Шеламов М.Д. Разработка, обучение и тестирование нейросети для обнаружения SQL-инъекций // Международный журнал информационных технологий и энергоэффективности. – 2024. –
Т. 9 № 2(40) с. 102–112

Международный журнал информационных технологий и энергоэффективности

Сайт журнала: http://www.openaccessscience.ru/index.php/ijcse/

УДК 004.056.57

РАЗРАБОТКА, ОБУЧЕНИЕ И ТЕСТИРОВАНИЕ НЕЙРОСЕТИ ДЛЯ ОБНАРУЖЕНИЯ SQL-ИНЪЕКЦИЙ

Шеламов М.Д
ФГАОУ ВО "РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА", Москва, Россия (127055, город Москва, ул. Образцова, д.9 стр.9), e-mail: maks.shelamov.00@mail.ru

Статья представляет собой исследование в области кибербезопасности, направленное на создание эффективного средства идентификации SQL-инъекций в веб-приложениях. В работе подробно рассматриваются этапы разработки нейросети, включая выбор архитектуры, сбор и подготовку данных, процесс обучения и тестирование модели. Автор акцентирует внимание на использовании языка программирования Python и библиотек машинного обучения TensorFlow и Keras для эффективной реализации проекта. Полученные результаты подтверждают высокую точность и надежность предложенной нейросетевой модели в обнаружении SQL-инъекций, что делает ее перспективным инструментом для повышения безопасности веб-приложений.

Ключевые слова: Нейросети, Python, SQL-инъекции, машинное обучение, безопасность веб-приложений, TensorFlow.

DEVELOPMENT, TRAINING AND TESTING OF A NEURAL NETWORK FOR SQL INJECTION DETECTION

Shalamov M.D.
RUSSIAN UNIVERSITY OF TRANSPORT, Moscow, Russia (127055, Moscow, Obraztsova st., 9, bldg. 9), e-mail: maks.shelamov.00@mail.ru

The article is a study in the field of cybersecurity aimed at creating an effective means of identifying SQL injections in web applications. The paper discusses in detail the stages of neural network development, including the choice of architecture, data collection and preparation, the learning process and model testing. The authors focus on using the Python programming language and machine learning libraries such as TensorFlow and Keras to effectively implement the project. The results obtained confirm the high accuracy and reliability of the proposed neural network model in detecting SQL injections, which makes it a promising tool for improving the security of web applications.

Keywords: Neural networks, Python, SQL injection, machine learning, web application security, TensorFlow

В современном цифровом ландшафте, где веб-приложения становятся неотъемлемой частью нашей повседневной активности, вопросы кибербезопасности приобретают особенное значение. Среди различных видов киберугроз выделяются SQL-инъекции – манипуляции с базой данных, целью которых является незаконный доступ к конфиденциальной информации [1]. Актуальность угроз SQL-инъекций подчеркивается не только их широким распространением, но и возможностью серьезных последствий для безопасности данных.

В данном контексте становится ясной важность своевременного обнаружения подобных атак, чтобы предотвратить утечку данных, сохранить конфиденциальность и обеспечить надежность веб-приложений. Однако, с учетом постоянно совершенствующихся методов атак и их скрытности, необходимо разработать эффективные инструменты для выявления SQL-инъекций [1].

В последние годы наблюдается стремительное развитие технологий машинного обучения, в частности, нейронных сетей. Эти методы демонстрируют уникальные способности в распознавании сложных паттернов и аномалий, что делает их перспективным решением для обнаружения новых форм киберугроз, включая SQL-инъекции. В данном контексте, создание инструмента на основе нейросетевых технологий представляет собой актуальный и перспективный шаг в направлении повышения кибербезопасности веб-приложений. Эта статья предлагает взгляд на разработку, обучение и тестирование нейросети, специально настроенной на выявление SQL-инъекций, в поиске эффективных решений для надежной защиты цифровых активов.[1]

Модель нейросети

Перед началом разработки нейросети критическое значение имеет выбор подходящей модели. Эффективность и результативность модели непосредственно зависят от ее способности корректно выявлять сложные закономерности в данных при минимальном количестве параметров. В этом контексте важно выбрать модель, которая сочетает в себе простоту и высокую производительность.[2]

Модель перцептрона предстает перед нами как оптимальное решение в данном контексте [2]. Перцептрон – это простая, но эффективная форма искусственной нейронной сети, идеально подходящая для задач бинарной классификации, таких как обнаружение SQL-инъекций. Его особенности делают его привлекательным выбором: минимальное количество слоев, быстрая обучаемость и адаптация к динамике изменяющихся сред.

Перцептрон включает в себя следующие элементы [2]:

Входной слой, состоящий из пяти нейронов, тесно связан с признаковым пространством данных, которое рассмотрено ниже. каждый признак представляет собой информацию, необходимую для выявления атак на базу данных.

Выходной слой, с двумя нейронами и функцией активации Softmax, стремится точно классифицировать сценарии: «атака есть» или «атаки нет».

Функция Softmax в общем случае выглядит следующим образом:

$$softmax(x_i) = \frac{e^{x_1}}{\sum_{j=1}^{n} e^{x_j}}$$

Функция Softmax для нашего случая (2 выхода) выглядит следующим образом:

$$softmax(y_1) = \frac{e^{y_1}}{e^{y_1} + e^{y_2}}$$

$$softmax(y_2) = \frac{e^{y_2}}{e^{y_1} + e^{y_2}}$$

Значения y_1 и y_2 вычисляются следующим образом:

$$y_1 = w_{11}x_1 + w_{12}x_2 + w_{13}x_3 + w_{14}x_4 + w_{15}x_5$$

$$y_2 = w_{21}x_1 + w_{22}x_2 + w_{23}x_3 + w_{24}x_4 + w_{25}x_5$$

Где w – вес признака, выбирающийся случайным образом для дальнейшей оптимизации.

Схема модели нейросети для обнаружения SQL-инъекции представлена на Рисунке 1.

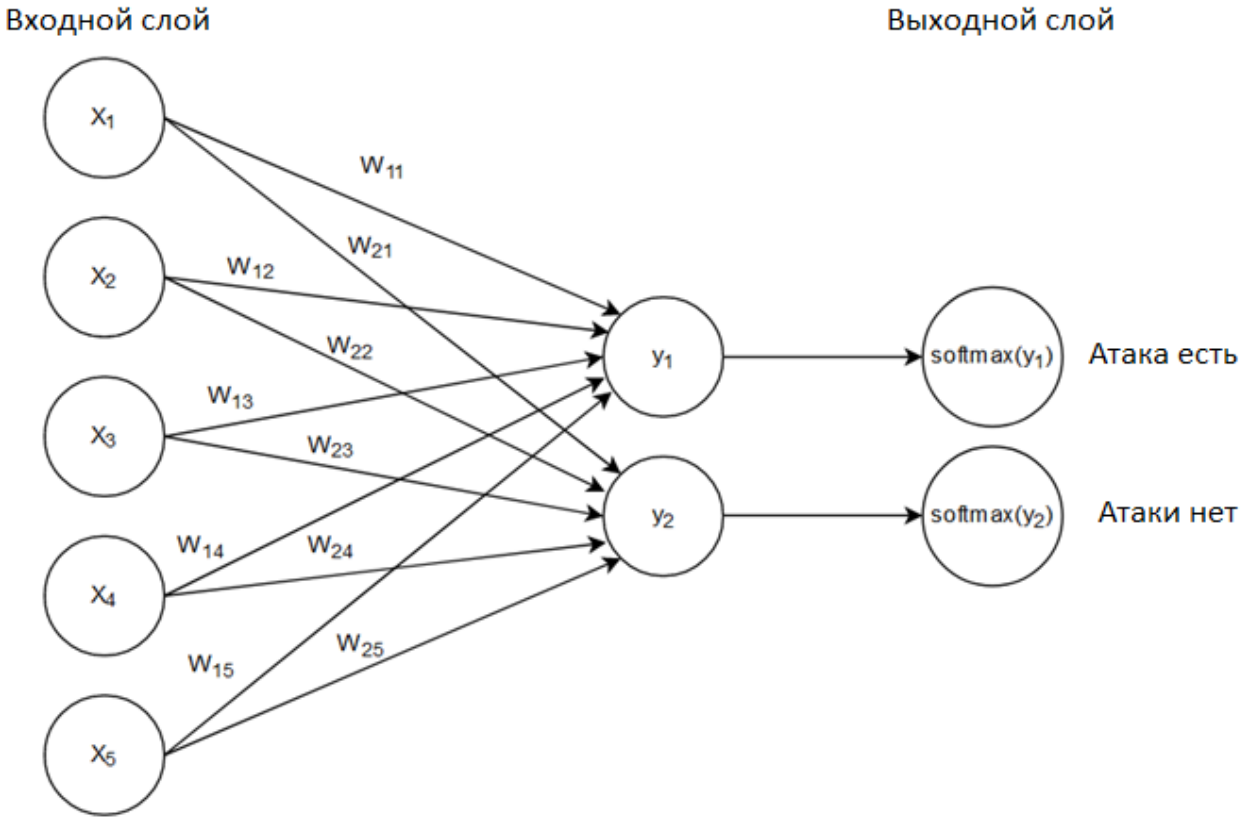

Рисунок 1 – Схема модели нейросети для обнаружения SQL-инъекций

Эта конфигурация модели обеспечивает высокую степень дифференциации между классами, необходимую для точного обнаружения потенциальных угроз SQL-инъекций.[3]

Признаковое пространство

Ключевой этап разработки модели перцептрона для обнаружения SQL-инъекций – описание признаков [3]. Этот этап занимает центральное положение в построении модели, поскольку от выбора и корректного определения признаков зависит способность модели выявлять потенциальные атаки на базы данных. Описывая признаки, мы определяем основные элементы, которые модель будет использовать для анализа данных и выявления аномалий.

Важным шагом при составлении признаков – нормализация [3]. Нормализация признаков – это процесс приведения значений признаков к общему стандартному виду или диапазону. Нормализация важна для обеспечения устойчивости и эффективности обучения моделей, предотвращения доминирования признаков с различными масштабами и повышения сходимости алгоритмов оптимизации. [4]Как правило, после нормализации признаки принимают значения в диапазоне от 0 до 1. В данной работе все признаки будут принимать значения в этом диапазоне.

Изменение объема базы данных

Увеличение или уменьшение объема данных может свидетельствовать о несанкционированном доступе к базе данных. Это может быть вызвано добавлением или удалением таблиц, столбцов, строк.

Нормализованное значение признака вычисляется по формуле:

$$\text{Нормализованное значение признака} = \frac{|V_2 - V_1|}{V_{max}}$$

где:

V_2 – объем БД после изменения,

V_1 – объем БД до изменения,

V_{max} – максимально допустимый объем изменений.

Пример: Добавление новой таблицы

Текущий размер базы данных: 0.5 ТВ

Размер базы данных после добавления новой таблицы: 0.7 ТВ

Максимально возможное изменение – 1 ТВ

Изменение объема: |0.7 – 0.5| = 0.2 ТВ

Нормализованное значение изменения: (0.2 ТВ / 1 ТВ) = 0.2

Увеличение объема трафика

Увеличение объема трафика может быть признаком слепых SQL-инъекций.

Допустим, мы сравниваем объем трафика через каждые 5 минут.

Нормализованное значение признака вычисляется по формуле:

$$\text{Нормализованное значение признака} = \min\left(1, \frac{V_2 - V_1}{V_{max}}\right)$$

Где:

V_2 – объем трафика после изменения,

V_1 – объем трафика до изменения,

V_{max} – максимально допустимое изменение трафика.[5]

Если разница в трафике больше максимально допустимого изменения, нормализованное значение будет ограничено до 1.

В данном случае нас интересует именно увеличение объема траффика. Т.е. если объем трафика уменьшился, принимаем значение признака равным 0.

Пример:

Текущий объем трафика: 500 Мбит/с

Размер объем трафика после изменения: 865 Мбит/с

Максимально возможное изменение – 300 Мбит/с

Нормализованное значение признака: $\min\left(1, \frac{865 - 500}{300}\right) = 1$

Увеличение числа запросов

Увеличение количества запросов может быть признаком эксплуатации слепых SQL-инъекций.

Для получения значения данного признака сравниваем число запросов в определенные промежутки времени.

Нормализованное значение признака вычисляется по формуле:

$$\text{Нормализованное значение признака} = \min\left(1, \frac{V_2 - V_1}{V_{max}}\right)$$

Где:

V_2 – число запросов после увеличения,

V_1 – число запросов до увеличения,

V_{max} – максимально допустимое увеличение числа запросов.

Если разница в количестве запросов больше максимально допустимого изменения, нормализованное значение будет ограничено до 1.

В данном случае нас интересует именно увеличение кол-ва запросов, т.е., если число запросов уменьшилось, принимаем значение признака равным 0.

Пример:

Текущее число запросов за выбранный промежуток времени(1 мин): 40

Число запросов после изменения: 200.

Максимально возможное изменение: 100 .

Нормализованное значение признака: $\min\left(1, \frac{200-40}{100}\right) = 1$

Ошибки базы данных

SQL-инъекции часто приводят к ошибкам/предупреждением БД.

Данный признак принимает булевы значения: 0 – ошибка/предупреждение не обнаружена и 1 – ошибка/предупреждение обнаружена.

Примеры:

Запрос:

```
SELECT first_name, last_name
FROM employees
WHERE department = 'Sales'
```

Ответ:

```
Timestamp: 2023-10-22 10:15:00
Query executed successfully.
Result:
```

first_name | last_name

```
---------------------
John       | Smith
Emily      | Johnson
...
```

Т.к. ответ на данный запрос не содержит ошибку/предупреждение, поэтому значение признака = 0.

Запрос:

```
SELECT * FROM users WHERE username = '' OR '1' = '1';
```

Ответ:

```
Timestamp: 2023-10-22 14:45:00
Query executed successfully.
Result:
username | email           | ...
--------------------------------
John     | john@email.com  | ...
Alice    | alice@email.com | ...
...
Warning: The query returned a large number of rows.
```

Ответ на данный запрос содержит ошибку/предупреждение, поэтому значение признака в данном случае = 1.

Необычные параметры и специальные символы

Существует ряд параметров и символов не характерных для обычных SQL-запросов, поэтому их наличие может свидетельствовать о SQL-инъекции.

Данный признак принимает булевы значения: 0 – необычный параметр/символ не обнаружен, 1 – был обнаружен необычный параметр/символ.

Пример:

```
user_input = ' OR 1 = SLEEP(10) --
```

Данный запрос содержит SQL-функцию, которая не является обычной. Следовательно, значение признака в данном случае – 1.

Пример:

Попытки ввода символов HTML или URL-кодирования (например, %3C или <) могут быть использованы для обхода фильтров и выполнения атаки.

```
user_input = ' OR 1=1%20--
```

В данном запросе обнаружены спецсимволы (`%20` и `--`), поэтому значение признака – 1.

Разработка нейросети на языке Python

Шаг 1: Подготовка данных

Прежде чем создать нейросеть для обнаружения SQL-инъекций нам необходимо подготовить данные. В данном случае, мы будем использовать библиотеку NumPy для создания массивов, представляющих собой входные данные (X) и соответствующие им метки классов (y).

Полный код будет представлен ниже. Теперь у нас есть данные, и мы готовы перейти ко второму шагу.

Шаг 2: Разделение данных

Далее мы хотим разделить наши данные на обучающую и тестовую выборки. Мы используем функцию train_test_split из библиотеки scikit-learn для этой задачи.

Теперь у нас есть обучающие и тестовые данные, и мы готовы приступить к следующему этапу.

Шаг 3: Преобразование меток классов

На следующем этапе мы преобразуем метки классов в формат one-hot encoding с помощью OneHotEncoder из scikit-learn.

Теперь у нас есть подготовленные данные для обучения нашей нейросети.

Шаг 4: Создание модели

Для построения модели мы используем библиотеки TensorFlow [4] и Keras [5]. Мы создадим последовательную модель и добавим к ней выходной слой с 2 нейронами и L2 регуляризацией.

Мы создали модель, но она ещё не готова к обучению. Перейдем к следующему шагу.

Шаг 5: Компиляция модели

На этом этапе мы компилируем нашу модель, указывая оптимизатор, функцию потерь и метрику, которую мы будем отслеживать.

Теперь наша модель готова к обучению.

Шаг 6: Обучение модели

Мы обучаем модель, используя обучающие данные, и сохраняем историю обучения для последующей визуализации.

Теперь наша модель обучена. Перейдем к оценке ее производительности на тестовых данных.

Шаг 7: Оценка модели

Мы оцениваем производительность модели на тестовых данных и выводим значения потерь и точности.

Теперь мы знаем, как хорошо работает наша модель. Давайте визуализируем результаты.

Шаг 8: Визуализация результатов

Используя библиотеку Matplotlib, мы строим графики потерь и точности на обучающей и тестовой выборках по эпохам. Эпоха – количество итераций обучения и тестирования нейросети.

Теперь мы завершили разработку нашей нейросети для обнаружения SQL-инъекций. Полная версия кода представлена ниже.

Код нейросети на языке Python

```
# Подключение необходимых библиотек
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.regularizers import l2

# Подготовка данных
# X - это ваши данные, y - это ваши метки классов
X = np.array([
    [0.1, 0.2, 0.3, 0.0, 1.0],  # пример обучения 1
    [0.2, 0.9, 0.8, 0.0, 0.0],  # пример обучения 2
    [0.3, 0.2, 0.1, 1.0, 1.0],  # пример обучения 3
    [0.1, 0.2, 0.3, 0.0, 0.0],  # пример обучения 4
    [0.1, 0.2, 0.2, 1.0, 0.0],  # пример обучения 5
    [0.34, 0.23, 0.0, 0.0, 1.0],
    [0.0, 0.46, 0.3, 0.0, 0.0],  # пример обучения 7
    [0.8, 0.3, 1.0, 0.0, 0.0],  # пример обучения 8
    [0.3, 0.4, 0.67, 1.0, 0.0],  # пример обучения 9
    [0.0, 0.22, 0.11, 0.0, 0.0],  # пример обучения 10
    [0.17, 0.04, 0.0, 0.0, 0.0],  # пример обучения 11
    [0.0, 0.04, 0.03, 1.0, 0.0],
    [0.0, 0.96, 0.87, 0.0, 0.0],  # пример обучения 13
    [0.16, 0.3, 0.03, 1.0, 1.0],  # пример обучения 14
    [0.12, 0.47, 0.36, 0.0, 0.0],  # пример обучения 15
    [0.07, 0.34, 0.4, 0.0, 0.0],  # пример обучения 16
    [0.79, 0.0, 0.1, 1.0, 0.0],  # пример обучения 17
    [0.25, 0.2, 0.0, 0.0, 0.0],
    [0.1, 0.2, 0.11, 0.0, 0.0],  # пример обучения 19
    [0.0, 0.15, 0.0, 0.0, 1.0],  # пример обучения 20
    [0.3, 0.16, 0.05, 0.0, 0.0],  # пример обучения 21
    [0.5, 0.2, 0.1, 1.0, 0.0],  # пример обучения 22
    [0.0, 0.1, 0.4, 1.0, 1.0],  # пример обучения 23
    [0.65, 0.2, 0.5, 0.0, 0.0],
    [0.06, 0.08, 0.25, 0.0, 0.0],  # пример обучения 25
    [0.53, 0.15, 0.1, 1.0, 0.0],  # пример обучения 26
    [0.0, 0.33, 0.51, 0.0, 0.0],  # пример обучения 27
```

```
    [0.0, 0.0, 0.04, 1.0, 1.0],  # пример обучения 28
    [0.2, 0.15, 0.0, 0.0, 0.0],  # пример обучения 29
    [0.0, 0.0, 0.0, 0.0, 0.0],
    [0.0, 0.17, 0.11, 0.0, 1.0],  # пример обучения 31
    [0.2, 0.3, 0.0, 0.0, 0.0],  # пример обучения 32
    [0.0, 0.12, 0.0, 1.0, 1.0],  # пример обучения 33
    [0.89, 0.45, 0.77, 0.0, 0.0],  # пример обучения 34
    [0.0, 0.01, 0.0, 1.0, 0.0],  # пример обучения 35
    [0.23, 0.0, 0.0, 0.0, 0.0],
    [0.0, 0.2, 0.22, 1.0, 0.0],  # пример обучения 37
    [0.02, 0.09, 0.32, 0.0, 0.0],  # пример обучения 38
    [0.88, 0.33, 0.56, 0.0, 0.0],  # пример обучения 39
    [0.0, 0.06, 0.0, 0.0, 0.0],  # пример обучения 40
    [0.0, 0.37, 0.45, 0.0, 0.0],  # пример обучения 41
    [0.0, 0.2, 0.03, 0.0, 1.0],
    [0.0, 0.05, 0.17, 0.0, 0.0],  # пример обучения 43
    [0.02, 0.03, 0.02, 1.0, 1.0],  # пример обучения 44
    [0.2, 0.15, 0.0, 0.0, 1.0],  # пример обучения 45
    [0.45, 0.0, 0.0, 0.0, 0.0],  # пример обучения 46
    [0.3, 0.5, 0.66, 0.0, 0.0],  # пример обучения 47
    [0.0, 0.3, 0.02, 1.0, 0.0],  # пример обучения 48
    [0.0, 0.03, 0.2, 0.0, 1.0],  # пример обучения 49
    [0.15, 0.0, 0.0, 1.0, 1.0],  # пример обучения 50

    [0.0, 0.37, 0.45, 1.0, 0.0],  # пример обучения 51
    [0.0, 0.37, 0.45, 0.0, 0.0],
    [0.1, 0.3, 0.17, 0.0, 0.0],  # пример обучения 53
    [0.0, 0.0, 0.0, 1.0, 0.0],  # пример обучения 54
    [0.2, 0.6, 0.65, 0.0, 0.0],  # пример обучения 55
    [0.45, 0.0, 0.0, 0.0, 0.0],  # пример обучения 56
    [0.3, 0.5, 0.66, 0.0, 1.0],  # пример обучения 57
    [0.15, 0.25, 0.02, 0.0, 0.0],  # пример обучения 58
    [0.6, 0.03, 0.0, 1.0, 0.0],  # пример обучения 59
    [0.15, 0.0, 0.0, 0.0, 0.0],  # пример обучения 60

    [0.0, 0.2, 0.75, 1.0, 0.0],  # пример обучения 61
    [0.0, 0.2, 0.3, 0.0, 1.0],
    [0.65, 0.67, 0.3, 0.0, 0.0],  # пример обучения 63
    [0.0, 0.51, 0.43, 1.0, 0.0],  # пример обучения 64
    [0.0, 0.1, 0.15, 0.0, 1.0],  # пример обучения 65
    [0.45, 0.44, 0.5, 0.0, 0.0],  # пример обучения 66
    [0.1, 0.0, 0.07, 1.0, 0.0],  # пример обучения 67
    [0.06, 0.25, 0.2, 0.0, 0.0],  # пример обучения 68
    [0.0, 0.5, 0.12, 0.0, 0.0],  # пример обучения 69
    [0.6, 0.0, 0.0, 0.0, 0.0],  # пример обучения 70
    # ...
])

y = np.array([1,  # метка для примера обучения 1 ("атаки нет")
    1,  # метка для примера обучения 2 ("атака есть")
```

```
1,0,0,1,0,1,1,0,0,0,1,1,0,0,1,0,0,1,0,1,1,1,0,1,0,1,0,0,1,0,0,1,1,
0,0,0,1,0,0,1,
0,1,1,0,1,0,1,1,1,0,0,0,1,0,1,0,1,0,1,1,1,1,1,1,1,0,0,0,0,])

# Разделение данных на обучающую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

# Преобразование меток классов в формат one-hot encoding
encoder = OneHotEncoder(sparse=False)
y_train = encoder.fit_transform(y_train.reshape(-1, 1))
y_test = encoder.transform(y_test.reshape(-1, 1))

# Создание модели
model = Sequential()

# Добавление выходного слоя с 2 нейронами и L2 регуляризацией
model.add(Dense(2, activation='softmax', input_shape=(5,),
kernel_regularizer=l2(0.01)))

# Компиляция модели
model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=['accuracy'])

# Обучение модели и сохранение истории обучения
history = model.fit(X_train, y_train, epochs=300, batch_size=32,
validation_data=(X_test, y_test))

# Оценка модели на тестовых данных
loss, accuracy = model.evaluate(X_test, y_test)

print(f'Loss: {loss}')
print(f'Accuracy: {accuracy}')

# Построение графиков функции потери и точности
plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)
plt.plot(history.history['loss'], label='Обучение')
plt.plot(history.history['val_loss'], label='Тесты')
plt.title('Изменение функции потери')
plt.xlabel('Эпоха')
plt.ylabel('Функция потери')
plt.legend()

plt.subplot(1, 2, 2)
plt.plot(history.history['accuracy'], label='Обучение')
plt.plot(history.history['val_accuracy'], label='Тесты')
plt.title('Изменение точности')
```

```
plt.xlabel('Эпоха')
plt.ylabel('Точность')
plt.legend()

plt.tight_layout()
plt.show()
```

Оценка эффективности нейросети

Графики, отражающие изменение метрик на протяжении обучения и тестирования представлены на Рисунке 2.

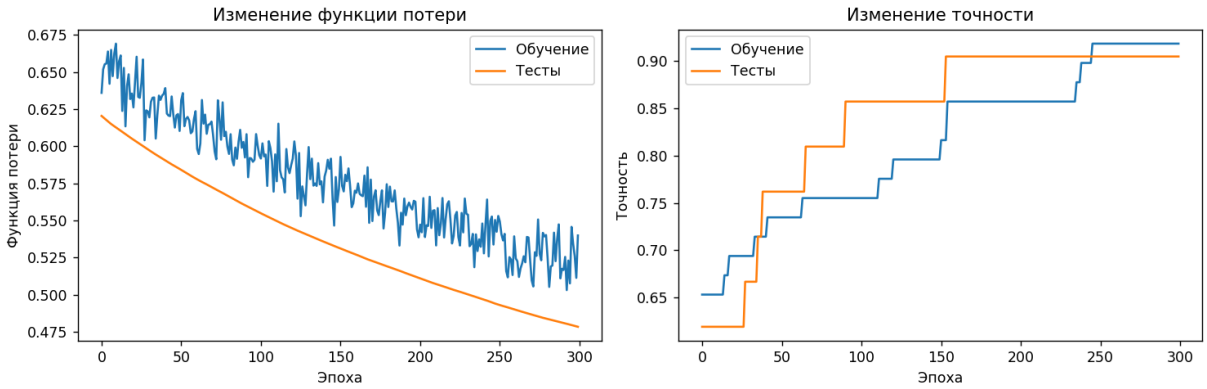

Рисунок 2 – Результаты обучения нейросети

В ходе обучения удалось добиться показателей точности определения наличия атаки >0.9, т.е., более чем в 90 процентах случаев атака будет идентифицирована нейросетью.

Выводы

В данной статье проведена разработка нейросети для обнаружения SQL-инъекций. Этот вопрос является крайне актуальным в контексте безопасности веб-приложений, и использование современных технологий, таких как нейронные сети, может стать ключом к успешному предотвращению атак. Разработанная нейросеть представляет собой мощный инструмент в арсенале средств обеспечения безопасности в области веб-разработки. Она подчеркивает важность использования передовых технологий для борьбы с постоянно эволюционирующими угрозами. Опыт в разработке этой модели вдохновит и мотивирует других исследователей и разработчиков в области кибербезопасности. Безопасность в сфере информационных технологий — это постоянная борьба, и инновационные решения, такие как использование нейросетей, могут сделать этот мир интернета более защищенным

Список литературы

1. Евтеев Д. SQL Injection от А до Я // Positive Technologies. 2022. С. 7-26. URL: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-Advanced-SQL-Injection.pdf (дата обращения: 11.01.2024).
2. Чачхиани Т.И., Серова М.Г. Алгоритм перцептрона: практикум. – Нижний Новгород: Нижегородский госуниверситет, 2015. – 25 с. С. 3-5. URL: http://www.unn.ru/books/met_files/Preceptron1.pdf (дата обращения: 11.01.2024).

3. Что такое машинное обучение? Обучающая выборка и признаковое пространство. URL: https://proproprogs.ru/ml/ml-chto-takoe-mashinnoe-obuchenie-obuchayushchaya-vyborka-i-priznakovoe-prostranstvo (дата обращения: 11.01.2024).
4. Библиотека для машинного обучения TensorFlow. URL: https://www.tensorflow.org (дата обращения: 11.01.2024).
5. Keras: Deep Learning for humans. URL: https://keras.io (дата обращения: 11.01.2024).

References

1. Evteev D. SQL Injection from A to Z // Positive Technologies. 2022. pp. 7-26. URL: https://www.ptsecurity.com/upload/corporate/ru-ru/analytics/PT-devteev-Advanced-SQL-Injection.pdf (date of reference: 11.01.2024).
2. Chachhiani T.I., Serova M.G. Perceptron algorithm: a practical course. – Nizhny Novgorod: Nizhny Novgorod State University, 2015. – 25 p. pp. 3-5. URL: http://www.unn.ru/books/met_files/Preceptron1.pdf (date of application: 11.01.2024).
3. What is machine learning? Training sample and feature space. URL: https://proproprogs.ru/ml/ml-chto-takoe-mashinnoe-obuchenie-obuchayushchaya-vyborka-i-priznakovoe-prostranstvo (accessed: 11.01.2024).
4. TensorFlow Machine Learning Library. URL: https://www.tensorflow.org (accessed: 11.01.2024).
5. Keras: Deep Learning for humans. URL: https://keras.io (date of application: 11.01.2024).