МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования

«Гродненский государственный университет имени Янки Купалы»

Факультет математики и информатики

Кафедра системного программирования и компьютерной безопасности

ШАПОРОВ БОРИС ИЛЬИЧ

**Разработка системы для обнаружения sql-иньекций методами машинного обучения**

Дипломная работа

студента 4 курса специальности

1-98 01 01-01 «Компьютерная безопасность

(математические методы и программные системы)»

дневной формы получения образования

|  |  |
| --- | --- |
| «Допустить к защите»  Заведующий кафедрой  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Кадан А.М.  \_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ 2022 г. | Научный руководитель  Косарева Екатерина Владимировна  Доцент кафедры системного программирования и компьютерной безопасности |

Гродно 2022

**РЕЗЮМЕ**

**Дипломная работа:** XX с., XX рис., XX листинга, XX источников.

ВЕБ АТАКИ, SQL, RFI, CSRF, SQL INJECTION, ПРОЕКТИРОВАНИЕ, ВЕБ-ПРИЛОЖЕНИЕ, ОБХОД СИСТЕМЫ, LSTM RNN SEQ2SEQ LEARNING, ОБУЧЕНИЕ С ПОДКРЕПЛЕНИЕМ, Q-ОБУЧЕНИЕ, PYTHON, JYTHON, PROCESSBUILDER API, JSR-223 SCRIPTING ENGINE, ИНФОРМАЦИОННАЯ СИСТЕМА, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ.

**Объект исследования:** приложение подверженное популярным атакам и модель машинного обучения для их выявления.

**Предмет исследования:** вредоносные SQL запросы при атаках на веб-приложения посредством внедрения их в код.

**Цель работы:** обучить модель машинного обучения, с применением sequence to sequence модели обучения, реализовать веб-приложение, для демонстрации использования SQL команд, с целью получения паролей всех пользователей. Внедрить полученную модель в тестовую среду для предотвращения вредоносных запросов.

**Методы исследования:** анализ, обобщение, проектирование приложения, метод и модель алгоритма рекуррентной нейронной сети.

**Исследования и разработки:** проведен анализ распространенных вредоносных запросов. Реализована тестовая среда. Проведено практическое использование уязвимостей в приложении на основе SQL инъекции. Реализована модель обучение seq-2-seq и внедрена в слабозащищённую среду.

**Область возможного практического применения:** веб-сервисы, обладающие высокими требованиями к безопасности системы. Обучение с использованием рекуррентной нейронной сети используют там, где задачей стоит не анализ данных, а выживание в реальной среде.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc101102190)

[1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ 5](#_Toc101102191)

[1.1. Примеры и виды распространенных запросов 5](#_Toc101102192)

[1.2. Постановка задачи и анализ предметной области 6](#_Toc101102193)

[1.3 Внедрение SQL операторов 8](#_Toc101102194)

[2. АНАЛИЗ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ 12](#_Toc101102195)

[2.1 Сверточные нейронные сети 12](#_Toc101102196)

[2.2 Сети с долговременной краткосрочной памятью (LSTM) 15](#_Toc101102197)

[2.3 Архитектура LSTM 19](#_Toc101102198)

[3. ПРИМЕНЕНИЕ LSTM ДЛЯ РАСПОЗНАВАНИЯ SQLI 21](#_Toc101102199)

[3.1 Данные для обучения 21](#_Toc101102200)

[3.2 Процесс обучения 22](#_Toc101102201)

[ЗАКЛЮЧЕНИЕ 25](#_Toc101102202)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 26](#_Toc101102203)

# ВВЕДЕНИЕ

Веб-атаки – это атаки, которые используют исключительно протокол HTTP / HTTPS. По данным Kaspersky Security Network было в 2022 году существенно (на 19 процентных пунктов по сравнению с 2021 годом) снизилась доля веб-приложений, содержащих уязвимости высокого уровня риска. Число уязвимостей, которое в среднем приходится на одно приложение, снизилось по сравнению с прошлым годом почти в полтора раза. Но, несмотря на это, общий уровень защищенности веб-приложений по-прежнему оценивается как низкий. 82% всех выявленных уязвимостей оказались обусловлены ошибками в коде.

На сегодняшний день 90% веб-приложений подвержены угрозе атак на клиентов. Как и в предыдущие годы, существенную роль в этом играет XSS и SQL-инъекция. Примерами атак на пользователей могут быть заражение компьютеров вредоносным ПО, фишинговые атаки для получения учетных или других важных данных, а также выполнение действий от имени пользователя с помощью обманной техники кликджекинг, в частности для накрутки лайков и просмотров.

В 2015 году команда GhostShell заявила, что взломала множество веб-сайтов с помощью атак с использованием SQL-инъекций, и раскрыл тысячи скомпрометированных данных учетной записи, включая электронные письма, имена пользователей, адреса, номера телефонов,

и другой конфиденциальной информации.

Важной причиной несовершенства веб-приложений является то, что многим программистам не хватает знаний о безопасном программировании, поэтому они оставляют приложения с недостатками, недоработками или банальными ошибками в самой логике.

Одной из самых популярных целей атак являются веб-серверы и веб-приложения. Пользователи веб-серверов и веб-приложений запрашивают и отправляют информацию с помощью запросов.

Объект исследования – приложения подверженные популярным атакам и модель машинного обучения для их выявления.

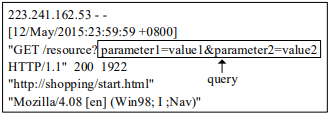
Предмет исследования – вредоносные SQL запросы при атаках на веб-приложения посредством внедрения их в код.

Цель работы – реализовать веб-приложение, для демонстрации использования SQL команд, с целью получения паролей всех пользователей. Рассмотреть существующую модель, основанную на алгоритме обучения с подкреплением.

# 1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

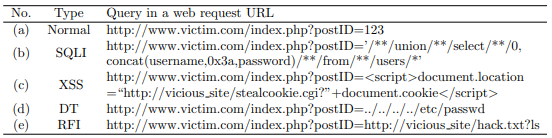
## Примеры и виды распространенных запросов

Запросы содержат большую часть пользовательского ввода и поэтому, атак, связанных с внедрением кода, становится все больше [1]. Запросы могут быть найденными в LOG журналах. На рис. 1.1 показана запись журнала веб-сервера в Common Log Format (CLF), состоящий из различных элементов, таких как исходный IP-адрес, временная отметка, код состояния ответа сервера, поле заголовка пользовательского агента и фактическая строка веб-запроса. Текст в рамке в строке запроса – это запрос, который рассматривается в данной работе. Запрос обозначается знаком "?" символ после указанного ресурса и перечисляет пары имен и значений параметров. Вредоносные запросы при атаках, посредством внедрения кода и обычные запросы, различаются по распределению и последовательности символов [5]. На рисунке 1.2 сравнивается обычный запрос и четыре типа атак внедрения кода: SQLI, XSS, DT и RFI.



**Рисунок. 1.1** – Запись в LOG журнале доступа в Веб-сервере

Примечание – Источник: [5].



**Рисунок 1.2** – Сравнение обычных и вредоносных запросов

Примечание – Источник: [5].

Запросы на рисунке 1.2 перенаправляются к уязвимому скрипту с именем «index.php» на веб-сайте, и скрипт может извлекать сообщения из базы данных и показать сообщение пользователю. «PostID» – это имя параметра для этого скрипта и принимает целочисленное входное значение в качестве ID сообщения при записи транзакции. Как показано на рисунке 1.2, (a) «postID = 123» – обычный запрос, а (b) - (e) – вредоносные запросы.

Запрос (b) – это типичная SQLI-атака, которая пытается получить конфиденциальные данные из закрытой таблицы.

Запрос (c) проводит XSS-атаку для получения файлов cookie пользователя с помощью Скрипт Common Gateway Interface (CGI) на хосте атакующего.

Запрос (d) использует эксплойт DT (Directory Traversal) для обнаружения файла «passwd» с использованием строки «../../../». Поскольку «passwd» является локальным файлом на уязвимом сервере (d) также называется атакой с подключением локального файла (Local file inclusion (LFI)).

Запрос (e) пытается загрузить файл «hack.txt» с удаленного вредоносного сервера и выполните команду ls в bashshell в случае атаки RFI. Также нужно отметить такие атаки как подделка межсайтовых запросов CSRF и отказ в обслуживании DDoS [11].

## Постановка задачи и анализ предметной области

Веб-приложения довольно привлекательные цели для кибератак. Например, SQL-инъекция, межсайтовый скриптинг (XSS) и удаленное выполнение кода являются самыми распространенными атаками, которые могут отключить веб-службы, украсть конфиденциальную информацию пользователей и привести к значительным финансовым потерям как для поставщиков услуг, так и для пользователей. Защитить веб-приложения от атак сложно. Несмотря на то, что программисты и исследователи разработали множество контрмер (таких как межсетевые экраны, системы обнаружения вторжений (IDS) и передовые методы защитного программирования) для защиты веб-приложений, веб-атаки остаются серьезной угрозой.

Исследователи обнаружили, что более половины веб-приложений во время сканирования 2019–2020 гг. содержали значительные уязвимости безопасности, такие как XSS или SQL Injection [11]. Более того, хакерские атаки обходятся средней американской фирме в 15.4 млн долларов в год. В результате взлома данных Equifax в 2019 году (использовавшего уязвимость в Apache Struts) была обнаружена конфиденциальная личная информация более 143 млн американских потребителей. Хотя уязвимость была обнаружена и исправлена в марте 2019 года, Equifax не предпринимала никаких действий около четырех месяцев спустя, что привело к предполагаемому ущербу в размере более 125 млн долларов [11].

Обычные системы обнаружения вторжения не работают так хорошо, как ожидалось, по ряду причин:

* **Ограниченность рабочей силы**. Для развертывания этих систем веб-разработчикам и операторам сетей необходимы глубокие знания в области веб-безопасности. Часто требуется опытный эксперт по безопасности, чтобы определить, какие функции необходимо извлечь из сетевых пакетов, двоичных файлов или других входных данных для систем обнаружения вторжений. Однако из-за большого спроса и относительно низкого барьера для входа в профессию программного обеспечения многим разработчикам не хватает необходимых знаний о методах безопасного кодирования.
* **Ограниченность классификации**. Многие системы обнаружения вторжений полагаются на стратегии, основанные на правилах или контролируемые алгоритмы машинного обучения, чтобы отличать обычные запросы от запросов атаки, что требует больших объемов данных для обучения машин. Однако получить эти данные для произвольных пользовательских приложений сложно и дорого. Кроме того, такого вида данные часто и сильно не сбалансированы, поскольку запросы на атаку для пользовательских систем труднее получить, чем обычные запросы, что создает проблемы для классификаторов. Более того, хотя подходы к обучению на основе правил или с учителем могут различать существующие известные атаки, постоянно появляются новые типы атак и уязвимостей, поэтому они могут быть неправильно классифицированы.
* **Ложноположительные ограничения**. Существуют алгоритмы, обеспечивающие приемлемую производительность, но вызывают слишком высокие показатели ложных срабатываний, которые на практике слишком высоки, например, увеличение количества ложных срабатываний на 1% может привести к тому, что система обнаружения вторжений будет неправильно отмечать тысячи законных пользователей. Поэтому очень важно уменьшить количество ложных срабатываний этих систем.

Помимо всех ограничений техники, персонала или самой системы существуют также некоторые барьеры, возникающие при разработке [14]:

* Проблема 1. *Атаки могут иметь существенно разные характеристики*. Различные типы веб-атак, такие как SQL-инъекции, межсайтовые сценарии, удаленное выполнение кода и уязвимости включения файлов, используют разные формы вектора атаки и используют различные уязвимости внутри веб-приложений. Поэтому эти атаки часто имеют совершенно разные характеристики. Например, внедрение SQL нацелено на базы данных, тогда как удаленное выполнение кода нацелено на файловые системы. Однако обычные системы обнаружения вторжений часто предназначены для обнаружения только одного типа атак. Например, анализ на основе грамматики, который работает с обнаружением SQL-инъекций, не будет работать с XSS.
* Проблема 2. *Мониторинг может иметь значительные потери в производительности*. Подходы статического анализа, которые анализируют исходный код и ищут потенциальные недостатки, имеют различные недостатки, включая уязвимость для неизвестных атак и необходимость доступа к исходному коду. Альтернативой является применение динамического анализа с помощью инструментальных приложений, что неизменно вызывает накладные расходы на мониторинг.
* Проблема 3. *Сбор данных для обучения маркированной атаке.* Системы обнаружения вторжений на основе машинного обучения полагаются на помеченные обучающие данные, чтобы узнать, что следует считать нормальным и ненормальным поведением. Сбор этих обучающих данных может быть трудным и дорогостоящим в крупномасштабных производственных веб-приложениях, поскольку маркировка данных требует значительных человеческих усилий и трудно охватить все возможные случаи. Например, обычные данные обучения запросов могут быть созданы с помощью инструментов нагрузочного тестирования, веб-сканеров или модульных тестов.
* Проблема 4: *Разработка систем обнаружения вторжений, не требующих от пользователей обширных знаний в области веб-безопасности.* Традиционные системы обнаружения вторжений применяют подход, основанный на правилах, когда пользователи должны иметь специфические знания в области веб-безопасности. Таким образом, необходимы опытные специалисты по безопасности, чтобы определить, какие функции необходимо извлечь из сетевых пакетов, двоичных файлов или других входных данных для систем обнаружения вторжений. Этот процесс выбора функций может быть утомительным, подверженным ошибкам и длительным, так что даже опытные инженеры часто полагаются на повторяющиеся процессы проб и ошибок. Более того, даже специалисты по веб-безопасности могут изо всех сил пытаться угнаться за последними уязвимостями из-за быстрых циклов обновления технологий и постоянного выпуска новых инструментов и пакетов.

## Внедрение SQL операторов

На сегодняшний день SQL-инъекция больше не является новой концепцией, однако все еще остается одним из наиболее популярных видом атак на веб-приложения. SQL-инъекция – это метод, который пользуется уязвимостями в поисковых запросах получения данных небезопасных веб-приложений. Для, краткости, далее будем называть SQL-инъекцию – SQLi. SQLi организована путем отправки вредоносных команд общения с базой данных на серверы веб-приложений, которые, в свою очередь, отправляют их в саму БД на получение ответа пользователю. Надо понимать, что любые входные данные сайта организаций (теги, файлы cookie, строки запроса и т.д.) могут быть использованы для отправки вредоносного кода.

Существуют распространенные типы ошибок SQLi:

* *Неправильно обращение*. Ошибки внедрения SQL такого типа обычно возникают из-за того, что программист или пользователь неясно определяет ввод данных или не выполняет этап проверки и фильтрации типа входных данных. Это может произойти, когда числовое поле используется в запросе SQL, но у программиста отсутствует проверка ввода для проверки типа данных, которые пользователь вводит как число.
* *Ошибка конфигурации СУБД на сервере*. Иногда уязвимости могут существовать в программном обеспечении базы данных сервера, как в случае с функцией mysql\_real\_ escape\_string () серверов MySQL. Это позволит злоумышленнику выполнить успешную атаку SQLi на основе необычных символов Юникода, даже когда ввод завершается.
* *Изменение значения условия запроса*. Этот тип ошибки позволяет злоумышленнику изменить значение условия в запросе, что искажает отображение приложения, содержащего эту ошибку.
* *Время запаздывания.* Этот тип ошибки внедрения SQL существует, когда время обработки одного или нескольких запросов SQL зависит от введенных логических данных или процесс обработки запросов механизма SQL занимает много времени.

Разработчику стоит помнить обо всех нюансах, и знать где, и в каком месте кода злоумышленник может провести SQLi, и заранее обезопасить себя и приложения используя, например, Prepare Statement для исправления ошибок, тогда данные от пользователя не будут выполняться в запросе.

Наиболее часто используемой операцией является извлечение данных с помощью команды SELECT, где запрос должен содержать имя таблицы, из которой выбираются данные, и имена столбцов, указывающие значения, которые должны быть выбраны. Ответ SQL представляет собой таблицу, содержащую выбранные столбцы и строки, удовлетворяющие критерию выбора, выраженному в предложении WHERE (например, значение столбца меньше, чем заданное значение или значение столбца, соответствующее регулярному выражению). Простым примером запроса, в котором из таблицы выбираются два столбца с условием по значению третьего столбца (листинг 1.1).

1. **SELECT** username, email **FROM** users **WHERE** **password** = 123;
3. **SELECT** u.username, ur.role **FROM** users **as** u JOIN user\_role **as** ur **on**
4. ur.user\_id = u.id **WHERE** email = example@mail.ru;

**Листинг 1.1** – Пример запросов с условием

Примечание – Источник: собственная разработка

SQLi происходит, когда сценарий на стороне сервера имеет неправильно обработанные входные данные, которые прямо или косвенно вставляются в SQL-запрос на стороне сервера. Из-за неправильной проверки, злоумышленник имеет полный или частичный контроль над запросом. В самом простом случае он может изменить вычисление выражения в предложении WHERE запроса, экранируя входную переменную и добавляя в запрос дополнительные команды (листинг 1.2).

1. **SELECT** username, email **FROM** users **WHERE** **password** = 123 OR 1 = 1;

**Листинг 1.2** – Использование условия ветвления для обхода проверки первого поля

Примечание – Источник: собственная разработка

В более сложных запросах злоумышленник может добавить оператор UNION и создать два запроса одновременно. В этих случаях злоумышленник должен найти или угадать количество и тип выбранных столбцов во втором запросе. В целом процесс использования уязвимости SQLi можно разложить на следующие шесть этапов:

1. Поиск уязвимого входного параметра. Веб-сайт может принимать несколько параметров с различными методами и различными переменными сеанса. Злоумышленник должен найти входной параметр, который вставляется в SQL-запрос сценарием с отсутствующей или неправильной проверкой ввода.
2. Определение типа уязвимого входного параметра. Злоумышленник должен выйти из исходного поля ввода запроса. Например, если сценарий помещает входной параметр между кавычками, то пользователь также должен использовать кавычку, чтобы выйти из нее; если исходный запрос был, например, **SELECT** Column1 **FROM** Table1 **WHERE** Column2 = 'input1'

тогда пользовательский ввод также должен начинаться с кавычки:

1’ OR ‘1’ = ‘1. Закрывающая кавычка будет добавлена уже на стороне сервера, как готовый SQL-запрос. Т.к. используется булевское сравнение, то 1 = 1 всегда будет истинно, следовательно, не смотря на неправильность первого условия, ответ от SQL будет положительным.

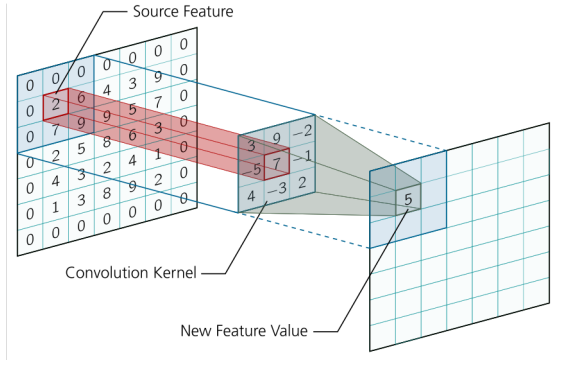
1. Продолжение SQL-запроса без синтаксической ошибки. Выход за пределы установленного сервером запроса предоставляет злоумышленнику возможность продолжить его. Синтаксис SQL должен соблюдаться с учетом возможных ограничений. Например, для выхода из строки требуется вставить новую строку, открывающую кавычку в конце ввода. Распространенный трюк заключается в использовании знака комментария в конце ввода, чтобы сделать недействительной остальную часть SQL-запроса в скрипте.
2. Получение ответа SQL в HTTP формат. После отправки своего SQL-запроса злоумышленник получает ответ через веб-сайт. Несмотря на то, что механизм SQL отвечает таблицей, этот необработанный вывод вряд ли будет виден. Сгенерированный HTTP-ответ с телом HTML, доставленным злоумышленнику, представляет собой функцию обработки неизвестного SQL-запроса кодом на стороне сервера. Иногда злоумышленник может увидеть одно или несколько полей из ответа.
3. Получение характеристик базы данных. Чтобы вставить значимые запросы в исходные данные, необходимо раскрыть имена таблиц или столбцов. Это может потребовать предварительного выбора значений из таблицы схемы информации. Если использовать подход UNION SELECT, нужно понимать какие типы данных выдают те или иные столбцы, чтобы можно было сопоставить информацию из второй таблицы в исходную.
4. Получение конфиденциальной информации. При получении необходимой части исходного запроса (тип ввода, структуру запроса) и всю информацию о БД (имена баз данных, имена таблиц, имена столбцов), остается только вывести все необходимые данные в таблицу.

# 2. АНАЛИЗ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ

## Сверточные нейронные сети

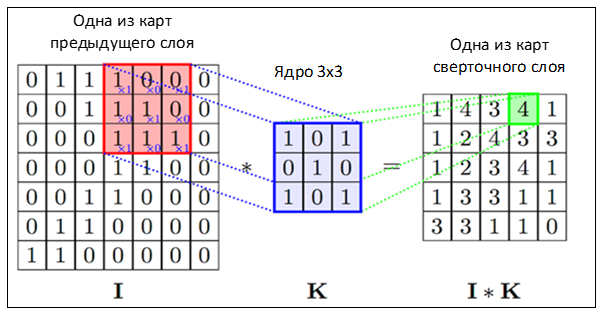
Сверточные нейронные сети (CNN) — это многоуровневые модели с прямой связью, которые предназначены для захвата пространственных характеристик. CNN обычно используются для таких приложений, как классификация изображений и обнаружение объектов, где требуется идентификация визуальной особенности. CNN состоит в основном из сверточных слоев, которые используются для цветовой субдискретизации (снижение цветового разрешения, при котором частота выборки цветоразностных сигналов может быть меньше частоты выборки яркостного сигнала) входных данных и обнаружения различных функций. Каждый сверточный слой выполняет операцию двумерной свертки между входными активациями (например, срез изображения, субдискретизированные функции) и параметрами (весами для этого слоя), где параметры состоят из одного или нескольких ядер (малых матрицы, которые применяют операцию к небольшой части ввода, например, повышение резкости среза изображения); и передает результат через нелинейную функцию активации (например, ReLU, sigmoid, tanh) для создания выходных активаций, обнаруженных функций. В конце модели полносвязные слои объединяют элементы, созданные из разных сверточных слоев, для выполнения окончательной классификации, но также может включать полносвязные, глубинные и точечные слои. CNN обычно принимает некоторый пространственно-ориентированный ввод (например, изображение, видео) и возвращает классификацию.

Сверточная нейронная сеть основана на математической операции свертки (линейное преобразование).



**Рисунок 2.1** **–** Двумерная свертка (2D convolution)

Примечание – Источник: собственная разработка



**Рисунок 2.2** – Операция свертки и получение значений сверточной карты

Примечание – Источник: собственная разработка

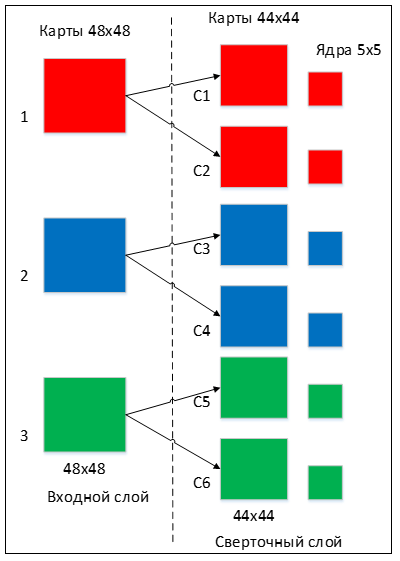
Ядро представляет из себя матрицу весов (weight matrix). Ядро “скользит” над двумерным изображением, поэлементно выполняя операцию умножения с той частью входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель. Ядро повторяет эту процедуру с каждой локацией, над которой оно “скользит”, преобразуя двумерную матрицу в другую все еще двумерную матрицу признаков. Признаки на выходе являются взвешенными суммами (где веса являются значениями самого ядра) признаков на входе, расположенных примерно в том же месте, что и выходной пиксель на входном слое. Размер ядра сверточной нейронной сети определяет количество признаков, которые будут объединены для получения нового признака на выходе.

Входные данные представляют из себя цветные изображения типа JPEG, размера 48х48 пикселей. Если размер будет слишком велик, то вычислительная сложность повысится, соответственно ограничения на скорость ответа будут нарушены, определение размера в данной задаче решается методом подбора. Если выбрать размер слишком маленький, то сеть не сможет выявить ключевые признаки лиц. Каждое изображение разбивается на 3 канала: красный, синий, зеленый. Таким образом получается 3 изображения размера 48х48 пикселей.

**Входной слой** учитывает двумерную топологию изображений и состоит из нескольких карт (матриц), карта может быть одна, в том случае, если изображение представлено в оттенках серого, иначе их 3, где каждая карта соответствует изображению с конкретным каналом (красным, синим и зеленым).

**Сверточный слой** представляет из себя набор карт (другое название – карты признаков, в обиходе это обычные матрицы), у каждой карты есть синаптическое ядро (сканирующее ядро или фильтр).

Количество карт определяется требованиями к задаче, если взять большое количество карт, то повысится качество распознавания, но увеличится вычислительная сложность. В большинстве случаев предлагается брать соотношение один к двум, то есть каждая карта предыдущего слоя (например, у первого сверточного слоя, предыдущим является входной) связана с двумя картами сверточного слоя, в соответствии с рисунком 3. Количество карт – 6.



**Рисунок 2.3** – Организация связей между картами сверточного слоя и предыдущего

Примечание – Источник: собственная разработка

**Подвыборочный слой** также, как и сверточный имеет карты, но их количество совпадает с предыдущим (сверточным) слоем, их 6. Цель слоя – уменьшение размерности карт предыдущего слоя. Если на предыдущей операции свертки уже были выявлены некоторые признаки, то для дальнейшей обработки настолько подробное изображение уже не нужно, и оно уплотняется до менее подробного. К тому же фильтрация уже ненужных деталей помогает не переобучаться.

В процессе сканирования ядром подвыборочного слоя (фильтром) карты предыдущего слоя, сканирующее ядро не пересекается в отличие от сверточного слоя. Обычно, каждая карта имеет ядро размером 2x2, что позволяет уменьшить предыдущие карты сверточного слоя в 2 раза. Вся карта признаков разделяется на ячейки 2х2 элемента, из которых выбираются максимальные по значению.

Последний из типов слоев это слой обычного **многослойного персептрона**. Цель слоя – классификация, моделирует сложную нелинейную функцию, оптимизируя которую, улучшается качество распознавания.

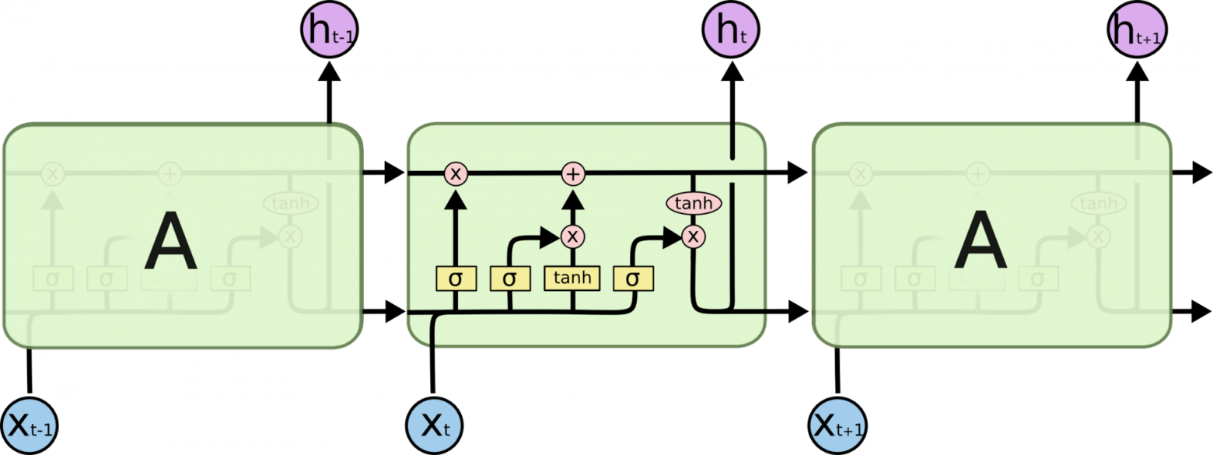
Нейроны каждой карты предыдущего подвыборочного слоя связаны с одним нейроном скрытого слоя. Таким образом число нейронов скрытого слоя равно числу карт подвыборочного слоя, но связи могут быть не обязательно такими, например, только часть нейронов какой-либо из карт подвыборочного слоя быть связана с первым нейроном скрытого слоя, а оставшаяся часть со вторым, либо все нейроны первой карты связаны с нейронами 1 и 2 скрытого слоя.

**Выходной слой** связан со всеми нейронами предыдущего слоя. Количество нейронов соответствует количеству распознаваемых классов, то есть 2 – лицо и не лицо. Но для уменьшения количества связей и вычислений для бинарного случая можно использовать один нейрон и при использовании в качестве функции активации гиперболический тангенс, выход нейрона со значением -1 означает принадлежность к классу “не лица”, напротив выход нейрона со значением 1 – означает принадлежность к классу лиц.

## Сети с долговременной краткосрочной памятью (LSTM)

Сети с долговременной краткосрочной памятью (LSTM) представляют собой многоуровневые модели с повторяющимися соединениями (т.e. данные из одной итерации повторно используются в последующей итерации того же уровня), которые эффективны при классификации последовательностей данных и прогнозирование будущих последовательностей. LSTM используются для таких приложений, как прогнозирование трафика, прогнозирование текстовых ответов и распознавание рукописного ввода.

Термин «долговременная кратковременная память» исходит из того, что простые рекуррентные нейронные сети имеют долговременную память в виде весов. Веса медленно меняются во время обучения, кодируя общие знания о данных. У них также есть кратковременная память в виде эфемерных активаций, которые передаются от каждого узла к последующим узлам. Модель LSTM вводит промежуточный тип хранения через ячейку памяти. Ячейка памяти ‑ это составная единица, построенная из более простых узлов в определенной схеме подключения, с новым включением мультипликативных узлов.

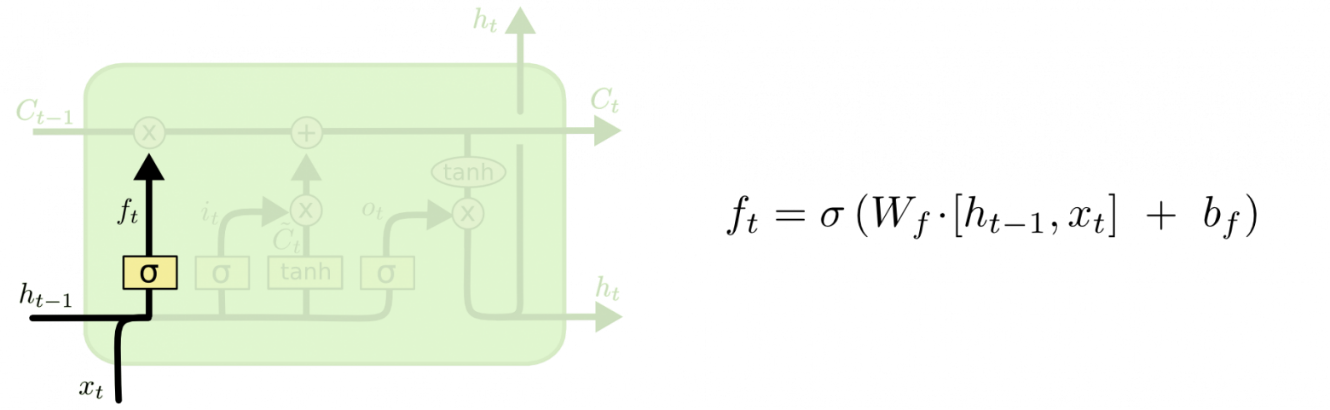


**Рисунок 2.4** – Модель LSTM

Примечание – Источник: собственная разработка

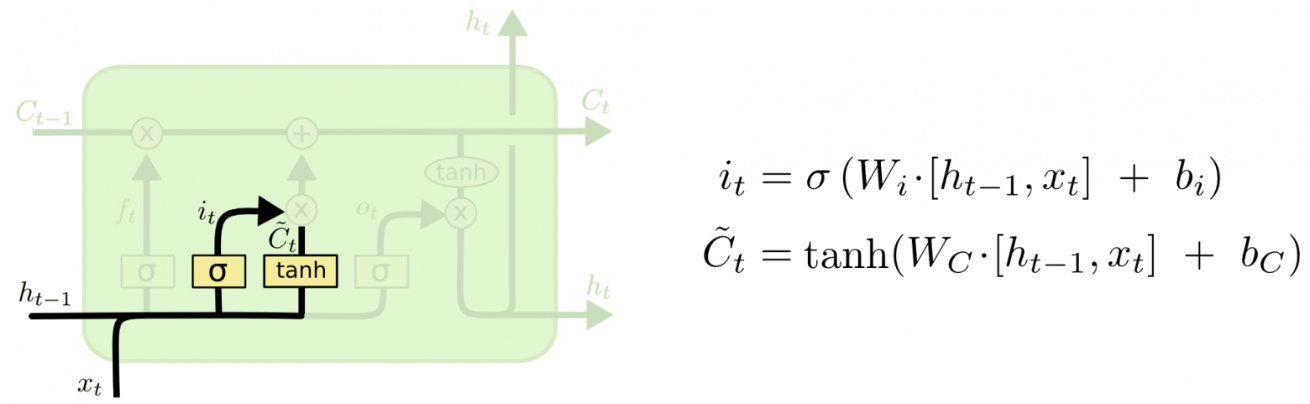
Сеть LSTM состоит из нескольких уровней, каждый из которых включает несколько ячеек. В каждой ячейке (рис. 2.4) есть три фильтра: input gate, forget gate, output gate.

1. Forget gate. Необходимо определить, какую информацию можно выбросить из состояния ячейки. Это решение принимает сигмоидальный слой, называемый “слоем фильтра забывания” (рис. 2.5). Он смотрит на ht–1 и xt и возвращает число от 0 до 1 для каждого числа из состояния ячейки Ct–1.1 означает “полностью сохранить”, а 0 – “полностью выбросить”.
2. Input gate. Основная задача заключается в том, что необходимо решить, какая новая информация будет храниться в состоянии ячейки. Этот этап состоит из двух частей. Сначала сигмоидальный слой под названием “слой входного фильтра” (input layer gate) определяет, какие значения следует обновить. Затем tanh-слой строит вектор новых значений которые можно добавить в состояние ячейки (рис. 2.6). Gate – это сигмоидальная единица, которая, как и входной узел, принимает активацию из текущей точки данных xt, а также из скрытого слоя на предыдущем временном шаге.
3. Output gate. Выходной фильтр отвечает за информацию, получаемую по итогу. Выходные данные будут основаны на состоянии ячейки, к ним будут применены некоторые фильтры. Сначала применяется сигмоидальный слой, который решает, какую информацию из состояния ячейки будет выводить. Затем значения состояния ячейки проходят через tanh-слой, чтобы получить на выходе значения из диапазона от -1 до 1, и перемножаются с выходными значениями сигмоидального слоя, что позволяет выводить только требуемую информацию (рис. 2.7).



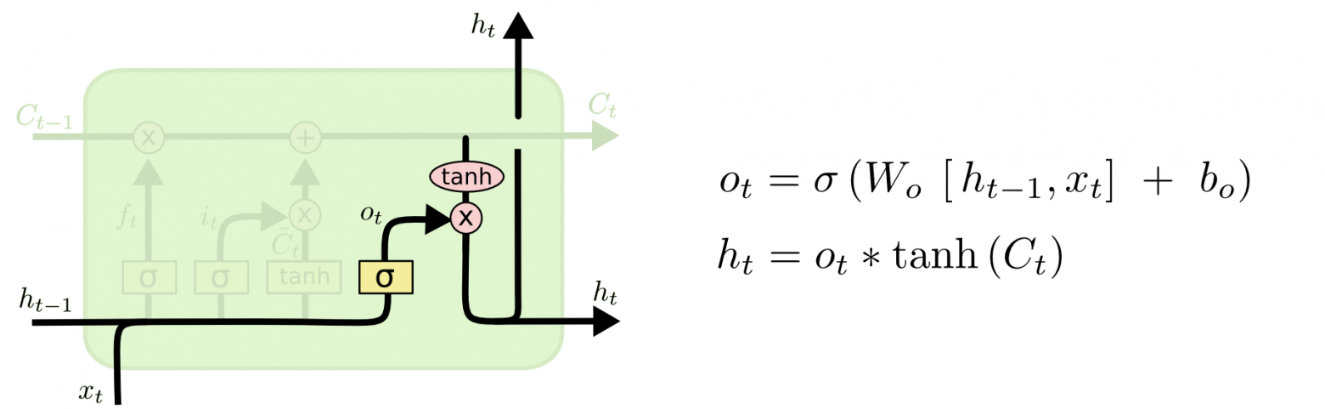
**Рисунок 2.5** – Модель фильтра забывания (Forget gate)

Примечание – Источник: собственная разработка



**Рисунок 2.6** – Модель входного фильтра (Input gate)

Примечание – Источник: собственная разработка



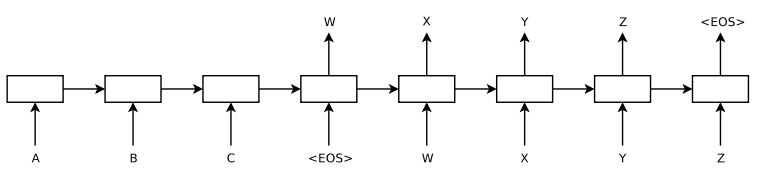
**Рисунок 2.7** – Модель выходного фильтра (Output gate)

Примечание – Источник: собственная разработка

Цель LSTM – оценить условную вероятность p(y1, . . . , yT′ |x1, . . . , xT), где (x1, ..., xT) – входная последовательность, а (y1, ..., yT′) – соответствующая выходная последовательность, длина T′ которой может отличаться от T. LSTM вычисляет эту условную вероятность, сначала получая фиксированное размерное представление *v* входной последовательности (x1, ..., xT), заданное последним скрытым состоянием LSTM, а затем вычисляя вероятность (y1, ..., yT′) со стандартной формулировкой LSTM-LM, начальное скрытое состояние которой установлено в представление *v* из x1, ..., xT формула (2.1)

(2.1)

В уравнении (2.1) каждое распределение представлено с softmax (преобразование каждого элемента коллекции, путем вычисления экспоненты каждого элемента, деленную на сумму экспонент всех элементов) по всем словам в словаре. Обязательным требованием, является наличие в конце специального символа «<EOS>», обозначающего конец строки (end - of - string), который позволяет модели определять распределение по последовательностям всех возможных длин. Общая схема представлена на рисунке 2.8, где вычисляется представление «A», «B», «C», «<EOS>», а затем использует это представление для вычисления вероятности «W», «X», «Y», «Z», «<EOS>»



**Рисунок 2.8** – Чтение строки ‘ABC’ выдавая ‘WXYZ’

Примечание – Источник: собственная разработка

Актуальные модели отличаются от приведенного выше описания по трем важным аспектам. Во-первых, использовали два разных LSTM: один для входной последовательности, а другой для выходной последовательности, потому что это увеличивает параметры числовой модели с незначительными вычислительными затратами и делает естественным обучение LSTM одновременно на нескольких языковых парах. Во-вторых, глубокие LSTM значительно превосходят мелкие LSTM, поэтому был выбран LSTM с четырьмя уровнями. В-третьих, необходимо изменить порядок слов во входном предложении. Так, например, вместо отображения предложения *a, b, c* на предложение α, β, γ, LSTM просят отобразить *c, b, a* на α, β, γ, где α, β, γ - перевод из *a, b, c*. Таким образом, *a* находится в непосредственной близости от α, *b* довольно близко к β и так далее, что позволяет легко «устанавливать связь» между входом и выходом. Таким образом простое преобразование данных значительно улучшает производительность LSTM.

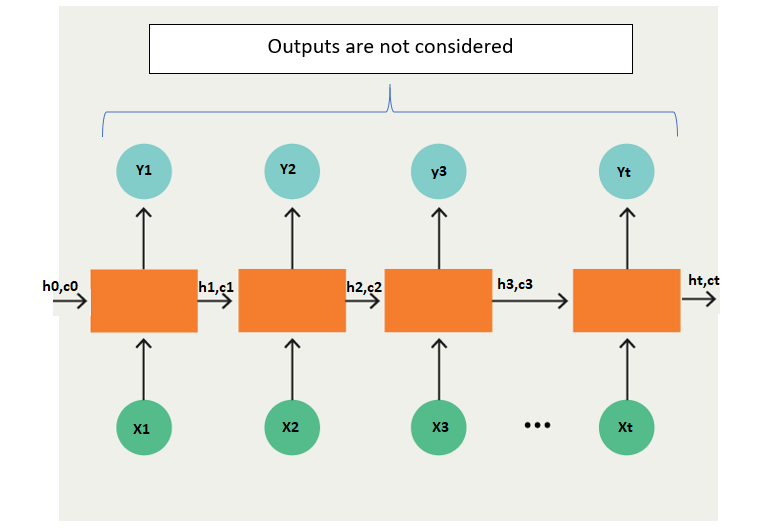
## Архитектура LSTM

Наиболее распространенной архитектурой, используемой для построения моделей Seq2Seq, является архитектура кодировщика-декодера. Как следует из названия, данная модель состоит из двух частей: кодировщик и декодер. Кодировщик:

* Является моделью LSTM (или иногда моделью GRU).
* Кодировщик считывает входную последовательность и суммирует информацию в так называемых векторах внутреннего состояния или векторе контекста (в случае LSTM они называются векторами скрытого состояния и состояний ячеек). Отбрасываются выходы энкодеры и сохраняем только внутренние состояния. Этот вектор контекста призван инкапсулировать информацию для всех входных элементов, чтобы помочь декодеру делать точные прогнозы.
* Скрытые состояния ht вычисляются по формуле 2.2

(2.2)

LSTM считывает данные в одну последовательность за другой (рис. 2.9). Таким образом, если входные данные представляют собой последовательность длины «t», то говорим, что LSTM считывает ее с временными шагами «t».



**Рисунок 2.9** – Последовательность считывание данных.

Примечание – Источник: собственная разработка

1. Xi – Входная последовательность на временном шаге i.
2. hi и ci – LSTM поддерживает два состояния («h» для скрытого состояния и «c» для состояния ячейки) на каждом временном шаге. В совокупности это внутреннее состояние LSTM на временном шаге i.
3. Yi – Выходная последовательность на временном шаге i. Yi на самом деле представляет собой распределение вероятностей по всему словарю, которое генерируется с помощью активации softmax. Таким образом, каждый Yi является вектором размера «vocab\_size», представляющим распределение вероятностей.

Декодер:

* Начальное состояние инициализируются конечными состояниями LSTM кодировщика, то есть вектор контекста последней ячейки кодировщика вводится в первую ячейку сети декодера. Используя эти начальные состояния, декодер начинает генерировать выходную последовательность, и эти выходные данные также принимаются во внимание для будущих выходных векторов.
* Стек из нескольких модулей LSTM, каждый из которых прогнозирует выход yt на временном шаге t.
* Каждый повторяющийся блок принимает скрытое состояние от предыдущего блока: производит и выводит данные, а также свое собственное скрытое состояние.
* Любое скрытое состояние hi вычисляется по формуле 2.3
* Выход yt на временном шаге t вычисляется по формуле 2.4
* вычисляются выходные вектора, используя скрытое состояние на текущем временном шаге вместе с соответствующим весом W(S). Softmax используется для создания вектора вероятности, который поможет нам определить окончательный результат (например, слово в задаче с ответами на вопросы).

(2.3)

(2.4)

Наиболее важным моментом является то, что начальные состояния (h0, c0) декодера устанавливаются в конечные состояния кодировщика. Это интуитивно означает, что декодер обучен запускать генерацию выходной последовательности в зависимости от полученной информации.

Потери рассчитываются на основе прогнозируемых выходных данных для каждого временного шага, и ошибки распространяются обратно во времени, чтобы обновить параметры сети. Обучение сети в течение более длительного периода с использованием достаточно большого количества данных дает довольно хорошие прогнозы.

# 3. ПРИМЕНЕНИЕ LSTM ДЛЯ РАСПОЗНАВАНИЯ SQLI

## Данные для обучения

В данной работе, в качестве dataset были использованы полностью сформированные запросы к серверу (листинг 3.1). Модель была обучена на подмножестве из 21991 запросов, состоящих из 425727 заголовков, участвующие в формировании запроса. Также в обучение участвовал dataset с 1097 аномальными (вредоносными) запросами. Поскольку типичные нейронные языковые модели полагаются на векторное представление каждого слова, был использован фиксированный словарь (листинг 3.2). Для это был задействован список из 105 возможных символов, встречающиеся в самом теле запроса. Каждое слово вне словарного запаса было заменено специальным токеном <UNK>.

ST@RT

Thu, 15 Mar 2018 14:45:52 INFO

POST /vulnbank/online/api.php HTTP/1.1

Host: 10.0.212.25

User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86\_64; rv:59.0) Gecko/20100101 Firefox/59.0

Accept: application/json, text/javascript, \*/\*; q=0.01

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://10.0.212.25/vulnbank/online/login.php

Content-Type: application/x-www-form-urlencoded; charset=UTF-8

X-Requested-With: XMLHttpRequest

Content-Length: 209

Cookie: PHPSESSID=mlacs0uiou344i3fa53s7raut6

Connection: keep-alive

type=user&action=create&username=Jack'+and+extractvalue(0x0a,concat(0x0a,(select version())))+and+'1'='1&password=passw0rd&firstname=first&lastname=last&birthdate=30-08-2017&email=eee%40mail.com&phone=747474747&account=DE44404419569750553340&creditcard=4556-9373-3913-6510

END

**Листинг 3.1** – Пример одного запроса из используемого в dataset

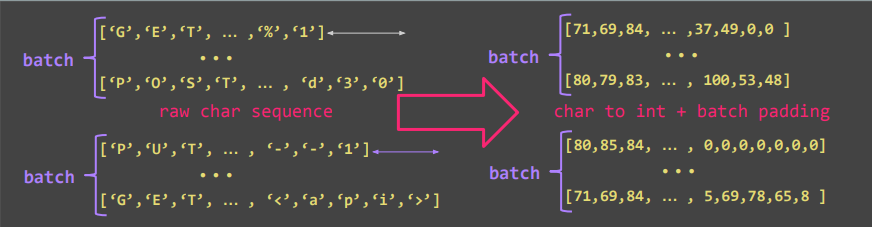
Примечание – Источник: собственная разработка

{"<EOS>": 3, "<UNK>": 1, "&": 71, "$": 69, "(": 73, ",": 77, "0": 4, "4": 8, …}

**Листинг 3.2** – Словарь для преобразования каждого символа в число

Примечание – Источник: собственная разработка

Входные данные преобразуются из строк разной длины в целые числа с использованием словаря (листинг 3.2) и дополняются до максимальной длины в группе (batch) (рис 3.1).



**Рисунок 3.1** – Преобразование строки в вектор

Примечание – Источник: собственная разработка

## Процесс обучения

Определяем составные части решения проблемы: инициализация модели, обучение, прогнозирование и проверка. Модель создается как экземпляр класса Seq2Seq, который имеет аргументы конструктора как показано в листинге 3.3.

Batch\_size – задается количество примеров в группе (рис 3.1)

Embed\_size - размер вложения пространства (меньше размера словаря)

Hidden\_size – количество скрытых состояний в lstm

Num\_layers – количество lstm блоков

Std\_factor - количество стандартных параметров, которое используется для определения порога модели

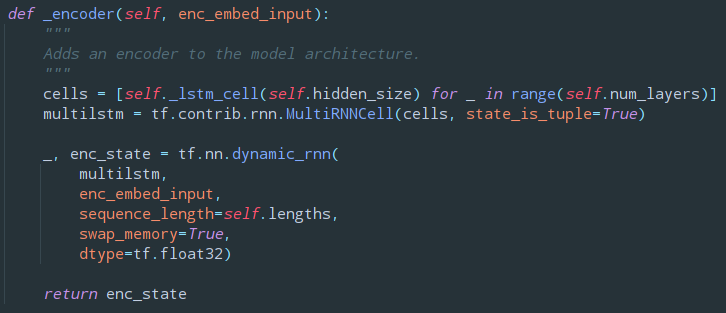
Dropout - вероятность того, что каждый элемент сохранен

Vacab – словарь (листинг 3.2)

**Листинг 3.3** – Аргументы конструктора LSTM

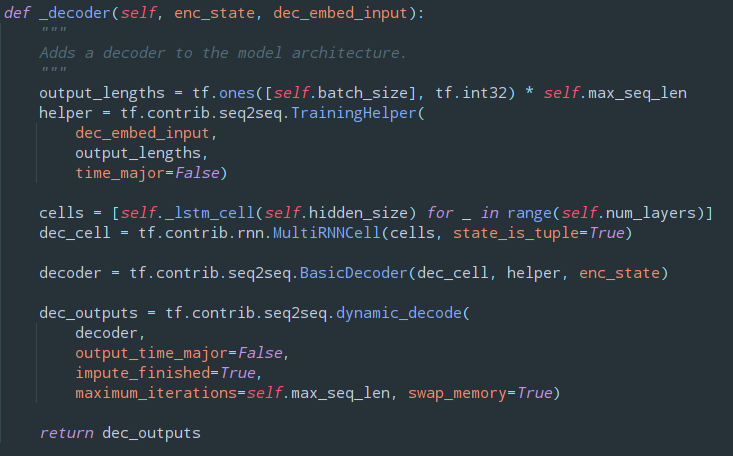
Примечание – Источник: собственная разработка

Следующим шагом идет инициализация слои автоэнкодера (рис 3.2, рис 3.3).



**Рисунок 3.2** – Инициализация слоя кодировщика

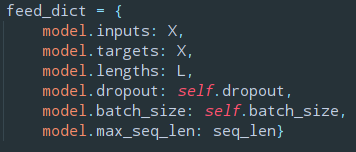
Примечание – Источник: собственная разработка



**Рисунок 3.3** – Инициализация слоя декодера

Примечание – Источник: собственная разработка

Так как решаемая проблема заключается в обнаружении аномалий, то целевые значения и входные данные совпадают (рис 3.4).

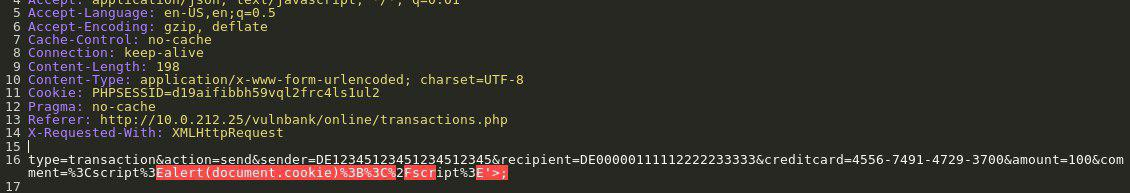


**Рисунок 3.4** – Обработка данных обучения и связанных меток на каждой итерации

Примечание – Источник: собственная разработка

После каждой эпохи лучшая модель сохраняется в качестве контрольной точки, которую затем можно загрузить. В целях тестирования было создано веб-приложение, которое было защищено моделью, чтобы проверить, окажутся ли реальные атаки успешными.

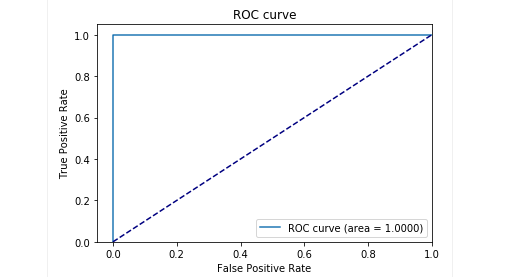
Также была сделана попытка визуализации (рис 3.5), применения к модели автоэнкодера, чтобы отмечать аномальные части данного запроса, но вероятности, выводимые из последнего слоя, работают лучше.



**Рисунок 3.5** – Визуализация работы автоэнкодера

Примечание – Источник: собственная разработка

На этапе тестирования на отложенной выборке были получены очень хорошие результаты: precision и recall близки к 0,99. И ROC-кривая стремится к 1 (рис 3.6).



**Рисунок 3.6** – Точности обучения модели

Примечание – Источник: собственная разработка

# ЗАКЛЮЧЕНИЕ

Таким образом, в ходе разработки курсового проекта были выполнены все запланированные задачи: был изучен материал по теме – рассмотрены основные виды атак на веб-приложения посредством использования HTTP-запросов, а также были разобраны определения, методы и классификация машинного обучения. Были рассмотрены фазы обучения, обнаружения атак и обнаружение аномалий на примере SQL-запросов.

Цель курсового проекта выполнена – обучена модель машинного обучения, с применением sequence to sequence модели, реализовано веб-приложение, для демонстрации использования SQL команд, с целью получения паролей всех пользователей. Внедрена полученная модель в тестовую среду для предотвращения вредоносных запросов.

В первой части курсового проекта были описаны самые популярные виды запросов, их внедрение и эффект. Также был подробно рассмотрен способ обхода приложения на основе внедрения SQL операторов в тело запроса к серверу, какие бывают распространенные типы ошибок SQLi, и описан процесс использования уязвимости SQL.

Во второй части курсового проекта был произведен анализ нейросетевых моделей: сверточные нейронные сети и сети с долговременной краткосрочной памятью. В данном проекте была разобрана модель машинного обучения, основания на Seq2Seq принципе. Был подробно описан каждый шаг внутри алгоритма, а также была рассмотрена архитектура кодировщика и декодера в LSTM. И наконец, рассмотрена реализация, обучение и применение LSTM модели на разработанном приложении. По итогу, модель показала достойные результаты в обнаружении вредоносных запросов.

Таким образом, в разработке любого приложения необходимо предусматривать все возможные варианты его работы, и заблаговременно находить места в коде, пользуясь которыми, злоумышленнику предоставляется полный доступ в системе. Реальные случаи внедрения SQL-кода представляют собой более высокий уровень, что является серьезной проблемой как для моделирования, так и для обучения. Дальнейшая работа может быть направлена на рассмотрение более реалистичных настроек, а также на развертывание более сложных агентов.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Нгуен, М.Т. Тестирование методов машинного обучения в задаче классификации http запросов с применением технологии tf-idf // Вестник ВГУ, серия: системный анализ и информационные технологии, 2019, № 4. – С.119-127.
2. Нгуен, М.Т. Применение метода машинного обучения и анализа атрибутов запроса в задачах обнаружения атак на веб-приложения // Вестник УрФО № 3. – 2019, C. 55–63.
3. Смирнова, К.В. Применимость машинного обучения для задач классификации атак на веб-системы / А.О. Смирнов, В.М. Плотников. – Одесса, 2018. – С.47-49.
4. Машинное обучение Nemesida WAF [Электронный ресурс]. – Режим доступа: <https://waf.pentestit.ru/about/4578>. – Дата доступа: 23.11.2020.
5. Adaptively Detecting Malicious Queries in Web Attacks [Электронный ресурс]. – Режим доступа: <https://arxiv.org/pdf/1701.07774>. – Дата доступа: 03.05.2021.
6. Дик, Д.И. Методы обнаружения аномалий в системах обнаружения вторжений для веб-приложений // Вестник УрФО № 2. 2017, С. 9–17.
7. Nilaykumar, K.S. Machine Learning in Application Security, 2017.
8. Detecting web attacks with end-to-end deep learning [Электронный ресурс]. – Режим доступа: <https://jisajournal.springeropen.com/articles/10.1186>. – Дата доступа: 03.05.2021.
9. Найденова, К.А. Машинное обучение в задачах обработки естественного языка: обзор современного состояния исследований. – Санкт-Петербург, 2008.
10. Jonghoon Lee. Cyber threat detection based on artificial neural networks using event profiles.
11. Клюшева, Е.В. Атаки на Веб-сайты в 2019 году: боты и простые уязвимости / Е. Килюшева, Е. Гнедин // Кибербезопасность 2019-2020: от итогов к прогнозам. – 2020. – С. 38–42.
12. Mishra, S. SQL Injection Detection Using Machine Learning: Master’s Theses and Graduate Research. – США, 2019. – С.51.
13. Мельников В. Г., Трифанов А. В. Методы обхода межсетевых экранов для приложений // Интерэкспо Сибирь. – 2017. – №. 2.
14. Bouzida, Y. Neural networks vs. decision trees for intrusion detection / Y. Bouzida, F. Cuppens // IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM). Vol. 28. – 2006. – С.29-32.
15. Justin Clarke. SQL Injection Attacks and Defense; под общ. ред. Kevvie Fowler. – 2-е изд. – USA: Waltham, 2012. – С. 27–83.
16. R.S. Sutton и A.G. Barto. Reinforcement learning: An introduction. – London, England, 2018. – 157 c.
17. F.M. Zennaro, A.A. Sommervoll. Simulating SQL Injection Vulnerability Exploitation Using Q-Learning Reinforcement Learning Agents. – Oslo, Norway, 2021. – 4 c.
18. FMZennaro / CTF-SQL [Электронный ресурс]. – Режим доступа: <https://github.com/FMZennaro/CTF-SQL>. – Дата доступа: 03.05.2021.