Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Курсова робота

освітній ступінь – бакалавр

на тему: «**Система виявлення зловмисних дій для фреймворку Spring з використанням елементів машинного навчання**»

Виконав: студент 3-го року навчання,
Спеціальності
122 «Комп'ютерні науки»

Боголій Владислав Валентинович

Керівник Бабич Т.А.,
магістр комп'ютерних наук, асистент
«____» _____ 20____ р.

Київ – 2021

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра інформатики

Освітній ступінь бакалавр

Спеціальність 122 «Комп'ютерні науки»

Освітня програма бакалавр

ЗАТВЕРДЖУЮ

Завідувач кафедри інформатики

Гороховський С. С.

"10" жовтня 2020 року

ЗАВДАННЯ
ДЛЯ КУРСОВОЇ РОБОТИ СТУДЕНТУ

Боголію Владиславу Валентиновичу

1. Тема роботи «**Система виявлення зловмисних дій для фреймворку Spring з використанням елементів машинного навчання**», керівник роботи Бабич Трохим Анатолійович, магістр комп'ютерних наук, асистент

2. Строк подання студентом роботи 17 травня 2021

3. План роботи:

    Анотація

    Вступ

    Розділ 1. Аналіз предметної області

    Розділ 2. Опис розробленої системи

    Розділ 3. Опис демонстраційного клієнтського застосунку

    Розділ 4. Тестування системи

    Висновки

Список використаних джерел

Додатки

ГРАФІК ПІДГОТОВКИ КУРСОВОЇ РОБОТИ ДО ЗАХИСТУ

№ з/п	ПЕРЕЛІК РОБІТ	Термін виконання	Дата ознайомлення наукового керівника	Підпис наукового керівника	Примітки
1.	Вибір теми, затвердження її на засіданні кафедри та закріплення наукового керівника Узгодження календарного графіка підготовки кваліфікаційної роботи. Ознайомлення студента з критеріями оцінювання кваліфікаційної роботи (п. 8.5).	01.10.2020			
2.	Вивчення джерел літератури, матеріалів архівів, періодичних видань, збір та узагальнення фактів, даних	05.11.2020			
3.	Складання плану каліф. роботи та узгодження з науковим керівником	21.10.2020			
4.	Написання розділів роботи *або* Постановка експерименту, аналіз отриманих результатів наукового дослідження	15.12.2020			
5.	Проміжний контроль виконання роботи	25.01.2021			
6.	Написання кваліфікаційної роботи в цілому, ознайомлення з її першим варіантом наукового керівника	12.02.2021			
	Розділ 1 (постановка проблеми, теоретичні основи, огляд літературних джерел)	07.03.2021			
	Розділ 2 (аналітично-дослідницька частина) (експериментальна частина для природничих і біологічних наук)	21.03.2021			
	Розділ 3 (проектно-рекомендаційна частина) (аналіз результатів експерименту для природничих і біологічних наук)	28.03.2021			
7.	Повне завершення написання кваліфікаційної роботи, оформлення її згідно з вимогами й подання на відгук науковому керівнику	16.05.2021			
8.	Подання кваліфікаційної роботи для перевірки письмових робіт студентів НаУКМА на відповідність вимогам академічної доброчесності,	17.05.2021			
9.	Подання на зовнішню рецензію				
10.	Підготовка до захисту кваліфікаційної роботи на засіданні кафедри: написання доповіді та виготовлення ілюстративного матеріалу	**до ____ травня**			
11.	Попередній захист кваліфікаційної роботи на засіданні кафедри	**до ____ травня**			
12.	Подання кваліфікаційної роботи на кафедру з усіма супроводжувальними документами	**до ____ травня**			
13.	Публічний захист кваліфікаційної роботи перед екзаменаційною комісією	згідно з розкладом роботи ЕК			

Графік узгоджено 01.10.2020 р.

Науковий керівник Бабич Трохим Анатолійович

Виконавець курсової роботи Боголій Владислав Валентинович

Зміст

Анотація 1

Вступ 2

Розділ 1. Аналіз предметної області 3

1.1. Відомі підходи та рішення 3

1.2. Обґрунтування обраного методу 3

Розділ 2. Опис розробленої системи 5

2.1. Програмне забезпечення 5

2.2. Експлуатація системи 5

2.3. База даних 6

2.4. Класифікатор 8

2.5. Серверна частина 8

2.5.1. Веб-інтерфейс 8

2.5.2. Збір даних 9

2.5.3. Тренування 11

2.5.4. Виявлення аномалій 11

Розділ 3. Опис демонстраційного клієнтського застосунку 15

3.1. Короткий опис 15

3.2. Програмне забезпечення 15

3.3. База даних 15

3.4. Серверна частина 16

Розділ 4. Тестування системи 18

Висновки 20

Список використаних джерел 21

Додатки 22
Додаток А 22
Додаток Б 23

Анотація

В даній роботі побудовано та успішно протестовано систему визначення злочинних дій. Розроблена система може бути використана для підвищення захисту веб-застосунків.

Вступ

В сучасному світі інтернет стає доступним із все більшої кількості точок на планеті, особливо після запуску проекту Starlink, який вселяє надію про можливість доступу до інтернету з будь-якої точки поверхні Землі. Все більше компаній розробляє власні веб-додатки, надає інтернет-послуги або принаймні має свій сайт.

В період пандемії з глобальним переходом в онлайн збільшилась кількість атак з боку хакерів. Як зазначено в звіті від FBI за 2020-й рік кількість скарг пов’язаних з інтернет-злочинами зросла на 69% в порівнянні з 2019-м роком, а заявлені втрати перевищили 4 100 мільйони доларів США [1].

Мета даної роботи – розробити систему виявлення зловмисних дій для застосунку на базі фреймворку Spring.

Актуальність теми:

Сучасний світ неможливо уявити без веб-додатків. Оскільки більшість додатків не ідеальні та мають вразливості, розробка систем захисту повинна постійно вдосконалюватися.

Об’єкт дослідження:

Виявлення зловмисних дій по відношенню до веб-застосунку.

Предмет дослідження:

Система, що зможе автоматично розпізнавати зловмисні дії.

Розділ 1. Аналіз предметної області

1.1. Відомі підходи та рішення

Популярними підходами до виявлення атак є наступні:

- Сигнатурний аналіз – метод, що базується на виявленні в даних раніше знайдених та описаних шаблонів атак. Сигнатурний аналіз характеризується високою точністю для відомих атак, але недієвий проти нових, оскільки їх сигнатури ще не виділені та не занесені в базу даних.
- Поведінковий аналіз – метод, що базується на виявленні ознак нормальних даних та аномалій за допомогою машинного навчання. Цей метод характеризується стійкістю до нових видів атак, а також більш вірогідними хибно-позитивними спрацьовуваннями.

Існує досить багато рішень, які спеціалізуються на безпеці веб-застосунків. Вважається, що найбільший рівень безпеки надають Web Application Firewall (WAF), які являють собою сукупність фільтрів та сканерів, що слідкують за HTTP-трафіком та блокують його у разі виявлення зловмисних дій. Одним з найпопулярніших та найвідоміших є WAF від Cloudflare, який використовує різні підходи для виявлення атак, включаючи сигнатурний аналіз та машинне навчання [2].

1.2. Обґрунтування обраного методу

Оскільки ми живемо у світі, що швидко змінюється, в якому постійно з’являються нові атаки, було прийнято рішення розробляти систему яка змогла б зрозуміти які дані є нормальними для

клієнтського застосунку та, на основі цього, виявляти аномалії в параметрах запитів та типах файлів, які завантажуються користувачами. Також слід зазначити, що в дослідженнях заснованих на даному методі досягалася висока точність виявлення атак [3].

Розділ 2. Опис розробленої системи

2.1. Програмне забезпечення

Через велику кількість доступних бібліотек, які реалізують алгоритми машинного навчання, для розробки системи було обрано мову Python. Для обробки HTTP-запитів та роботи з базою даних було використано фреймворк Django. У якості СКБД використано PostgreSQL.

Для тренування класифікатора даних, отриманих від користувача системи, використано бібліотеку fastText. Основними перевагами цієї бібліотеки є можливість розбиття слова на n-грами, що дозволяє класифікувати слова, які відсутні в словнику, та наявність автоматичного підбору гіперпараметрів для моделі [4, 5].

2.2. Експлуатація системи

Після встановлення необхідного програмного забезпечення, застосунок готовий до використання в локальній мережі. Після запуску сервер має наступні основні етапи роботи:

1. Збір даних – система накопичує дані, які отримує від клієнтського застосунку та зберігає їх у базу даних для подальшого опрацювання.
2. Обробка даних – на цьому етапі система аналізує збережені дані та робить припущення про їх природу.
3. Виявлення аномалій – всі нові дані, що надходять до системи від клієнтського застосунку, проходять перевірку на співпадіння з очікуваним значенням, аналіз даних надсилається у відповідь.

2.3. База даних

Для збереження інформації про запити, а також збереження результатів аналізу зібраних даних було розроблено схему бази даних (рисунок 2.3.1).

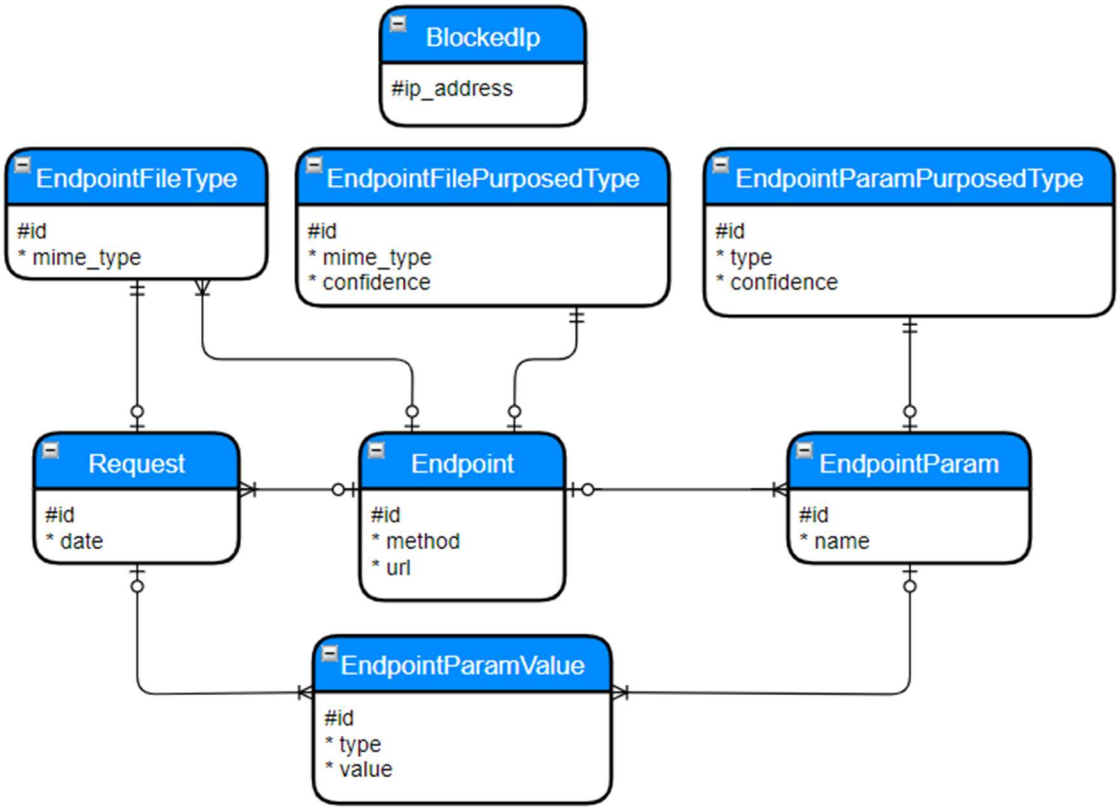


Рисунок 2.3.1 Схема бази даних

Всі таблиці мають id як первинний ключ, виключенням є табличка BlockedIp, у якої первинним ключем є ip-адреса. Розглянемо таблиці більше детально:

- Endpoint:
  - url – url або шлях до ресурсу в межах домену на який було надіслано запит
  - method – HTTP-метод (GET, POST …)
- EndpointParam:
  - name – назва параметру запиту

- EndpointParamValue:
    - type – мітка, визначена системою для значення, що міститься у value
    - value – значення конкретного параметра, який надійшов з запитом
- EndpointParamPurposedType:
    - type – мітка, що визначена як очікувана для конкретного параметра
    - confidence – впевненість системи у тому, що мітку визначено правильно. Наразі обраховується як кількість запитів з найбільш частою міткою поділена на загальну кількість запитів з даним параметром
- Request:
    - date – дата та час запиту
- EndpointFileType:
    - mime_type – визначений MIME-тип файлу, який був надісланий запитом
- EndpointFilePurposedType:
    - mime_type – MIME-тип, що визначений як очікуваний для конкретного значення в таблиці Endpoint
    - confidence – впевненість системи у тому, що MIME-тип визначено правильно. Наразі обраховується як кількість запитів з найбільш частим типом поділена на загальну кількість запитів завантаження файлу, що відносяться до конкретного запису в таблиці Endpoint
- BlockedIp:
    - blocked_ip – ip-адреса, запити з якої були визначені як аномальні, і тому ігноруються

2.4. Класифікатор

Для надання міток значенням параметрів, що отримані як інформація про запит, за допомогою бібліотеки fastText було натреновано класифікатор.

Для тренування було зібрано та розмічено дані. Частина даних для XML та JSON форматів були згенеровані спеціальним сервісом [6]. Для SQL-ін’єкцій та звичайних рядків було використано набір даних з платформи Kaggle [7]. Для XSS-ін’єкцій дані були зібрані з кількох репозиторіїв на платформі GitHub.

Зібрані дані були розбиті випадковим чином на тренувальний (80%) та тестовий (20%) набори. Після десяти хвилин тренування модель переставала навчатися. Було проведено більше двадцяти спроб натренувати модель та обрано найкращу. Для визначення яка з моделей краща було обраховано точність, повноту, а також визначено F1-міру [8].

Точність та повнота, для визначення яких було використано вбудований функціонал бібліотеки fastText, найкращої моделі становлять: на тренувальному наборі (31336 екземплярів) 0.99978, а на тестовому (8432 екземпляри) – 0.99704.

2.5. Серверна частина

2.5.1. Веб-інтерфейс

Після запуску, спілкування клієнта з системою проходить через веб-інтерфейс (рисунок 2.5.1.1).

```
urlpatterns = [
    path('detector/collect/params/', controller.collect_params),
    path('detector/collect/file/', controller.collect_file),
    path('detector/train/', controller.train),
    path('detector/verify/params/', controller.verify_params),
    path('detector/verify/file/', controller.verify_file),
    path('detector/reload-blocked-ips/', controller.reload_blocked_ips),
    path('detector/update-security-params/', controller.update_security_params),
]
```

Рисунок 2.5.1.1 Веб-інтерфейс застосунку

Всі запити до сервера мають надсилатися методом POST, а дані запитів, якщо вони є, у форматі JSON.

У системи наявна панель адміністратора (додаток А), яка дозволяє переглядати накопичені дані в зручному вигляді, а також додавати та видаляти їх з бази даних. Доступ до панелі налаштовується перед запуском застосунку.

2.5.2. Збір даних

На першому етапі клієнтський застосунок повинен надсилати системі інформацію про запити, які отримує. Окремо надсилається інформація про завантаження файлів.

Інформація про запит має містити наступні поля:

- method – HTTP-метод яким було надіслано запит
- endpointUrl – url або шлях до ресурсу в межах домену на який було надіслано запит
- params – параметри, що надійшли із запитом у форматі JSON-об’єкту

Рисунок 2.5.2.1 Приклад надання інформації про запит

Після отримання інформації про запит, кожному з параметрів система ставить у відовідність мітку, що характеризує його дані. Система розрізняє мітки, що були підібрані відповідно до основних типів даних: __label__bool, __label__int, __label__float, __label__string; основних форматів передачі даних: __label__json, __label__xml; а також популярних та небезпечних атак [9]: __label__sql_inj_attack, __label__xss_attack.

Для міток __label__bool, __label__int та __label__float використовуються стандартні методи Python: для перевірки на бульовий тип використовується функція «isinstance», для перевірки на ціле число та число з плаваючою крапкою використовується зведення типу рядка до відповідного типу. Для визначення всіх інших міток використовується заздалегідь натренований класифікатор.

Для виявлення очікуваних типів файлів, здійснюється збір наступної інформації:

- method – HTTP-метод яким було надіслано запит
- endpointUrl – url або шлях до ресурсу в межах домену на який було надіслано запит
- base64fileBytes – перші 2048 байтів файлу або весь файл, якщо він меншого розміру, що закодовані в рядок символів формату base64

Останнє поле передається для визначення типу файлу, оскільки розширення можна легко змінити. Закодований рядок декодується в масив байтів, після чого за допомогою бібліотеки python-magic визначається MIME-тип файлу. Отримана інформація зберігається в базу даних.

2.5.3. Тренування

При отриманні запиту на тренування система завантажує накопичену інформацію з бази даних та для кожного набору з URL, HTTP-методу та назви параметра знаходить частоту кожної з міток, після чого зберігає мітку, яка зустрічається найчастіше, як ту, що відповідає даному набору.

Для файлів використовується подібний підхід: набору з URL та HTTP-методу ставиться у відповідність MIME-тип, що найбільш часто зустрічається для цього набору.

2.5.4. Виявлення аномалій

При отриманні запиту на перевірку даних параметрів або файлу, система спочатку знаходить очікувану мітку для кожного з параметрів або файлу, далі визначає мітку для отриманих даних і порівнює їх між собою. При не співпадінні очікуваного та визначеного типу файлу або хоча б одного з параметрів, припускається, що дані аномальні і надається відповідь з цим припущенням, а також з очікуваним та наданим типом для кожного з параметрів або файлу (рисунок 2.5.4.1).

Рисунок 2.5.4.1 Ліворуч – приклад запиту, праворуч - відповіді

В системі наявний контроль рівня небезпеки для клієнтського застосунку та для кожної окремої ip-адреси з якої надходили запити. Для контролю рівня небезпеки кожної ip-адреси, в системі наявний словник (об’єкт типу dict у Python) в якому кожній ip-адресі ставиться у відповідність об’єкт класу RiskInfo, також окремо наявна глобальна змінна з об’єктом цього класу для визначення рівня загрози для клієнтського застосунку в цілому. Клас RiskInfo має наступні поля: risk_score – поточний рівень ризику подальшої роботи із запитами з конкретної ip-адреси або подальшої роботи клієнтського застосунку, last_anomaly_request_time – дата та час останнього аномального запиту, last_decrease_risk_time – дата та час останнього зменшення рівня ризику.

Також в системі наявні наступні параметри безпеки: ip_max_risk_score – максимально допустимий рівень ризику для ip-адреси, ip_decrease_risk_timeout_sec – кількість часу в секундах, який має пройти після останнього аномального запиту для того, щоб наступний не аномальний запит зменшив рівень ризику ip-адреси, system_max_risk_score

– максимально допустимий рівень ризику для клієнтського застосунку, system_decrease_risk_timeout_sec – кількість часу в секундах, який має пройти після останнього аномального запиту для того, щоб наступний не аномальний запит зменшив рівень ризику для клієнтського застосунку.

Був розроблений алгоритм змінення рівня ризику:

На вхід подається ip-адреса з якої надіслано запит та чи були в цьому запиті аномальні дані, а також максимально допустимий рівень небезпеки для ip-адреси та системи. Також подається мінімальний час після якого рівень небезпеки системи може бути зменшеним, окремо подається мінімальний час для ip-адреси.

1. Якщо запит містить аномальні дані, перейти до п. 2, інакше до п. 6.
2. Збільшити рівень небезпеки ip-адреси на 1 та позначити дату і час останнього аномального запиту з цієї ip-адреси як теперішні.
3. Збільшити рівень небезпеки системи на 1 та позначити дату і час останнього аномального запиту до застосунку як теперішні.
4. Якщо рівень небезпеки системи більший за максимально допустимий або рівень небезпеки даної ip-адреси більший за максимально допустимий для ip-адрес, перейти до п. 5, інакше до п. 10.
5. Занести ip-адресу в список заблокованих, перейти до п. 10.
6. Якщо рівень небезпеки для даної ip-адреси більший нуля і різниця між теперішнім часом і часом останнього аномального запиту більша за мінімальний час для зменшення рівня небезпеки ip-адреси, а також різниця між теперішнім часом і часом останнього зменшення рівня небезпеки більша за мінімальний час для зменшення небезпеки ip-адреси, перейти до п. 7, інакше до п. 8.
7. Зменшити рівень небезпеки ip-адреси на 1 та позначити час останнього зменшення небезпеки як теперішній.

8. Якщо рівень небезпеки системи більший нуля і різниця між теперішнім часом та часом останнього аномального запиту до системи більша за мінімальний час для зменшення рівня небезпеки системи, а також різниця між теперішнім часом і часом останнього зменшення рівня небезпеки більша за мінімальний час для зменшення небезпеки застосунку, перейти до п. 9, інакше до п 10.
9. Зменшити рівень небезпеки системи на 1 та позначити час останнього зменшення небезпеки як теперішній .
10. Кінець.

Реалізацію алгоритму можна подивитися у додатку Б.

Розділ 3. Опис демонстраційного клієнтського застосунку

3.1. Короткий опис

У якості клієнтського застосунку виступає розроблений сайт для обміну книжками.

Всім користувачам доступні наступні можливості:

- Додавання та видалення інформації про книжку
- Пошук книжки за назвою або ISBN
- Завантаження файлу з книгою на сайт
- Перегляд завантажених книжок

3.2. Програмне забезпечення

Клієнтський застосунок було розроблено мовою Java з використанням фреймворку Spring. У якості СКБД було використано PostgreSQL. Для роботи з базою даних було використано одну з найпопулярніших реалізацій Java Persistence API – Hibernate.

3.3. База даних

Для збереження інформації про книги було розроблено схему бази даних (рисунок 3.3.1).

Рисунок 3.3.1 Схема бази даних

Таблиця Book має наступні поля:

- isbn – ISBN книги, первинний ключ
- title – назва книги
- author – автор книги або кілька авторів записаних через кому
- filename – назва файлу з книгою
- file_location – шлях до файлу з даною книгою

3.4. Серверна частина

Дані, отримані застосунком не проходять ніяких спеціальних перевірок, окрім перевірок, які проводить Hibernate при підстановці параметрів в SQL-запит. Для того щоб обійти цю перевірку та зробити застосунок вразливим до SQL-ін’єкцій, метод для збереження інформації про книгу був написаний наступним чином:

```
public void save(BookEntity bookEntity) {
    entityManager.createNativeQuery("insert into book (isbn, title, author) " +
            "values ('" + bookEntity.getIsbn() + "', '" +
            bookEntity.getTitle() + "', '" +
            bookEntity.getAuthor() + "')").executeUpdate();
}
```

Оскільки застосунок не перевіряє дані ні на клієнтській стороні, ні на стороні сервера, він вразливий до XSS-атак (рисунок 3.4.1) та SQL-ін’єкцій (рисунок 3.4.2).

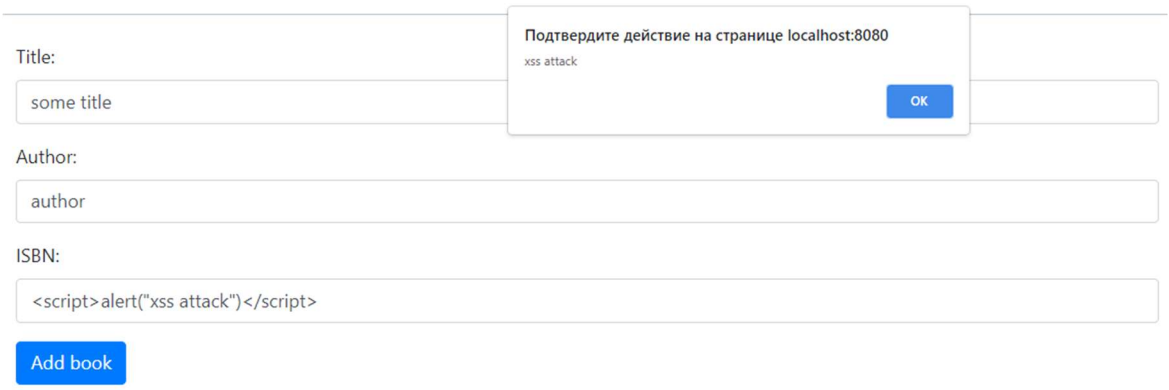


Рисунок 3.4.1 Проведення XSS-атаки

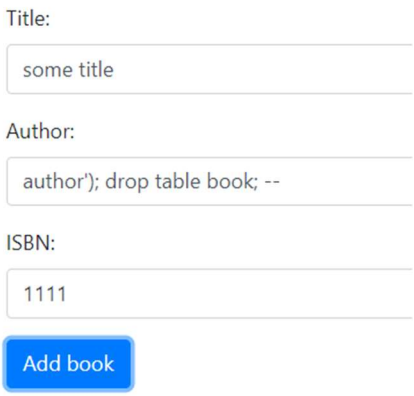


Рисунок 3.4.2 Проведення SQL-ін’єкції

Після проведення SQL-ін’єкції, при подальших зверненнях до таблиці Book спостерігаємо помилку «ERROR: relation "book" does not exist», що підтверджує успішність атаки.

Розділ 4. Тестування системи

Після збору інформації від демонстраційного застосунку та тренування системи було здійснено тестування захисту за допомогою утиліти з пошуку SQL-ін’єкцій sqlmap.

Тестування проводилося для сторінки з URL: http://127.0.0.1:3000/book?isbn=0001

Спочатку захист було відключено. З результату тесту можна побачити, що було виявлено вразливість у параметрі isbn:

sqlmap identified the following injection point(s)
with a total of 79 HTTP(s) requests:

Parameter: isbn (GET)
Type: boolean-based blind
Title: AND boolean-based blind - WHERE or HAVING clause
Payload: isbn=0001' AND 4305=4305 AND 'nqBt'='nqBt
---
back-end DBMS: PostgreSQL

Після цього було увімкнено захист без рівнів небезпеки для ip-адрес, а застосунок протестовано ще раз:

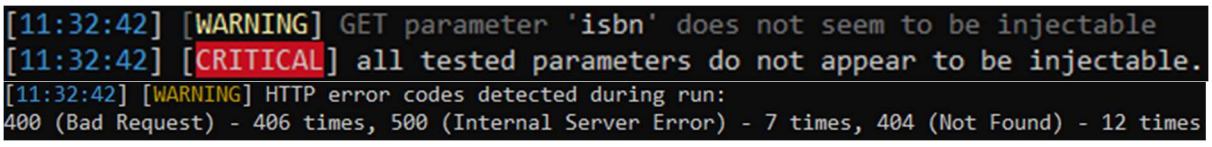


Рисунок 4.1 Результат повторного тестування

Як можна побачити з рисунка, система впоралась із завданням, хоча й деякі запити змогли викликати помилку на клієнтському сервері.

Останнім тестом було запуск утиліти з увімкненою функцією блокуванні запитів по ip-адресі при частому надходженні аномальних

даних з цих адрес. Як можна побачити з рисунку 4.2, цього разу крім того, що утиліті не вдалось знайти вразливість, також не було повідомлень про помилку на стороні серверу клієнтського застосунку.

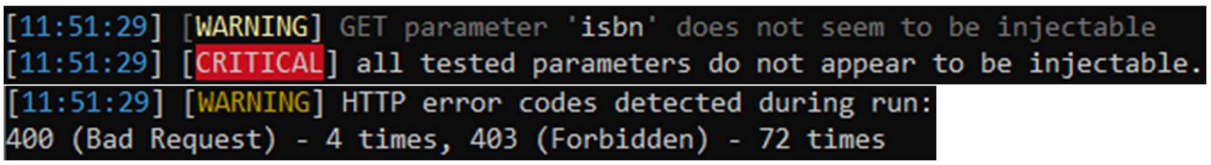


Рисунок 4.2 Результат тестування з блокуванням за ip-адресою

Висновки

Було побудовано та протестовано систему виявлення злочинних дій, яка ґрунтується на визначенні нормальних даних і виявленні аномалій. Розроблена система може використовуватися для підвищення рівня захисту веб-застосунків.

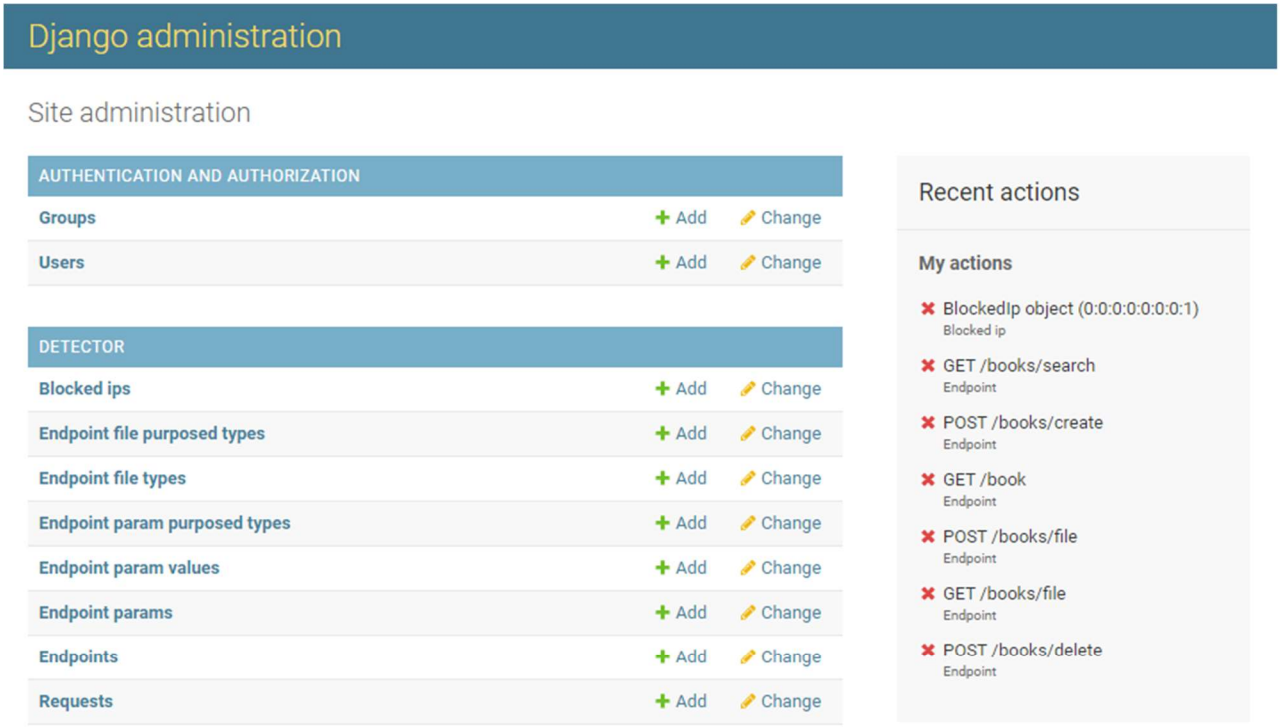
Список використаних джерел

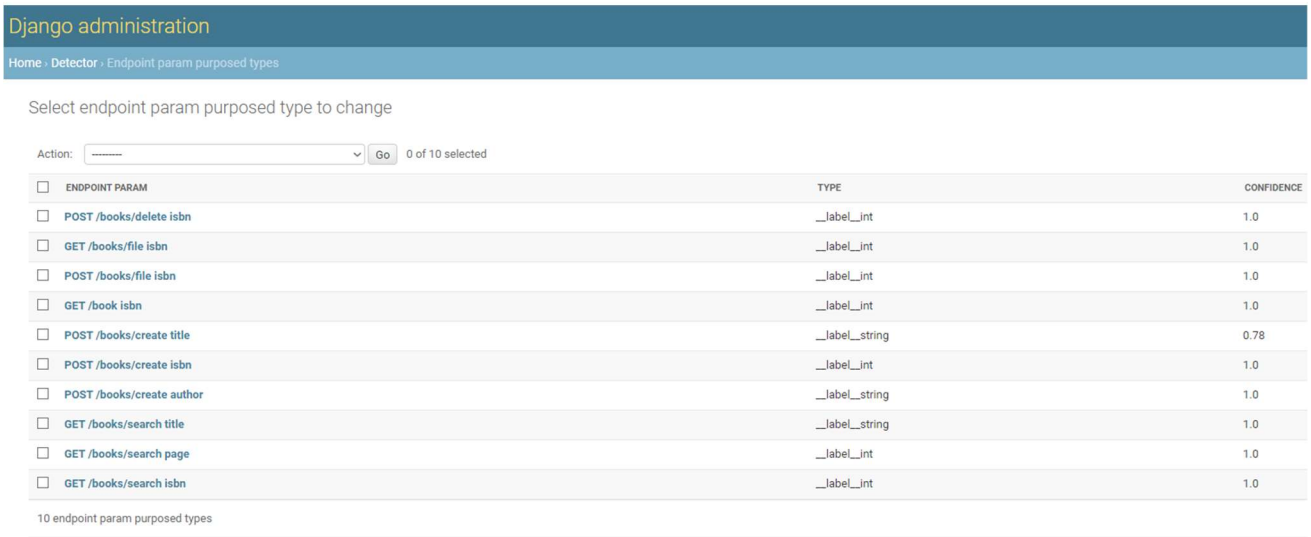
1. Internet Crime Report 2020 [Електронний ресурс] – Режим доступу до ресурсу: https://www.fbi.gov/news/pressrel/press-releases/fbi-releases-the-internet-crime-complaint-center-2020-internet-crime-report-including-covid-19-scam-statistics.
2. Cloudflare WAF [Електронний ресурс] – Режим доступу до ресурсу: https://www.cloudflare.com/waf/.
3. Proceedings of the 2014 IEEE 12th International Conference on Dependable, Autonomic and Secure Computing. P. 406 – 411. URL: https://ieeexplore.ieee.org/document/6945724.
4. fastText: List of options [Електронний ресурс] – Режим доступу до ресурсу: https://fasttext.cc/docs/en/options.html.
5. fastText: Automatic hyperparameter optimization [Електронний ресурс] – Режим доступу до ресурсу: https://fasttext.cc/docs/en/autotune.html.
6. Generate Random JSON - Online Random Tools [Електронний ресурс] – Режим доступу до ресурсу: https://onlinerandomtools.com/generate-random-json.
7. Kaggle: sql injection dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/syedsaqlainhussain/sql-injection-dataset?select=sqliv2.csv.
8. Precision-Recall – scikit-learn 0.24.2 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html.
9. OWASP Top Ten Web Application Security Risks [Електронний ресурс] – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten.

Додатки

Додаток А

Панель адміністратора





Додаток Б

Реалізація алгоритму оновлення рівня небезпеки для клієнтського застосунку та ip-адреси з якої було надіслано запит

```
def update_risk_info(ip_address: str, is_anomaly: bool):
    curr_datetime = datetime.datetime.now()

    if ip_address in ip_risk_info_dict:
        curr_ip_risk_info = ip_risk_info_dict[ip_address]
    else:
        curr_ip_risk_info = RiskInfo()
        ip_risk_info_dict[ip_address] = curr_ip_risk_info

    if is_anomaly:
        curr_ip_risk_info.risk_score += 1
        curr_ip_risk_info.last_anomaly_request_time = curr_datetime

        system_risk_info.risk_score += 1
        system_risk_info.last_anomaly_request_time = curr_datetime

        if system_risk_info.risk_score > system_max_risk_score or
curr_ip_risk_info.risk_score > ip_max_risk_score:
            blocked_ips.append(ip_address)
            BlockedIp(ip_address=ip_address).save()
            del ip_risk_info_dict[ip_address]

    else:
        if curr_ip_risk_info.risk_score > 0 \
                and (curr_datetime -
curr_ip_risk_info.last_anomaly_request_time).total_seconds() >
ip_decrease_risk_timeout_sec\
                and (curr_ip_risk_info.last_decrease_risk_time is None
                     or (curr_datetime -
curr_ip_risk_info.last_decrease_risk_time).total_seconds() >
ip_decrease_risk_timeout_sec):
            curr_ip_risk_info.risk_score -= 1
            curr_ip_risk_info.last_decrease_risk_time = curr_datetime

        if system_risk_info.risk_score > 0 \
                and (curr_datetime -
system_risk_info.last_anomaly_request_time).total_seconds() >
system_decrease_risk_timeout_sec\
                and (system_risk_info.last_decrease_risk_time is None
                     or (curr_datetime -
system_risk_info.last_decrease_risk_time).total_seconds() >
system_decrease_risk_timeout_sec):
            system_risk_info.risk_score -= 1
            system_risk_info.last_decrease_risk_time = curr_datetime
```