывая полезную нагрузку наблюдаемых пакетов. Одна из первых систем, использующих для обнаружения аномалий полезную нагрузку (payloads) пакета, была представлена в [50]. Предлагаемая система использует статистическое обнаружение аномалий в сочетании со специальными знаниями для поиска ранее неизвестных вредоносных пакетов, предназначенных для сетевых служб. На этапе обучения модели нормального трафика строятся отдельно для каждого из защищенных сетевых сервисов, что имеет ключевое значение для получения полезных профилей, которые не дадут слишком много ложных срабатываний. Значительные отклонения отслеживаемого трафика от полученных моделей нормального поведения указывают на потенциальные атаки. Статистические профили формируются только на основе пакетов запросов на обслуживание, оценка аномалии для которых вычисляется с использованием типа и длины запроса и распределения полезной нагрузки. Существуют также и другие решения, использующие аналогичные подходы, например, [73] и [6]. В [6] представлена система обнаружения аномалий на основе контента (PAYL), основанная на моделировании доброкачественного трафика. В PAYL вычисляется целый набор моделей, основанных на распределении байтов полезных нагрузок пакетов. При этом распределения байтов должны быть получены отдельно для

разных комбинаций хоста, порта и длины пакета, так как разные хосты и сетевые службы имеют разные шаблоны трафика, а для одного и того же протокола разные диапазоны длины пакетов имеют разные типы полезной нагрузки. После фазы обучения, при нормальной работе PAYL вычисляет сходство распределения полезных нагрузок входящих пакетов с моделями распределения байтов нормального поведения, используя расстояние. Однако обнаружение SQL-injection атак в тестирование системы не входило.

В работе [70], основанной на трудах [49] описан подход, основанный на обнаружении аномалий, который использует распределение символов определенных разделов HTTP-запросов для обнаружения SQL-инъекций. Подход не требует взаимодействия с пользователем, изменений или доступа к базе данных или исходному коду самого веб-приложения.

1.4.3 Методы интеллектуального анализа данных

Применение методов интеллектуального анализа данных (Data Mining или ИАД) – популярное в научно-практической сфере направление, получившее широкое распространение в системах обнаружения вторжений благодаря достаточно высокой результативности.

Задача, которую выполняет ИАД, заключается в выявлении значимых корреляций, тенденций, неочевидных, объективных, и полезных на практике закономерностей в разнородных, сложно структурированных данных большого объема. Под неочевидными закономерностями понимаются те, которые не обнаруживаются стандартными методами обработки информации или субъективным экспертным путём. Решение этой задачи крайне актуально для всеобъемлющего, глубокого, точного анализа и описания объектов, построения моделей, классификации и систем принятия решений. Способность самостоятельно находить такие закономерности и строить гипотезы о взаимосвязях, что само по себе является чуть ли не самой сложной задачей в этой области, выгодно отличает ИАД от других методов «грубого» и поверхностного анализа.

В общем, ИАД состоит из трех стадий: исследование набора данных с последующим выявлением закономерностей и их валидации, использование найденных закономерностей для классификации и/или прогнозирования, анализ исключений для выявления и интерпретации аномалий.

Среди наиболее популярных методов (ИАД) выделяют искусственные нейронные сети, деревья решений, символьные правила, методы ближайшего соседа и k-ближайших соседей, метод опорных векторов (SVM), байесовские сети, линейную регрессию, корреляционно-регрессионный анализ, иерархические и неиерархические методы кластеризации (например, k-средних и k-медиан), алгоритмы поиска ассоциативных правил (например, Apriori), методы ограниченного перебора, эволюционное программирование, генетические алгоритмы, а также различные способы визуализации данных и множество других подходов [59]. Тем не менее, все методы и алгоритмы ИАД в общем виде можно условно разделить на две категории: статистические и кибернетические [39].

В статистических методах можно выделить четыре основные группы:

– Дескриптивный анализ и описание исходных данных;

– Анализ связей (корреляционный, регрессионный, факторный, дисперсионный анализ);

– Многомерный статистический анализ (компонентный анализ, дискриминантный анализ, многомерный регрессионный анализ, канонические корреляции и др.);

– Анализ временных рядов (динамические модели и прогнозирование).

К кибернетическим методам относятся:

– Искусственные иммунные системы и нейронные сети (ИНС) [36];

– Эволюционное программирование;

– Генетические алгоритмы;

– Ассоциативная память;

– Деревья решений;
– Нечеткая логика;
– Системы обработки экспертных знаний.

Среди всех упомянутых методов особого внимания заслуживают подходы, основанные на нейронных сетях. Эти технологии имитируют работу биологических нейронных сетей головного мозга и обладают такими характеристиками, как способность к обучению на основе прошлого опыта, выявлению скрытых закономерностей в данных и обобщению известных примеров для применения в новых ситуациях. Известно множество разновидностей искусственных нейронных сетей (ИНС) [37], различающихся по числу промежуточных слоев, наличию или отсутствию обратных связей, а также по типу связности нейронов в скрытых слоях – они могут быть полносвязными или иметь произвольные связи.

Нейронные сети в контексте обнаружения аномалий сетевой активности могут использоваться для построения модели нормального профиля сетевого трафика с последующим решением задач распознавания, классификации образов и обнаружением аномального поведения, которое не удается обнаружить распространенными СОВ на основе сигнатур. В процессе обучения на параметрах, описывающих нормальное функционирование сети в условиях отсутствия атак, нейронная сеть адаптируется к заданным характеристикам и сохраняет это состояние как эталонное. Впоследствии, при работе с текущими параметрами состояния сети, она способна обнаруживать любые отклонения от заданного уровня или критериев «нормального» поведения.

Подводя итог, можно отметить, что процесс создания СОВ с применением нейронных сетей включает несколько ключевых шагов. На первом этапе выполняется сбор и предварительная обработка данных, а именно перехват сетевого трафика и извлечение нужной информации из заголовков пакетов. Далее «сырые» данные приводятся к виду, который сможет принять классификатор. На этом этапе осуществляется обработка или

вычисление параметров для формирования обучающей выборки. Третий этап заключается в построении и обучении нейронной сети. И наконец на последнем этапе происходит тестирование сети, анализ результатов и обнаружение сетевых аномалий трафика.

Зачастую СОВ на основе нейросетевых технологий ориентированы на построение шаблона поведения пользователя, используя для обучения нейронной сети такие параметры как время, набор узлов, с которым обычно работает пользователь, характеристики использования ресурсов системы [39]. При этом выбранные параметры подаются на вход сети обратного распространения ошибки (backpropagation neural network, BPNN). Нейронная сеть обучается с учителем на парах («нормальные» параметры, 0) и («аномальные» параметры, 1).

При разработке СОВ для выявления и анализа сетевых атак часто применяется многослойный персептрон (MLP), обучение которого выполняется с использованием метода обратного распространения ошибки. Альтернативным подходом может служить искусственная нейронная сеть (ИНС) встречного распространения, состоящая из двух слоев: нейронов Кохонена, которые способны выявлять статистические закономерности в данных, и нейронов Гроссберга, ответственных за обобщение и уточнение результатов. Обучение такой сети осуществляется в два этапа: слой Кохонена тренируется без учителя, а нейроны Гроссберга настраиваются на основе данных первого слоя с применением комбинации алгоритма обратного распространения ошибки и метода Коши.

Необходимо отметить, что ключевой задачей при построении таких СОВ является выбор наиболее значимых параметров для обучения и успешного обнаружения того или иного типа атак, структуры ИНС и метода обучения, от которых зависит качество обучения сети, сходимость и успех решения поставленной задачи.

Чтобы выявить ключевые параметры и основные компоненты, чаще всего применяется рециркуляционная нейронная сеть (RNN). Она

представляет собой многослойный персептрон, который выполняет линейное или нелинейное преобразование входных данных, сжимая их через «узкое место» в скрытом слое.

Подробнее о применении нейронных сетей в СОВ можно узнать в [39].

В контексте обнаружения аномалий в сетевом трафике нельзя не отметить перспективность и распространенность применения методов машинного обучения [15].

Подводя итоги, приведем классификацию моделей обнаружения аномалий в сетевой активности по режиму обнаружения (рисунок 1) и по способу формирования нормального профиля (рисунок 2), а также классификацию основных методов обнаружения аномалий сетевой активности на рисунке 3, расширенную с учетом материалов [1], [16].

Рисунок 1 - Классификация моделей обнаружения аномалий по режиму обнаружения

Рисунок 2 - Классификация моделей обнаружения аномалий по способу формирования нормального профиля

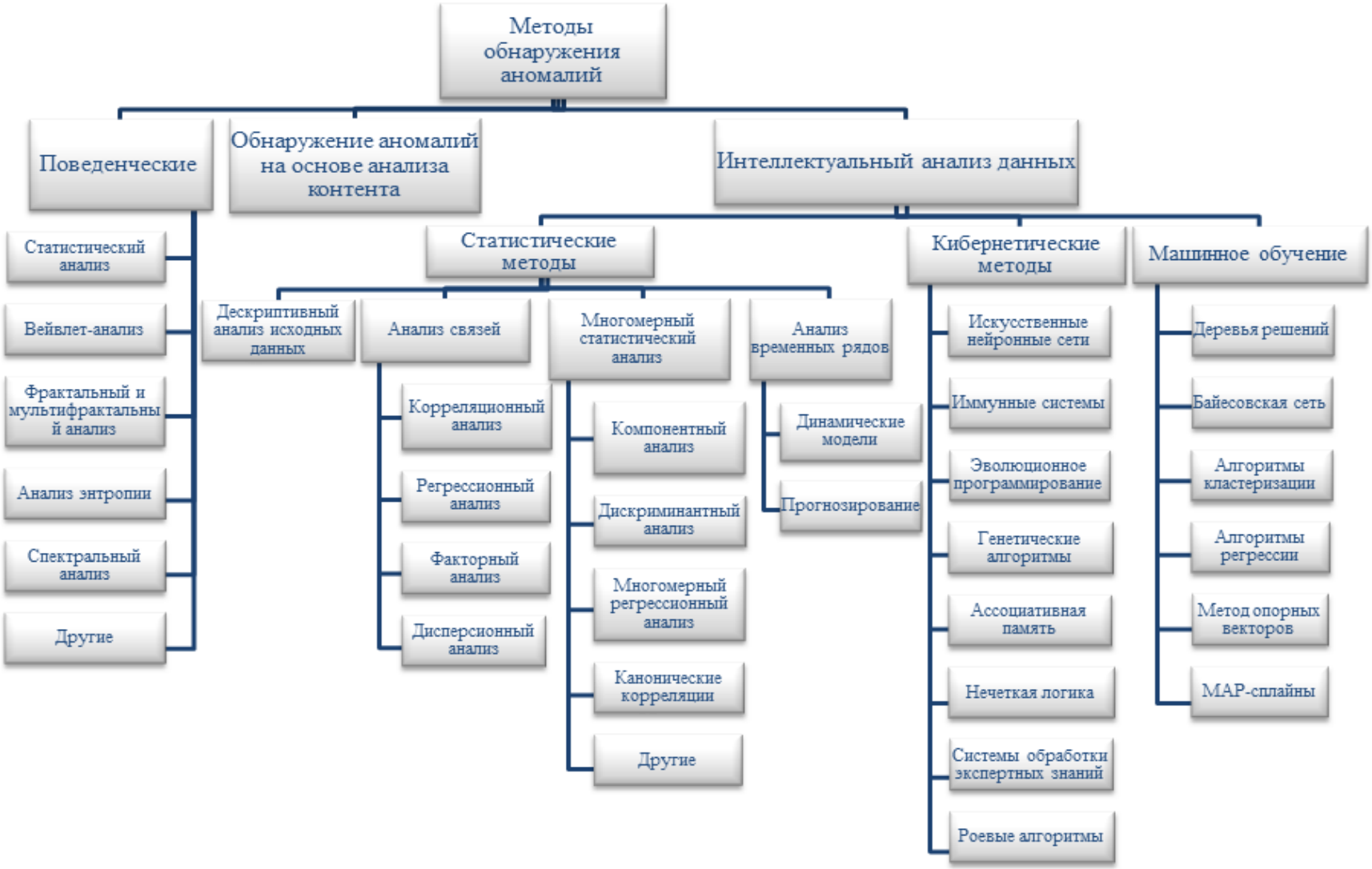


Рисунок 3 - Классификация основных методов обнаружения аномалий

Следует заметить, что большинство современных решений не ограничиваются строгим использованием какого-то одного метода. Напротив, зачастую использование гибридных методов оказывается гораздо эффективнее в силу объединения достоинств и взаимного устранения недостатков. Однако список самых опасных рисков (уязвимостей) веб-приложений от 2013 до 2017 года [76] неизменно возглавляет уязвимость внедрения кода, что говорит о том, что, не смотря на огромное количество предлагаемых решений и практик, универсального средства, позволяющего эффективно защитить веб-сайты от атак внедрения кода не разработано до сих пор. В связи с этим задача разработки нового метода обнаружения самого распространенного вида атаки на веб-ресурсы является более чем актуальной.

1.5 Итоги и постановка задачи

По результатам систематизации и анализа материалов обзора существующих подходов и методов обнаружения атак типа SQL-injection и Buffer Overflow, а также подходов обнаружения аномалий, составленного по публикациям последних лет, можно заключить следующее.

Большинству используемых в настоящее время систем обнаружения вторжений в целом свойственны следующие недостатки:

– Обработка больших объемов данных в реальном времени требует в значительной степени мощных аппаратных и вычислительных ресурсов;

– Для эффективной работы сигнатурных СОВ необходимо поддерживать базу сигнатур в актуальном состоянии, что само по себе трудозатратно. При этом обновление происходит лишь по прошествии некоторого времени, требующегося на обнаружение, анализ новых атак, написание и добавление в базу соответствующих сигнатур. В связи с этим, такой тип СОВ абсолютно не эффективен против новых атак (атак «нулевого дня»). Поддержка работы СОВ на основе сигнатур с учетом большого количества ежедневно появляющихся сетевых угроз и атак требует все больше

ресурсов и становится серьезной проблемой, которая подвергает сомнению дальнейшую целесообразность их использования;

– Учитывая, что сетевые СОВ анализируют данные, полученные из сети, то они точно так же подвержены влиянию различных сетевых угроз и аномалий;

– Эффективные СОВ, созданные для использования на конкретном оборудовании, имеют внушительные габариты и подходят по большей части только для крупных предприятий и организаций. При этом в условиях активно развивающегося Интернета вещей, подверженного множеству уязвимостей, большинство устройств остаются не защищенными в силу маломощных платформ IoT и нецелесообразности использования для защиты не портативных СОВ;

– Высокая стоимость коммерческих готовых аппаратных решений.

К методам выявления аномалий относятся несколько значительных недостатков:

– Большое количество ложных срабатываний, что может быть вызвано, например, сбоями в работе программного обеспечения;

– Сложность определения конкретной причины возникновения аномалии;

– В сетях с высокой пропускной способностью обнаружение вторжений в реальном времени с требуемой точностью может быть крайне затруднено из-за ограниченных ресурсов;

– Для эффективного выявления угроз необходим полный и надежный анализ сетевого трафика. В современных сетях с гигабитными скоростями для этого может потребоваться сотни гигабайт дискового пространства ежедневно;

– Многие бесплатные открытые решения сложны в установке и настройке, что делает их использование возможным только при наличии специалистов высокой квалификации.

Например, популярные межсетевые экраны часто генерируют тысячи предупреждений о подозрительных событиях, которые необходимо анализировать вручную для выявления реальных угроз. Чтобы избежать этой проблемы, многие предпочитают отключать большинство сигнатур, связанных с веб-приложениями [78]. Однако такой подход часто приводит к серьезным и необратимым последствиям.

Существующим системам обнаружения вторжений, основанным на сигнатурных методах, все сложнее справляться с постоянно растущим числом новых способов организации сетевых атак, приводящих к утечке данных. Новые эксплойты разрабатываются непрерывно, а инструменты обфускации (запутывания) позволяют с легкостью уклоняться от систем обнаружения вторжений на основе сигнатур. Учитывая вышеуказанные недостатки, логично предположить, что в скором времени существующие системы потеряют свою актуальность и станут малоэффективны по причине огромных затрат на ресурсы для того, чтобы предвидеть все возможные векторы атак и вручную задавать правила распознавания всех уже известных и вновь появляющихся сетевых атак. В то время как подходы, основанные на обнаружении аномалий, которые могут обеспечить своевременное обнаружение аномальных активностей для предотвращения ранее неизвестных атак, также не лишены недостатков.

В связи с этим, способы и системы обнаружения вторжений должны постоянно исследоваться и совершенствоваться в целях решения существующих проблем.

Сравнивая значимость указанных недостатков и учитывая прогнозируемые перспективы для существующих СОВ, можно сказать, что дальнейшее совершенствование существующих и разработка новых методов обнаружения подозрительной сетевой активности в работе, как сетевых сервисов, так и «подключенных» устройств является актуальной и перспективной задачей. Наиболее перспективным представляется исследование и разработка алгоритма выявления аномалий, который позволил

бы осуществлять обнаружения нештатных ситуаций и сетевых атак на ранних этапах, а также исключить проблемы, связанные с требованиями к ресурсам, частотой ложных срабатываний и сложностью настройки и реализации.

В итоге, в данной работе ставится задача, направленная на создание алгоритма, который потенциально будет обладать возможностью обнаружения аномалий и сетевых атак на сетевые сервисы, функционировать на малоресурсных платформах, а также в случае адекватного создания нормального профиля сетевого трафика будет генерировать малое число ложных срабатываний, и при этом будет прост и экономичен в установке и эксплуатации.

Выводы

В данной главе приведены и систематизированы основные угрозы безопасности веб-сайтов. Дано подробное описание атак SQL-injection и Buffer Overflow, а также представлен обширный обзор и систематизация существующих подходов обнаружения атак такого типа. Кроме того, в главе рассмотрены существующие подходы выявления атак на основе обнаружения аномалий. Весь рассмотренный материал был классифицирован и проанализирован. Сформулирована задача исследования.

Глава 2 Нейросетевой метод обнаружения аномалий

Как уже было отмечено ранее, актуальность настоящей работы обусловлена тем, что большая часть сетевых информационных ресурсов тесно связана с конфиденциальными данными пользователей и постоянно подвергается попыткам взлома и несанкционированного доступа, от которых требуется серьезная защита.

С недавнего времени появилась новая тенденция применения нейронных сетей в решениях обеспечения информационной безопасности. Нейросетевые технологии позволяют решать такие задачи, решение которых классическими формальными методами затруднено или даже невозможно. Наиболее актуальным представляется исследование возможных реализаций и внедрение технологий искусственных нейронных сетей (ИНС) в системы обнаружения аномалий для выявления сетевых атак на сетевые сервисы [11]. Это обусловлено тем, что искусственные нейронные сети – это один из методов обработки информации, позволяющий достичь хороших результатов в решении таких вычислительно сложных задач как распознавание образов, классификация, прогнозирование, с последующим обобщением. Главными преимуществами решений, основанных на обнаружении аномалий с помощью нейросетей, является гибкость, адаптивность, скорость, отсутствие необходимости в постоянном пополнении базы сигнатур известных атак, а также возможность построить профиль нормальной активности пользователей, устройства или сайта. Используемая поведенческая аналитика позволяет выявлять сетевые атаки, не обнаруживаемые стандартными системами. А в связи со способностью искусственных нейронных сетей в процессе обучения выявлять сложные неочевидные свойства и зависимости в данных, которые отсутствовали в явном виде, они являются перспективным инструментом, который может заменить подсистемы анализа, базы правил и сигнатур для обнаружения атак в современных СОВ, что приведет к существенному снижению затрачиваемых материальных, временных,

человеческих и вычислительных ресурсов. Снижение нагрузки функционирования подсистемы анализа с помощью ИНС совместно с применением статистических методов анализа данных в рамках функционирования одной системы должно привести к повышению эффективности выявления сетевых атак. Более того, поскольку для защиты сайтов требуется своевременное выявление атак, скорость обработки нейронной сети может обеспечить реагирование на вторжение до того, как будет нанесен непоправимый ущерб системе.

2.1 Автоассациативные нейронные сети

В настоящей работе в основе предлагаемого метода лежит использование автоассоциативной нейронной сети (АНС) для обучения и последующего обнаружения аномалий. Это обусловлено тем, что автоассоциативная сеть (обычно многослойный персептрон) способна сжимать информацию и понижать размерность данных, поданных на вход, в промежуточном слое, имеющем меньшую размерность, выделяя лишь основные связи и признаки, из которых впоследствии восстанавливаются входные данные в выходном слое. Автоассоциативная память – это внутренняя самообучающаяся часть нейронной сети. В общем случае АНС должна содержать как минимум три скрытых слоя нейронов. В ней число нейронов в выходном слое равно числу нейронов во входном слое. Средний слой, называемый «узким горлышком», состоит из небольшого количества нейронов и предназначен для формирования компактного представления входных данных в процессе обучения. Первый скрытый слой отвечает за кодирование информации, а последний – за поиск декодера, который сможет восстановить исходные данные из их сжатой формы. Обучение сети направлено на точное воспроизведение входных сигналов: ответ считается корректным, когда выходные значения совпадают с соответствующими входными данными. Минимизация ошибки воспроизведения сетью входной

информации после «сжатия» данных на выходе эквивалентна оптимальному кодированию в «узком горле» сети. Слои, расположенные после «узкого горлышка», представляют собой декодер, который восстанавливает данные, используя только ту информацию, которая была закодирована в этом «узком горлышке». Эффективность восстановления оценивается с помощью условной энтропии [2]. Чем ниже её значение, тем меньше неопределённость и тем качественнее происходит воспроизведение данных. Привлекательной чертой такого подхода к сжатию информации является его общность. Автоассоциативная память является «контентно-адресной»: каждая часть или черта образа хранится в отдельной клетке, то есть представление эталонных образов не локализовано в памяти, а распределено по всей сети. Поэтому она способна опознавать входящий образ, даже если он представлен частью информации или в искаженном, видоизмененном виде.

Трехслойная автоассоциативная сеть сначала преобразует входные данные в пространство меньшей размерности через промежуточный слой, а затем восстанавливает их в исходную форму на выходе. Такая архитектура фактически реализует метод главных компонент. Чтобы выполнить нелинейное снижение размерности, требуется пятислойная сеть. В ней средний слой отвечает за уменьшение размерности, а соседние с ним слои выполняют нелинейные преобразования для связи с входным и выходным слоями. Если удалить два последних слоя из обученной пятислойной сети, останется структура, способная проецировать входные данные в пространство, размерность которого соответствует количеству нейронов в третьем слое [9]. В работе [22] подробно описаны принципы АНС.

2.2 Модель сетевого сервера как динамической системы

Абстрактно можно показать, что модель любого сетевого сервера, обрабатывающего конечное число различных запросов от пользователя может быть представлена в виде динамической системы, которая под действием

различных внешних и внутренних воздействий изменяет во времени свое состояние. Такой вывод обусловлен тем, что сервер в отличие от пользовательского компьютера обрабатывает внешние сетевые запросы в соответствии с возможностями его ПО и назначением, при этом выход сервера – ответ на запрос, зависит от входа – запроса и некоторой информации из памяти сервера. Такое функционирование напоминает динамическую модель системы и позволяет ввести понятие «состояние».

Эволюция динамической системы определяется детерминированной функцией, то есть через заданный интервал времени система примет конкретное состояние, зависящее от текущего состояния [12]. То есть у сетевого сервера, связанного с сайтом, есть конечное множество предписанных разработчиками функций – штатных состояний. Далее под штатным состоянием понимается корректное использование пользователем возможностей и функций сайта, что приводит к корректной обработке запроса и выдаче корректного результата. Например, если пользователь на страничке авторизации вводит корректные, существующие в БД, логин и пароль, то через определенное время зарегистрированный пользователь проходит авторизацию и перенаправляется на главную страницу сайта, получает санкционированный доступ к функциям сайта, то есть система переходит через определенное время в штатное состояние. В случае если злоумышленник пытается получить несанкционированный доступ и вместо логина и пароля вводит вредоносный SQL-код, позволяющий получить конфиденциальную информацию из БД, то система, получая нештатное входное воздействие, после обработки запроса переходит в нештатное состояние, не предусмотренное разработчиками сайта. При этом предполагается, что время обработки корректного запроса и объем соответствующего трафика должны отличаться от времени обработки и объема трафика при некорректном использовании сайта. Эти отличия опосредованно используются в предлагаемом методе для обнаружения аномалий в поведении сервера, которые будут отстоять от сформированной области допустимых состояний сервера.

Состояние динамической системы в любой момент времени может быть описано множеством вещественных чисел (или векторов), соответствующим определённой точке в пространстве состояний. В разработанном методе в качестве формализуемого признака для описания нормального профиля поведения сетевого сервера в штатных условиях предлагается использовать корреляционную функцию отклика сетевого сервера на запрос, рассчитываемую на некотором ограниченном интервале данных об интенсивности трафика. Таким образом, в качестве характеристик для описания всевозможных штатных состояний конкретного сервиса предлагается использовать взаимные корреляционные функции (КФ), рассчитанные методом скользящего окна по данным интенсивности входящего и исходящего трафика в каждый дискретный момент времени. Эта идея обусловлена следующими соображениями. Учитывая предположение об отличии объема трафика и времени получения динамического отклика сервера на штатные запросы от объема трафика и времени получения динамического отклика в процессе или в результате атаки, необходима характеристика, учитывающая соотношение объема переданных байт при запросе и ответе сервера, а также длительность выдачи ответа на запрос (производительность сервера). КФ, вычисляемые на ограниченном интервале равномерно дискретизированных по времени рядов интенсивности входящего и исходящего трафика, содержат в себе эту информацию, а значит, отличия в объеме трафика и времени обработки запросов будут отражаться на форме КФ. В данной постановке для обнаружения аномалий необходимо решить задачу обнаружения нетипичных форм КФ. Так как задачу обнаружения аномалий можно рассматривать, как задачу распознавания образов (или задачу классификации), то для её решения применяются нейронные сети.

2.3 Нейросетевой метод обнаружения аномальных динамических откликов

Предлагаемый нейросетевой метод позволяет обнаружить аномальный динамический отклик относительно типичных откликов для данного сетевого сервиса без инспекции содержимого трафика. Методом, основанным на распознавании образов динамического отклика сетевого сервера на запрос, предположительно можно диагностировать сетевые атаки, например, атаки типа Code Injection, провоцирующие аномальное поведение сетевого сервера. Предполагается возможным рассмотрение метода, как эффективной и простой в эксплуатации технологии обнаружения вторжений.

Метод был разработан в предположении об отличии динамического отклика сервера на запросы при штатном использовании сервиса от динамического отклика в процессе или в результате атаки, которое можно использовать для обнаружения аномальных откликов. В основе нейросетевого подхода к обнаружению аномалий в сетевом трафике лежит классифицирующая модель, на базе которой принимается решение об отнесении предъявляемого ей фрагмента сетевого трафика, представляющего собой рассчитанные по интенсивности входящего и исходящего трафика КФ, к нормальной сетевой активности или к аномальной.

Метод состоит из двух этапов. Первый этап заключается в накоплении и вычислении информации о динамическом отклике на всевозможные допустимые запросы. После накопления данных реализуется второй этап метода, основанный на использовании автоассоциативной нейронной сети оптимальной архитектуры, которая обучается на образцах корреляционных функций и предназначается для распознавания образов, отвечающих штатному поведению сетевого трафика.

Итак, распознаваемым классом является множество всех корреляционных функций, соответствующих нормальному поведению

сетевого трафика. Рассмотрим алгоритм реализации одноклассового классификатора на основе автоассоциативной нейронной сети.

По определению, автоассоциативная нейросеть производит тождественное преобразование входного вектора значений в выходной вектор. При этом архитектура нейронной сети формируется так, чтобы в одном из скрытых слоев происходило сжатие данных. В процессе этого преобразования сеть выделяет ключевые и наиболее характерные особенности входного вектора. В автоассоциативной нейронной сети типа многослойный персептрон сжатие обеспечивается в «узком горле» (рисунок 4).

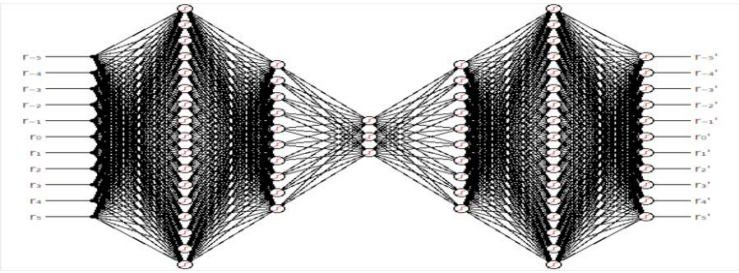


Рисунок 4 - Структура автоассоциативной нейросети для T=5

Обозначим функцию, выполняемую многослойным персептроном $N(r)$, где r – текущая функция взаимной корреляции трафика на входе нейросети в виде вектора значений $r(-T), r(-T+1), \dots, r(0), \dots, r(T)$. Тогда в результате вычислений для заданного входного вектора будет получен выходной вектор $r' = N(r)$. Назовем ошибкой реконструкции величину, вычисляемую по формуле среднеквадратичной ошибки между входным и выходным вектором нейросети (1).

$$e = \left(\frac{1}{2T+1} \sum_{t=-T}^{T} (r(t) - r'(t))^2 \right)^{\frac{1}{2}} \qquad (1)$$

Если нейросеть обучена осуществлять тождественное преобразование на некотором множестве векторов r из обучающей выборки, то для нового

вектора r ошибка реконструкции будет небольшой, если он близок к одному из векторов обучающей выборки. И наоборот, если входной вектор незнаком нейросети, то ошибка реконструкции будет большой.

Настройка автоассоциативной нейросети производится одним из методов обучения с учителем, например, методом Левенберга-Марквардта. Для обучения используется множество векторов корреляционных функций $r_{xy}(\tau)$, полученных в разные моменты времени штатной эксплуатации защищаемого сервиса. В процессе обучения весовые коэффициенты нейронов корректируются для минимизации среднеквадратической ошибки выхода нейросети относительно целевых значений.

В рабочем режиме нейронная сеть выдает малую ошибку реконструкции для знакомых ей из обучающей выборки корреляционных функций и большую для корреляционных функций аномальных фрагментов трафика сервера. То есть, если вновь предъявляемый нейронной сети образ текущей корреляционной функции будет близок к образу из обучающей выборки, ошибка реконструкции будет минимальна, иначе ошибка реконструкции будет велика. На основании графика ошибки реконструкции можно сделать вывод о нормальной работе или наличии аномалий в текущем поведении сервера.

Таким образом, в предлагаемом методе критерием обнаружения аномалии является превышение значением ошибки реконструкции для текущей корреляционной функции заданного для конкретного защищаемого сервиса критического уровня.

После завершения настройки нейронная сеть может использоваться для обнаружения аномалий в сетевом трафике или аномального поведения обрабатывающего запросы сервера, что вполне может быть и сетевой атакой, и попыткой злоумышленника получить несанкционированный доступ.

2.4 Принципы формирования обучающей выборки для обнаружения аномалий сетевых сервисов

Формирование обучающей выборки – это важнейший этап в задачах обучения нейронной сети, от которого зачастую зависит успех решения задач обучения и распознавания, поэтому к ее построению необходимо подходить систематически.

Процедура настройки алгоритма предполагает сбор сетевого трафика при как можно более разнообразных вариантах штатного использования защищаемого веб-сайта и его нагрузки. Для накопления исчерпывающей информации, описывающей нормальную эксплуатацию веб-сайта, необходимо использовать все его штатные возможности при различной нагрузке, для чего, в некоторых случаях, может потребоваться достаточно длительное время. Реализация прохождения всего функционала веб-сайта может быть выполнена вручную или с помощью специально разработанного инструмента автоматизации выполнения основных запросов, который мог бы генерировать запросы в течение длительного времени для сбора достаточного количества трафика нормальной, штатной работы обрабатывающего сервера.

Важно отметить, что ключевым аспектом в успешном обучении нейронной сети и обнаружения ею аномалий или атак является выбор оптимального множества наиболее значимых параметров (признаков) для описания нормального профиля.

Как уже было сообщено ранее, в проведённых исследованиях в качестве признаков, составляющих обучающую выборку, было принято множество корреляционных функций, последовательно рассчитываемых каждую десятую секунду скользящим окном шириной, равной 10, по рядам интенсивности входящего и исходящего трафика, снятого во время выполнения пользователем всевозможных штатных обращений к веб-сайту. Способ получения необходимой информации для последующих расчетов КФ и формирования нормального пространства признаков, которое будет лежать

в основе обучения нейросети, более подробно изложено ниже, в п.п. 2.4.1-2.4.2 настоящей работы.

Для обучения и проверки качества обучения нейронной сети полученные данные разбиваются на обучающую и проверочную выборку. При этом обучающая выборка содержит в себе исключительно нормальные образцы, соответствующие штатному использованию функций сайта, а в состав проверочной выборки включаются независимые данные, не содержащие аномалий, что также заведомо известно.

Итак, для построения нормального профиля защищаемого сетевого объекта используется свободный от аномалий набор данных, на основе которого формируется пространство нормальных признаков – обучающая выборка, которая в свою очередь используется только для настройки нейронной сети. Проверочная выборка необходима для того, чтобы убедиться, что нейронная сеть успешно обучилась распознавать нормальные образы, частично похожие на те, по которым она обучалась, сохранив способность к обобщению полученных знаний и опыта на последующие, новые данные.

Способность нейронной сети распознавать данные из нормального множества, которые не использовались в процессе обучения, демонстрирует её умение обобщать знания.

Задача настройки весов искусственной нейронной сети (ИНС) во время обучения заключается в поиске такой комбинации значений весов, которая позволит наилучшим образом воспроизводить последовательность обучающих, "нормальных" характеристик (НКФ) на выходе. При этом важно учитывать взаимосвязь между количеством весов (степенями свободы) и числом обучающих примеров [29]. Если целью обучения является только запоминание обучающих выборок, то их количество может быть равно числу весов. В таком случае каждый вес будет соответствовать конкретной обучающей паре, но сеть лишится способности к обобщению и сможет только воспроизводить уже известные данные. Чтобы развить это свойство, сеть должна обучаться на избыточном наборе данных, где веса адаптируются к

статистическим усреднениям, а не к уникальным выборкам. Таким образом, для усиления способности к обобщению необходимо не только оптимизировать структуру сети, стремясь к её минимизации, но и использовать достаточно большой объем обучающих данных. После обучения на некотором наборе примеров сеть сможет корректно обрабатывать входные данные, принадлежащие тому же множеству, но не участвовавшие в процессе обучения.

Даже если обучение нейросети кажется успешным, её качество необходимо проверять на тестовых примерах, которые не использовались в процессе обучения. При этом количество тестовых данных должно увеличиваться с ростом точности обучения. Например, если вероятность ошибки сети приближается к одной миллиардной, то для подтверждения такого уровня надёжности потребуется порядка миллиарда тестовых примеров [17]. Вследствие этого проверка высокоточных нейронных сетей представляет собой сложную задачу.

Сбор исчерпывающей информации о трафике для формирования обучающей выборки представляет собой ресурсоемкую процедуру, однако, ее требуется провести лишь единожды без необходимости постоянно дообучать нейронную сеть. Дообучение может понадобиться лишь в случае добавления нового функционала и обновления веб-сайта.

2.4.1 Алгоритм получения и обработки сетевого трафика

Для формирования обучающей выборки необходимо собрать сетевой трафик во время штатной эксплуатации защищаемого веб-сайта и получить информацию, включающую статистику принятых и переданных данных, о трафике localhost, который был передан через локальный интерфейс loopback (lo). В качестве инструмента для захвата, записи и анализа сетевого трафика используется Wireshark. Wireshark часто используется для проведения исследований сетевых приложений и протоколов, для поиска проблем в работе сети и причин их вызывающих. Кроме того, он работает с подавляющим большинством известных протоколов, имеет понятный и логичный

графический интерфейс и мощнейшую систему фильтров. Ядром Wireshark служит библиотека для захвата сетевых пакетов libpcap.

Перед захватом трафика выбирается интерфейс lo, пакеты с которого будут захвачены и после начала захвата можно приступать к штатной эксплуатации веб-сайта в соответствии с заранее разработанной схемой. После многократного выполнения всех основных допустимых сценариев использования сайта захват трафика прекращается, а захваченные пакеты сохраняются в файл со стандартным форматом pcap (packet capture).

С помощью функции Statistics> IO Graph можно визуализировать интересующий трафик, применяя фильтрацию. Wireshark позволяет отбирать интересующие пакеты по условиям – фильтрам.

Следующая задача состоит в формировании временных рядов отдельно для входящего и исходящего трафика (inputbytes(time) и outputbytes(time)), собранного в процессе эксплуатации веб-сайта и содержащего штатные запросы и ответы. С помощью специальных инструментов захваченный сетевой трафик представляется в удобном для дальнейшего использования в расчетах виде – два временных ряда, состоящих из количества переданных байт при запросах (входящий трафик) и их обработке (исходящий трафик) за определённый период времени. Шаг дискретизации временных рядов составляет 0,1 сек.

Для получения временных рядов inputbytes(time) и outputbytes(time) использовался tshark, консольная версия Wireshark. Tshark – это инструмент для захвата, чтения и анализа сетевых пакетов с интерфейсом командной строки. Он может выводить содержимое трафика и необходимую информацию о нём в понятном человеку формате, а также записывать ее в текстовый файл для последующего анализа и расчетов.

Так, например, для получения временных рядов из захваченного сетевого трафика traffic.pcap можно применить следующие команды:

- LC_ALL=C tshark -n -q –r traffic.pcap -z 'io,stat,0.1,tcp.srcport==80' >io_stat_0.1_out.txt;

- LC_ALL=C tshark -n -q -r traffic.pcap -z 'io,stat,0.1,tcp.dstport==80' >io_stat_0.1_in.txt.

«LC_ALL=C» позволяет tshark интерпретировать число 0.1 (значение шага дискретизации) в строке «io,stat,0.1», как число с плавающей точкой. Файл traffic.pcap – это входной файл захваченного трафика, а файл io_stat_0.1_out.txt – выходной (рисунки 5-6).

```
===================================================
| IO Statistics                                   |
|                                                 |
| Interval size: 0.1 secs                         |
| Col 1: Frames and bytes                         |
|     2: tcp.dstport==80                          |
|-------------------------------------------------|
|                   |1              |2            |
| Interval          | Frames | Bytes | Frames | Bytes |
|-------------------------------------------------|
|    0.0 <>    0.1 |      5 |   964 |      3 |   824 |
|    0.1 <>    0.2 |      0 |     0 |      0 |     0 |
|    0.2 <>    0.3 |     18 |  1284 |     12 |   840 |
|    0.3 <>    0.4 |      0 |     0 |      0 |     0 |
|    0.4 <>    0.5 |      0 |     0 |      0 |     0 |
|    0.5 <>    0.6 |      0 |     0 |      0 |     0 |
```

Рисунок 5 - Пример файла на выходе tshark (io_stat_0.1_in.txt)

```
===================================================
| IO Statistics                                   |
|                                                 |
| Interval size: 0.1 secs                         |
| Col 1: Frames and bytes                         |
|     2: tcp.srcport==80                          |
|-------------------------------------------------|
|                   |1              |2            |
| Interval          | Frames | Bytes | Frames | Bytes |
|-------------------------------------------------|
|    0.0 <>    0.1 |      5 |   964 |      2 |   140 |
|    0.1 <>    0.2 |      0 |     0 |      0 |     0 |
|    0.2 <>    0.3 |     18 |  1284 |      6 |   444 |
|    0.3 <>    0.4 |      0 |     0 |      0 |     0 |
|    0.4 <>    0.5 |      0 |     0 |      0 |     0 |
|    0.5 <>    0.6 |      0 |     0 |      0 |     0 |
```

Рисунок 6 - Пример файла на выходе tshark (io_stat_0.1_out.txt)

Далее, функцией awk можно исключить форматирование, оставив только необходимые для расчетов КФ временные ряды inputbytes(time) и outputbytes(time), записанные в файлы «in_0.1.dat» и «out_0.1.dat» соответственно (рисунок 7) с помощью следующих команд:

- tail -n +12 io_stat_0.1_out.txt | awk '$1=="|" {b=12; print $2,$b}' >out_0.1_all.dat

- tail -n +12 io_stat_0.1_in.txt | awk '$1=="|" {b=12; print $2,$b}' >in_0.1_all.dat

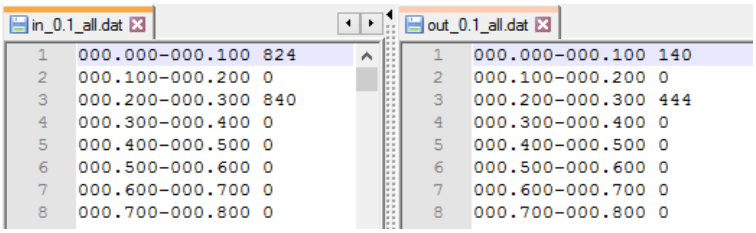


Рисунок 7 - Пример файлов после обработки (in_0.1_all.dat и out_0.1_all.dat)

Также для обработки трафика и получения необходимой информации из него можно воспользоваться инструментом pcap2matlab (), который позволяет импортировать возможности анализатора сетевого трафика TShark в MATLAB. Функция может работать в двух режимах: выполнять прямой захват сетевых пакетов и читать готовые файлы *.pcap прямо из рабочей области MATLAB.

После решения задач сбора и обработки сетевого трафика для получения обучающей выборки предлагается к полученной и обработанной сетевой информации применить статистический метод анализа и расчета образцов.

2.4.2 Алгоритм расчета образцов для обучения и проверки работы АНС

Расчет корреляционных функций, их визуализация и формирование на их основе файлов обучающей и проверочной выборок, по которым будет проведено обучение и проверка нейронной сети, реализовано с помощью специально написанных программ в MATLAB (см. Приложение А).

Расчет КФ для формирования обучающей и проверочной выборки был реализован методом скользящего окна. КФ рассчитываются каждую десятую долю секунды по данным двух файлов in_0.1_all.dat и out_0.1_all.dat, при этом ширина окна составляет 10 отсчетов и охватывает 1 секунду трафика. Такие

корреляционные функции содержат в себе информацию о размере запроса, времени обработки запроса веб-сервером и размере ответа сервера на запрос. Выбор такого способа расчета позволяет не только решить проблему разбиения аномальной активности в трафике по двум соседним КФ, но и видеть мгновенные изменения в поведении трафика, характеризующемся потоком КФ. То есть, расчет КФ скользящим окном потенциально должен увеличивать быстродействие обнаружения аномалий нейросетью.

Полученные корреляционные функции нормируются и, за исключением нулевых, записываются в файл обучающей выборки в виде, приведённом на рисунке 8.

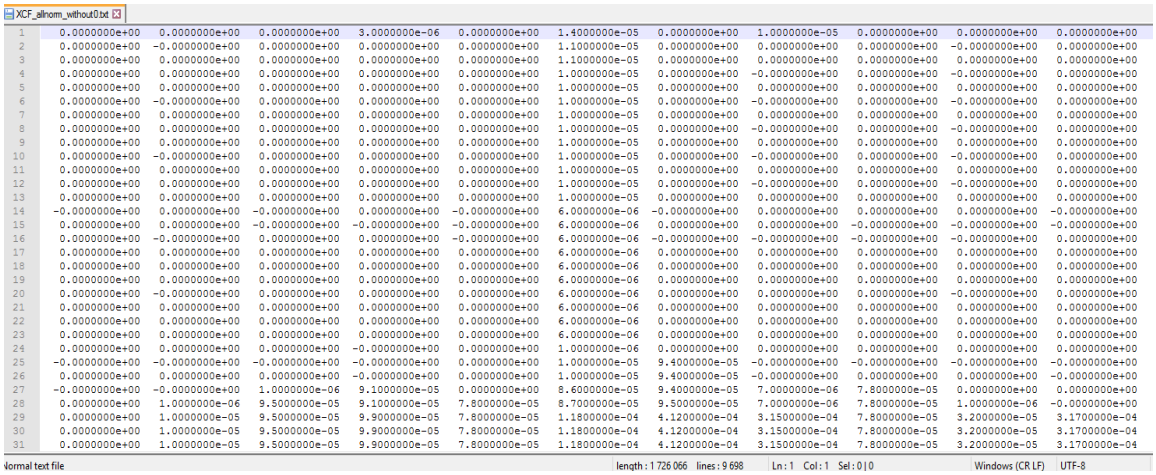


Рисунок 8 - Пример части файла обучающей выборки

В целом, можно сказать, что такой выбор характеристики описания трафика помогает решить проблему больших данных при создании СОВ. Для эффективного выявления вторжений требуется полная и достоверная информация о сетевом трафике. В условиях современных компьютерных сетей с гигабитными скоростями передачи данных может возникнуть необходимость в использовании сотен гигабайт дискового пространства ежедневно. Данный метод использует информацию, содержащуюся только в заголовках пакетов. Отсутствие инспекции «полезной загрузки» (payload) делает предложенный метод сравнительно экономичным.

Необходимо понимать, что от адекватности полученных данных зависит возможность построения алгоритма интеллектуального анализа трафика, направленного на обнаружение отклонений в откликах сервера от типового поведения и идентификацию аномалий. Далее, имея в распоряжении показатели нормального поведения трафика и шаблон для обучения АНС, критерий обнаружения аномалии, можно ставить более высокие задачи, связанные с обучением нейронной сети, идентификацией несанкционированных воздействий извне, с целью организации атаки, по выявленным отклонениям.

Итак, в текущей главе сформулирован подход для обнаружения аномального поведения сетевого сервера, идея которого состоит в запоминании автоассоциативной нейронной сетью различных типовых вариантов корреляционных функций, соответствующих разнообразным вариантам штатной эксплуатации защищаемого сервиса и рассчитанных по рядам интенсивности входящего и исходящего трафика методом скользящего окна. Таким образом, алгоритм обнаружения аномалий основан на интеллектуальном анализе обученной нейронной сетью подобия поданных на ее вход форм корреляционных функций тем, по которым она была обучена. Для обнаружения нейронной сетью аномальных корреляционных функций задействуется основное её свойство, состоящее в идентификации образов, которые не похожи на те, по которым она обучалась, а признаком выявления аномалии является превышение значением ошибки реконструкции для текущего образа некоторого заданного порогового значения.

После завершения настройки нейронная сеть может использоваться для обнаружения аномального поведения сервера, что вполне может быть попыткой злоумышленника, например, получить несанкционированный доступ или вывести систему из строя.

Далее, будет продемонстрирована работа метода в условиях, приближенных к реальным.

Выводы

В данной главе описаны перспективы и преимущества использования нейронных сетей в задачах обнаружения аномалий в сетевом трафике.

Было предложено и обосновано рассмотрение модели сетевого сервера как динамической системы, изменяющей свое состояние в зависимости от входных воздействий – запросов пользователя. Выдвинуто предположение о том, что реакция сервера, как динамической системы, на штатные запросы отличается от реакции на несанкционированное, непредвиденное разработчиками сетевого ресурса воздействие. Была поставлена задача обнаружения поведенческих аномалий в ответах сервера без проведения инспекции содержимого трафика и предложено решение. В предложенном подходе в качестве признака, характеризующего работу сервера, «запрос-ответ», используется взаимная корреляционная функция. На основе выдвинутого предположения был разработан нейросетевой метод обнаружения аномалий в сетевом трафике, свидетельствующих о потенциальных сетевых атаках. Метод основан на применении автоассоциативной нейронной сети, обученной на значениях корреляционных функций (КФ), рассчитываемых каждую 0,1 сек. по блоку данных об интенсивности входящего и исходящего трафика методом скользящего окна, что позволяет видеть мгновенные изменения в контролируемых характеристиках.

Обоснован выбор использования автоассоциативной нейронной сети (АНС) и признаков, характеризующих необходимую информацию о связке «запрос-ответ», для целей обучения и обнаружения аномалий.

Предложены алгоритмы и выбран инструментарий для получения и обработки сетевого трафика, конвертации полученных данных в необходимый формат в целях получения признаков для описания нормального трафика.

Изложены основные принципы обучения и проверки работы нейронной сети. Предложена методика формирования обучающей выборки. В рамках

разработанного метода АНС обучается на КФ, рассчитанных по трафику, генерируемому штатной деятельностью пользователя на сайте, то есть при разрешенном использовании функционала сайта. С учетом того, что АНС настраивает весовые коэффициенты своих нейронов воспроизводить на выходном слое данные обучающей выборки, характеризующей легитимное использование ресурса и легитимное поведение пользователя, был разработан критерий обнаружения аномалий. Введено определение и метод расчета ошибки реконструкции, которая позволяет оценить, насколько поданный на вход настроенной АНС образ похож на те, по которым она была обучена. Критерием обнаружения аномалий является превышение значением ошибки реконструкции для входного образа некоторого порогового значения, выбранного по результатам обучения и тестирования нейронной сети на выборках признаков, рассчитанных по данным штатного трафика.

АНС может быть настроена единожды для конкретного сервиса, и, не требуя мощных аппаратных и вычислительных ресурсов, выполнять свои функции обнаружения аномалий.

Предполагается возможным апробировать разработанный метод для обнаружения реальных атак на типичный веб-сайт, например, распространенных SQL-injection и flood-атак.

Глава 3 Проверка работоспособности метода, анализ полученных результатов

В свете широкого распространения типовой архитектуры коммуникационных сервисов наиболее перспективным объектом для демонстрации простоты проведения атак на современные сетевые сервисы и их обнаружения разработанным нейросетевым алгоритмом представляется типовой сетевой сервер, например, веб-сайт, построенный на основе некоторого типового движка. В качестве такого объекта был выбран достаточно распространённый и доступный веб-форум MyBB (MyBulletinBoard), написанный на языке PHP. MyBB – это бесплатный сервис для создания тематических форумов, пользователи которого создают личные аккаунты и могут обмениваться информацией по интересующей их тематике.

Для того, чтобы продемонстрировать работоспособность нейросетевого метода на примере обнаружения атак на реальный сетевой сервис, по данным трафика при штатной эксплуатации MyBB была получена обучающая выборка для настройки нейронной сети и проверочная выборка для проверки качества ее обучения распознавать нормальные образцы. При использовании алгоритма в целях обнаружения атак на сетевой сервис качественно обученная нейронная сеть позволит выделять аномалии и атаки в сетевом трафике, если они будут встречаться.

Рассмотрим подробное описание процесса получения данных для обучения.

3.1 Формирование обучающей выборки

Специально для создания пространства нормальных признаков, описывающих штатные запросы и нормальные реакции сервера, была разработана схема основных сценариев использования MyBB. Под сценариями понимаются лишь основные и типовые варианты штатных

запросов пользователя к веб-сервису MyBB, во время генерации и обработки которых, должен быть организован захват трафика. После чего, на основе данных захваченного трафика будут рассчитаны КФ, которые будут составлять обучающую и проверочную выборку. Схема эксплуатации сервиса приведена в таблице 1.

Реализация прохождения схемы может быть выполнена вручную или с помощью специально разработанного инструмента автоматизации выполнения основных запросов, который мог бы генерировать запросы из схемы в течение длительного времени для сбора достаточного количества трафика нормальной, штатной работы сервера MyBB.

Ручное выполнение позволяет гибко адаптировать поведение пользователя под реальные условия и обеспечивает контроль за каждым этапом взаимодействия с системой. При этом подходе сбор данных осуществляется через стандартные средства мониторинга сетевого трафика, такие как Wireshark и Tshark, с последующей их фильтрацией и анализом для формирования обучающей выборки.

Таблица 1 - Основные сценарии использования MyBB

Описание	Действия	Параметры действий
Просмотр данных, касающихся системной платы. Добавление других примечаний	Ввод и сохранение примечаний	Примечание из 20 символов
		Примечание из 200 символов
		Примечание из 2000 символов
Управление собственными настройками панели управления администратора, ввод и сохранение произвольных заметок	Ввод и сохранение заметок	Заметка из 20 символов
		Заметка из 200 символов
		Заметка из 2000 символов
Открывается страница документации, где можно прочитать учебники и онлайн-документацию, найти решения проблем	Перейти на страницу https://docs.mybb.com/	–
Редактирование профиля (изменения пароля, Email, аватара)	Загрузить аватар	Фотография 2кБ
Управление объявлениями, которые показываются на всех выбранных форумах	Ввод и сохранение объявления (Add Announcement)	Объявление из 200 символов
		Объявление из 2000 символов
Создание новой темы	Ввести наименование новой темы	–
Просмотр новых постов	Просмотр новых постов	–
Ответ на пост, опубликованный одним из пользователей	Ввод и публикация ответа	Ответ из 200 символов
		Ответ из 2000 символов
		Ответ из 20 символов и картинки в 400кБ
Подписка на данную тему	Подписаться на данную тему	–
Поиск интересующей темы	Ввод и поиск темы	–
Отправка личного сообщения пользователю	Ввести и отправить сообщение администратору	Сообщение из 200 символов
		Сообщение из 2000 символов
Просмотр личной информации и личного блокнота	Заполнить персональный блокнот	Текст из 2000 символов
Просмотр полученных сообщений	Просмотр полученных сообщений	–

Сценарии из схемы, описанной в таблице 1, были выполнены неоднократно вручную, при этом захват сетевого трафика проводился с помощью Wireshark. Общее время наблюдения за сервером при реализации описанных в таблице действий составило 3 часа. Сетевой трафик был зафиксирован и сохранен в файл «traffic.pcap». Каким именно образом данные о трафике были обработаны было сообщено ранее в п.п. 2.4.1-2.4.2. После получения временных рядов интенсивности входящего и исходящего трафика (inputbytes(time) и outputbytes(time)) был рассчитан массив корреляционных функций, соответствующих нормальному сетевому трафику. Данные о размерности массива, характеризующего нормальное пространство признаков, приведены в таблице 2.

Таблица 2 - Данные об обучающей выборке

Время наблюдения за штатной работой MyBB, с	Количество КФ, описывающих нормальную работу	Размерность обучающей выборки после удаления нулевых КФ
10441,05	104402	9697

Эксперименты показали, что для обучения нейронной сети достаточно использовать лишь половину имеющегося набора данных. В связи с этим общий массив был разделён на две части: одна предназначена для обучения сети, а вторая – для тестирования и оценки качества её работы.

3.2 Обучение нейронной сети алгоритмом обучения Левенберга-Марквардта

Обучение (training) нейронной сети – это этап, в процессе которого на вход нейронной сети поочередно поступают данные из обучающей выборки с целью корректировки весовых коэффициентов синоптических связей для

получения наиболее адекватного сигнала на выходе нейронной сети, в случае автоассоциативной сети выход сети должен максимально соответствовать входу. Другими словами, весовые коэффициенты нейронов слоев сети подстраиваются таким образом, чтобы минимизировать ошибку между данными на входе и выходе нейронной сети.

В рамках настоящей работы обучение проводилось в предварительном режиме (off-line training), то есть, когда коррекция весов производится один раз за период обучения – после предъявления всех векторов обучающей выборки, с помощью алгоритма обучения без учителя (unsupervised learning), при котором набор эталонов, то есть «правильный» ответ на выходе нейронной сети, отсутствует.

Настройка автоассоциативной нейросети производится методом Левенберга-Марквардта, одним из градиентных методов оптимизации второго порядка, который применяется для обучения нейронных сетей [13]. Выбор этого алгоритма обучения обусловлен тем, что для него характерно достижение высокой точности обучения нейронной сети [31]. Кроме того считается, что алгоритму обучения Левенберга-Марквардта не присущи недостатки свойственные, например, одному из самых популярных методов - методу обратного распространения ошибки, а именно медленная сходимость и негативное влияние локальных минимумов [32]. Подробное описание метода Левенберга-Марквардта изложено в [30].

Для проведения экспериментов применялся специальный модуль MATLAB – Neural Network Toolbox, который предоставляет инструменты для проектирования, моделирования, разработки и визуализации нейронных сетей. Этот инструментарий поддерживает широкий спектр стандартных архитектур нейросетей и обладает гибкой, открытой структурой. В состав пакета входят как командные функции для работы через терминал, так и графический интерфейс, позволяющий быстро и поэтапно создавать нейросетевые модели.

Для обучения используется множество векторов корреляционных функций $r_{xy}(\tau)$, полученных в разные моменты времени интервала настройки. В процессе обучения весовые коэффициенты нейронов корректируются для минимизации среднеквадратической ошибки выхода нейросети относительно целевых значений. Архитектура нейросети была определена с учетом ширины окна корреляции и экспертных оценок возможностей многослойных персептронов для автоассоциативной памяти. В ходе экспериментов применялась сеть, содержащая 11 входов и 11 выходов, с последовательно соединенными полносвязными слоями, включающими 15, 10, 6, 3, 5, 10 и 15 нейронов соответственно (рисунок 9). Обучение проводилось в пакете MATLAB методом Левенберга-Марквардта (TRAINLM) с автоматической выборкой из обучающих данных подмножества для обучения, верификации и проверки.

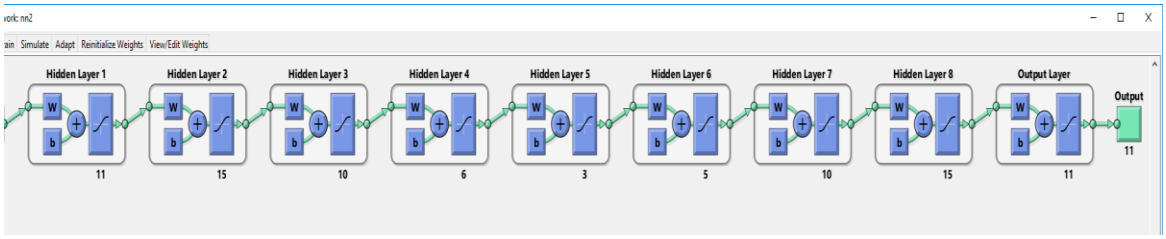


Рисунок 9 - Структура нейронной сети

Сеть была обучена достаточно быстро (рисунок 10), достаточно было 39 итераций, а минимум среднеквадратичной ошибки на валидационном наборе данных составил всего 1.1911×10^{-8} (рисунок 11).

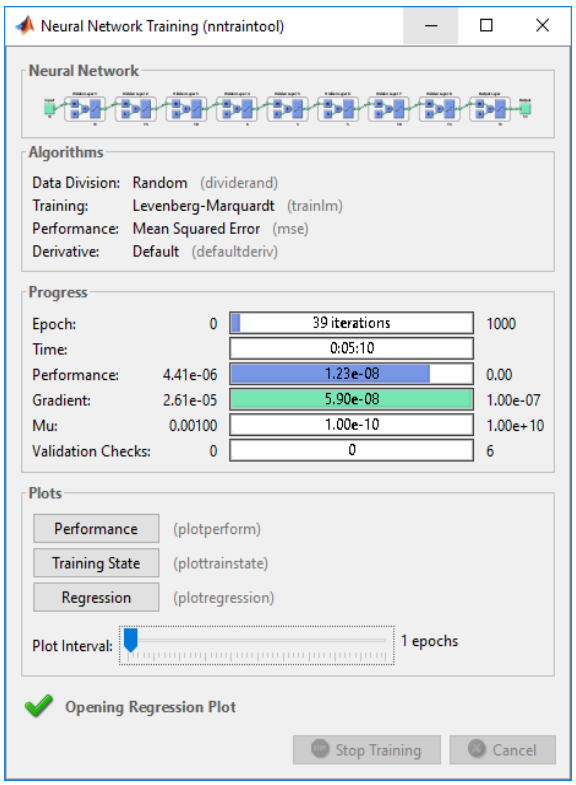


Рисунок 10 - Мониторинг прогресса обучения

Ниже на рисунках 11-13 представлены статистические результаты и отображены оценки качества обучения.

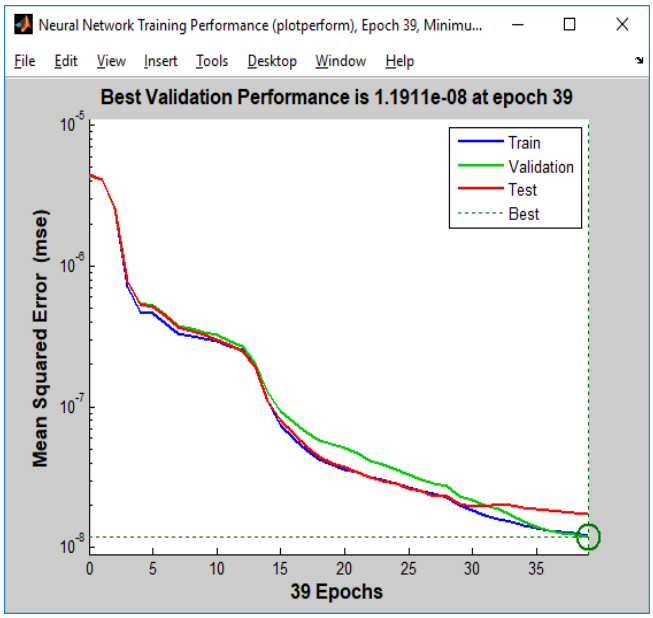


Рисунок 11 - Среднеквадратичная ошибка на валидационном наборе данных для последовательных эпох обучения

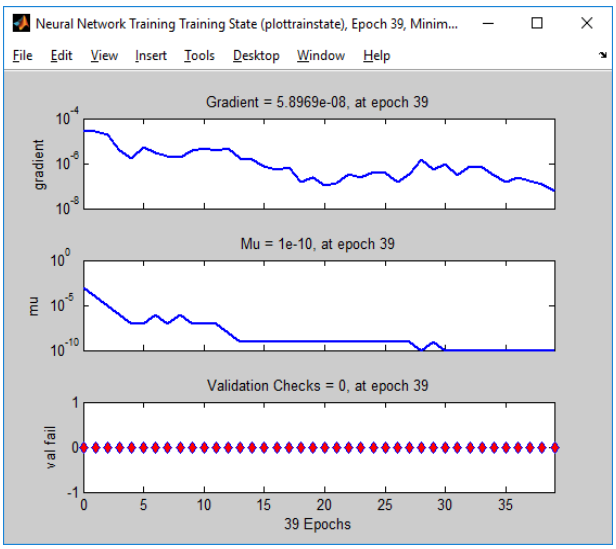


Рисунок 12 - Training State

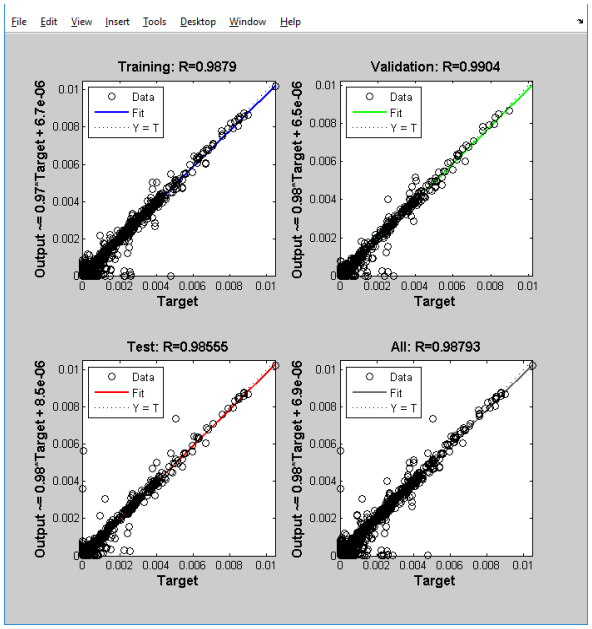


Рисунок 13 – Regression

Наилучшие оценки качества обучения были получены именно для данной структуры нейронной сети.

Обучение нейросети представляет собой ресурсоемкую процедуру в случае обучающей выборки большого объема. В частности, в рамках проводимых экспериментов на современном компьютере обучение длилось 5 минут. Однако работа настроенной нейросети требует совсем мало ресурсов и может производиться на процессорах с малой вычислительной мощностью и энергопотреблением.

3.3 Тестирование качества обучения нейросети и определение критерия обнаружения аномалий

Тестирование – это этап проверки работоспособности нейронной сети, в течение которого на вход сети подаются данные, которые не были использованы в процессе обучения. Важно отметить, что в процессе тестирования весовые коэффициенты нейронной сети не изменяются.

Итак, обучающая выборка (training sample) нужна для алгоритма настройки весовых коэффициентов, а наличие проверочной выборки (control sample) нужно для оценки эффективности обученной нейронной сети в распознавании КФ, отвечающих штатным запросам к сайту.

Проверочная выборка представляла собой совокупность КФ, рассчитанных по контрольному трафику (control traffic), нормальному, но отличному от того, по которому рассчитывались КФ для обучающей выборки. Тестирование необходимо для верификации того, что нейронная сеть не переобучилась и сохранила свойства обобщённости, то есть способность распознавать образцы, частично отличающиеся от образцов из обучающей выборки, но принадлежащие к классу нормальных.

После подачи на вход настроенной нейронной сети проверочной выборки был экспортирован массив ошибок между целевыми и выходными данными, на основе которого была рассчитана ошибка реконструкции для элементов проверочной выборки. Для сравнения на рисунке 14 приведен график ошибки реконструкции, рассчитанной для каждого элемента обучающей выборки.

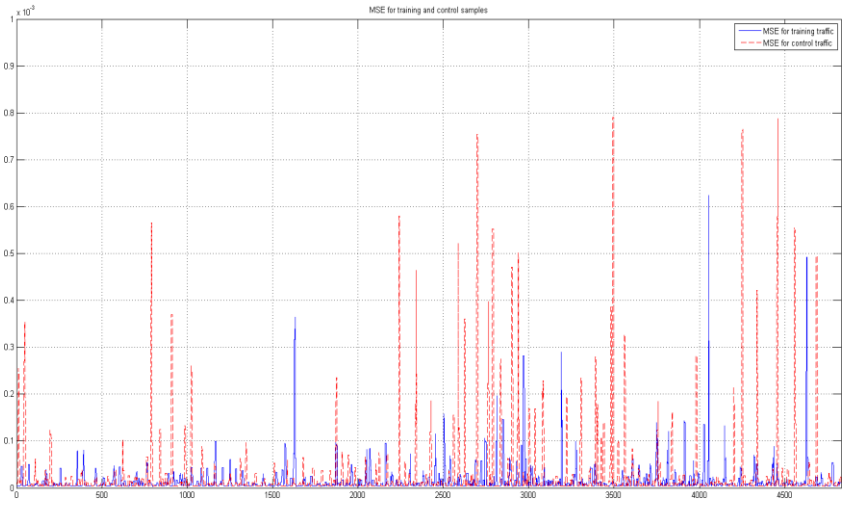


Рисунок 14 - Ошибка реконструкции для обучающей и проверочной выборки

Анализ графиков, изображенных на рисунке 14 позволяет с уверенностью сказать, что эффекта переобучения - ситуации, когда нейронная сеть хорошо распознает только образцы обучающей выборки, при этом ошибки сети очень малы, а на новых независимых нормальных данных они неоправданно возрастают, нет. Напротив, большую часть КФ проверочной выборки нейронная сеть нашла нормальными и только для четырех образов из проверочной выборки ошибка реконструкции оказалась немного больше, чем максимальная ошибка для элементов из обучающей выборки. Это говорит о том, что нейронная сеть обладает свойством обобщения, то есть способностью распознавать КФ, которые принадлежат классу нормальных данных и частично похожи на обучающие данные, но не были использованы на этапе обучения.

Так как величиной, определяющей степень несоответствия поданной на вход нейросети корреляционной функции, является ошибка реконструкции, рассчитанная по формуле (1) для обучающей и проверочной выборки, то она может использоваться для определения порогового значения, при превышении которого будет диагностироваться аномальная корреляционная функция.

Выбор порогового значения для критерия обнаружения аномалий может быть определен в диапазоне от $0{,}8 \times 10^{-3}$ до 3×10^{-3} , это обусловлено максимальными значениями ошибки реконструкции для образцов проверочной выборки.

3.4 Применение нейросетевого метода для обнаружения атак на сетевой сервис

После обучения и выбора критерия обнаружения аномалий проводились испытания нейронной сети на корреляционных функциях, полученных по данным сетевого трафика, содержащего, в частности, различные сетевые атаки. Набор таких КФ назовем рабочей выборкой.

Для тестирования работоспособности нейронной сети и эффективности обнаружения ею неопознанных, то есть аномальных образцов, соответствующих тем или иным аномалиям в трафике, в частности, несанкционированным вторжениям, формируются рабочие выборки. Рабочие выборки, содержащие в себе признаки, как трафика, полученного в условиях штатной эксплуатации сетевого сервиса, так и в условиях сетевых атак состоят как из нормальных, так и из аномальных образцов.

В рамках проведенных исследований рабочие выборки были сформированы по данным трафика, захваченного в условиях штатной эксплуатации веб-сайта и в условиях сетевых атак на обрабатывающий веб-сервер, в частности, при внедрении вредоносного SQL-кода для получения несанкционированного доступа к информации из БД и некоторых типов flood-атак.

На основании результатов этого этапа будет определена статистика и анализ эффективности НС по критериям качества распознавания аномалий и наличию ложных срабатываний (когда нормальное обращение принимается за атаку).

Продемонстрируем применение нейросетевого метода для обнаружения каждой из них.

3.4.1 Демонстрация атаки code injection на сетевой сервис

Заведомо известно, что CMS (Content Management System – система управления содержимым) сервиса для создания форумов, MyBB 1.8.6, содержит уязвимость внедрения SQL-кода. Именно эта уязвимость MyBB 1.8.6 будет использована в дальнейшем для апробации нейросетевого метода на основе анализа подобия корреляционных функций. Уязвимость может быть использована для получения доступа к информации в базе данных. Удалённый пользователь может с помощью специально сформированного запроса выполнить произвольные команды в базе данных уязвимого приложения.

Уязвимость существует из-за недостаточной фильтрации входных данных в компоненте forumdisplay.php:

$perpage = $mybb->settings['threadsperpage']; $query = $db->query("SELECT t.*, {$ratingadd}t.username AS threadusername, u.username FROM ".TABLE_PREFIX."threads t LEFT JOIN ".TABLE_PREFIX."users u ON (u.uid = t.uid) WHERE t.fid='$fid' $tuseronly $tvisibleonly $datecutsql2 $prefixsql2 ORDER BY t.sticky DESC, {$t}{$sortfield} $sortordernow $sortfield2 LIMIT $start, $perpage");

Итак, ниже приведён эксплойт, то есть последовательность команд, использующая уязвимость в программном обеспечении и применяемая для проведения SQL-injection в MyBB [53], [95]:

1) Выполнение произвольного SQL-запроса возможно, например, на странице настроек http://localhost/admin/index.php?module=config-settings&action=change&gid=7, где хаккер может установить значение Threads Per Page: 20 procedure analyse(extractvalue(rand(),concat(0x3a,version())),1);

2) При открытии страницы http://localhost/forumdisplay.php?fid=3 будет выведена ошибка, содержащая версию MySQL:

SQL Error: 1105 - XPATH syntax error: ':5.5.33-1' Query: SELECT t.*, (t.totalratings/t.numratings) AS averagerating, t.username AS

threadusername, u.username FROM mybb_threads t LEFT JOIN mybb_users u ON (u.uid = t.uid) WHERE t.fid='3' AND t.visible IN (-1,0,1) ORDER BY t.sticky DESC, t.lastpost desc LIMIT 0, 20 procedure analyse(extractvalue(rand(),concat(0x3a,version())),1);

Для того чтобы провести атаку SQL-injection на сервис MyBB достаточно ввести в поле Threads Per Page строку «20 procedure analyse(extractvalue(rand(),concat(0x3a,version())),1);» (рисунок 15). Однако это оказалось невозможным, так как это поле имеет числовой тип данных (type="number") (рисунок 9), а для проведения атаки в это поле требуется записать SQL-код. Преодолеть это препятствие оказалось не сложно - нужно снять валидатор формата данных прямо на веб-странице. В браузере Firefox с помощью встроенных средств отладки веб-страниц (Tools> Web developer> Inspector) любой человек может исследовать код страницы, в том числе, в виде иерархической структуры HTML- документа DOM (Document Object Model), а также изменять произвольным образом эту структуру.

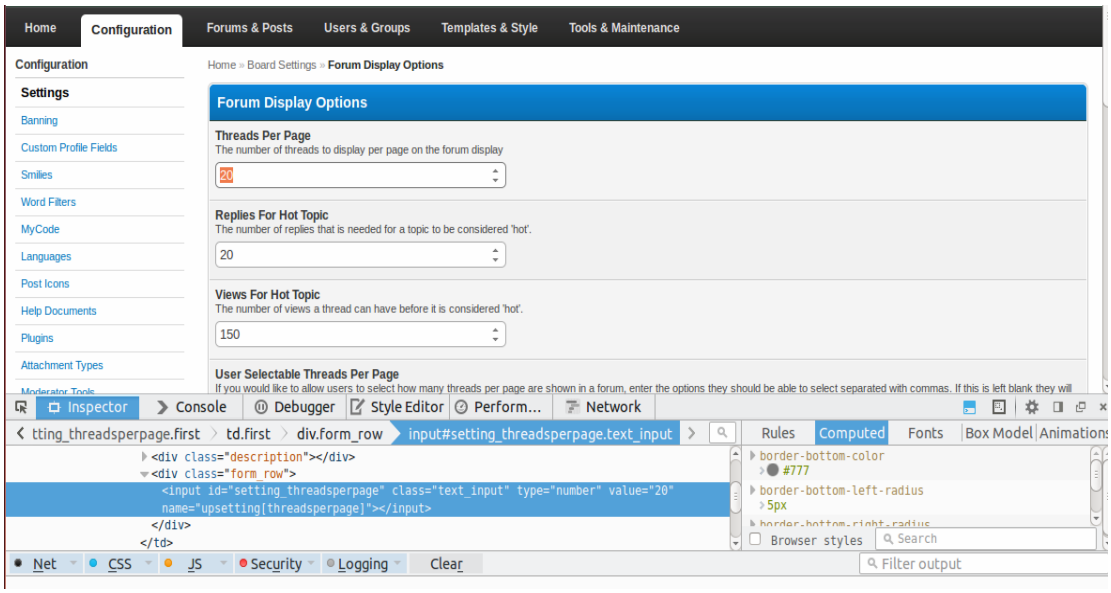


Рисунок 15 - Исходный код для поля ввода значений Threads Per Page

Для проведения SQL-injection реализован следующий алгоритм:

1) Мышью выбирается поле ввода значения Threads Per Page с валидатором формата на странице настроек;

2) В нижней области окна стирается атрибут type="number" (рисунок 16). При этом на страничке у поля ввода исчезают характерные стрелочки инкремента/декремента числового значения и поле становится доступным для ввода SQL-кода. Важно, что сам файл HTML при этом не меняется.

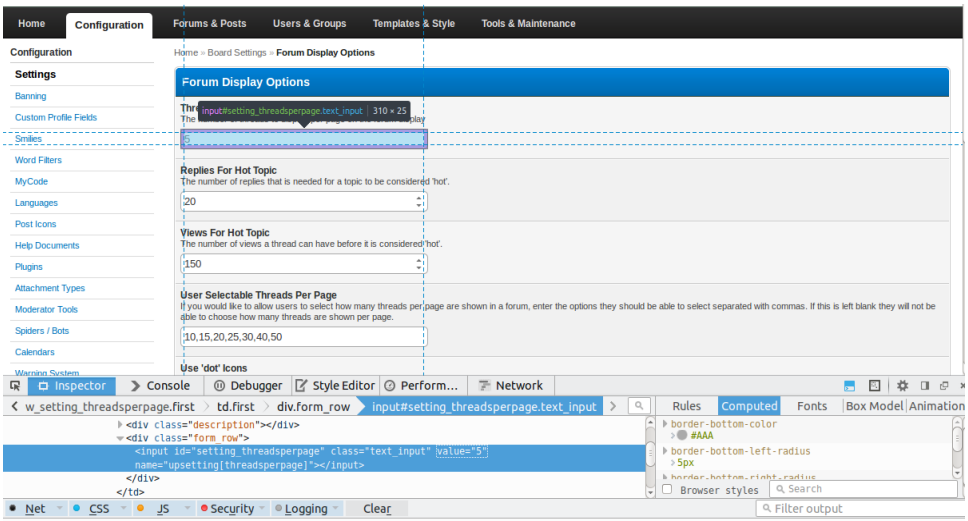


Рисунок 16 - Внесение изменений в код для поля ввода значений Threads Per Page

3) После того, как валидатор формата данных снят, можно внедрить на сервер строку с SQL-injection (рисунок 17).

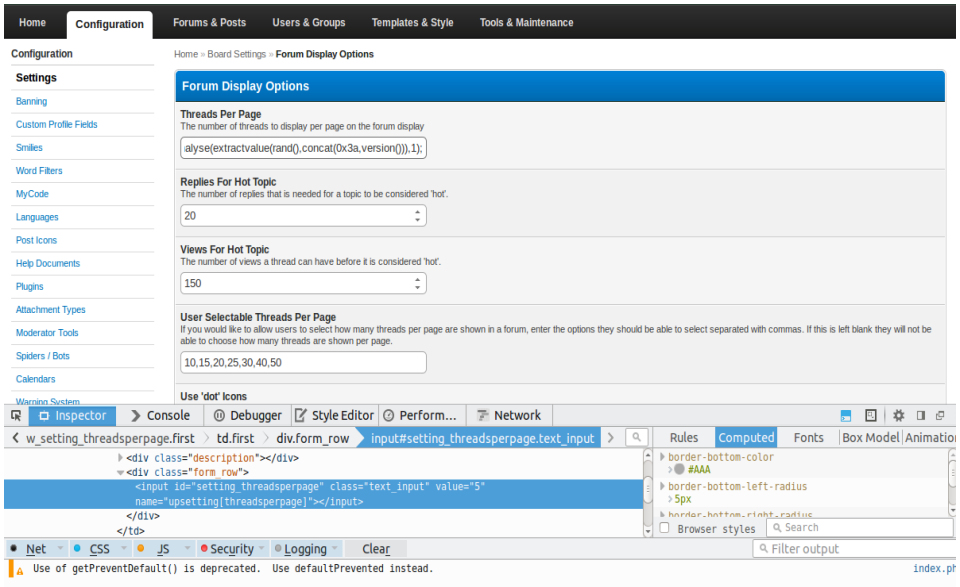


Рисунок 17 - Внедрение SQL- кода в поле ввода значений Threads Per Page

4) Далее, настройки сохраняются нажатием соответствующей кнопки и SQL-injection успешно реализована (рисунки 18-19).

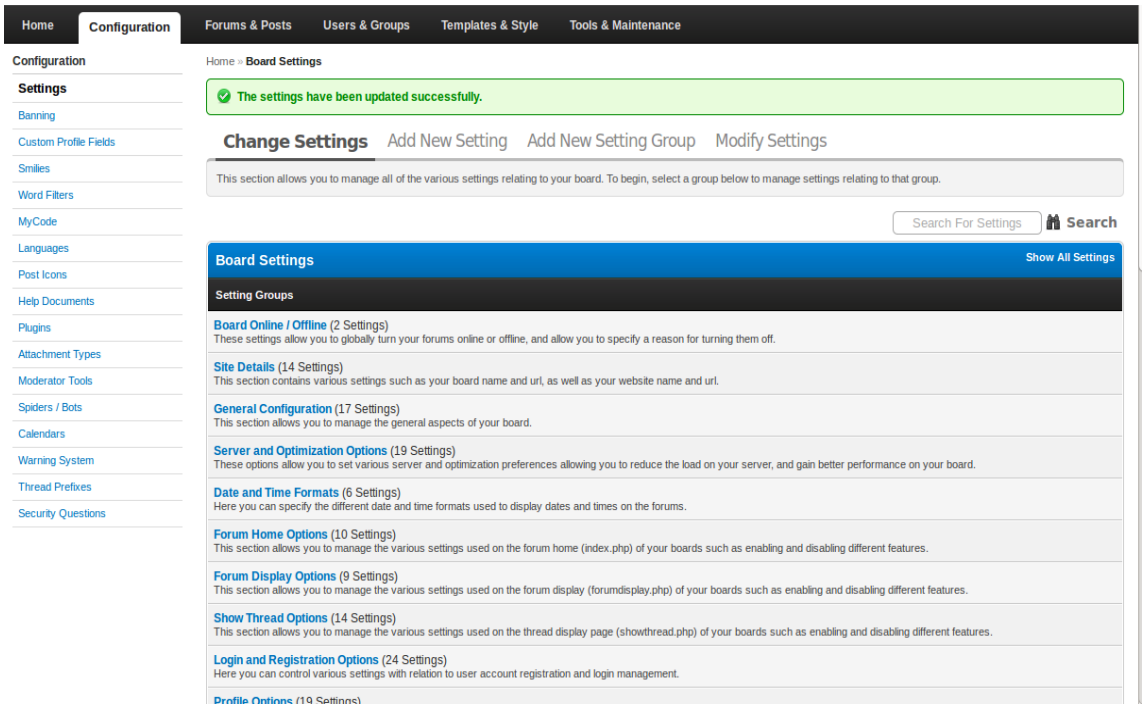


Рисунок 18 - Успешное внедрение SQL- кода в поле ввода значений Threads Per Page

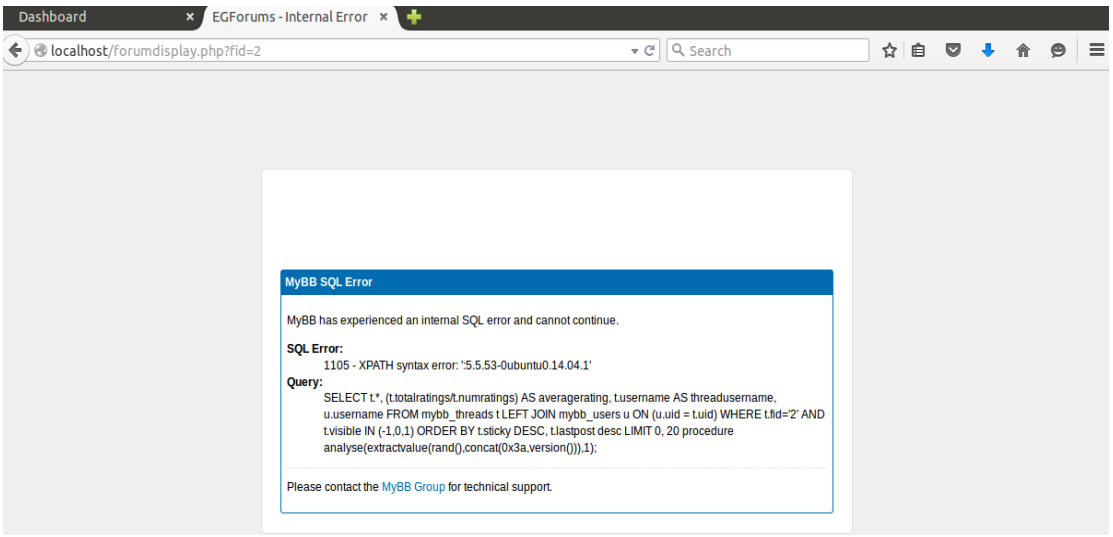


Рисунок 19 - Результат проведения атаки SQL-injection

При проведении SQL-injection удалось извлечь некоторые данные из базы данных, версию MySQL.

3.4.2 Тестирование метода для обнаружения SQLIA на сервис MyBB

Для апробации разработанного нейросетевого метода обнаружения аномалий в целях выявления реальной, специальным образом смоделированной атаки SQL-injection с помощью обученной нейросети была получена рабочая выборка (test sample), состоящая из значений корреляционных функций «запрос-ответ» и содержащая в себе уникальные характеристики типовых запросов и ответов MyBB, а также вредоносных запросов и аномальных реакций сервера.

Для формирования рабочей выборки, получение и предъявление на вход нейросети которой позволит продемонстрировать не только распознавание ею нормальных образов, но и выявления аномалий, была разработана схема как штатной, так и нештатной эксплуатации возможностей MyBB. Схема действий для формирования рабочей выборки для обнаружения SQLIA описана в таблицах 2-4. Для записи трафика, по которому в последствии будет вычислена рабочая выборка, были реализованы действия из таблицы 2, также согласно таблице 3 генерировались штатные запросы на странице настроек (http://localhost/admin/index.php?module=config) и на просмотр страницы, отображающей форум, My Forum (http://localhost/forumdisplay.php?fid=2). После чего, как показано в таблице 4, в уязвимое поле Threads Per Page помимо разрешенных числовых значений в разные моменты времени 3 раза был введен текст SQL-injection, то есть было проведено три SQLIA, согласно эксплойту, изложенному в п. 3.4.1.

Таблица 3 - Сценарий штатной эксплуатации уязвимого поля MyBB

Наименование			Назначение	Действия	Параметры действий
Configuration	Settings (http://localhost/admin/index.php?module=config)	Forum Display Options (http://localhost/admin/index.php?module=config-settings&action=change&gid=7)	Управление параметрами отображения форума (forumdisplay.php)	Установка значения в поле Threads Per Page	Установить 1
Save Settings (http://localhost/admin/index.php?module=config-settings)			Сохранение введенных настроек	–	–
EGForums	MyCategory	My Forum (http://localhost/forumdisplay.php?fid=2)	Отображение страницы форума	–	–
Configuration	Settings (http://localhost/admin/index.php?module=config)	Forum Display Options (http://localhost/admin/index.php?module=config-settings&action=change&gid=7)	Управление параметрами отображения форума (forumdisplay.php)	Установка значения в поле Threads Per Page	Установить 2
Save Settings (http://localhost/admin/index.php?module=config-settings)			Сохранение введенных настроек	–	–
EGForums	MyCategory	My Forum (http://localhost/forumdisplay.php?fid=2)	Отображение страницы форума	–	–
Configuration	Settings (http://localhost/admin/index.php?module=config)	Forum Display Options (http://localhost/admin/index.php?module=config-settings&action=change&gid=7)	Управление параметрами отображения форума (forumdisplay.php)	Установка значения в поле Threads Per Page	Установить 100
Save Settings (http://localhost/admin/index.php?module=config-settings) -			Сохранение введенных настроек	–	–
EGForums	MyCategory	My Forum (http://localhost/forumdisplay.php?fid=2)	Отображение страницы форума	–	–

Таблица 4 - Сценарий эксплуатации уязвимого поля MyBB для формирования тестовой выборки

Наименование			Назначение	Действия	Параметры действий
Save Settings (http://localhost/admin/index.php?module=config-settings)			Сохранение введенных настроек	–	–
EGForums	MyCategory	My Forum (http://localhost/forumdisplay.php?fid=2)	Отображение страницы форума	–	–
Повторение при установке различных числовых значений в поле Threads Per Page				x73	
Configuration	Settings (http://localhost/admin/index.php?module=config)	Forum Display Options (http://localhost/admin/index.php?module=config-settings&action=change&gid=7)	Управление параметрами отображения форума (forumdisplay.php)	Установка значения в поле Threads Per Page	Установить 75
Save Settings (http://localhost/admin/index.php?module=config-settings) -			Сохранение введенных настроек	–	–
Configuration	Settings (http://localhost/admin/index.php?module=config)	Forum Display Options (http://localhost/admin/index.php?module=config-settings&action=change&gid=7)	Управление параметрами отображения форума (forumdisplay.php)	Ввод SQLIA в поле Threads Per Page	Ввести 20 procedure analyse(extractvalue(rand(), concat(0x3a,version())),1);
Save Settings (http://localhost/admin/index.php?module=config-settings)			Сохранение введенных настроек	–	–
EGForums	MyCategory	My Forum (http://localhost/forumdisplay.php?fid=2)	Отображение страницы форума	–	–
Повторение шагов при установке различных числовых значений в поле Threads Per Page				x5	x7

Из КФ, вычисленных ранее описанным способом по трафику, зафиксированному во время выполнения нормальных действий пользователем и внедрения SQL-кода, согласно схемам 2, 3 и 4, составляется рабочая выборка. Предполагается, что КФ, рассчитанные в момент проведения SQLIA, не типичны по отношению к корреляционным функциям, описывающим штатные запросы и их обработку, и это отличие может быть обнаружено нейросетью, обученной на корреляционных функциях, характеризующих исключительно штатную эксплуатацию MyBB.

Таким образом, рабочая выборка содержит как признаки, характеризующие область нормальных запросов и реакций сервера, в частности, при вводе разрешенных числовых значений в поле Threads Per Page и действий из таблицы 2, так и аномальные реакции сервера после проведения атак SQL-injection attack (SQLIA), приводящие к ошибке работы MyBB.

Подав на вход обученной нейросети элементы рабочей выборки, проверим, сможет ли нейронная сеть их отличить, распознав КФ, похожие на знакомые ей КФ из обучающей выборки, и выделив непохожие.

Графики ошибки реконструкции, рассчитанной для элементов обучающей (training sample) и рабочей выборки (test sample), среди образцов которой находятся образцы, соответствующие времени проведения SQLIA, показаны на рисунке 20.

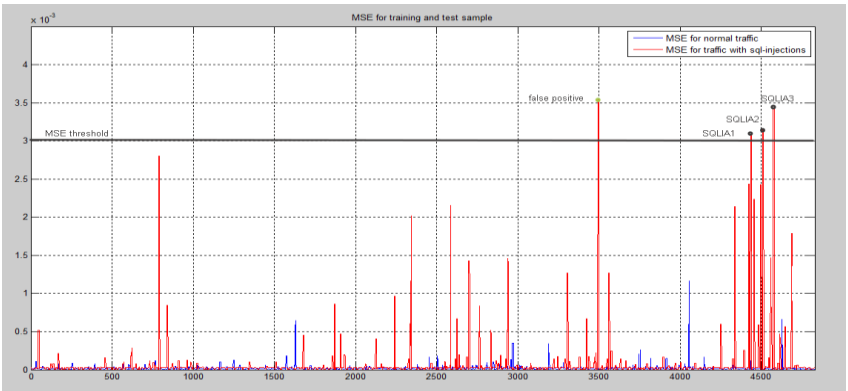


Рисунок 20 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка

Получена ошибка реконструкции для корреляционных функций, описывающих штатный трафик MyBB, и ошибка реконструкции, когда на вход обученной нейронной сети были поданы корреляционные функции, описывающие еще и трафик, содержащий информацию о проведении трех атак SQL-injection.

Максимум ошибки реконструкции для обучающей выборки составляет 1.2×10^{-3}, а максимум ошибки реконструкции для рабочей выборки составляет 3.5×10^{-3}. В данном случае порог ошибки реконструкции может быть выбран равным 0.003, что в несколько раз больше, чем максимальное значение ошибки реконструкции для обучающей выборки.

Критерием обнаружения аномалии является превышение значением ошибки реконструкции для конкретного элемента заданного порога ошибки. В таком случае можно утверждать, что поданная на вход корреляционная функция не опознана нейронной сетью и является аномальной, нетипичной для данного сетевого обрабатывающего устройства и вполне может быть признаком несанкционированной деятельности на сайте.

Учитывая, что большинство значений ошибки реконструкции по рабочей выборке не превышают пороговое значение, то можно сделать вывод, что нейронная сеть успешно распознает НКФ в рабочей выборке, соответствующие штатным запросам.

Обращая внимание на превышающие порог значения ошибки для рабочей выборки, можно предположить, что нейронная сеть успешно обучилась обнаруживать КФ, непохожие на КФ, по которым она была обучена.

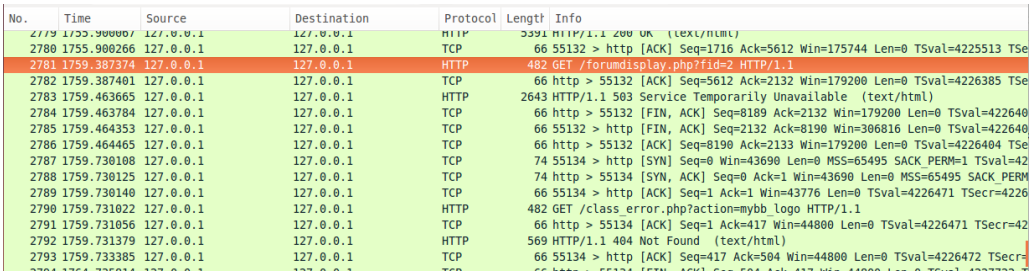
Для идентификации, являются ли КФ, которые нейронная сеть выделяет, как аномальные, соответствующими моменту проведения атак, была написана программа (см. Приложение Б), которая по индексу ошибки реконструкции, превышающей допустимый порог, определяла соответствующее время в трафике. Благодаря этому можно верифицировать, действительно ли аномальная КФ, обнаруженная нейросетью, была рассчитана по трафику в

момент SQLIA. Алгоритм программы заключается в следующем. Индекс ошибки реконструкции на графике равен индексу элемента в рабочей выборке, то есть КФ, рассчитанной по тому или иному отрезку времени. Учитывая, что в рабочей выборке участвовали только не нулевые КФ, то необходимо найти индекс данной аномальной КФ в исходном файле, где есть однозначное соответствие между порядковым номером КФ и отрезком времени трафика, в который она была рассчитана.

Результат проведения сопоставления представлен на рисунке 21.

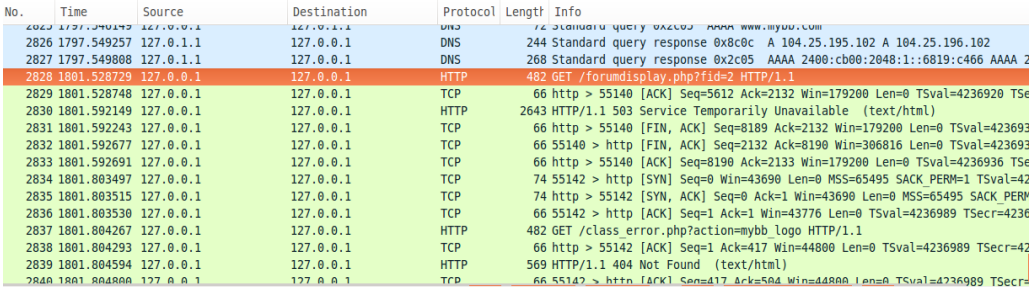
Рисунок 21 - Результат работы программы

Оказалось, что нейронной сети удалось правильно распознать значительное число нормальных примеров из рабочей выборки, распознать три КФ, рассчитанные в момент проведения трех SQLIA, что выделено на рисунке 20, как SQLIA1, SQLIA2 и SQLIA3, при этом было лишь одно ложное срабатывание, соответствующее значение маркировано на рисунке 20, как «false positive». На рисунках 22-24 представлен трафик в моменты времени, соответствующие аномальным КФ.



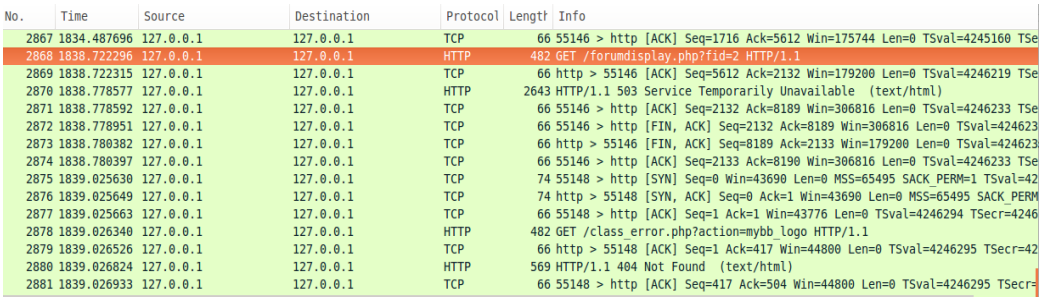
No.	Time	Source	Destination	Protocol	Lengt	Info
2779	1755.900067	127.0.0.1	127.0.0.1	HTTP	5391	HTTP/1.1 200 OK (text/html)
2780	1755.900266	127.0.0.1	127.0.0.1	TCP	66	55132 > http [ACK] Seq=1716 Ack=5612 Win=175744 Len=0 TSval=4225513 TSe
2781	1759.387374	127.0.0.1	127.0.0.1	HTTP	482	GET /forumdisplay.php?fid=2 HTTP/1.1
2782	1759.387401	127.0.0.1	127.0.0.1	TCP	66	http > 55132 [ACK] Seq=5612 Ack=2132 Win=179200 Len=0 TSval=4226385 TSe
2783	1759.463665	127.0.0.1	127.0.0.1	HTTP	2643	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
2784	1759.463784	127.0.0.1	127.0.0.1	TCP	66	http > 55132 [FIN, ACK] Seq=8189 Ack=2132 Win=179200 Len=0 TSval=422640
2785	1759.464353	127.0.0.1	127.0.0.1	TCP	66	55132 > http [FIN, ACK] Seq=2132 Ack=8190 Win=306816 Len=0 TSval=422640
2786	1759.464465	127.0.0.1	127.0.0.1	TCP	66	http > 55132 [ACK] Seq=8190 Ack=2133 Win=179200 Len=0 TSval=4226404 TSe
2787	1759.730108	127.0.0.1	127.0.0.1	TCP	74	55134 > http [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=42
2788	1759.730125	127.0.0.1	127.0.0.1	TCP	74	http > 55134 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM
2789	1759.730140	127.0.0.1	127.0.0.1	TCP	66	55134 > http [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=4226471 TSecr=4226
2790	1759.731022	127.0.0.1	127.0.0.1	HTTP	482	GET /class_error.php?action=mybb_logo HTTP/1.1
2791	1759.731056	127.0.0.1	127.0.0.1	TCP	66	http > 55134 [ACK] Seq=1 Ack=417 Win=44800 Len=0 TSval=4226471 TSecr=42
2792	1759.731379	127.0.0.1	127.0.0.1	HTTP	569	HTTP/1.1 404 Not Found (text/html)
2793	1759.733385	127.0.0.1	127.0.0.1	TCP	66	55134 > http [ACK] Seq=417 Ack=504 Win=44800 Len=0 TSval=4226472 TSecr=

Рисунок 22 – Пакеты SQLIA1, описываемые АКФ 4433



No.	Time	Source	Destination	Protocol	Lengt	Info
2825	1797.540149	127.0.0.1	127.0.1.1	DNS	72	Standard query 0x2c05 AAAA www.mybb.com
2826	1797.549257	127.0.1.1	127.0.0.1	DNS	244	Standard query response 0x8c0c A 104.25.195.102 A 104.25.196.102
2827	1797.549808	127.0.1.1	127.0.0.1	DNS	268	Standard query response 0x2c05 AAAA 2400:cb00:2048:1::6819:c466 AAAA 2
2828	1801.528729	127.0.0.1	127.0.0.1	HTTP	482	GET /forumdisplay.php?fid=2 HTTP/1.1
2829	1801.528748	127.0.0.1	127.0.0.1	TCP	66	http > 55140 [ACK] Seq=5612 Ack=2132 Win=179200 Len=0 TSval=4236920 TSe
2830	1801.592149	127.0.0.1	127.0.0.1	HTTP	2643	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
2831	1801.592243	127.0.0.1	127.0.0.1	TCP	66	http > 55140 [FIN, ACK] Seq=8189 Ack=2132 Win=179200 Len=0 TSval=423693
2832	1801.592677	127.0.0.1	127.0.0.1	TCP	66	55140 > http [FIN, ACK] Seq=2132 Ack=8190 Win=306816 Len=0 TSval=423693
2833	1801.592691	127.0.0.1	127.0.0.1	TCP	66	http > 55140 [ACK] Seq=8190 Ack=2133 Win=179200 Len=0 TSval=4236936 TSe
2834	1801.803497	127.0.0.1	127.0.0.1	TCP	74	55142 > http [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=42
2835	1801.803515	127.0.0.1	127.0.0.1	TCP	74	http > 55142 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM
2836	1801.803530	127.0.0.1	127.0.0.1	TCP	66	55142 > http [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=4236989 TSecr=4236
2837	1801.804267	127.0.0.1	127.0.0.1	HTTP	482	GET /class_error.php?action=mybb_logo HTTP/1.1
2838	1801.804293	127.0.0.1	127.0.0.1	TCP	66	http > 55142 [ACK] Seq=1 Ack=417 Win=44800 Len=0 TSval=4236989 TSecr=42
2839	1801.804594	127.0.0.1	127.0.0.1	HTTP	569	HTTP/1.1 404 Not Found (text/html)
2840	1801.804800	127.0.0.1	127.0.0.1	TCP	66	55142 > http [ACK] Seq=417 Ack=504 Win=44800 Len=0 TSval=4236989 TSecr=

Рисунок 23 – Пакеты SQLIA2 и описываемые АКФ 4508



No.	Time	Source	Destination	Protocol	Lengt	Info
2867	1834.487696	127.0.0.1	127.0.0.1	TCP	66	55146 > http [ACK] Seq=1716 Ack=5612 Win=175744 Len=0 TSval=4245160 TSe
2868	1838.722296	127.0.0.1	127.0.0.1	HTTP	482	GET /forumdisplay.php?fid=2 HTTP/1.1
2869	1838.722315	127.0.0.1	127.0.0.1	TCP	66	http > 55146 [ACK] Seq=5612 Ack=2132 Win=179200 Len=0 TSval=4246219 TSe
2870	1838.778577	127.0.0.1	127.0.0.1	HTTP	2643	HTTP/1.1 503 Service Temporarily Unavailable (text/html)
2871	1838.778592	127.0.0.1	127.0.0.1	TCP	66	55146 > http [ACK] Seq=2132 Ack=8189 Win=306816 Len=0 TSval=4246233 TSe
2872	1838.778951	127.0.0.1	127.0.0.1	TCP	66	55146 > http [FIN, ACK] Seq=2132 Ack=8189 Win=306816 Len=0 TSval=424623
2873	1838.780382	127.0.0.1	127.0.0.1	TCP	66	http > 55146 [FIN, ACK] Seq=8189 Ack=2133 Win=179200 Len=0 TSval=424623
2874	1838.780397	127.0.0.1	127.0.0.1	TCP	66	55146 > http [ACK] Seq=2133 Ack=8190 Win=306816 Len=0 TSval=4246233 TSe
2875	1839.025630	127.0.0.1	127.0.0.1	TCP	74	55148 > http [SYN] Seq=0 Win=43690 Len=0 MSS=65495 SACK_PERM=1 TSval=42
2876	1839.025649	127.0.0.1	127.0.0.1	TCP	74	http > 55148 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0 MSS=65495 SACK_PERM
2877	1839.025663	127.0.0.1	127.0.0.1	TCP	66	55148 > http [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSval=4246294 TSecr=4246
2878	1839.026340	127.0.0.1	127.0.0.1	HTTP	482	GET /class_error.php?action=mybb_logo HTTP/1.1
2879	1839.026526	127.0.0.1	127.0.0.1	TCP	66	http > 55148 [ACK] Seq=1 Ack=417 Win=44800 Len=0 TSval=4246295 TSecr=42
2880	1839.026824	127.0.0.1	127.0.0.1	HTTP	569	HTTP/1.1 404 Not Found (text/html)
2881	1839.026933	127.0.0.1	127.0.0.1	TCP	66	55148 > http [ACK] Seq=417 Ack=504 Win=44800 Len=0 TSval=4246295 TSecr=

Рисунок 24 – Пакеты SQLIA3 и описываемые АКФ 4573

Выделенный пакет и следующие за ним, характеризуют результат проведения SQLIA – вызов ошибки работы MyBB с выводом на экран информации из БД.

Для визуализации распознанных элементов выборки и выделяющихся аномальных была создана нейронная сеть, аналогичная обученной (рисунок 25), но содержащая только слои сжатия размерности данных и «узкое горло», по выходу которого можно судить о выделении нейронной сетью основных кластеров из входных данных. Для этого все настроенные весовые коэффициенты из полной нейронной сети были перенесены на новую нейронную сеть (рисунок 26), соответствующую левой части обученной ранее нейронной сети (nn2).

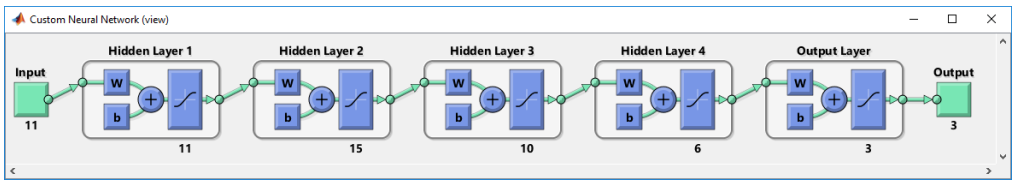


Рисунок 25 - Структура нейронной сети nn2half

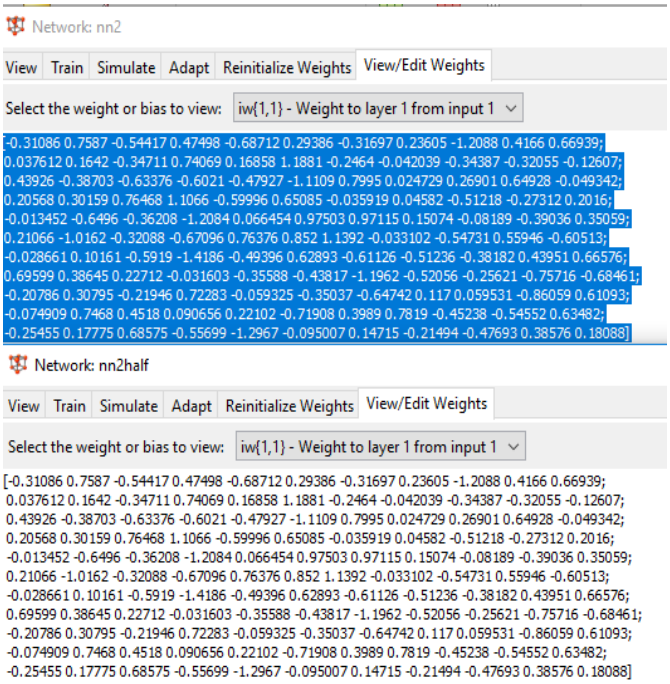


Рисунок 26 - Пример переноса весовых коэффициентов первого слоя обученной нейронной сети nn2 в первый слой nn2half

После создания нейронной сети, в выходном слое которой три нейрона, на ее вход были поданы обучающая и рабочая выборка для визуализации их представления на выходе «узкого горла». Нейронная сеть явно относит практически все данные проверочной выборки к одному кластеру, кластеру нормальных данных, из которых состоит обучающая выборка, однако есть и отдаленные от центра их локализации точки - аномалии.

На рисунке 27 синим цветом изображено пространство нормальных КФ, по которым нейронная сеть была обучена, а красным цветом изображены КФ из рабочей выборки, большинство из которых нейросеть распознала и отнесла к кластеру нормальных данных.

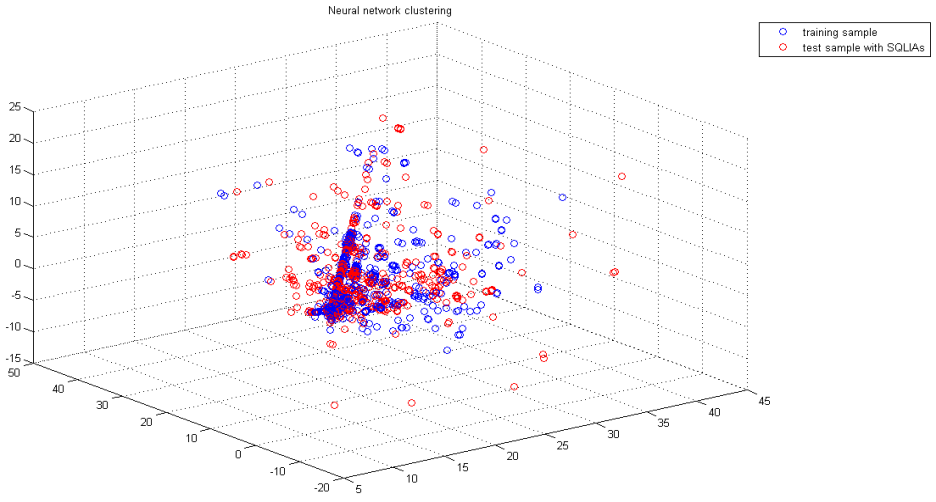


Рисунок 27 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с SQLIAs

Рассмотрим вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были выявлены номера отсчётов с максимальной ошибкой реконструкции, которые соответствуют аномальным корреляционным функциям. Для сравнительного анализа на рисунке 28 продемонстрирована разница между нормальными КФ, отвечающими нормальным запросам и ответам, и аномальными КФ, рассчитанными в момент проведения SQLIA.

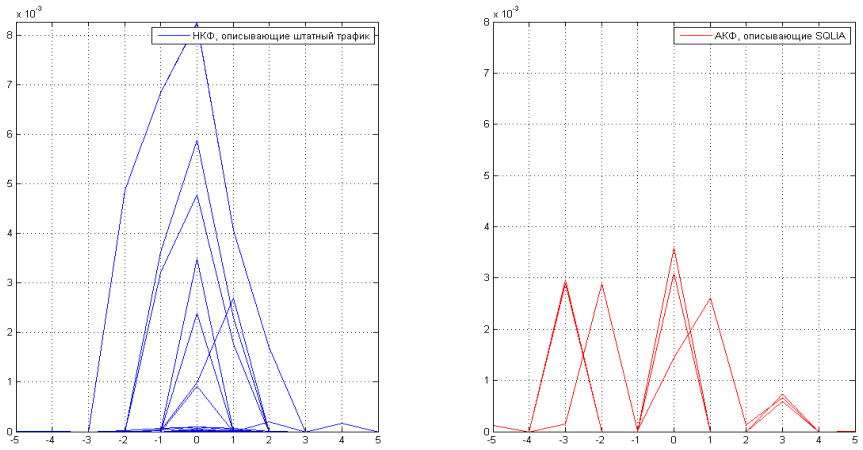


Рисунок 28 - Нормальные КФ из обучающей выборки и аномальные КФ, соответствующие трем SQLIA

Основное отличие между этими КФ, которые после визуального анализа может заметить человек – это мультимодальный характер КФ, соответствующих внедрению SQL-кода и аномальной реакции сервера на запрос.

Полученные результаты позволяют всерьез рассматривать вопрос о применимости нейросетевого метода для обнаружения реальных атак, например, таких как SQL-injection, для сервисов типа myBB и внушают перспективы для исследования задачи распространения метода для обнаружения других сетевых атак.

3.4.3 Демонстрация flood-атак на сетевой сервис

Наряду с атаками выполнения кода с точки зрения значимости их своевременного обнаружения и предотвращения по-прежнему остаются DDoS- атаки. Атаки типа DoS (Denial of Service) и DDoS (Distributed DoS) являются одними из самых распространенных и опасных атак для прикладных и промышленных систем, поскольку их цель – выведение системы из строя, что в ряде случаев приводит к существенным материальным потерям и ущербу для атакованной организации. Кроме того, такие атаки все чаще становятся оружием конкурентной борьбы, направленным на крупные сетевые сервисы.

Выделяют достаточно много типов DDoS-атак. Наиболее известными представителями такого типа атак как flood являются атаки сетевого уровня SYN-flood и UDP-flood, также встречаются ICMP-flood [55] и TCP-flood, и атаки прикладного уровня- DNS-flood и HTTP-flood, атака, которую довольно непросто предотвратить [20].

Например, при атаке SYN-flood создается массовый поток запросов на подключение к серверу, что делает невозможным их обработку. При получении каждого SYN-пакета система выделяет ресурсы в памяти, формирует ответ SYN+ACK с криптографическими данными, проверяет таблицы сессий и выполняет другие операции, требующие процессорного времени. Обычно отказ в обслуживании происходит при интенсивности

атаки от 100 до 500 тысяч пакетов в секунду [28]. Однако злоумышленник с доступом к гигабитному каналу может направить до 1,5 миллионов пакетов за это же время. Атака UDP-flood реализуется с использованием фрагментированных UDP-пакетов небольшого размера, обработка и сборка которых также требуют значительных системных ресурсов. Защита от flood-атак чаще всего обеспечивается DPI-системами (глубокий анализ трафика), которые могут фильтровать трафик, блокировать нерелевантные протоколы или ограничивать их пропускную способность. Однако такие системы установлены не повсеместно.

Как ни странно, не смотря на наличие широкого выбора средств защит от различных видов DDoS-атак, они все еще представляют серьезную опасность для служб и приложений. Кроме того, DDoS-атаки нередко применяются как отвлекающий маневр, чтобы скрыть другие виды атак, такие как взлом сайтов или веб-приложений.

Актуальность данной проблемы подчеркивается серией DDoS-атак, направленных на крупнейшие российские банки, включая Сбербанк, Альфа-Банк, ВТБ, Банк Москвы и Росбанк [3]. Главной задачей киберпреступников было парализовать работу сайтов этих организаций. Для этого они применяли многовекторные атаки, такие как syn-flood (направленные на истощение ресурсов операционной системы) и http-flood (нацеленные на перегрузку веб-сервера и блокирование доступа к сервисам). Эти атаки отличались высокой скоростью и сложностью, достигая пиковой интенсивности около 2 ГБ/с. Потери от атак типа «отказ в обслуживании» затрагивают не только финансовые показатели банков, но и наносят серьезный ущерб их репутации [4].

Основываясь на данных «Лаборатории Касперского», Securelist также отмечает рост популярности сложных атак (на уровне приложений, HTTPS). В качестве примера приводится комбинированная атака (SYN + TCP Connect + HTTP Flood + UDP Flood) на Московскую фондовую биржу [18].

Сети ботнетов, созданные на основе IoT-устройств, представляют серьезную угрозу и создают значительные трудности для экспертов в области кибербезопасности, поскольку трафик, который они производят по протоколу TCP, крайне сложно отличить от легального. Например, ботнет Mirai создавал потоки атак, используя TCP ACK+push и TCP SYN flood. Исследователи считают, что ботнет Necurs, сеть которого насчитывает более 6 млн. ботов, модернизирован и сможет заставлять подконтрольные машины генерировать HTTP- или UDP-флуд на ресурс жертвы [25].

Также из [79] известно, что службы и приложения все чаще подвергаются flood-атакам, кроме того, в ближайшем будущем ожидается активное появление еще более крупных ботнетов, чем Mirai, способных к flood-атакам даже без использования amplification-протоколов.

Например, недавно был обнаружен новый тип ботнета WireX, который проводит атаки методом UDP-flood, используя устройства на базе Android. Согласно анализу исследователей, изученный экземпляр бота запускает 50 параллельных потоков, каждый из которых способен отправлять до 10 миллионов UDP-пакетов размером 512 байт [71].

Из всего этого следует, что задача своевременного обнаружения признаков начинающейся flood-атаки не только на сетевые сервисы, но и на незащищенные платформы промышленного интернета вещей, датчики, от показаний которых может зависеть управление критически важным промышленным объектом, носит принципиально важный характер.

Для реализации таких атак в целях тестирования веб-серверов в одном из дистрибутивов Linux – Kali Linux, созданном, в частности, для специалистов информационной безопасности, разработан ряд утилит и инструментов.

Например, для проведения атаки типа flood на целевой веб-сервер можно задействовать утилиту hping3. Команда может выглядеть следующим образом:

- hping3 -S <IP-адрес сервера> -i u1 --rand-source -p 80

Здесь hping3 отправляет SYN-запросы на порт 80 (-p). Атакуемый сервер пытается отвечать пакетами SYN+ACK, но из-за использования случайного IP-источника (--rand-source) соединение не завершается. В итоге сервер перегружается обработкой этих запросов, что мешает работе легальных пользователей.

Примером TCP flood служит атака TCP RST, которую можно организовать так:

- hping3 <IP-адрес сервера> -p 80 -R --rand-source –faster

В этом случае сервер получает поток RST-пакетов (-R), что создает чрезмерную нагрузку на систему.

Примеры команд для ICMP и UDP flood [97]:

- hping3 --udp <IP-адрес сервера> -i u1 --rand-source -p 80

- hping3 --icmp <IP-адрес сервера> -i u1 --rand-source

Цель таких атак – вызвать нестабильность системы, спровоцировать потерю пакетов или перегрузить сеть слишком большим объемом трафика.

В рамках настоящих экспериментов для генерации SYN-flood, TCP-flood, UDP-flood, ICMP-flood атак на целевой веб-сервер [57] были использованы соответственно следующие команды:

- sudo hping3 --faster -S -p 80 --rand-source 127.0.0.1;

- sudo hping3 -RSA -d 500 -p 80 --faster --rand-source 127.0.0.1;

- sudo hping3 -q -n -a 192.168.0.102 --udp -s 53 --keep -p 80 --faster 127.0.0.1;

- sudo hping3 --icmp -d 1000 -p 80 --faster -a 192.168.0.102 127.0.0.1.

Используемые параметры команд:

-S – отправка только пакетов SYN;

-a 10.0.9.150 – в роли источника пакета установить адрес 192.168.0.102;

-flood – режим флуда для отправки пакетов так быстро, как это возможно, без отображения входящих пакетов;

--faster – режим отправки 100 пакетов в секунду;

-R или --rst – установка RST флага;

-P или --push – установка PUSH флага;

-A или --ack – установка ACK флага;

-p 80– порт назначения 80;

--rand-source – использование рандомных IP адресов источника;

127.0.0.1 - целевой IP адрес;

- q – тихий режим;

-n – только цифровой выход;

-1 или --icmp – включение ICMP-режима;

-2 или --udp – включение UDP-режима.

- d 1000 – установка размера пакета;

- s 53 – установка порта источника 53;

--keep – фиксирует порт отправления, иначе он будет увеличиваться на 1 для каждого следующего пакета.

SYN-flood - атака на превышение максимального количества полуоткрытых сессий (рисунок 29). ICMP-flood используется вместе с большим размером пакетов, чтобы попробовать исчерпать входящий канал атакуемого сервера. Атака UDP-flood так же насыщает полосу пропускания. Атака TCP-flood представляла собой пакеты с установленными флагами RST, SYN, ACK (рисунок 30).

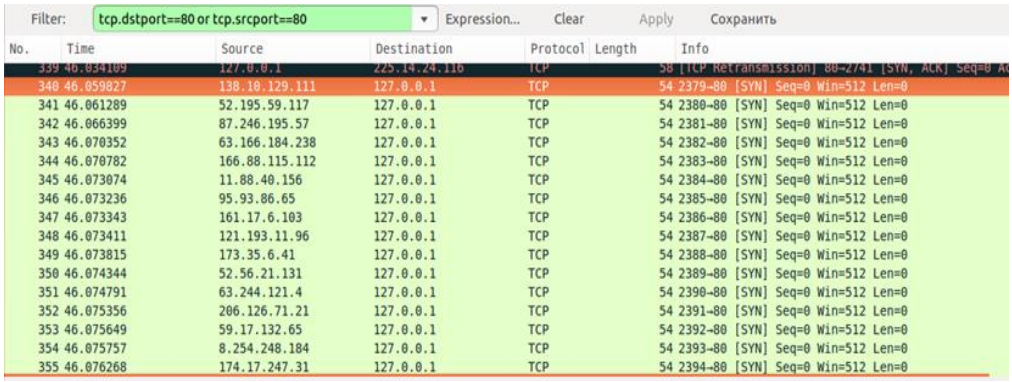


Рисунок 29 - Направление SYN-пакетов

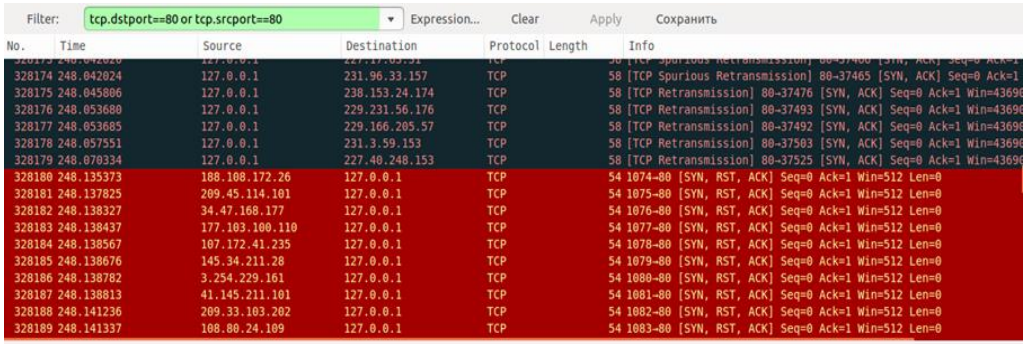


Рисунок 30 - TCP-flood

Также были организованы следующие атаки непосредственно на веб-сервер MyBB.

Одна из них – атака на медленные подключения [45]. Смысл данной атаки в том, что при подключении и очень медленном запросе страницы – соединение не сбрасывается, а у любого веб-сервера есть лимит на количество параллельных, одновременных подключений.

Для организации такой атаки были использованы следующие команды:

- slowhttptest -c 1000 -H -i 20 -r 200 -t GET -u http://127.0.0.1 -x 24 -p 3;

- slowhttptest -c 1000 -H -i 20 -r 200 -t POST -u http://127.0.0.1 -x 24 -p 3.

Используемые параметры:

-c – общее суммарное количество подключений;

-i – пауза между сессиями для загрузки части страницы (в рамках одного подключения);

-r – количество подключений в секунду;

-t GET – использовать GET (POST) запросы;

-u – URL. Поддерживает HTTP и HTTPS ссылки;

-x – количество загружаемых байт из части страницы за одну сессию (в рамках одного подключения);

-p – время ожидания при проверке подключения. Если ответ от сервера не получен за это время, то сервер считается недоступным;

-H – атака slow headers a.k.a. Slowloris.

Так же возможна генерация и других типов атак:

-B – атака slow body a.k.a R-U-Dead-Yet;

-R – атака range attack a.k.a Apache killer;

-X – атака Slow Read.

Результат, выводимый программой во время тестирования и содержащий основные характеристики проведенной атаки, представлен на рисунке 31. Как видно по правому рисунку, сервис MyBB становится недоступным достаточно быстро.

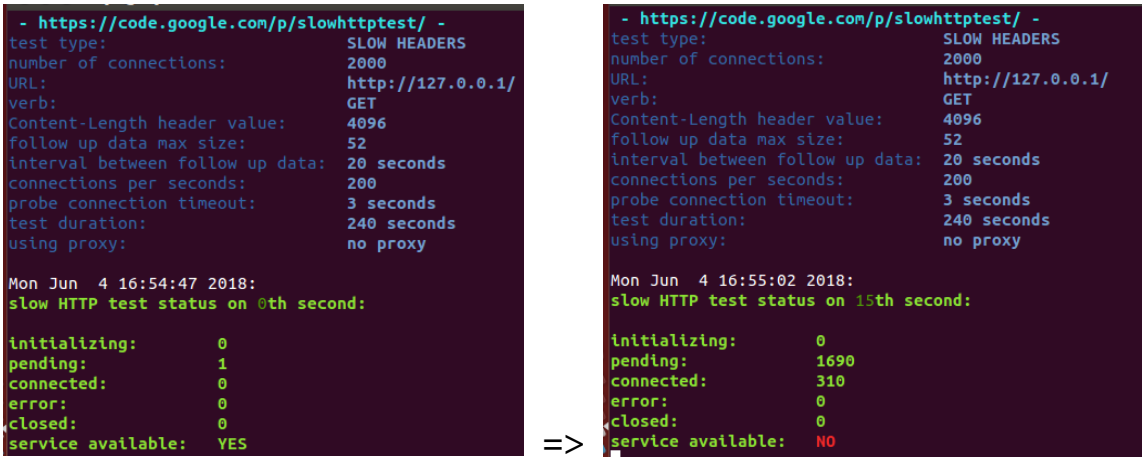


Рисунок 31 - Демонстрация реализации атаки slow headers

Для организации HTTP-flood было использована утилита для нагрузочного тестирования веб-серверов – siege, которая просто запрашивает в огромное количество потоков страницы сайта, после чего происходит исчерпание либо ресурсов сервера (если запрашивать тяжелые станицы), либо ресурсов исходящего канала (если запрашивать тяжелые файлы). Команда выглядела следующим образом:

- siege -i -c 250 http://localhost/admin/index.php?module=config-settings.

Используемые параметры:

-i – симулировать обычного пользователя;

-c – количество подключений;

page – запрашиваемая страница.

При использовании данной команды генерируется одновременная деятельность 250 пользователей, запрашивающих указанную страницу (рисунок 32).

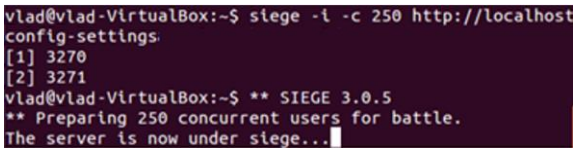


Рисунок 32 - Использование утилиты siege для нагрузочного тестирования

Так же можно данной утилите передать флагом -R файл, в котором будет перечислен список страниц для запроса.

3.4.4 Тестирование метода для обнаружения flood-атак

Ранее отмечалось, что flood-атаки заключаются в массовой отправке бесполезных пакетов с целью нарушения работы системы либо перегрузки её огромным объемом данных, который невозможно обработать, в силу предельной загруженности полосы пропускания. В любом случае, это приводит к отказу в обслуживании законных пользователей системы и задача обнаружения таких атак крайне важна, и для сетевых сервисов, и для веб-приложений, а также для IoT-устройств, характеризующихся ограниченными вычислительными мощностями и ресурсами. Поэтому работа метода для обнаружения именно этого типа атак также важна.

Для расчета рабочих выборок был осуществлен захват трафика при воспроизведении ранее захваченного штатного трафика на прослушиваемом программой Wireshark интерфейсе, а через некоторое время с адреса машины с установленным дистрибутивом Kali Linux были сгенерированы flood-атаки на целевой сервер способом, продемонстрированным выше.

Для получения рабочей выборки описанным ранее способом были сформированы временные ряды и рассчитаны корреляционные функции, часть которых включают в себя информацию о несанкционированном воздействии и аномальной реакции сервера. То есть рабочая выборка помимо

КФ, описывающих нормальную работу сайта, содержит еще КФ, которые являются потенциально аномальными по отношению к типовым корреляционным функциям из обучающей выборки. Именно это и должно позволить обученной нейросети обнаружить аномалии, соответствующие проведенным flood-атакам.

Итак, на вход настроенной нейросети были поданы поочередно рабочие выборки, содержащие помимо КФ, рассчитанных по сетевому трафику, захваченному во время штатной эксплуатации, КФ, рассчитанные по трафику, захваченному во время проведения различных flood-атак.

Результат работы нейросети при подаче на ее вход рабочей выборки (test sample), последовательности признаков штатного трафика (control trafic), а затем SYN-flood атак, представлен на рисунке 33. Для сравнения на графике отображен уровень ошибки реконструкции для элементов обучающей выборки (training sample).

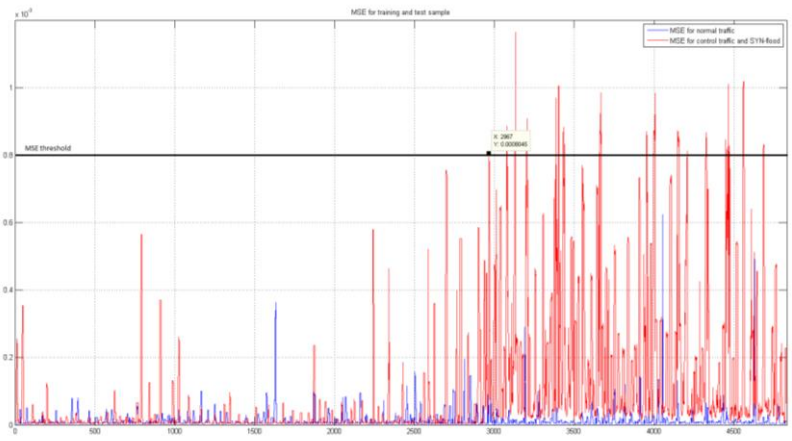


Рисунок 33 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка с признаками SYN-flood (красный цвет)

Согласно установленному в п. 3.3. пороговому значению, превышение которого означает обнаружение аномалии, на рисунке 33 установлена метка отсчета, начиная с которого деятельность определяется нейросетью как аномальная и подозрительная, учитывая продолжительное и частое

превышение ошибкой реконструкции установленного порога, нейросетью может быть диагностирована атака. Ошибка реконструкции для 2967 элемента рабочей выборки составляет $8{,}045 \cdot 10^{-3}$. С помощью программы удалось установить время, в течение которого были переданы пакеты, соответствующие данному признаку рабочей выборки (рисунок 34).

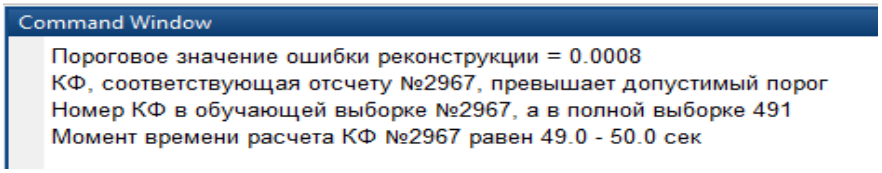


Рисунок 34 - Результат сопоставления ошибки реконструкции и времени в файле захваченного трафика

КФ с порядковым номером 2967 была рассчитана по трафику, зафиксированному с 49.0 сек. по 50.0 сек. Трафик в этот отрезок времени, действительно, представляет собой поток SYN-пакетов и изображен на рисунке 35.

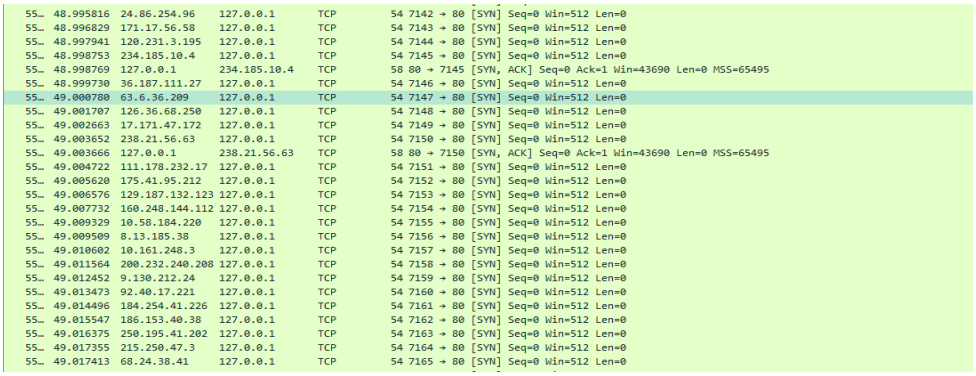


Рисунок 35 - Момент обнаружения SYN-flood атаки

Кроме того, нижняя граница ошибки реконструкции, начиная с отсчета номер 2932, заметно выросла по сравнению со значениями ошибки для нормального трафика. То есть, качественные изменения в поведении ошибки реконструкции стали заметны раньше, чем ошибка реконструкции превысила пороговое значение. Аномальное поведение было замечено нейронной сетью,

начиная с 45,5 сек. Однако, по трафику, изображенному на рисунке 36, видно, что на самом деле SYN-flood атака началась с 26,3 сек.

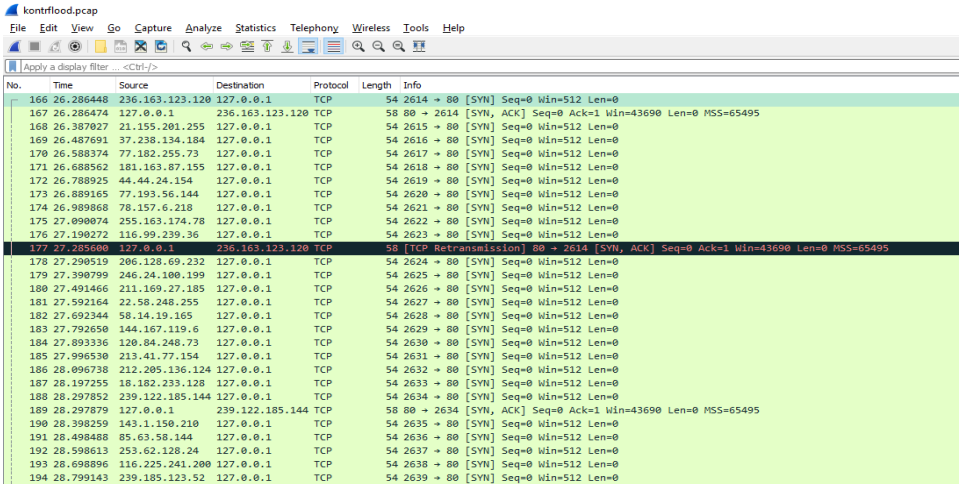


Рисунок 36 - Момент начала SYN-flood атаки

Учитывая, что TCP-пакеты с установленным флагом SYN сами по себе не являются вредоносными, более того, имеют тот же размер, что и обычные запросы, то задержка их обнаружения очевидна.

Покажем теперь, насколько нейронной сети удалось выделить во входных данных, обучающей (training sample) и рабочей выборки (sample with SYN-flood), основной нормальный кластер и аномалии. На рисунке 37 изображены данные на выходе «узкого горла» нейронной сети в трехмерном пространстве.

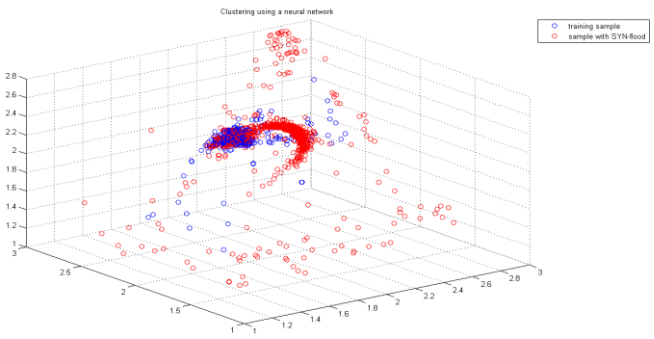


Рисунок 37 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с SYN-flood

По графику видно, что основная часть рабочей выборки локализована нейронной сетью в кластере нормальных данных обучающей выборки, а аномальные значения в значительной степени отстоят от центра кластера.

Рассмотрим, вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были построены некоторые КФ из обучающей выборки и некоторые КФ из рабочей выборки, начиная с момента обнаружения нейронной сетью аномального поведения (рисунок 38).

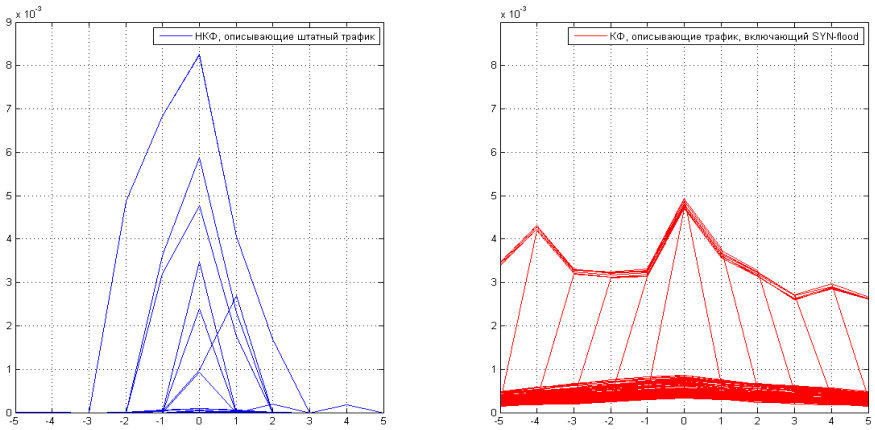


Рисунок 38 - Нормальные КФ из обучающей выборки и КФ рабочей выборки после начала SYN-flood атаки

Нейронной сети, очевидно, удалось обнаруживать аномальные КФ, соответствующие SYN-flood атаке, заметно отличающиеся по виду от нормальных КФ.

Аналогичные эксперименты были проведены для обнаружения TCP-flood атаки. После предъявления на вход обученной нейронной сети рабочей выборки (test sample), содержащей помимо признаков независимого нормального трафика (control traffic) признаки трафика, генерируемого при TCP-flood атаке, были получены значения ошибки реконструкции для всех элементов рабочей выборки. Для сравнения на рисунке 39 представлены значения ошибки реконструкции и для элементов обучающей выборки (training sample).

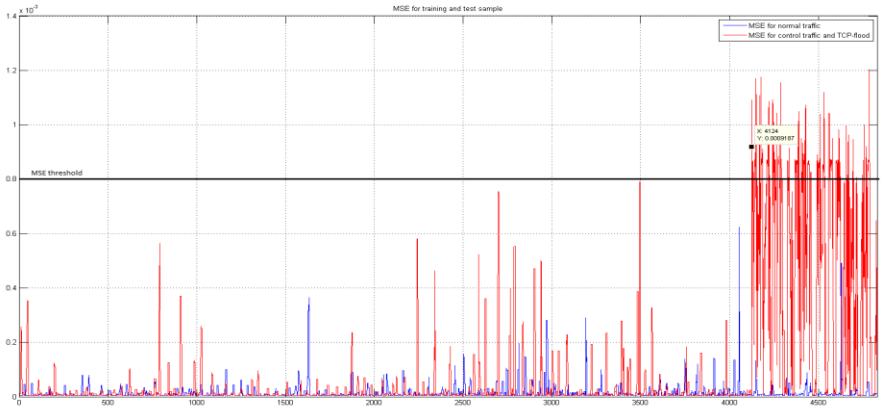


Рисунок 39 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка с признаками TCP-flood (красный цвет)

Согласно установленному в п. 3.3. пороговому значению, превышение которого означает обнаружение аномалии, на рисунке 39 установлена метка отсчета, начиная с которого деятельность определяется нейросетью как аномальная и подозрительная, учитывая продолжительное и частое превышение ошибкой реконструкции установленного порога, нейросетью может быть диагностирована атака. Ошибка реконструкции для элемента рабочей выборки с индексом 4124 составляет $9{,}187 \cdot 10^{-3}$. С помощью программы удалось установить время, в течение которого были переданы пакеты, соответствующие данному признаку рабочей выборки (рисунок 40).

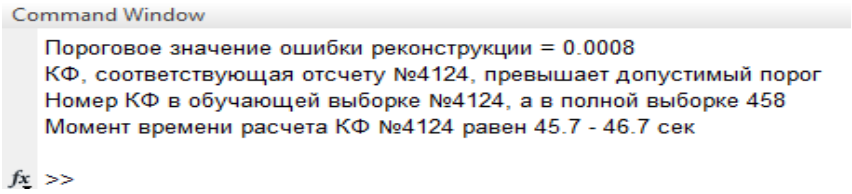


Рисунок 40 - Результат сопоставления ошибки реконструкции и времени в файле захваченного трафика

КФ с порядковым номером 4124 была рассчитана по трафику, зафиксированному с 45.7 сек. по 46.7 сек. Трафик в этот отрезок времени, действительно, представляет собой поток TCP-пакетов и изображен на рисунке 41.

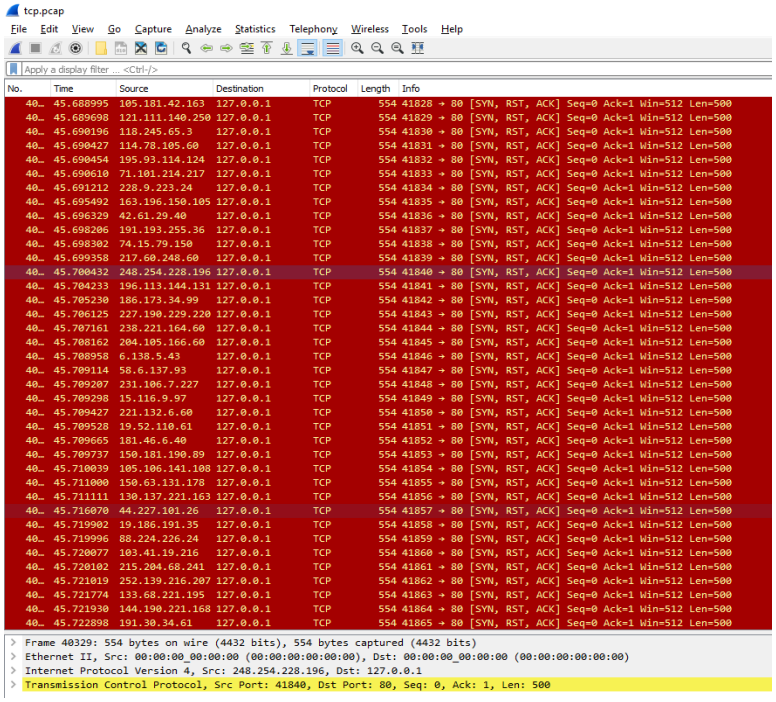


Рисунок 41 - Момент обнаружения TCP-flood атаки

Однако, по трафику, изображенному на рисунке 42, видно, что на самом деле TCP-flood атака началась с 17,39 сек.

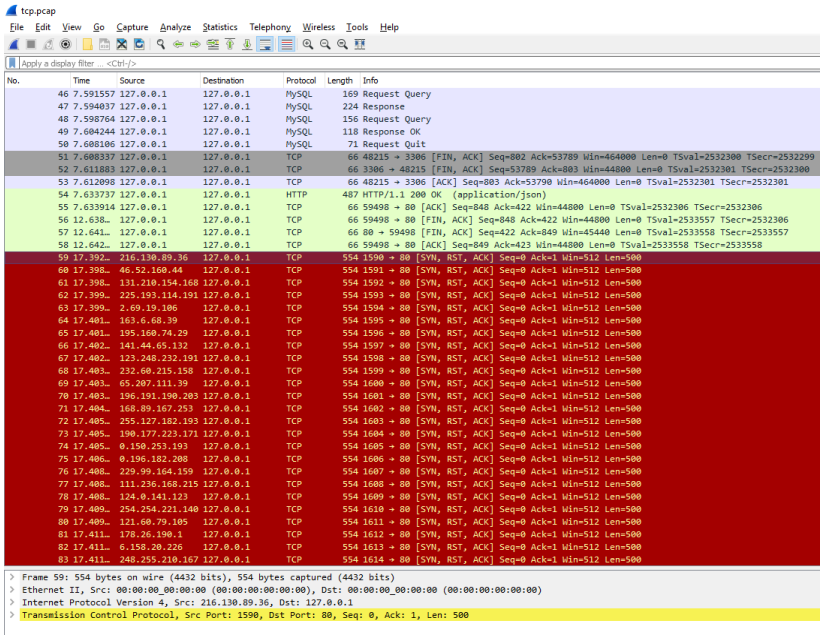


Рисунок 42 - Момент начала TCP-flood атаки

Учитывая, что TCP-пакеты сами по себе не являются вредоносными, то задержка их обнаружения очевидна.

Покажем теперь, насколько нейронной сети удалось выделить во входных данных, обучающей (training sample) и рабочей выборки (sample with TCP-flood), основной нормальный кластер и аномалии. На рисунке 43 изображены данные на выходе «узкого горла» нейронной сети в трехмерном пространстве.

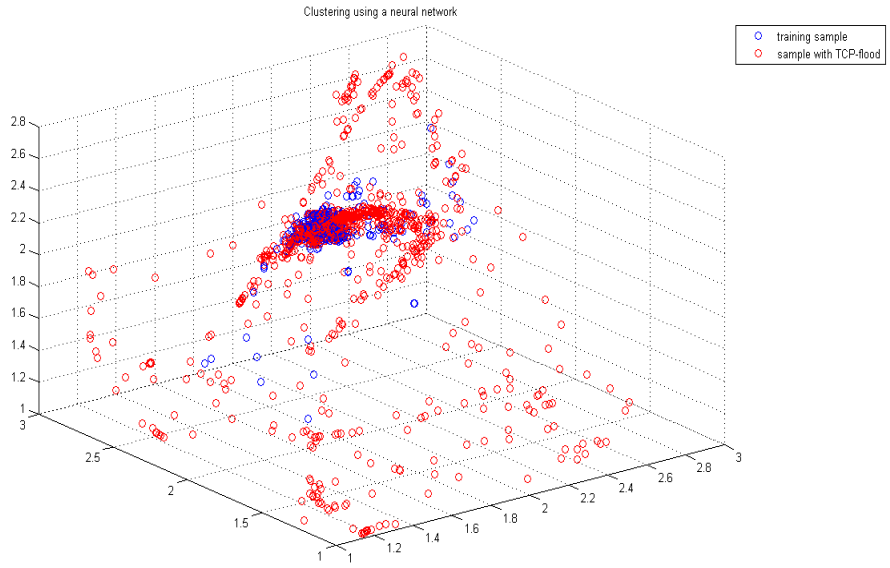


Рисунок 43 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с TCP-flood

По графику видно, что основная часть рабочей выборки локализована нейронной сетью в кластере нормальных данных обучающей выборки, а аномальные значения в значительной степени отстоят от центра кластера.

Рассмотрим, вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были построены некоторые КФ из обучающей выборки и некоторые КФ из рабочей выборки, начиная с момента обнаружения нейронной сетью аномального поведения (рисунок 44).

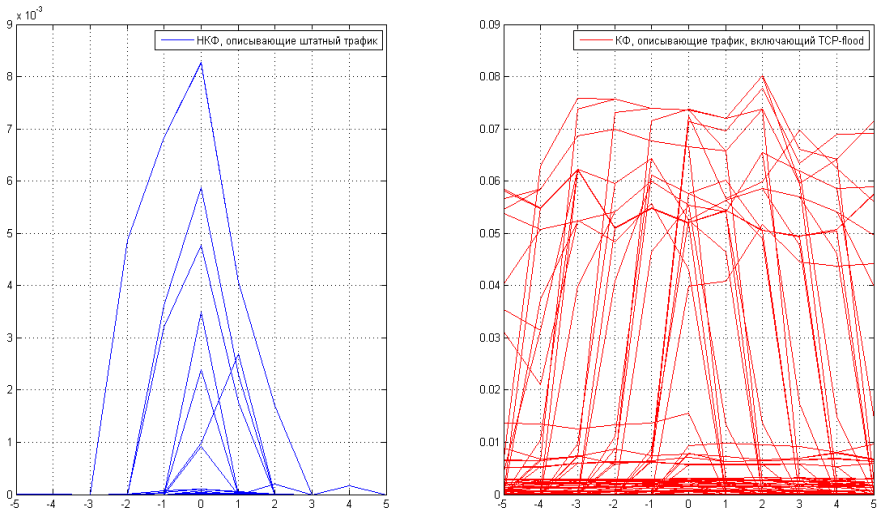


Рисунок 44 - Нормальные КФ из обучающей выборки и КФ рабочей выборки после начала SYN-flood атаки

Нейронной сети, очевидно, удалось обнаруживать аномальные КФ, соответствующие TCP-flood атаке, заметно отличающиеся по виду от нормальных КФ.

Следующие эксперименты были проведены для обнаружения UDP-flood атаки. После предъявления на вход обученной нейронной сети рабочей выборки (test sample), содержащей помимо признаков независимого нормального трафика (control traffic) признаки трафика, генерируемого при UDP-flood атаке, были получены значения ошибки реконструкции для всех элементов рабочей выборки. Для сравнения на рисунке 45 представлены значения ошибки реконструкции и для элементов обучающей выборки (training sample).

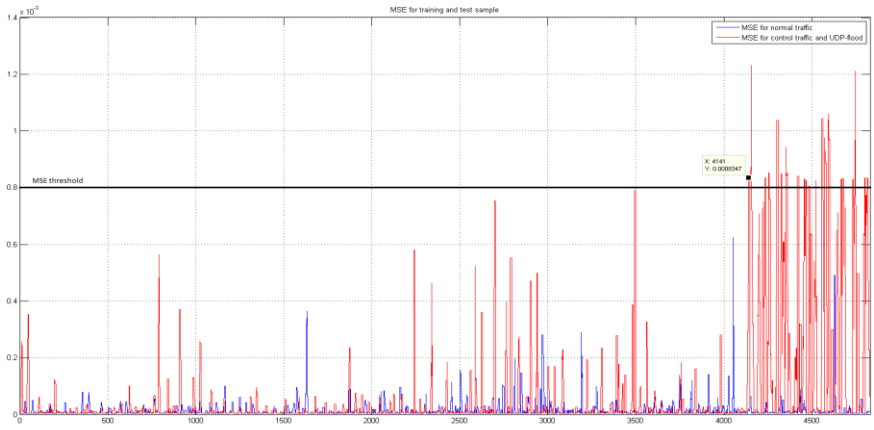


Рисунок 45 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка с признаками UDP-flood (красный цвет)

Согласно установленному в п. 3.3. пороговому значению, превышение которого означает обнаружение аномалии, на рисунке 45 установлена метка отсчета, начиная с которого деятельность определяется нейросетью как аномальная и подозрительная, учитывая продолжительное и частое превышение ошибкой реконструкции установленного порога, нейросетью может быть диагностирована атака. Ошибка реконструкции для элемента рабочей выборки с индексом 4141 составляет $8{,}347 \cdot 10^{-3}$. С помощью программы удалось установить время, в течение которого были переданы пакеты, соответствующие данному признаку рабочей выборки (рисунок 46).

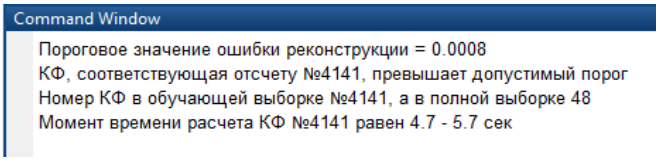


Рисунок 46 - Результат сопоставления ошибки реконструкции и времени в файле захваченного трафика

КФ с порядковым номером 4141 была рассчитана по трафику, зафиксированному с 4.7 сек. по 5.7 сек. Трафик в этот отрезок времени, действительно, представляет собой поток UDP-пакетов и изображен на рисунке 47.

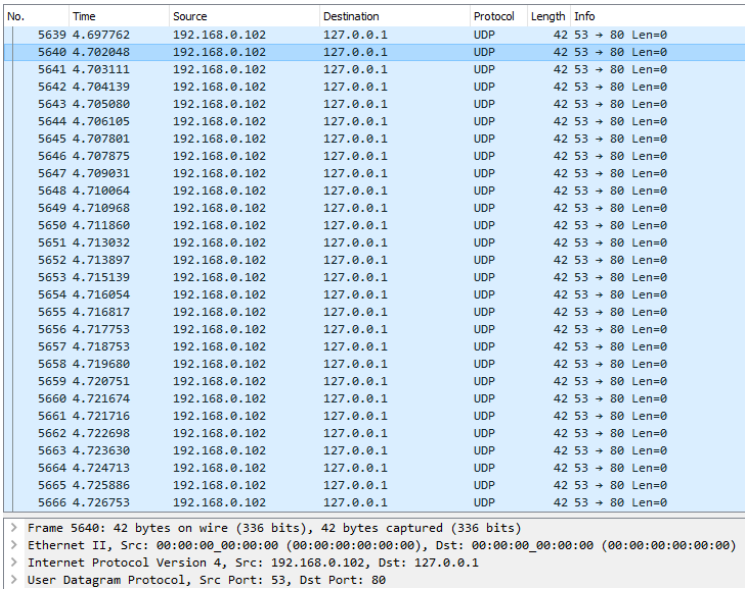


Рисунок 47 - Момент обнаружения UDP-flood атаки

Однако, по трафику, изображенному на рисунке 48, видно, что на самом деле UDP-flood атака началась с 0,0 сек.

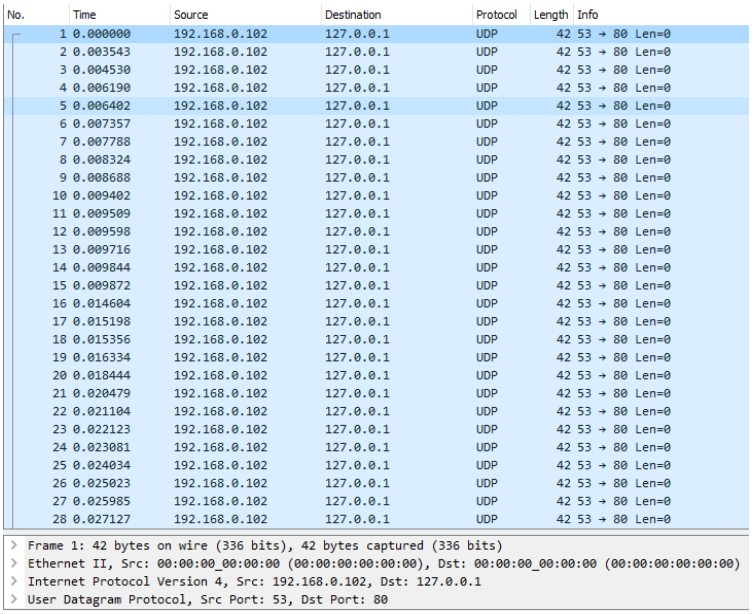


Рисунок 48 - Момент начала UDP-flood атаки

Учитывая, что UDP-пакеты сами по себе не являются вредоносными, то задержка их обнаружения очевидна.

Покажем теперь, насколько нейронной сети удалось выделить во входных данных, обучающей (training sample) и рабочей выборки (sample with

UDP-flood), основной нормальный кластер и аномалии. На рисунке 49 изображены данные на выходе «узкого горла» нейронной сети в трехмерном пространстве.

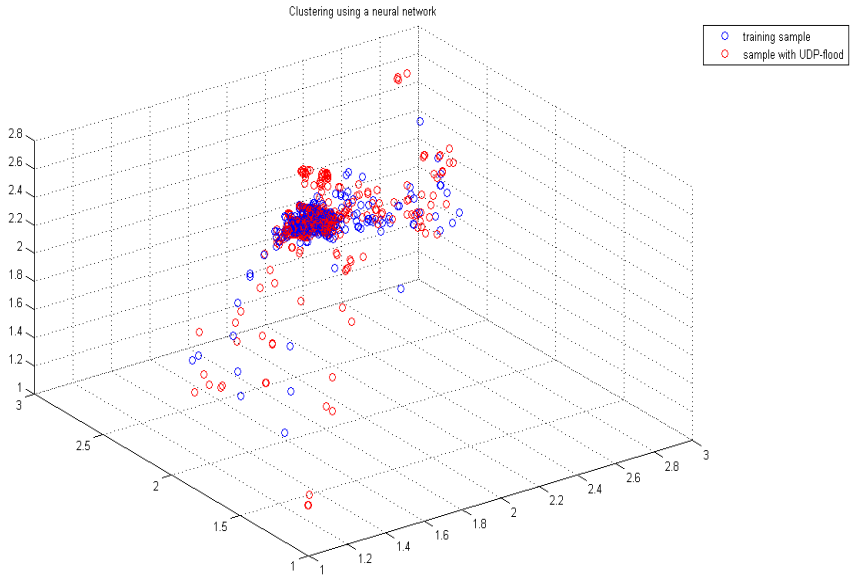


Рисунок 49 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с UDP-flood

По графику видно, что основная часть рабочей выборки локализована нейронной сетью в кластере нормальных данных обучающей выборки, а аномальные значения в значительной степени отстоят от центра кластера.

Рассмотрим, вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были построены некоторые КФ из обучающей выборки и некоторые КФ из рабочей выборки, начиная с момента обнаружения нейронной сетью аномального поведения (рисунок 50).

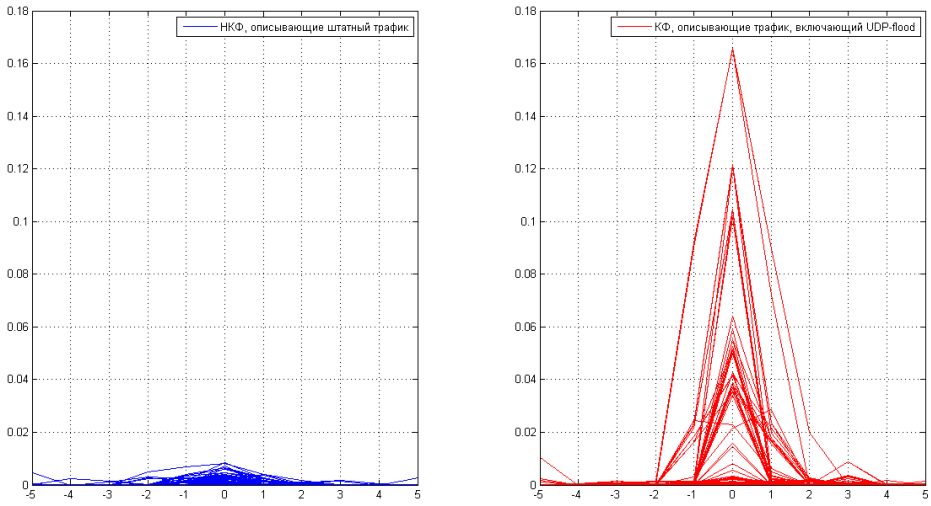


Рисунок 50 - Нормальные КФ из обучающей выборки и КФ рабочей выборки после начала UDP-flood атаки

Нейронной сети, очевидно, удалось обнаруживать аномальные КФ, соответствующие UDP-flood атаке, заметно отличающиеся по виду от нормальных КФ.

Следующие эксперименты были проведены для обнаружения ICMP-flood атаки. После предъявления на вход обученной нейронной сети рабочей выборки (test sample), содержащей помимо признаков независимого нормального трафика (control traffic) признаки трафика, генерируемого при ICMP-flood атаке, были получены значения ошибки реконструкции для всех элементов рабочей выборки. Для сравнения на рисунке 51 представлены значения ошибки реконструкции и для элементов обучающей выборки (training sample).

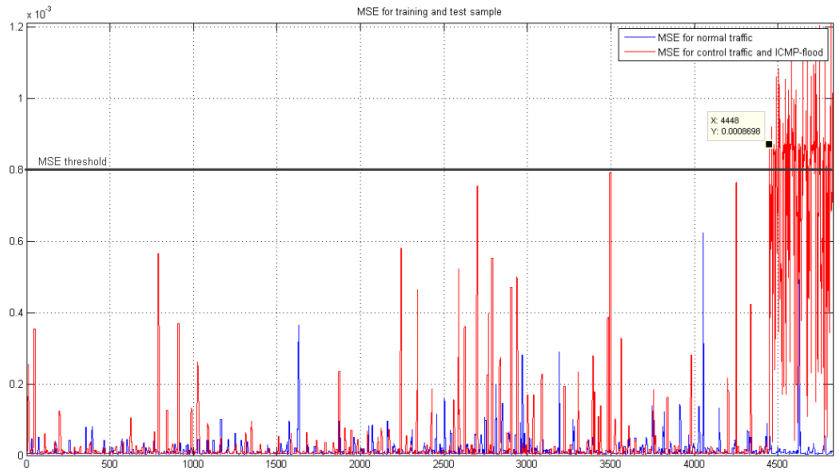


Рисунок 51 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка с признаками ICMP-flood (красный цвет)

Согласно установленному в п. 3.3. пороговому значению, превышение которого означает обнаружение аномалии, на рисунке 51 установлена метка отсчета, начиная с которого деятельность определяется нейросетью как аномальная и подозрительная, учитывая продолжительное и частое превышение ошибкой реконструкции установленного порога, нейросетью может быть диагностирована атака. Ошибка реконструкции для элемента рабочей выборки с индексом 4448 составляет $8{,}698 \cdot 10^{-3}$. С помощью программы удалось установить время, в течение которого были переданы пакеты, соответствующие данному признаку рабочей выборки (рисунок 52).

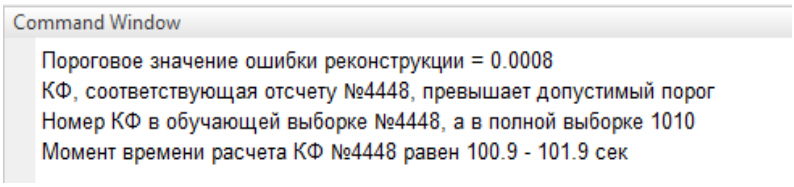


Рисунок 52 - Результат сопоставления ошибки реконструкции и времени в файле захваченного трафика

КФ с порядковым номером 4448 была рассчитана по трафику, зафиксированному с 100,9 сек. по 101.9 сек. Трафик в этот отрезок времени,

действительно, представляет собой поток ICMP-пакетов и изображен на рисунке 53.

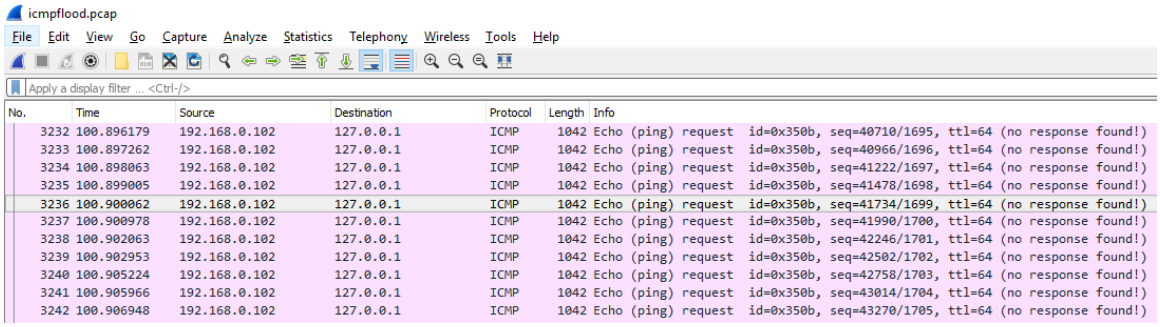


Рисунок 53 - Момент обнаружения ICMP-flood атаки

Однако, по трафику, изображенному на рисунке 54, видно, что на самом деле ICMP-flood атака началась с 99,6 сек.

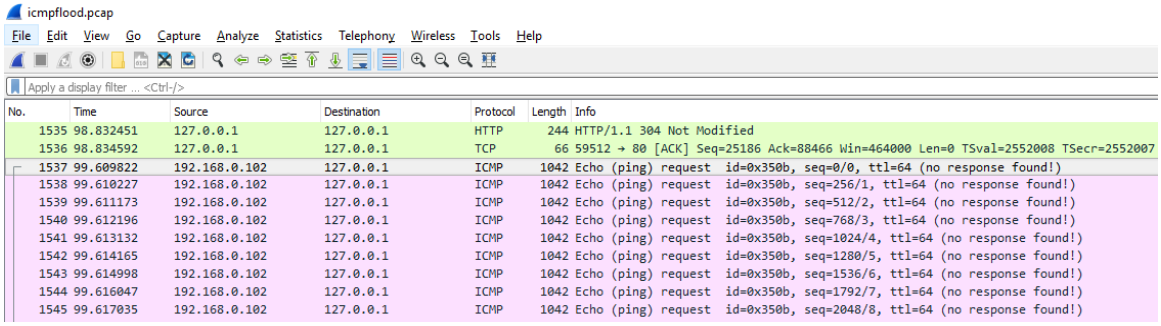


Рисунок 54 - Момент начала ICMP-flood атаки

В данном случае момент обнаружения нейронной сетью аномального поведения отстает от реального момента начала атаки на 1,3 сек.

Покажем теперь, насколько нейронной сети удалось выделить во входных данных, обучающей (training sample) и рабочей выборки (sample with ICMP-flood), основной нормальный кластер и аномалии. На рисунке 55 изображены данные на выходе «узкого горла» нейронной сети в трехмерном пространстве.

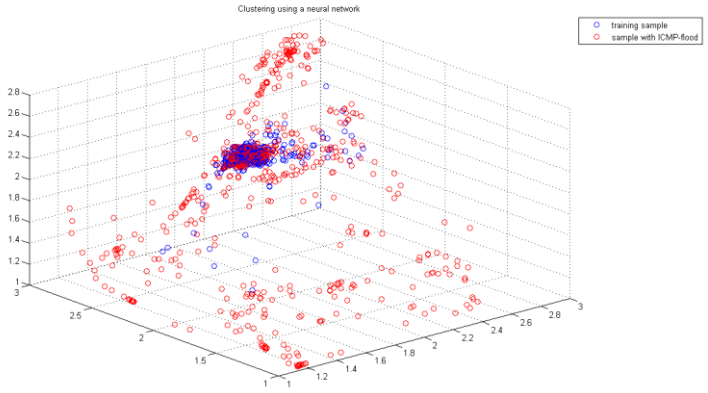


Рисунок 55 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с ICMP-flood

По графику видно, что основная часть рабочей выборки локализована нейронной сетью в кластере нормальных данных обучающей выборки, а аномальные значения в значительной степени отстоят от центра кластера.

Рассмотрим, вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были построены некоторые КФ из обучающей выборки и некоторые КФ из рабочей выборки, начиная с момента обнаружения нейронной сетью аномального поведения (рисунок 56).

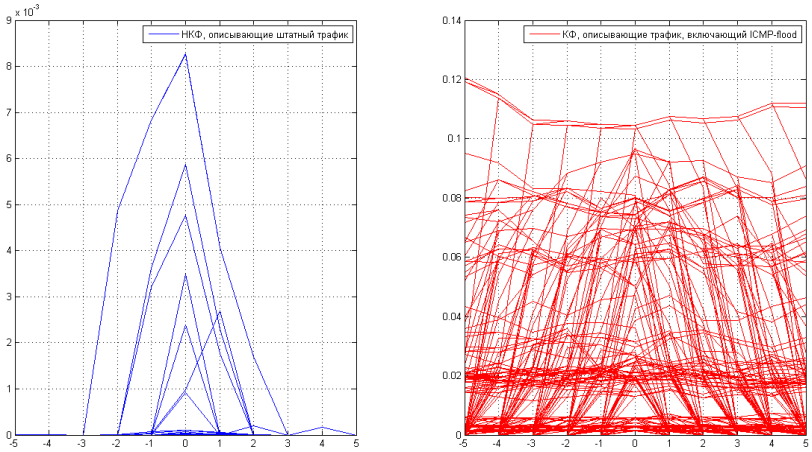


Рисунок 56 - Нормальные КФ из обучающей выборки и КФ рабочей выборки после начала ICMP-flood атаки

Нейронной сети, очевидно, удалось обнаруживать аномальные КФ, соответствующие ICMP-flood атаке, заметно отличающиеся по виду от нормальных КФ, соответствующих штатному трафику веб-сайта.

И наконец, последний эксперимент был проведен для обнаружения HTTP-flood атаки. После предъявления на вход обученной нейронной сети рабочей выборки (test sample), содержащей помимо признаков независимого нормального трафика (control traffic) признаки трафика, генерируемого при HTTP-flood атаке, были получены значения ошибки реконструкции для всех элементов рабочей выборки. Для сравнения на рисунке 57 представлены значения ошибки реконструкции и для элементов обучающей выборки (training sample).

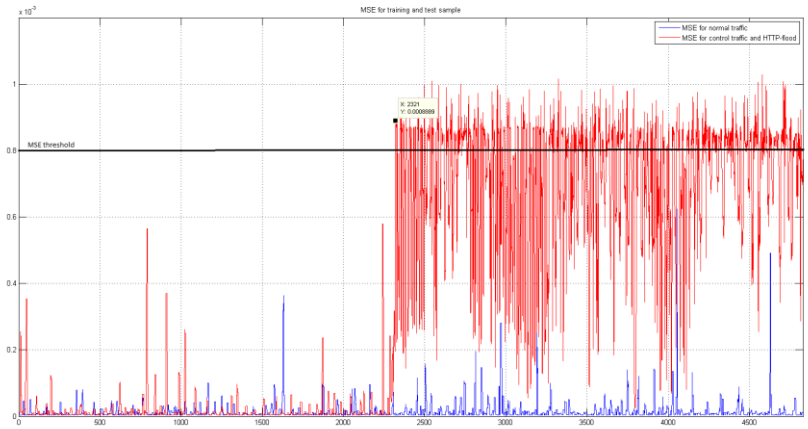


Рисунок 57 - Ошибка реконструкции, когда на входе НС обучающая выборка из нормальных КФ (НКФ) (синий цвет) и ошибка реконструкции, когда на входе НС рабочая выборка с признаками HTTP-flood (красный цвет)

Согласно установленному в п. 3.3. пороговому значению, превышение которого означает обнаружение аномалии, на рисунке 57 установлена метка отсчета, начиная с которого деятельность определяется нейросетью как аномальная и подозрительная, учитывая продолжительное и частое превышение ошибкой реконструкции установленного порога, нейросетью может быть диагностирована атака. Ошибка реконструкции для элемента рабочей выборки с индексом 2321 составляет $8{,}889 \cdot 10^{-3}$. С помощью

программы удалось установить время, в течение которого были переданы пакеты, соответствующие данному признаку рабочей выборки (рисунок 58).

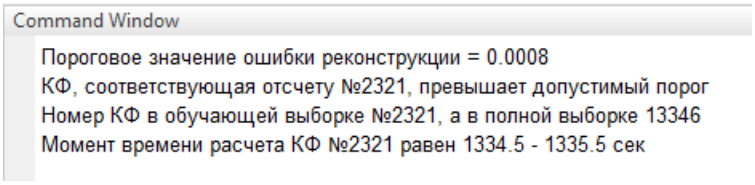


Рисунок 58 - Результат сопоставления ошибки реконструкции и времени в файле захваченного трафика

КФ с порядковым номером 2321 была рассчитана по трафику, зафиксированному с 1334,5 сек. по 1335,5 сек. Трафик в этот отрезок времени, действительно, представляет собой многократные запросы страницы http://localhost/admin/index.php?module=config-settings и изображен на рисунке 59.

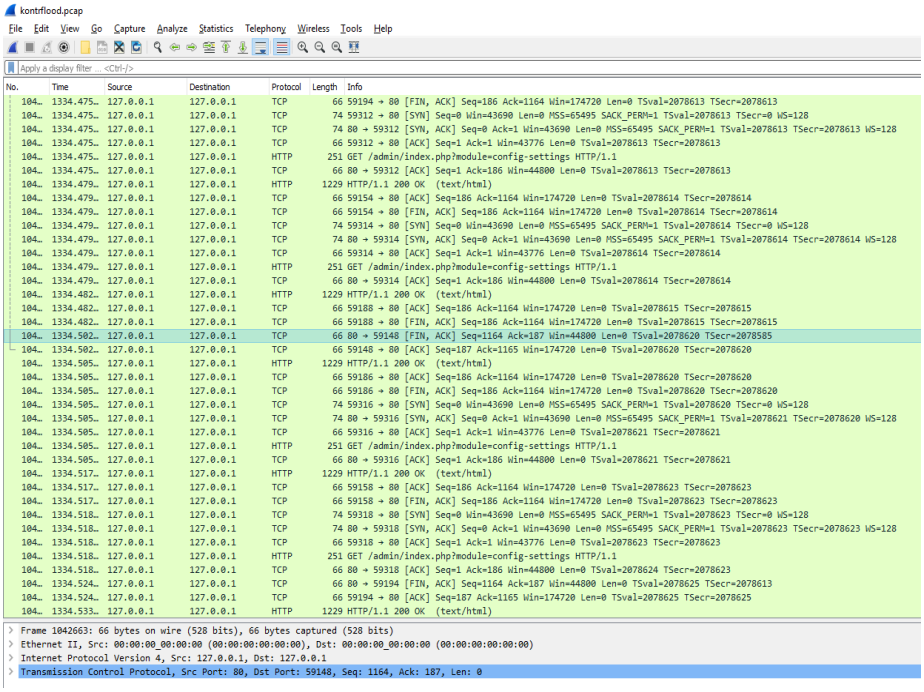


Рисунок 59 - Момент обнаружения HTTP-flood атаки

Однако, по трафику, изображенному на рисунке 60, видно, что на самом деле HTTP-flood атака началась с 1326,25 сек.

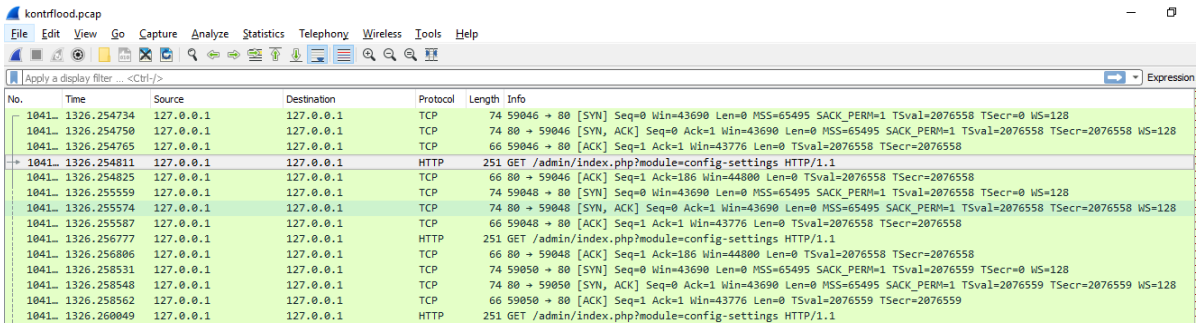


Рисунок 60 - Момент начала HTTP-flood атаки

Учитывая, что запросы страницы настроек сайта сами по себе не являются вредоносными, то задержка их обнаружения очевидна.

Покажем теперь, насколько нейронной сети удалось выделить во входных данных, обучающей (training sample) и рабочей выборки (sample with HTTP-flood), основной нормальный кластер и аномалии. На рисунке 61 изображены данные на выходе «узкого горла» нейронной сети в трехмерном пространстве.

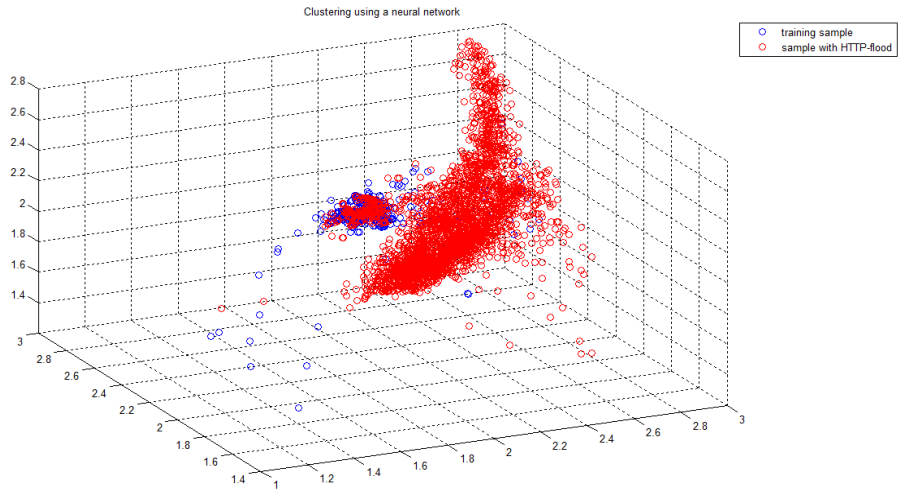


Рисунок 61 - Выходные данные «узкого горла» АНС для обучающей и рабочей выборки с HTTP-flood

По графику видно, что основная часть рабочей выборки локализована нейронной сетью в кластере нормальных данных обучающей выборки, а аномальные значения сконцентрированы в конкретной области отдельно от нормального кластера.

Рассмотрим, вид неопознанных нейронной сетью корреляционных функций и сравним с теми, ошибка реконструкции для которых мала. Для этого были построены некоторые КФ из обучающей выборки и некоторые КФ из рабочей выборки, начиная с момента обнаружения нейронной сетью аномального поведения (рисунок 62).

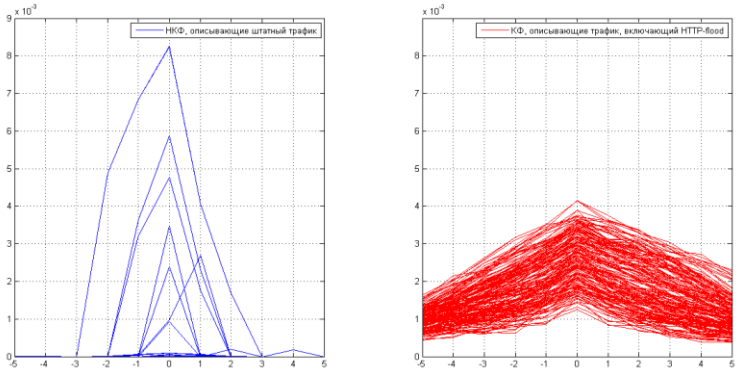


Рисунок 62 - Нормальные КФ из обучающей выборки и КФ рабочей выборки после начала HTTP-flood атаки

Нейронной сети, очевидно, удалось обнаруживать аномальные КФ, соответствующие HTTP-flood атаке, заметно отличающиеся по виду от нормальных КФ, соответствующих трафику штатной эксплуатации веб-сайта.

3.5 Анализ результатов и перспективы применения нейросетевого метода

Представленные результаты применения метода демонстрируют подтверждение предположения об отличии КФ, рассчитанных по нормальному трафику, от КФ, рассчитанных по трафику, включающему атаки. Результаты также подтверждают способность нейронной сети обучаться на примерах КФ обучающей выборки, выделять в проверочной выборке нормальный кластер – примеры КФ, похожие на те, по которым она была обучена, но не участвующие в процессе обучения. Кроме того, нейронной сети удалось распознавать нормальное поведение сетевого сервера,

диагностировать аномальное поведение и вредоносное действие сетевых атак, описанное в рабочих выборках. При этом распознанные нормальные признаки, характеризующие штатный трафик, при визуализации выхода нейросети в трехмерном пространстве, локализуются в одной области, а аномальные выбросы существенно отстоят от области нормальных признаков или концентрируются в отдельно стоящий кластер аномальных признаков (рисунок 61). Результаты проведенных экспериментов по обнаружению сетевых атак на сервис MyBB представлены в сводной таблице 5.

Таблица 5 - Результаты проверки эффективности алгоритма

Наименование атаки	Пороговое значение ошибки реконструкции	Обнаружение	Задержка обнаружения, с	Степень превышения порога	false positive
SQLIA	3×10^{-3}	3/3	0	1.03 – 1.14	1
SYN-flood	0.8×10^{-3}	да	19.2	1.01 – 1.454	0
TCP-flood	0.8×10^{-3}	да	28.31	1.08 – 1.505	0
UDP-flood	0.8×10^{-3}	да	4.7	1.03 – 1.54	0
ICMP-flood	0.8×10^{-3}	да	1.3	1.09 – 1.514	0
HTTP-flood	0.8×10^{-3}	да	8.25	1.12 – 1.285	0

По таблице видно, что нейронной сети удалось распознать все рассмотренные типы атак. Причем при проверке обнаружения трех атак SQLIA было зафиксировано только одно ложное срабатывание. При этом отсутствие задержки обнаружения SQLIA осуществлено за счёт способа расчета КФ. Расчет КФ «скользящим окном» увеличивает быстродействие обнаружения аномалий нейросетью, поскольку расчет характеристик осуществляется каждую 0,1 сек. и позволяет видеть мгновенные изменения в поведении обрабатывающего сетевого устройства на ранних этапах. Задержка в выявлении flood-атак связана с тем, что отправляемые в большом объеме пакеты не содержат явных признаков вредоносности и воспринимаются сервером и системой обнаружения аномалий как обычный трафик. Однако со временем такие атаки начинают влиять на сетевой стек, и по мере загрузки

канала связи и истощения ресурсов становится ясно, что этот трафик является частью DDoS-атаки, а не легитимными запросами. Предложенный подход к обнаружению аномалий эффективно выявляет длительное нестандартное поведение, вызванное flood-атакой, в среднем через 12,3 секунды после его начала.

В общем, подводя итоги, можно сказать, что поставленная цель – разработка и апробация нейросетевого метода обнаружения аномалий сетевого трафика для обнаружения сетевых атак – достигнута, не избегая, однако, слабых мест, присущих методам обнаружения аномалий.

Из недостатков метода, помимо характерных для поведенческих методов, можно указать его не универсальность – необходима настройка для защиты каждого конкретного сервиса или ресурса. То есть метод эффективен только в пределах того объекта, по штатному трафику которого была обучена нейронная сеть, что существенно ограничивает универсальную применимость метода (только локальная устойчивость) и не обеспечивает требований к быстроте и простоте настройки.

Представленные результаты позволяют предполагать распространить технологию для обнаружения и других типов атак, и для защиты других типов сетевых объектов, например, стремительно набирающих популярность наравне с ростом числа уязвимостей, IoT-устройств. Нейронные сети являются адаптивными и имеют сравнительно низкую вычислительную сложность. Настроенная нейронная сеть работает очень быстро и не требует значительных вычислительных ресурсов, что позволяет допустить возможность применения метода в тиражируемых устройствах IoT. Однако есть основания полагать, что алгоритм обладает значительно большим потенциалом для устройств со сложными протоколами взаимодействия. Преимущества применения метода обусловлены его простотой и экономичностью.

Итак, предложенное в работе решение может оказаться полезным не только для многочисленных сетевых ресурсов, но и в контексте широкого распространения типовой архитектуры коммуникационных сервисов и

устройств Интернета Вещей. Применение предложенного алгоритма экономично, поэтому в качестве наиболее потенциального объекта защиты могут быть рассмотрены типовые сетевые сервера, также сетевые устройства ограниченной функциональности. Примерами сетевого сервера являются веб-сайты, построенные на основе некоторого типового «движка», микросервисные архитектуры на основе Docker, а также устройства Интернета вещей, обладающие внешним сетевым интерфейсом для доступа к ним, и устройства, предназначенные для применения в системах промышленной автоматизации (M2M). Алгоритм может быть полезен для разработчиков и владельцев безопасных веб-сервисов и приложений. Ведь самая ресурсоемкая процедура по сбору данных для обучения нейросети проводится единожды и может быть организована еще на этапе тестирования веб-приложения, когда тестируются и эксплуатируются все возможности сайта. И после настройки алгоритма для обнаружения аномалий в использовании конкретного ресурса, его владелец может контролировать правомерность использования веб-приложения.

Выводы

В данной главе был выбран объект исследований – сетевой трафик при эксплуатации существующего сервиса для создания форумов в Интернете – MyBB. Подробно изучены возможности и способы проведения атак SQLIA, а также инструментарий, позволяющий в целях тестирования веб-ресурса, моделировать flood-атаки.

Была подробно представлена методика формирования обучающих, проверочных и рабочих выборок для обучения, проверки качества обучения нейросети и эффективности обнаружения нейронной сетью как нормальных, так и аномальных образцов в предъявляемых данных.

Показана методика и результаты обучения нейронной сети и проверки ее работы на независимых нормальных данных.

Проведены целевые атаки и получены рабочие выборки для апробации нейросетевого алгоритма обнаружения аномалий для выявления реальных атак SQL–injection и flood-атак на веб-сервер MyBB. Настроенная нейронная сеть продемонстрировала хорошие результаты распознавания нормальной активности в MyBB и всех проведенных атак. Однако имело место быть ложное срабатывание при обнаружении атак SQLIA и объяснимые задержки обнаружения flood-атак.

Таким образом, нейросетевой подход успешно применен для обнаружения трех атак SQLIA, пяти вариантов flood-атак на тестовый стенд сервиса MyBB.

Предложены перспективы использования разработанного нейросетевого метода обнаружения сетевых атак на сетевые сервисы.

Полученные результаты дают основания для проведения дальнейших исследований, в частности, распространения технологии для обнаружения и других типов атак на сетевые ресурсы ограниченного функционала, в том числе на веб-приложения набирающих популярность устройств Интернета вещей (IoT).

Заключение

В процессе выполнения работы были рассмотрены и классифицированы основные и актуальные в настоящее время угрозы безопасности веб-сайтов. Выявлено, что атаки типа «внедрение кода» представляют значительную опасность для веб-сайтов. Более подробно описаны и изучены атаки типа Buffer Overflow и SQL-injection, а также был приведен обширный обзор, систематизация и анализ существующих способов обнаружения этих атак, а также подходов, основанных на обнаружении аномалий.

Оказалось, что, несмотря на значительное число проведенных исследований в данной области, проблема выявления сетевых атак остается актуальной, а существующие решения все еще имеют ряд нерешенных недостатков.

В связи с этим обозначенной целью данной работы являлась разработка метода обнаружения атак на основе выявления аномалий без проведения инспекции содержимого трафика и демонстрация его применения для обнаружения реальных сетевых атак на сетевой сервис.

В настоящей работе предложен метод обнаружения атак, в котором обнаружение аномалий в поведении сетевого сервера как системы с динамическим откликом на запрос осуществляется за счет применения автоассоциативной нейронной сети, реконструирующей КФ, рассчитанные по блокам данных об интенсивности входящего и исходящего трафика и, таким образом, содержащие в себе информацию о времени и размере ответа сервера на тот или иной запрос. Предложен критерий выявления аномалий, позволяющий обнаруживать корреляционные функции, не опознанные нейронной сетью и являющиеся аномальными для данного сетевого обрабатывающего устройства, что вполне может быть выявлением признака несанкционированной деятельности на сайте.

Метод апробирован в ряде имитационных экспериментов для обнаружения различных типов сетевых атак. На сегодняшний день уязвимость

к атакам «внедрения кода» (SQL-injection), позволяющим получить несанкционированный доступ к базе данных, а в некоторых случаях к серверу веб-ресурса, и «отказа в обслуживании» является чуть ли не самой распространённой и не до конца решенной проблемой веб-сайтов. Поэтому на примере обнаружения именно этих атак на сетевой сервис была проверена и показана эффективность предложенного алгоритма.

Можно сказать, что поставленные в работе задачи и цели были достигнуты, предложенный нейросетевой метод обнаружения аномалий успешно продемонстрировал обнаружение SQL-injection атак и flood-атак на веб-форум MyBB.

Рассмотрены достоинства, недостатки и перспективы применения предложенного метода, ориентированного на работу в составе поведенческой СОВ.

В качестве будущей работы можно обозначить исследования по распространению метода для обнаружения и других типов атак, исследования возможности его использования в маломощных IoT-устройствах, а также исследования, направленные на повышение точности обнаружения и снижение числа ложных срабатываний.

Список используемой литературы и используемых источников

1. А. БРАНИЦКИЙ и И. КОТЕНКО, «КЛАССИФИКАЦИЯ МЕТОДОВ ОБНАРУЖЕНИЯ СЕТЕВЫХ АТАК» Труды СПИИРАН, т. 2(45), 2016.

2. А. Ежов и С. Шумский, «Автоассоциативные сети» 2015. – URL: http://helpiks.org/3-10930.html.

3. А. Королева, «Российские банки под DDoS-атакой» 2016. – URL: https://test.expert.ru/2016/11/11/rossijskie-banki-pod-ddos-atakoj/.

4. А. Лямин, «Сетевые атаки: банки под прицелом» 2017. – URL: http://bankir.ru/publikacii/20171012/setevye-ataki-banki-pod-pritselom-10009243/.

5. А. Олейникова, «Система обнаружения сетевых атак нового поколения» 7 Март 2018. – URL: http://www.jetinfo.ru/stati/sistema-obnaruzheniya-setevykh-atak-novogo-pokoleniya.

6. Александр И. ИССЛЕДОВАНИЕ СОБСТВЕННЫХ СЕТЕВЫХ ПРОТОКОЛОВ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ //Безопасность информационных технологий. – 2023. – Т. 30. – №. 2. – С. 80-88.

7. Безопасность и защита сайтов – URL: http://www.infosecurity.ru/iprotect/websec/classification/.

8. В. П. Шкодырев, К. И. Ягафаров, В. А. Баштовенко и Е. Э. Ильина, «Обзор методов обнаружения аномалий в потоках данных» – URL: http://ceur-ws.org/Vol-1864/paper_33.pdf.

9. В. Шитиков, Г. Розенберг и Н. Костина, «Формальная постановка задачи визуализации данных» 2005. – URL: http://www.ievbras.ru/ecostat/Kiril/Article/A16/Volgabas1/Volgabas1.htm.

10. Д. Борчев, «Как устроены дыры в безопасности: переполнение буфера» 2015. – URL: https://habr.com/post/266591/.

11. Д. Буйновский, «Применение нейронных сетей для обнаружения сетевых атак» 2015. – URL: http://stud.wiki/programming/3c0b65635a3bd69b5c43b89421316c26_0.html.

12. Динамическая система – URL: https://ru.wikipedia.org/wiki/Динамическая_система.

13. Е. Борисов, «О методах обучения многослойных нейронных сетей прямого распространения. Часть 3: Градиентные методы второго порядка» 2016. – URL: http://mechanoid.kiev.ua/neural-net-backprop3.html.

14. Еремин Е. О. Обзор открытых наборов данных для выявления атак на веб-приложения //International Journal of Open Information Technologies. – 2024. – Т. 12. – №. 3. – С. 106-113.

15. И. Бауман и Ю. Коновалов, «ОБЗОР МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ В КОНТЕКСТЕ РЕШЕНИЯ ЗАДАЧИ ОБНАРУЖЕНИЯ АНОМАЛИЙ В СЕТЕВОМ ТРАФИКЕ» «Научно-практический электронный журнал Аллея Науки» №3(19) Alley-science.ru, 2018.

16. И. Кожевникова, «Анализ методов обнаружения аномалий для обнаружения сканирования портов» Молодой ученый, № 14, pp. 31-34, 2017.

17. Искусственная нейронная сеть 2025. – URL: https://ru.wikipedia.org/wiki/%D0%98%D1%81%D0%BA%D1%83%D1%81%D1%81%D1%82%D0%B2%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D0%BD%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C#Проверка_адекватности_обучения.

18. ИТ-ГРАД, «В 2017 году зафиксировано пятикратное увеличение DDoS-атак» 2017. – URL: https://habr.com/company/it-grad/blog/330340/.

19. К. К. Моокхи и Н. Бургхате, «Обнаружение SQL инъекций и CSS атак» 2004. – URL: https://www.securitylab.ru/analytics/216344.php.

20. Колесникова, В. В. Исследование DDOS-атак и методов борьбы с ними / В. В. Колесникова, А. С. Романенко, А. С. Любухин // Информационные системы, экономика и управление: Ученые записки. – Ростов-на-Дону:

Ростовский государственный экономический университет (РИНХ), 2023. – С. 52-56. – EDN GJNBYM.

21. М. Офер и А. Шалман, «Внедрение SQL кода с завязанными глазами» 2004. – URL: http://www.securitylab.ru/analytics/216332.php.

22. НЕЙРОННЫЕ СЕТИ ПРИ ИЗУЧЕНИИ КОГНИТИВНЫХ ПРОЦЕССОВ – URL: http://ecsocman.hse.ru/data/948/676/1219/gradoselskaya_7.pdf.

23. О. Маор и А. Шалман, «Внедрение SQL кода с завязанными глазами, часть 2» 2004. – URL: https://www.securitylab.ru/analytics/216333.php.

24. О. Шелухин и Р. Судариков, «Анализ информативных признаков в задачах обнаружения аномалий трафика статистическими методами» T-Comm - Телекоммуникации и Транспорт, 2014.

25. Орехов, А. В. Автоматическое обнаружение аномалий сетевого трафика при DDoS-атаках / А. В. Орехов, А. А. Орехов // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. – 2023. – Т. 19, № 2. – С. 251-263. – DOI 10.21638/11701/spbu10.2023.210. – EDN XYNCXN.

26. Переполнение буфера – URL: http://ru-wiki.org/wiki/Переполнение_буфера.

27. Поздняк, И. С. Использование алгоритмов машинного обучения для обнаружения аномального поведения трафика / И. С. Поздняк, И. С. Макаров // Инфокоммуникационные технологии. – 2023. – Т. 21, № 3. – С. 20-27. – DOI 10.18469/ikt.2023.21.3.04. – EDN HQNGLZ.

28. Российский разработчик DPI-системы СКАТ, «Немного о типах DDoS-атак и методах защиты» 2016. – URL: https://habr.com/company/vasexperts/blog/313562/.

29. С. Осовский, «Нейронные сети для обработки информации. Подбор оптимальной архитектуры сети» 2002. – URL: http://stu.scask.ru/book_ns.php?id=33.

30. С. Осовский, Нейронные сети для обработки информации, Москва: Финансы и статистика, 2002, pp. 65-67.

31. С. Пархоменко и Т. Леденёва, «ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ МЕТОДОМ ЛЕВЕНБЕРГА-МАРКВАРДТА В УСЛОВИЯХ БОЛЬШОГО КОЛИЧЕСТВА ДАННЫХ» ВЕСТНИК ВГУ, СЕРИЯ: СИСТЕМНЫЙ АНАЛИЗ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ, 2014.

32. С. Хайкин, «Преимущества и ограничения обучения методом обратного распространения» в Нейронные сети, Москва, Вильямс, 2006, p. 304–314.

33. С. Ю. Бойков, А. В. Никитенко, Д. А. Луговой, Ю. Н. Шулева SQL-инъекции: виды и методы защиты от них // Защита информации. Инсайд. – 2024. – № 4(118). – С. 24-27. – EDN GQGJJI.

34. Семыкина, Н. А. Особенности применения методов искусственного интеллекта при решении задачи мониторинга сетевого трафика с целью обнаружения атак / Н. А. Семыкина, Н. М. Садовникова // Инженерный вестник Дона. – 2023. – № 4(100). – С. 296-302. – EDN UNZCDS.

35. ТОП-10 уязвимостей от OWASP – URL: https://pikabu.ru/story/top10_uyazvimostey_ot_owasp_4253368.

36. Тураев, С. Э. Анализ сетевого трафика (АСТ) в кибербезопасности / С. Э. Тураев, Д. А. Заколдаев // Наука и бизнес: пути развития. – 2024. – № 5(155). – С. 61-65. – EDN POJQFY.

37. Уфимский Государственный Авиационный Технический Университет, «Классификация нейронных сетей» 2014 – URL: https://studfiles.net/preview/986645/page:7/.

38. Ф. Заболотный, «Почему SQL Injection – это самый опасный вид уязвимостей?» 29 Январь 2018. – URL: https://acribia.ru/articles/why_sql_injection_is_the_most_dangerous_type_of_vulnerability.

39. Чаругин, В. В. Анализ и формирование наборов данных сетевого трафика для обнаружения компьютерных атак / В. В. Чаругин, А. Н. Чесалин

// International Journal of Open Information Technologies. – 2023. – Т. 11, № 6. – С. 100-106. – EDN HZDNHW.

40. Ю. В. Жуков, Основы веб-хакинга: нападение и защита., Спб.: Питер, 2011.

41. A. One, «Smashing the Stack for Fun and Profit» – URL: http://insecure.org/stf/smashstack.html.

42. A. S. Choudhary, «CIDT: Detection of Malicious Code Injection Attacks on Web Application» International Journal of Computer Applications (0975 – 8887) Volume 52– No.2, 2012.

43. B. Maroš и I. Homoliak, «Detection of network buffer overflow attacks: A case study».

44. B. McCorkendale и P. Ször, «Code red buffer overflow» Virus Bulletin, pp. 4-5, September 2001.

45. Black Diver, «DoS и DDoS атаки на Web сервера. Стресс-тестирование» 2015. – URL: https://blackdiver.net/it/security/3903.

46. Buffer Overflows – URL: https://www.owasp.org/index.php/Buffer_Overflows.

47. C. Basta, A. elfatatry и S. Darwish, «Detection of SQL Injection Using a Genetic Fuzzy Classifier System» (IJACSA) International Journal of Advanced Computer Science and Applications, т. 7, № 6, 2016.

48. C. C. J. Johansen, S. Beattie и P. Wagle, «Pointguard: Protecting pointers from buffer overflow vulnerabilities» 2003.

49. C. Kruegel и G. Vigna, «Anomaly detection of web-based attacks» в Proceedings of the 10th ACM conference on Computer and communication security, 2003.

50. C. Kruegel, T. Toth и E. Kirda, «Service specific anomaly detection for network intrusion detection» в In Proceedings of the 2002 ACM symposium on Applied computing (SAC), 2002.

51. Cesar Cerrudo, «Управление Microsoft SQL Server используя SQL инъекции» 2005. – URL: https://www.securitylab.ru/analytics/216396.php.

52. Cowan, C. Pu, D. Maier, M. Hinton, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle и Q. Zhang, «Stackguard: Automatic adaptive detection and prevention of buffer-overflow attacks» 1998.

53. Curesec Research Team, «MyBB 1.8.6: SQL Injection» 2016. – URL: https://www.curesec.com/blog/article/blog/MyBB-186-SQL-Injection-159.html.

54. D. Jonathan Day, Z. Zhao и M. Ma, «Detecting Return-to-libc Buffer Overflow Attacks Using Network Intrusion Detection Systems» 2010.

55. DoS и DDoS атаки – URL: http://wiki.vspu.ru/users/goodbendz/%D1%81%D1%80%D1%811.

56. F. S. Rietta, «Application Layer Intrusion Detection for SQL Injection» – URL: https://rietta.com/papers/rietta_acmse2006.pdf.

57. hping3, Флудим по своему сайту – URL: https://allwebstuff.info/hping3-флудим-по-своему-сайту/.

58. I. Lee, S. Jeong, S. Yeo и J. Moon, «A novel method for SQL injection attack detection based on removing SQL query attribute values» Mathematical and Computer Modelling, т. 55, № 1-2, pp. 56-58, 2012.

59. ITI Capital, «Топ-10 data mining-алгоритмов простым языком» 2015. – URL: https://habr.com/company/iticapital/blog/262155/.

60. J. Mazel, «Unsupervised network anomaly detection» – URL: https://tel.archives-ouvertes.fr/tel-00667654/document.

61. J. P. Singh, «Analysis of SQL Injection Detection Techniques» – URL: https://arxiv.org/ftp/arxiv/papers/1605/1605.02796.pdf.

62. K. R. Muthu и B. Sujitha.T.V, «Intrusion Detection System in Web Services» International Journal of Science and Research (IJSR), т. 2, № 2, 2013.

63. K. S. Provos, K. Srinivas и Fabia, «ShellOS: Enabling fast detection and forensic analysis of code injection attacks» 2011.

64. Keeper, «Buffer Overflows and IDS Basics» Декабрь 2012. – URL: https://www.hackthis.co.uk/articles/buffer-overflows-and-ids-basics.

65. Kumar A., Dutta S., Pranav P. Analysis of SQL injection attacks in the cloud and in WEB applications //Security and Privacy. – 2024. – Т. 7. – №. 3. – С. e370.

66. M. A. Kathwate и D. K. Chitre, «Code-injection Buffer Overflow Attack Blocker» International Journal of Electronics Communication and Computer Engineering, т. 3, № 4.

67. M. Cobb, «SQL injection detection tools and prevention strategies» – URL: https://www.computerweekly.com/tip/SQL-injection-detection-tools-and-prevention-strategies.

68. M. Conover и w00w00 Security, «w00w00 on Heap Overflow» 1999– URL: http://www.opennet.ru/base/usersoft/17.txt.html.

69. M. E. Donaldson, «INSIDE THE BUFFER OVERFLOW ATTACK: MECHANISM, METHOD, & PREVENTION» GSEC Version 1.3, Апрель 2002. – URL: https://www.sans.org/reading-room/whitepapers/securecode/buffer-overflow-attack-mechanism-method-prevention-386.

70. M. Kiani, A. Clark и G. Mohay, «Evaluation of Anomaly Based Character Distribution» в The Third International Conference on Availability, Reliability and Security, 2008.

71. M. Mimoso, «Новая разновидность ботнета WireX может проводить атаки UDP-flood» 2017. – URL: https://threatpost.ru/wirex-variant-capable-of-udp-flood-attacks/22256/.

72. M. Polychronakis, «Generic Detection of Code Injection Attacks using Network-level Emulation» 2009. – URL: https://www3.cs.stonybrook.edu/~mikepo/papers/mikepo_thesis.pdf.

73. M. V. Mahoney, «Network traffic anomaly detection based on packet bytes» в In Proceedings of the 2002 ACM symposium on Applied computing (SAC), 2003.

74. N. Moradpoor Sheykhkanloo, «A Pattern Recognition Neural Network Model for Detection and Classification of SQL Injection Attacks» International Journal of Computer and Information Engineering, т. 9, № 6, 2015.

75. Nasereddin M. et al. A systematic review of detection and prevention techniques of SQL injection attacks //Information Security Journal: A Global Perspective. – 2023. – Т. 32. – №. 4. – С. 252-265.

76. Pentestit, «OWASP Top 10 2017 RC» 2017. – URL: https://habr.com/company/pentestit/blog/326272/.

77. ptsecurity, «Исследование кибератак 2017 года: 47% атак направлены на инфраструктуру компаний» – URL: https://habr.com/company/pt/blog/351228/.

78. ptsecurity, «Чем защищают сайты, или зачем нужен WAF?» – URL: https://habr.com/company/pt/blog/269165/.

79. Qrator Labs, «Отчет за 2017 год компании Qrator Labs» Москва, 2018.

80. R. Auger, «Buffer Overflow» – URL: http://projects.webappsec.org/w/page/13246916/Buffer%20Overflow.

81. R. Garde, D. R. Anekar, M. Ghadge и N. Kulkarni, «A System to Detect and Block SQL Injection with the help of Multi Agent System using Artificial Neural Network» International Journal of Computer Applications, т. 71, № 12, 2013.

82. S. Andersson, A. Clark и G. Mohay, «Detecting Network-based Obfuscated Code Injection Attacks Using Sandboxing» 2005.

83. S. Andersson, A. Clark и G. Mohay, «Network-Based Buffer Overflow Detection by Exploit Code Analysis».

84. S. Andersson, A. J. Clark, G. M. Mohay, B. Schatz и J. Zimmermann, «A framework for detecting network-based code injection attacks targeting Windows and UNIX» в In Proceedings of the 21st Annual Computer Security Applications Conference, Tucson, Arizona, 2005.

85. S. Ding, H. B. Kuan Tan и H. Zhang, «ABOR: An Automatic Framework for Buffer Overflow Removal in C/C++Programs» в ICEIS 2014: Enterprise Information Systems, 2014.

86. S. Ezzat Salama, M. I. Marie, L. M. El-Fangary и Y. K. Helmy, «Web Anomaly Misuse Intrusion Detection Framework for SQL Injection Detection»

International Journal of Advanced Computer Science and Applications (IJACSA), т. 3, № 3, p. 123, 2012.

87. S. Manmadhan и T. Manesh, «A METHOD OF DETECTING SQL INJECTION ATTACK TO SECURE WEB APPLICATIONS» International Journal of Distributed and Parallel Systems (IJDPS), т. 3, № 6, 2012.

88. S. Son, «Diglossia: Detecting Code Injection Attacks with Precision and Efficiency» Berlin, Germany, 2013.

89. S. Swamynathan и V. Shanmughaneethi, «Detection of SQL Injection Attack in Web Applications using Web Services» IOSR Journal of Computer Engineering (IOSRJCE), т. 1, № 5, pp. 13-20, 2012.

90. SophosLabs, Sophos Plc, «SQL-инъекция» Декабрь 2007. – URL: https://yandex.ru/support/webmaster/protecting-sites/sql-injection.html.

91. SQL Slammer – URL: http://ru-wiki.org/wiki/SQL_Slammer.

92. SQL инъекция и ORACLE 2003. – URL: https://www.securitylab.ru/analytics/216253.php.

93. T. Barabosch, N. Bergmann, A. Dombeck и E. Padilla, «Quincy: Detecting Host-Based Code Injection Attacks in Memory Dumps» в DIMVA 2017: Detection of Intrusions and Malware, and Vulnerability Assessment, 2017.

94. T. Barabosch, S. Eschweiler и E. Gerh, «Bee Master: Detecting Host-Based Code Injection Attacks».

95. T. Coen, «MyBB 1.8.6 SQL Injection» 2016. – URL: https://vulners.com/packetstorm/PACKETSTORM:138744.

96. T. Ye, L. Zhang, L. Wang и X. Li, «An Empirical Study on Detecting and Fixing Buffer Overflow Bugs» в IEEE International Conference on Software Testing, Verification and Validation, 2016.

97. UDP-Flood» 2014. – URL: http://netwild.ru/udp-flood/.

98. Web Application Security Consortium, «WASC THREAT CLASSIFICATION» – URL: http://projects.webappsec.org/f/WASC-TC-v2_0.pdf.

99. X. Wang, C.-C. Pan, P. Liu и S. Zhu, «SigFree: A Signature-Free Buffer Overflow Attack Blocker» IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, т. 5, № 4, 2008.

100. X. Xiao, Y. Ruibo, Y. Runguo, L. Qing, P. Sancheng и J. Yong, «Detection and Prevention of Code Injection Attacks on HTML5-Based Apps» в Advanced Cloud and Big Data, 2015 Third International Conference on, 2015.

101. Y. Ruibo, X. Xiao, G. Hu, S. Peng и Y. Jiang, «New deep learning method to detect code injection attacks on hybrid applications» Journal of Systems and Software, т. 137, pp. 67-77, 2018.

Приложение А

Программа для получения рабочих выборок для обнаружения TCP, UDP, ICMP-flood

```
a = load('tcpin_0.1.dat', ' ');
b = load('tcpout_0.1.dat', ' ');
in = a(:,2);
out = b(:,2);
lag=5;       % correlation base
gap=10;   % 1 second
% Number of points in the traffic measurement series
m=length(in)-gap+1;

% Plot general correlation for the whole in/out series
%subplot(1,3,1);
G_cor=xcorr(out, in, lag);
%plot([-lag:1:lag], G_cor, 'r');
%title('Обобощенная корреляционная функция по обучающей выборке', 'FontWeight' ,'bold');

% Amplitude at zero delay
maxG=G_cor(lag+1);
% Plot several correlation functions for every 3-hours of in/out series
% Normalized to the general correlation
%subplot(1,3,2);
for i=1:m
    n1=(i-1)+1;
    n2=n1+gap-1;
    F_cor=xcorr(out(n1:n2), in(n1:n2), lag)/maxG;
  %  plot([-lag:1:lag], F_cor, 'b');
    %hold on;

 %Получение файла с КФ
id=fopen('TCP_flood.txt', 'a');
fprintf(id, '%1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f\n', F_cor);
 fclose(id);
end
clear;
a = load('udpin_0.1+.dat', ' ');
b = load('udpout_0.1+.dat', ' ');

in = a(:,2);
out = b(:,2);

lag=5;       % correlation base
gap=10;   % 1 second

% Number of points in the traffic measurement series
m=length(in)-gap+1;
% Plot general correlation for the whole in/out series
%subplot(1,3,1);
G_cor=xcorr(out, in, lag);
%plot([-lag:1:lag], G_cor, 'r');
%title('Обобощенная корреляционная функция по обучающей выборке', 'FontWeight' ,'bold');
% Amplitude at zero delay
maxG=G_cor(lag+1);
```

```
% Plot several correlation functions for every 3-hours of in/out series
% Normalized to the general correlation
%subplot(1,3,2);
for i=1:m
   n1=(i-1)+1;
   n2=n1+gap-1;
   F_cor=xcorr(out(n1:n2), in(n1:n2), lag)/maxG;
 %  plot([-lag:1:lag], F_cor, 'b');
   %hold on;

 %Получение файла с КФ
id=fopen('UDP_flood.txt', 'a');
fprintf(id, '%1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f\n', F_cor);
 fclose(id);
end
clear;
a = load('icmpin_0.1.dat', ' ');
b = load('icmpout_0.1.dat', ' ');

in = a(:,2);
out = b(:,2);

lag=5;       % correlation base
gap=10;   % 1 second

% Number of points in the traffic measurement series
m=length(in)-gap+1;
% Plot general correlation for the whole in/out series
%subplot(1,3,1);
G_cor=xcorr(out, in, lag);
%plot([-lag:1:lag], G_cor, 'r');
%title('Обобощенная корреляционная функция по обучающей выборке', 'FontWeight' ,'bold');

% Amplitude at zero delay
maxG=G_cor(lag+1);
% Plot several correlation functions for every 3-hours of in/out series
% Normalized to the general correlation
%subplot(1,3,2);
for i=1:m
   n1=(i-1)+1;
   n2=n1+gap-1;
   F_cor=xcorr(out(n1:n2), in(n1:n2), lag)/maxG;
 %  plot([-lag:1:lag], F_cor, 'b');
   %hold on;

 %Получение файла с КФ
id=fopen('ICMP_flood.txt', 'a');
fprintf(id, '%1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f %1.6f\n', F_cor);
 fclose(id);

end
```

Приложение Б

Программа для сопоставления момента времени обнаружения TCP-flood атаки по индексу ошибки реконструкции, превышающей пороговое значение

```
q=find(mse_tcp>=0.0008);
for i = 1:1
 kf=KVtcp(:,q(i));
 kf=kf';
 tcp=load('TCP_flood.txt');
  for j=1:11
eval(['n' num2str(j) '=find(tcp(:,j)==kf(j))']);
clc;
  end
 fprintf('Пороговое значение ошибки реконструкции = 0.0008\n')
 fprintf('КФ, соответствующая отсчету №%.0f, превышает допустимый порог\n', q(i))
  n={n1 n2 n3 n4 n5 n6 n7 n8 n9 n10 n11};
  %поиск максимальной длины столбца
  maxLength=max(cellfun(@(x)numel(x),n));
  %дополнение нулями и формирование одной матрицы
  out=cell2mat(cellfun(@(x)cat(1,x,zeros(maxLength-length(x),1)),n,'UniformOutput',false));
  %поиск одинаковых индексов, встречающихся во всех столбцах
  unqA = unique(out);
  indexx = unqA(all(any(bsxfun(@eq,out,permute(unqA,[2,3,1])),1),2));
  fprintf('Номер КФ в обучающей выборке №%.0f, а в полной выборке %.0f\n', q(i), indexx(1))
  fprintf('Момент времени расчета КФ №%.0f равен %.1f - %.1f сек\n', q(i), (indexx(1)-
1)/10,((indexx(1)-1)/10)+1)
  fprintf(' \n')
end
```