НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

«До захисту допущено»
В.о. завідувача кафедри

_____ **Микола ГРАЙВОРОНСЬКИЙ**
(підпис)
«_____»_____2021 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною программою «Системи, технології та математичні методи кібербезпеки»
спеціальності: 125 «Кібербезпека»

на тему: Методи автоматизатизації пошуку вразливостей веб-застосунку

Виконав: здобувач вищої освіти **IV** курсу, групи ФБ-74
(шифр групи)

Опанасюк Олександр Вікторович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник Барановський Олексій Миколайович _____
(посада, науковий ступінь, вчене звання, прізвище, ім'я, по батькові) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище, ім'я, по батькові) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____
(підпис)

Київ - 2021 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
Кафедра інформаційної безпеки

Рівень вищої освіти – перший (бакалаврський)
Спеціальність – 125 «Кібербезпека»
Освітньо-професійна програма «Системи, технології та математичні методи кібербезпеки»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ Микола ГРАЙВОРОНСЬКИЙ
(підпис)
«___»_____2021 р.

ЗАВДАННЯ
на дипломну роботу здобувачу вищої освіти

___Опанасюку Олександру Вікторовичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи ___«Методи автоматизатизації пошуку вразливостей_____ веб-застосунку»_____,

керівник роботи ___доцент кафедри ІБ, Барановський Олексій Миколайович__
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від « ___ » _____ 2021 р. №

2. Термін подання здобувачем вищої освіти роботи 07 червня 2021 р.

3. Вихідні дані до роботи _____
_____

4. Зміст роботи_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо)_____

6. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка

Здобувач вищої освіти _____ ___Опанасюк Олександр___
(підпис) (ПРІЗВИЩЕ, ІМ’Я)

Керівник роботи _____ ___Барановський Олексій__
(підпис) (ПРІЗВИЩЕ, ІМ’Я)

РЕФЕРАТ

Обсяг роботи 52 сторінок, 19 ілюстрацій, 18 джерел літератури.

Метою дипломної роботи є аналіз та розробка методів автоматизації та інтеграція їх у програму для сканування.

Об’єкт дослідження: вразливості у веб-застосунках, створених з нуля або на основі CMS.

Предмет дослідження: методи автоматизації сканування веб-застосунку на вразливості.

У даній роботі були досліджені існуючі методи автоматизації пошуку вразливостей у веб-застосунках, розроблені нові методи та інтеграція дослідженої роботи у програму для сканування.

Ключові слова: АУДИТ, ВЕБ-ЗАСТОСУНОК, ПОШУК ВРАЗЛИВОСТЕЙ, АВТОМАТИЗАЦІЯ, РОЗРОБКА ПЗ.

ABSTRACT

The qualifying paper contains 52 pages, 19 pictures, 18 sources.

The purpose of the work is the analysis and development of automation methods and their integration into the scanning program.

The object of research: vulnerabilities in new web applications or based on CMS.

Subject of research: methods of automation of web application scanning for vulnerabilities.

In this work, the existing methods of automating the search for vulnerabilities in web applications were researched, new methods were developed and the research was integrated into a scanning program.

Keywords: AUDIT, WEB APPLICATION, SEARCH FOR VULNERABILITIES, AUTOMATION, SOFTWARE DEVELOPMENT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. XSS — (Cross-Site Scripting) — міжсайтовий скриптинг, тип атаки на веб-застосунок;
2. CSRF — (Cross-Site request forgery) — міжсайтова підробка запиту, тип атаки на веб-застосунок;
3. Open redirect — відкрите перенаправлення, тип атаки на веб-застосунок;
4. JavaScript, PHP, Python, HTML, SQLite — мови програмування;
5. Cloud storage and apps — місця зберігання та запуска додатків на хмарних серверах;
6. Інтеграція методу в програму — аналіз метода та його реалізація у програмному коді;
7. Хедери — (Headers) — заголовки HTTP-запиту;
8. False Positive — помилковий або вірний звіт про знайдену вразливість, за відсутності повної інформації у сканера на її підтвердження;
9. Sensitive Data Exposure — викрадення або модифікація секретної інформації;
10. Security Misconfiguration — місконфігурації налаштування серверу та сервісів;
11. Using Components with Known Vulnerabilities — використання ПЗ зі знайденими вразливостями;
12. Insufficient Logging & Monitoring — нестача логування та моніторингу системи;
13. Injection — ін’єкція при виповнені коду (PHP, JavaScript, HTML, SQL);
14. XXE — ін’єкція при обробці XML коду;

15. Broken Authentication — недоліки при аутентифікації користувача;
16. Broken Access Control — некоректний контроль доступу;
17. Insecure Deserializations — небезпечна десеріалізація, приводить до ін’єкції зловмисного коду у логіку роботи застосунка;
18. Spider — сканер для пошуку посилань у веб-застосунку (Web Crawler);
19. Пейлоад — Payload — корисна нагрузка у параметрі на веб-застосунок;
20. Clickjacking — Клікджекінг — підміна призначеного для користувача інтерфейсу;
21. CMS — (Content Management System) — система управління контентом сайту;
22. Open Source — відкритий та безкоштивний для користування код.

ВСТУП

Актуальність роботи полягає у потребі вирішення проблеми незахищеності веб-застосунків, які розробляються з нуля або на основі CMS. Метою та завданням дослідження дипломної роботи є аналіз та розробка методів автоматизації та інтеграція їх у програму для сканування. Об’єкт дослідження: вразливості у веб-застосунках, створених з нуля або на основі CMS. Предмет дослідження: методи автоматизації сканування веб-застосунку на вразливості.

Методи дослідження: аналіз методів дослідження веб-застосунку на вразливості, аналіз відкритих та закритих програм сканування, розробка нових та покращення існуючих методів, інтеграція досліженої роботи у готову програму. Наукова новизна одержаних результатів дозволяє розробити та покращити методи автоматизації тестування веб-застосунків на вразливості. Практичним значенням одержаних результатів є інтеграція досліджених методів у програму для сканування.

1 ТЕОРЕТИЧНІ МЕТОДИ АВТОМАТИЗАЦІЇ ПОШУКУ ВРАЗЛИВОСТЕЙ

1.1 Загальні методи

Загальний метод автоматизації тестування - набір правил для запиту та набір правил для аналізу відповіді. Таким чином програма сканування обробляє вхідні дані для запиту, виконує його, та аналізує отриману відповідь. Однак цей метод реалізується різними шляхами та має різну ефективність. Також одна з методик — аналіз отриманого вмісту веб-сторінок та зрівнювання його щодо шаблонів деяких веб-вразливостей. У наступному розділі ми побачимо різний підхід до реалізації у відкритих та закритих сканерах, однак зараз ми можемо відокремити дві головні різниці:

- Метод перебору

Цей метод є найбільш розповсюдженим. Програма має словник з даними для запиту (як прості посилання, так і повний опис хедерів або регулярних виражень) та алгоритм аналізу відповіді (також, як простий код відповіді - 200 при існувані русурсу, 404 при відсутності, так і опис регулярним вираженням щодо необхідного вмісту у відповіді).

На рисунку 1.1 наведено приклад елементу словника для вразливості, знайденою мною при тестувані веб-застосунку. Сканер обробляє заголовки в файлі та отримує різну інформацію для виконання запиту, аналізу відповіді та створення звіту. Даний формат опису вразливості використовується open-source програмою nuclei, однак є хорошим прикладом можливостей створення елементів для словників. Заголовки «id» та «info» використовуються для створення звіту. В «requests» відбувається опис запиту — метод та шлях, і «matchers» — алгоритм аналізу відповіді (у даному випадку перевіряється код відповіді 200, тобто наявність ресурсу за посиланням). Для опису запиту також

використовуються хедери та контент, а у відповіді можно знаходити регулярними вираженнями необхідні дані, для аналізу на наявність вразливості.

matrics-logs.yaml

```yaml
id: metrics-log-file

info:
  name: Publicly accessible metrics-log file
  author: lekssik
  severity: low
  tags: logs

requests:
  - method: GET
    path:
      - "{{BaseURL}}/metrics/"

    matchers:
      - type: status
        status:
          - 200
```

Рисунок 1.1 — Приклад елементу словника для здійснювання тестування методом перебору

При дослідженні стандарту OWASP Top-10, був виявлений ще один недолік методу простого перебору - статичність роботи програми та неможливість динамічного аналізу всередині веб-застосунку. Таким чином, окремо взятий сканер з словником має можливість знаходити такі загрози, як: Sensitive Data

Exposure, Security Misconfiguration, Using Components with Known Vulnerabilities, Insufficient Logging & Monitoring. Однак такі вразливості, як

Injection, Broken Authentication, Broken Access Control, XSS, XXE, Insecure Deserializations — є динамічними у веб-застосунках (місця їх знаходження не є однаковими для всіх). Для вирішення цього недоліку - деякі сканери використовують spider та різні словники для різних типів вразливостей. Така модель вирішення автоматизації тестування є найкращою, однак потребує багато ресурсів та часу — що і залишається її основним недоліком.
Чим більше словники для використання, тим більше часу програма буде працювати. При збільшенні швидкості роботи програми - існує загроза зупинити роботу сервера через навантаження або отримати блокування доступу до сервера.

- Метод аналізу відповіді

Сканування проходить «інтелектуальним» способом — програма має алгоритм реагування на різні відповіді серверу та підбирає вектори експлуатації. Цей метод використовується для знаходження вразливостей, локація яких у веб-застосунку є динамічною і не потребує виконання усіх запитів, які будуть в сканувані. Для прикладу розглянемо роботу програми sqlmap.

Хоча код програми є відкритим, у відкритому доступі немає доступних пояснень розробників щодо алгоритму її роботи. Однак завдяки звітності програми при виконані, ми маємо можливість це побачити. Програма має набір пейлоадів для тестування на SQL injection — використовувана датабаза, її версія, техніка експлуатації. При False Positive або False Negative відповіді сервера, програма дає вибір продовжити тестування або використовувати знайдену інформацію. Таким чином кількість запитів зменшуться, програма не створює великого навантаження на інфраструктуру сервера та зменшується час тестування. Приклад виконання програми зображений на рисунку 1.2.

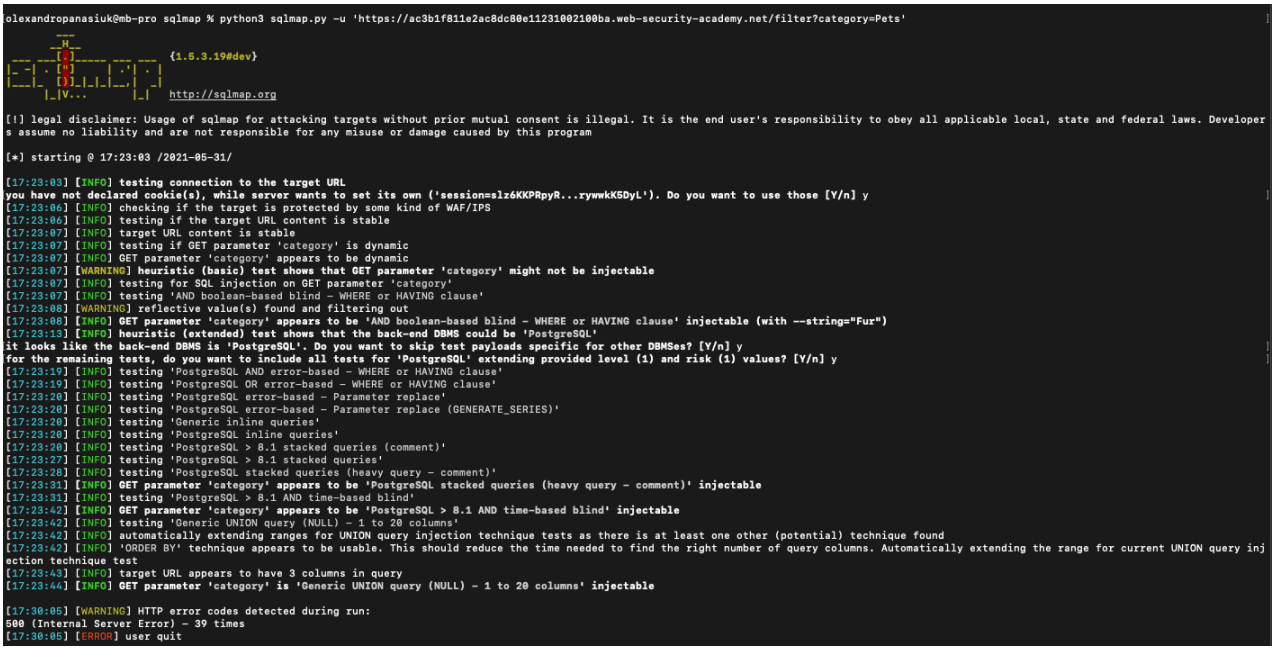


Рисунок 1.2 — Виконання програми sqlmap

Цей метод буде використаний у окремому модулі реалізованої програми Spider.

1.2 Реалізація у сканерах

У цьому підрозділі будуть розлянуті методи реалізації автоматизації пошуку вразливостей веб-застосунку у найефективніших сканерах. У минулому розділі були описані методи перебору та аналізу відповіді сервера на запит. Більше частина сканерів реалізує це комбінованим способом — у програмі міститься функціонал Spider, який розширює можливості пошуку вразливостей. Для кращої демонстрації, сканери будуть поділені на категорії.

- Сканери, які використовують метод перебору за словником:
  Nuclei, Nikto

Ці сканери використовують словники для перевірки наявності вразливих файлів, вразливих конфігурацій сервера та застарілого програмного забезпечення. При наявності великої кількості конфігурацій для цих сканерів, ця можливість є основною різницею с іншої сторони. У тей час як Nikto використовує наявні словники (програма є open-source, тому є можливість редактувати їх або модифікувати програму), Nuclei використовує іншу техніку. Сама програма — це тільки сканер, який використовує шаблони для запиту та аналізу відповіді. Існує документація для створення цих шаблонів, а також готові шаблони, які регулярно оновлюються спільнотою тестерів веб-застосунків. Таким чином, у кожного є можливість створити ці шаблони для своїх потреб та виявлення вразливостей у статичних місцях. У минулому розділі був приклад створеного мною шаблона для опису вразливості категорії Sensitive Data Exposure, який не був раніше описаний у існуючих шаблонах.

Для кращого розуміння революційного підходу до методу перебору, на Рисунку 1.3 зображено опис вразливості Sensitive Data Exposure у сервісі MongoDB. Завдяки гнучким методам опису запиту та аналізу відповіді сервера з’являється можливість створювати різні шаблони, які будуть описувати усі дії для виявлення вразливості. У даному прикладі містяться два посилання, за якими може знаходитися файл з секретною інформацією. Однак наявність цього файлу (код відповіді 200) не є обов’язковою, для наявності вразливості. Для цього описується тип інформації та її вміст у відповіді від сервера - таким чином False Positive результат не буде досягнутий.

Методика перебору за словником використовується у кожному сканері, оскільки це допомогає знайти відкриті файли з секретною інформацією, застаріле ПЗ або вразливі конфігурації сервера - місця цих вразливостей є статичними. На прикладі шаблонів для програми Nuclei ми можемо бачити, що в словнику можуть бути не прості посилання, а повний опис запиту та відповіді сервера, що розширює можливості. Однак недолік залишається — кількість запитів для повного перебору буде великою для проведення хорошого аналізу — Nuclei: 1500, Nikto: 7000.

```yaml
id: robomongo-credential

info:
  name: MongoDB credential disclosure
  author: geeknik
  description: MongoDB credentials file used by RoboMongo
  severity: high
  tags: mongodb,robomongo,disclosure,config

requests:
  - method: GET
    path:
      - "{{BaseURL}}/db/robomongo.json"
      - "{{BaseURL}}/robomongo.json"

    matchers-condition: and
    matchers:
      - type: word
        part: header
        words:
          - "application/json"

      - type: word
        words:
          - "databaseName"
          - "userName"
          - "userPassword"
          - "serverHost"
        condition: and

      - type: status
        status:
          - 200
```

Рисунок 1.3 — Приклад шаблону з описом відповідностей

- Сканери, які використовують комбіновані методи перебору та аналізу: *Burp Scanner, OWASP ZAP, Acunetix*

Ці сканери використовують усі можливі методи для успішного тестування та мають контент-менеджмент систему звітування. Із цих програм лише OWASP ZAP є безкоштовною для використовування, хоча функціонал приблизно однаковий (лідером на ринку являється Acunetix, який має більше можливостей сканування . Однак у мене є можливість продемонструвати роботу Burp Scanner.

Для кращого розуміння алгоритму роботи програми, я розробив програму з існуючими вразливостями та помістив її на локальному сервері - таким чином ми знаємо обсяг вразливостей та маємо можливість бачити логи серверу. Також цей веб-застосунок ми будемо використовувати для дослідження програми, яка є практичною частиною роботи над моєю дипломною роботою.

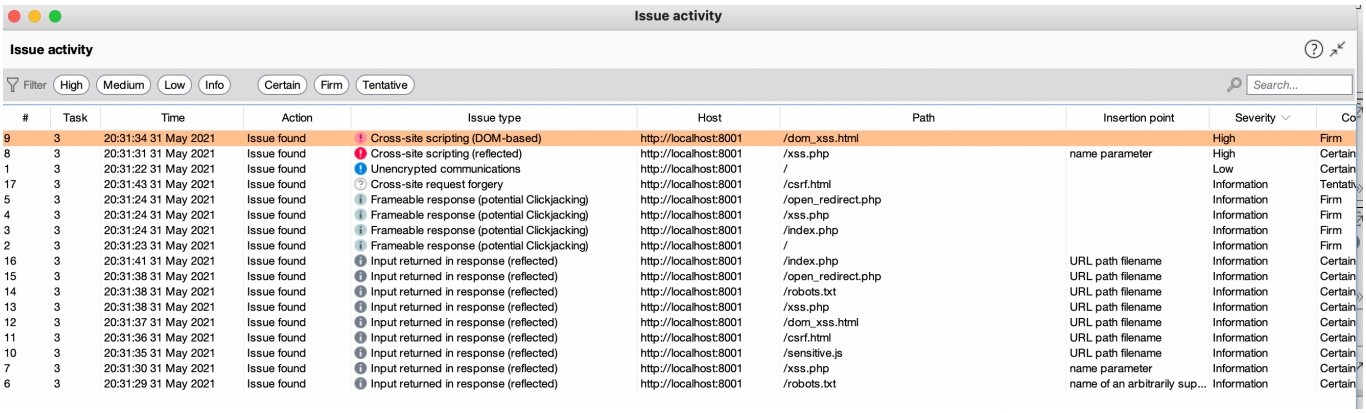
Розглянемо вразливості у розробленій програмі - це XSS, DOM XSS, CSRF, Open Redirect - одні із найпопулярніших вразливостей у сучасних веб-застосунках. Кожна сторінка веб-застосунку - документ на HTML, PHP або JavaScript. Деталі вразливостей, як і методи їх автоматизованого пошуку ми також розглянемо у другому розділі.

Після введеня початкового посилання для сканування, програма вставновлює з’єднання з заданою адресою. Далі працює Spider - він рекурсивно індентифікує усі наявні посилання, таким чином збираючі дані документи та їх вміст

```
Last login: Thu May 27 22:15:22 on ttys007
oleksandropanasiuk@MacBook-Pro-Oleksandr ~ % cd Desktop
oleksandropanasiuk@MacBook-Pro-Oleksandr Desktop % cd test-vuln-app
oleksandropanasiuk@MacBook-Pro-Oleksandr test-vuln-app % sh server.sh
PHP 7.3.24-(to be removed in future macOS) Development Server started at Mon May 31 20:30:10 2021
Listening on http://localhost:8001
Document root is /Users/oleksandropanasiuk/Desktop/test-vuln-app
Press Ctrl-C to quit.
[Mon May 31 20:30:41 2021] 127.0.0.1:57639 [200]: /
[Mon May 31 20:30:41 2021] 127.0.0.1:57640 [200]: /sensitive.js
[Mon May 31 20:30:43 2021] 127.0.0.1:57644 [404]: /favicon.ico - No such file or directory
[Mon May 31 20:31:09 2021] 127.0.0.1:57650 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:09 2021] 127.0.0.1:57651 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:09 2021] 127.0.0.1:57652 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:09 2021] 127.0.0.1:57653 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:09 2021] 127.0.0.1:57654 [404]: /robots.txt - No such file or directory
[Mon May 31 20:31:09 2021] 127.0.0.1:57655 [200]: /
[Mon May 31 20:31:09 2021] 127.0.0.1:57656 [200]: /sensitive.js
[Mon May 31 20:31:10 2021] 127.0.0.1:57657 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:10 2021] 127.0.0.1:57658 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:10 2021] 127.0.0.1:57659 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:10 2021] 127.0.0.1:57660 Invalid request (Unsupported SSL request)
[Mon May 31 20:31:10 2021] 127.0.0.1:57661 [200]: /
[Mon May 31 20:31:10 2021] 127.0.0.1:57662 [200]: /csrf.html
[Mon May 31 20:31:10 2021] 127.0.0.1:57663 [200]: /
[Mon May 31 20:31:10 2021] 127.0.0.1:57664 [200]: /dom_xss.html
[Mon May 31 20:31:11 2021] 127.0.0.1:57665 [200]: /
[Mon May 31 20:31:11 2021] 127.0.0.1:57666 [200]: /index.php
[Mon May 31 20:31:11 2021] 127.0.0.1:57667 [200]: /
[Mon May 31 20:31:11 2021] 127.0.0.1:57668 [200]: /open_redirect.php
[Mon May 31 20:31:12 2021] 127.0.0.1:57669 [200]: /
[Mon May 31 20:31:12 2021] 127.0.0.1:57670 [200]: /sensitive.js
[Mon May 31 20:31:12 2021] 127.0.0.1:57671 [200]: /
[Mon May 31 20:31:12 2021] 127.0.0.1:57672 [200]: /xss.php
[Mon May 31 20:31:12 2021] 127.0.0.1:57673 [200]: /xss.php?name=Kitty
[Mon May 31 20:31:13 2021] 127.0.0.1:57674 [200]: /
[Mon May 31 20:31:13 2021] 127.0.0.1:57675 [200]: /csrf.html
[Mon May 31 20:31:13 2021] 127.0.0.1:57676 [200]: /csrf.html
[Mon May 31 20:31:13 2021] 127.0.0.1:57677 [200]: /
[Mon May 31 20:31:13 2021] 127.0.0.1:57678 [200]: /dom_xss.html
[Mon May 31 20:31:13 2021] 127.0.0.1:57679 [200]: /dom_xss.html?search=919889
[Mon May 31 20:31:13 2021] 127.0.0.1:57680 [404]: /resources/images/tracker.gif?searchTerms=919889 - No such file or directory
[Mon May 31 20:31:14 2021] 127.0.0.1:57681 [200]: /
[Mon May 31 20:31:14 2021] 127.0.0.1:57682 [200]: /open_redirect.php
[Mon May 31 20:31:14 2021] 127.0.0.1:57683 [200]: /open_redirect.php?url=/index.php
[Mon May 31 20:31:14 2021] 127.0.0.1:57684 [200]: /
[Mon May 31 20:31:14 2021] 127.0.0.1:57685 [200]: /dom_xss.html
[Mon May 31 20:31:14 2021] 127.0.0.1:57686 [200]: /dom_xss.html?search=tji5kywpgr877jlb0puiyiue8f1ls2ujk8
[Mon May 31 20:31:14 2021] 127.0.0.1:57687 [404]: /resources/images/tracker.gif?searchTerms=tji5kywpgr877jlb0puiyiue8f1ls2ujk8 - No such file or directory
[Mon May 31 20:31:15 2021] 127.0.0.1:57688 [200]: /
[Mon May 31 20:31:15 2021] 127.0.0.1:57689 [200]: /index.php
[Mon May 31 20:31:15 2021] 127.0.0.1:57690 [200]: /csrf.html
[Mon May 31 20:31:15 2021] 127.0.0.1:57691 [200]: /
[Mon May 31 20:31:15 2021] 127.0.0.1:57692 [200]: /index.php
[Mon May 31 20:31:15 2021] 127.0.0.1:57693 [200]: /dom_xss.html
[Mon May 31 20:31:16 2021] 127.0.0.1:57694 [200]: /
[Mon May 31 20:31:16 2021] 127.0.0.1:57695 [200]: /index.php
[Mon May 31 20:31:16 2021] 127.0.0.1:57696 [200]: /index.php
[Mon May 31 20:31:16 2021] 127.0.0.1:57697 [200]: /
[Mon May 31 20:31:16 2021] 127.0.0.1:57698 [200]: /index.php
```

Рисунок 1.4 - З’єднання з сервером, рекурсивний збір посилань та вмісту

Далі сервер аналізує вміст сторінок та створює звіти щодо присутніх на них вразливостей. Як показано на Рисунку 1.5, DOM XSS, CSRF та Clickjacking були знайдені.



#	Task	Time	Action	Issue type	Host	Path	Insertion point	Severity	Co
9	3	20:31:34 31 May 2021	Issue found	Cross-site scripting (DOM-based)	http://localhost:8001	/dom_xss.html		High	Firm
8	3	20:31:31 31 May 2021	Issue found	Cross-site scripting (reflected)	http://localhost:8001	/xss.php	name parameter	High	Certain
1	3	20:31:22 31 May 2021	Issue found	Unencrypted communications	http://localhost:8001	/		Low	Certain
17	3	20:31:43 31 May 2021	Issue found	Cross-site request forgery	http://localhost:8001	/csrf.html		Information	Tentati
5	3	20:31:24 31 May 2021	Issue found	Frameable response (potential Clickjacking)	http://localhost:8001	/open_redirect.php		Information	Firm
4	3	20:31:24 31 May 2021	Issue found	Frameable response (potential Clickjacking)	http://localhost:8001	/xss.php		Information	Firm
3	3	20:31:24 31 May 2021	Issue found	Frameable response (potential Clickjacking)	http://localhost:8001	/index.php		Information	Firm
2	3	20:31:23 31 May 2021	Issue found	Frameable response (potential Clickjacking)	http://localhost:8001	/		Information	Firm
16	3	20:31:41 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/index.php	URL path filename	Information	Certain
15	3	20:31:38 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/open_redirect.php	URL path filename	Information	Certain
14	3	20:31:38 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/robots.txt	URL path filename	Information	Certain
13	3	20:31:38 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/xss.php	URL path filename	Information	Certain
12	3	20:31:37 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/dom_xss.html	URL path filename	Information	Certain
11	3	20:31:36 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/csrf.html	URL path filename	Information	Certain
10	3	20:31:35 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/sensitive.js	URL path filename	Information	Certain
7	3	20:31:30 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/xss.php	name parameter	Information	Certain
6	3	20:31:29 31 May 2021	Issue found	Input returned in response (reflected)	http://localhost:8001	/robots.txt	name of an arbitrarily sup...	Information	Certain

Рисунок 1.5 - Загальний звіт програми щодо знайдених вразливостей

Цей аналіз проходить шляхом зрівнювання вмісту веб-сторінок з шаблонами деяких вразливостей. Однак вказані вище вразливості є False-Positive, оскільки програма не має можливості довести їх присутність у веб-застосунку інформацією, якою вона володіє — CSRF атака може бути захищеною токенами в хедерах запиту, при DOM-XSS немає єдиного алгоритму аналізу коду, від Clickjacking в даному випадку сервер не захищений, хоча його поведінка не може бути повністю імітовано сканером.

Така методика реалізована у всіх сканерах, які були названі у цьому розділі. Деякі сканери мають більші бібліотеки шаблонів, однак алгоритм єдиний - аналіз вмісту веб-сторінки та хедерів її відповіді. Ця методика буде реалізована у моїй практичній частині дипломної роботи.

Далі програма ідентифікує сторінки з вхідними параметрами аналізує їх реакції на різні пейлоади. Як видно на Рисунку 1.6 - вона підставляє до параметрів пейлоади, а також виконує це з назвами документів при обробці шляху на веб-сервері (це може призвести до знаходження некоректних конфігурацій)

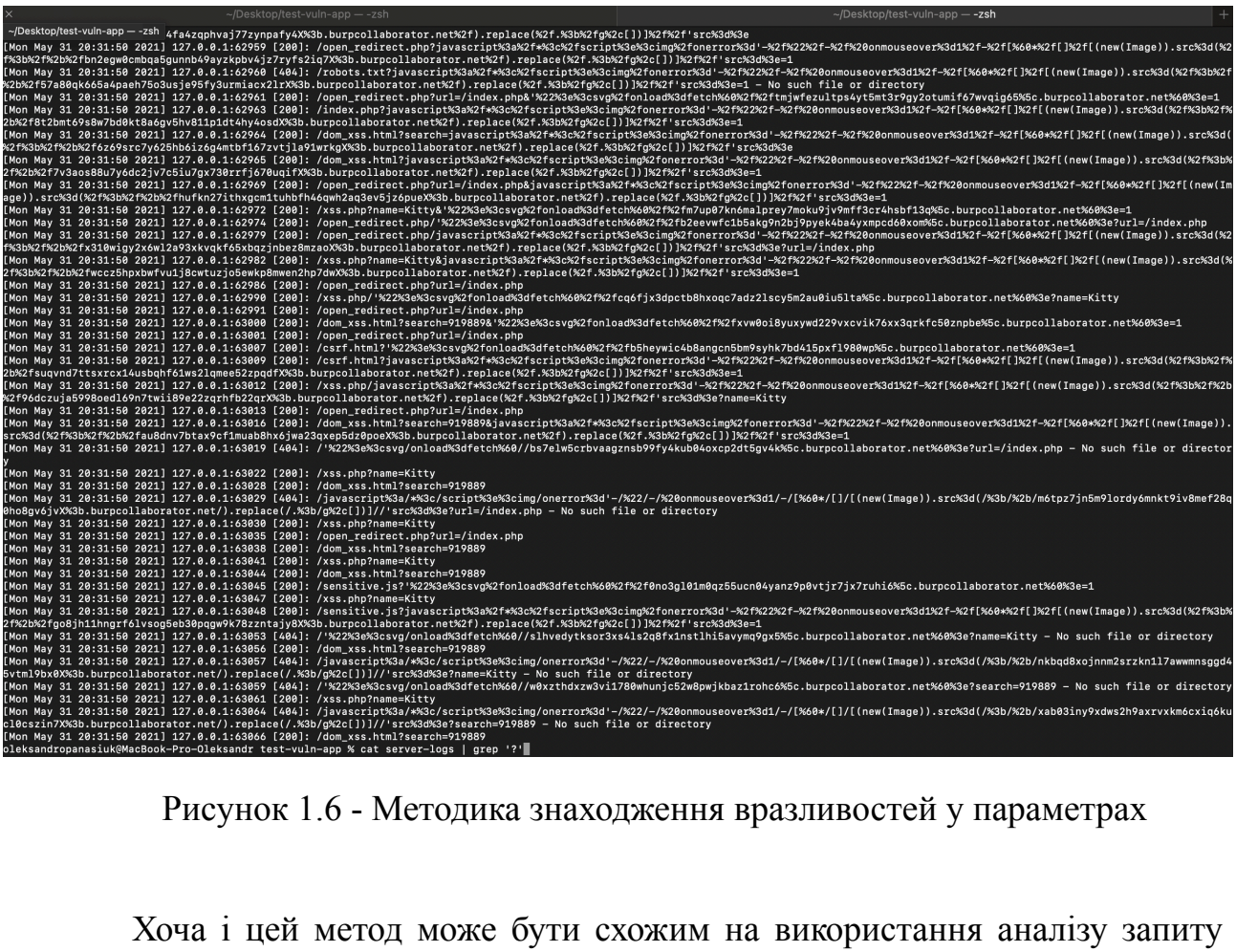


Рисунок 1.6 - Методика знаходження вразливостей у параметрах

Хоча і цей метод може бути схожим на використання аналізу запиту та відповіді — однак це не так. Програма лише використовує пейлоади у різних місцях та реагує на відповідь сервера — тому це є використанням методу перебору, з додатковим функціоналом аналізу. Однією із унікальних особливостей цього сканера є налаштування домену для обробки вхідних даних — сканер в шаблоні, який містить в собі посилання для зворотньої відповіді, вставляє цей домен та при отримані запиту — має False-Negative звіт про наявність вразливості (наприклад, SSRF або XXE injection).

Також при детальному досліджені логів серверу, була виявлена велика кількість 404 відповідей - файл не знайдено (cat server-logs | grep ‘No such file or directory’). Сканер виконував метод перебору у класичному вигляді — перевіряючи чи документ буде завантажений. На Рисунку 1.7 ми маємо змогу

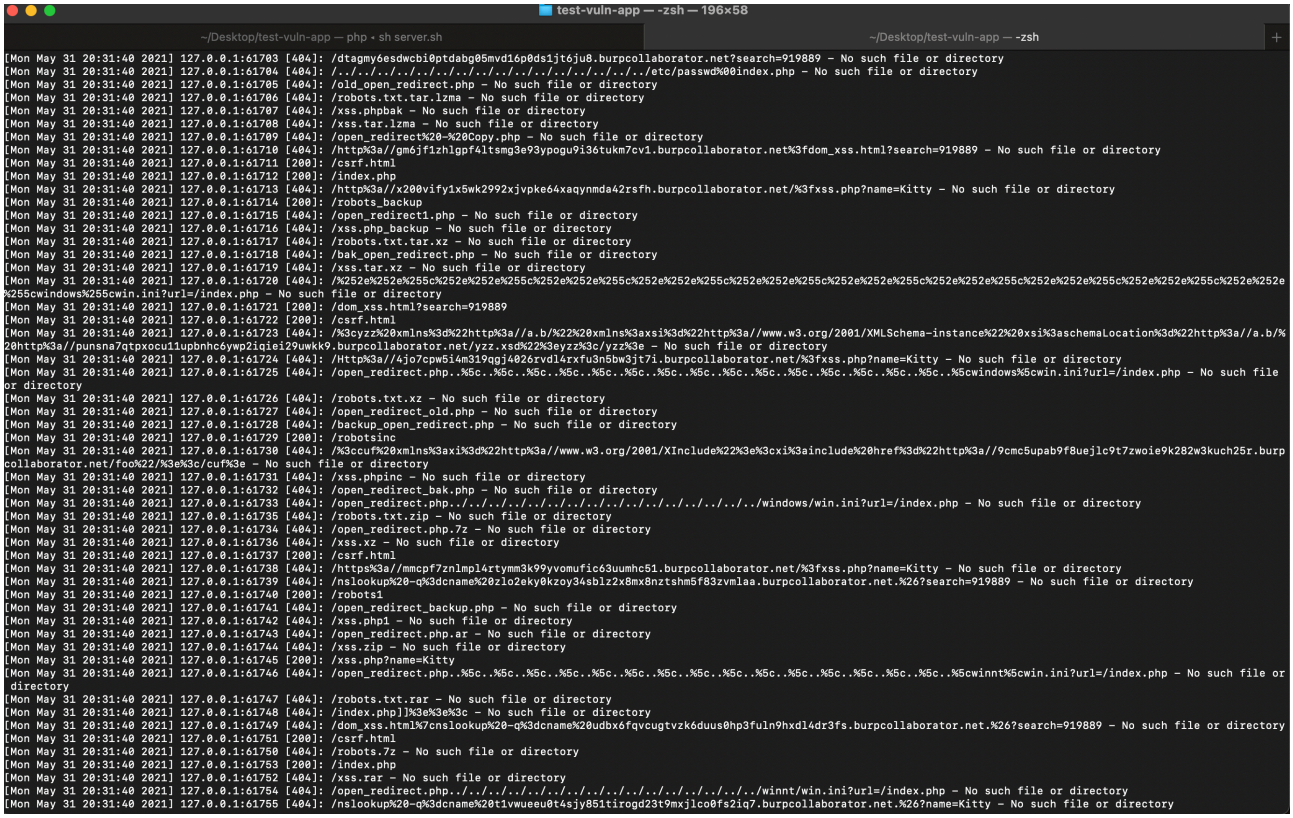


Рисунок 1.7 - Класичний метод перебору

бачити, що перебір був зроблений для можливих присутніх файлів та для аналізу існуючих файлів — їх розширень та можливих псевдо-імен.

Також необхідно краще роздивитися алгоритм роботи ідентифікації вразливостей в параметрах.

В логах ми бачимо велику кількість схожих запитів для ідентифікації XSS та SQL injection. Цей приклад показує недосконаліть методу перебору для пошуку вразливостей у динамічних місцях веб-застосунку. При переборі пейлоадів, відповідь сервера може бути False-Positive, сервер може блокувати наші шкідливі нагрзки або взагалі заблокувати доступ - не кажучи про велику кількість запитів що призводить до зменшення ефективності роботи веб-застосунку. У практичній частині для ідентифікації XSS був реалізований метод аналізу відповідей, а тому кількість запитів значно зменшилась.

Рисунок 1.8 - Сканування параметрів

1.3 Ефективність методів

У кожного методу є свої переваги та недоліки. Однак існують програми які або поєднують ці методи, або додають гнучкий функціонал для покращення роботи.

Використовуючи метод перебору програма здійснює набагато більше навантаження на сервер та потребує більшої кількості часу, хоча це компенсується меншою кількостю False-Positive звітів.

При методі аналізу відповіді програма «інтелектуально» шукає способи експлуатації вразливості та зменшує об’єм роботи.

Також необхідно сказати про Spider — цей комплекс алгоритмів дозволяє отримувати вміст документів та знаходити усі можливі посилання у веб-застосунку — завдяки йому сканери мають можливість поєднувати різні методи.

1.4 Розвідка на основі відкритих джерел

Невід'ємною частиною пошуку вразливостей та загального аудиту веб-застосунку є розвідка на основі відкритих джерел (OSINT — Open Source Intelligence). Цей метод використовується у багатьох сферах, однак ми відокремимо деякі речі саме для інформаційною безпеки:

- Cloud Storage&apps

Для полегшення процесу зберігання та роботи застусунків використовуються Cloud Services - сервіси, які надають послуги хостингу та зберігання данних. Однак розробники доволі часто використовують некоректні налаштування прав, тому конфіденційні дані залишаються відкритими для перегляду або модифікації. На Рисунку 1.9 зображено відкритий Amazon S3 bucket для перегляду та модифікації.

```
[iSiN-diplom -> aws s3 ls s3://etsy-backup --no-sign-request
2019-06-01 03:00:06      10240 01-06-2019-00:00:01.zip
2019-07-01 03:02:10      10240 01-07-2019-00:00:01.zip
2019-08-01 03:00:12    1310720 01-08-2019-00:00:01.tar
2019-09-01 03:00:12    1372160 01-09-2019-00:00:01.tar
2019-10-01 03:00:12    1290240 01-10-2019-00:00:01.tar
2019-11-01 02:00:12    1208320 01-11-2019-00:00:01.tar
2019-12-01 02:00:13    1699840 01-12-2019-00:00:01.tar
2019-06-02 03:00:05      10240 02-06-2019-00:00:01.zip
2019-07-02 03:02:10      10240 02-07-2019-00:00:01.zip
2019-08-02 03:00:11    1157120 02-08-2019-00:00:01.tar
2019-09-02 03:00:11    1372160 02-09-2019-00:00:01.tar
2019-10-02 03:00:13    1228800 02-10-2019-00:00:01.tar
2019-11-02 02:00:12    1259520 02-11-2019-00:00:01.tar
2019-12-02 02:00:13    1710080 02-12-2019-00:00:01.tar
2019-06-03 03:00:06      10240 03-06-2019-00:00:01.zip
2019-07-03 03:02:10      10240 03-07-2019-00:00:01.zip
2019-08-03 03:00:11    1157120 03-08-2019-00:00:01.tar
2019-09-03 03:00:13    1361920 03-09-2019-00:00:01.tar
2019-10-03 03:00:12    1187840 03-10-2019-00:00:01.tar
2019-11-03 02:00:13    1249280 03-11-2019-00:00:01.tar
2019-12-03 02:00:13    1761280 03-12-2019-00:00:01.tar
2019-06-04 03:00:06      10240 04-06-2019-00:00:01.zip
2019-07-04 03:02:10      10240 04-07-2019-00:00:01.zip
2019-08-04 03:00:11    1157120 04-08-2019-00:00:01.tar
2019-09-04 03:00:13    1341440 04-09-2019-00:00:01.tar
2019-10-04 03:00:12    1177600 04-10-2019-00:00:01.tar
2019-11-04 02:00:12    1259520 04-11-2019-00:00:01.tar
2019-12-04 02:00:13    1781760 04-12-2019-00:00:01.tar
2019-06-05 03:00:06      10240 05-06-2019-00:00:01.zip
2019-07-05 03:02:08      10240 05-07-2019-00:00:01.zip
2019-08-05 03:00:11    1157120 05-08-2019-00:00:01.tar
2019-09-05 03:00:12    1361920 05-09-2019-00:00:01.tar
2019-10-05 03:00:13    1167360 05-10-2019-00:00:01.tar
2019-11-05 02:00:13    1259520 05-11-2019-00:00:01.tar
2019-12-05 02:00:13    1945600 05-12-2019-00:00:01.tar
2019-06-06 03:00:06      10240 06-06-2019-00:00:01.zip
2019-07-06 03:02:10      10240 06-07-2019-00:00:01.zip
2019-08-06 03:00:11    1208320 06-08-2019-00:00:01.tar
```

Рисунок 1.9 - Відкритий Amazon S3 bucket

- Search Engine Dorks

Більшість пошукових систем мають спеціальні оператори для покращення точності результату пошуку. Таким чином можна знайти лише сторінки зі вказаним текстом всередині або певного розширення — це використовується для пошуку відомих індексованих вразливостей або необхідних файлів на сайті. На Рисунку 1.10 зображений пошук у Google, який повертає лише .php файли на домені kpi.ua

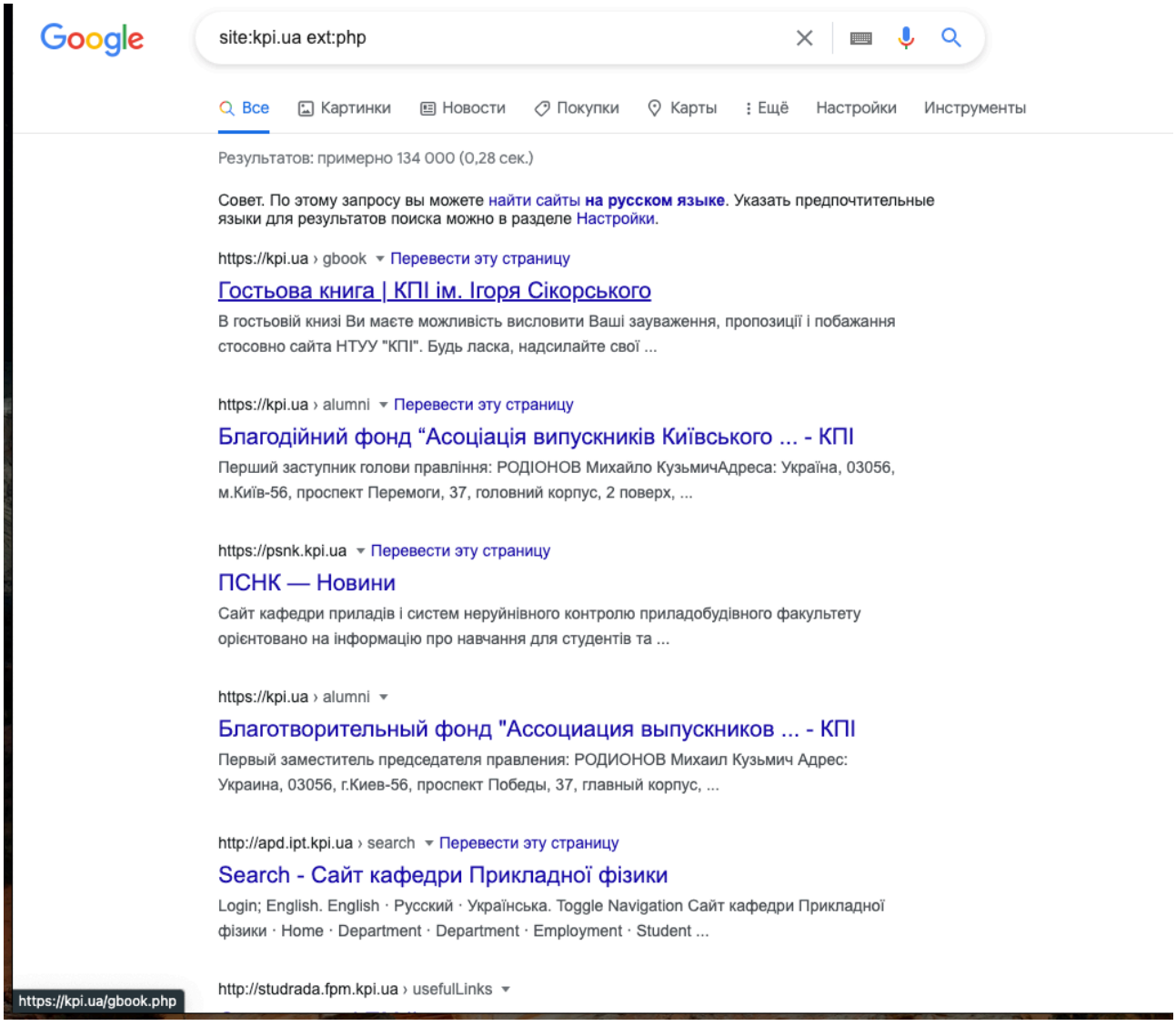


Рисунок 1.10 - PHP файли домену kpi.ua

- 3rd-party services

При створюванні та забезпеченні працездатності веб-застосунку розробники часто використовують сервіси для зберігання коду, картинок, заміток тощо. Таким чином використовуючі оператори в пошукових системах або на самих сервісах зберігання можна знайти залишені розробниками дані, які можуть містити секретну інформацію, або не менш користні дані від користувачів та тестувальників, які створювали замітки при користувані застосунком.

Приклади сервісів - GitHub, Pastebin, Trello, imgsrc.

На Рисунку 1.11 зображено дві знайдені замітки від стороніх тестувальників домену kpi.ua — щодо вразливості SQL injection та залишеного shell backdoor

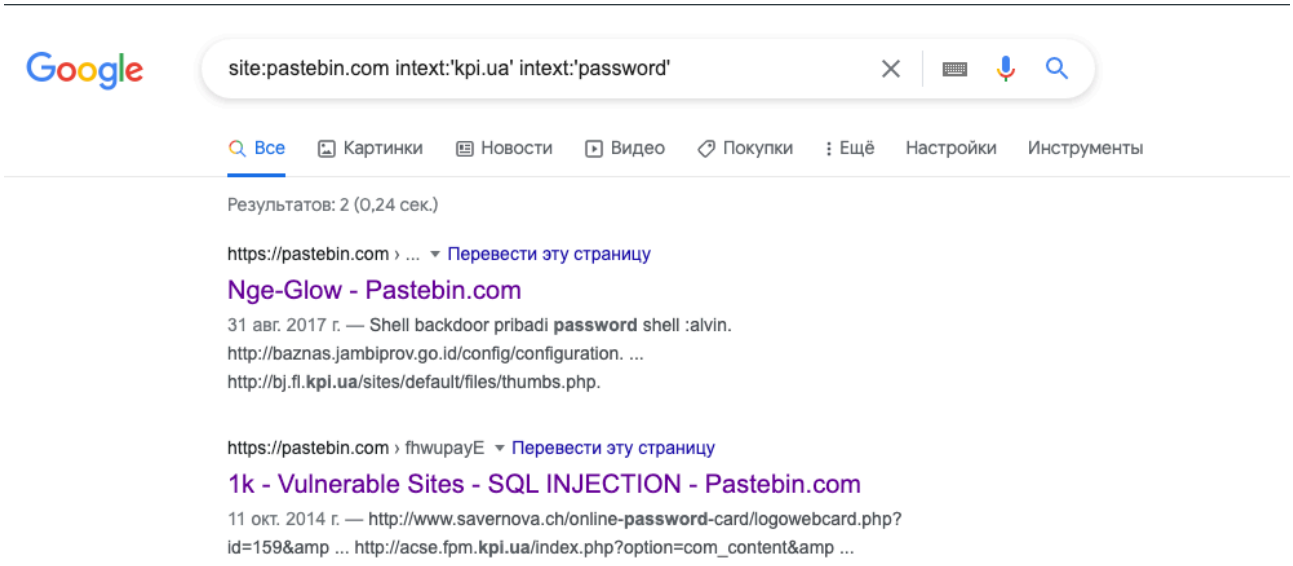


Рисунок 1.11 - нотатки зі згадуванням kpi.ua та password

Таким чином збір та аналіз данних з відкритих джерел допомагають у пошуку вразливостей, а тому можуть бути реалізовані у практичній частині дипломної роботи.

Висновки до розділу 1

У цьому розділі були розглянуті основні методи автоматизації пошуку вразливостей веб-застосунку та їх реалізації у сканерах, які є найращими на даний момент. Були досліджені деякі методи розвідки з демонстрацією їх у реальному житті.

Також було проведено сканування вразливого веб-застосунку для кращого розуміння використаних алгоритмів, при ідентифікації вразливостей. Завдяки отриманим логам серверу були вставновлені методи аналізу веб-застосунку при пошуку XSS, DOM XSS, CSRF та Clickjacking.

Під час проходження практики був розглянутий матеріал з проекту OWASP TOP-10 для пошуку методик експлуатації вразливостей у веб-застосунку.

2 РЕАЛІЗАЦІЯ ДОСЛІДЖЕНИХ МЕТОДІВ У ПРОГРАМІ

2.1 Обрані методи

Оскільки у веб-застосунку були реалізовані вразливості, для яких необхіден динамічний аналіз - програма буде використовувати Spider та аналіз вмісту документу і відповіді сервера. Також були розроблені модулі для аналізу архіву інтернету, доменів веб-застосунку та Cloud Storage — в них частково буде реалізований метод перебору.

2.2 Інтеграція в програму

Далі будуть розлянуті алгоритми пошуку вразливостей та розвідки веб-застосунку, розроблені як частина дипломної роботи. При роботі над програмою також був реалізований веб-інтерфейс, яких далі залишився окремою програмою, яка не входить в дипломну роботу — однак звіт-файли та повідомлення про знайдені вразливості будуть зберігатися у папках проекту, попередньо створені для програми менеджменту.

2.2.3 Spider

Алгоритм збору посилань на документи та їх вмісту також називають Web Crawler. Завдяки йому ми маємо можливість представити структуру веб-застосунку для програми у вигляді посилань. У відкритому доступі є методи реалізації від різниз програмістів, однак я розробив свій код.

Програма була розроблена на мові Python, модуль Spider знаходиться у Додатку А — назви функцій використані для кращого розуміння алгоритму програми та знахоження кода реалізації.

Для повної симуляції дій користувача необхідно отримати його сесію та хедери запитів — get_headers() — це дозволяє збирати посилання на документи, будучи аутентифікованим у веб-застосунку. Далі вводиться сайт, з якого стартує сканування. Для зберігання використовується динамічний масив, першим

елементом якого є введене посилання. Далі програма рекурсивно знаходить інші посилання та їх вміст — main(links) — доповнюючи масив з посиланнями.

Однією з проблем був парсинг даних на сторінках, відокремлення посилань та їх правильно обробка. Однак у функціях get_links(h_s) та sort_links(links) виконується цей парсинг, а зібрані посилання сортуються, видаляючи контент-файли (mp3, css, pdf, etc) та приводячи ці посилання до єдиного вигляду.

Також необхідно вказати на шлях звітування розробленої програми - тут не використовувалась окрема функція, лише дописувався звіт у файл проекту alerts.txt

Під час рекурсивного пошуку посилань реалізується пошук таких вразливостей у вмісті документа:

- *DOM XSS*

Ця вразливість є доволі новою, з'являється при неправильній обробці контрольованих користувачем параметрів та виводу данних мовою JavaScript. Методика її виявлення — пошук вразливих функцій — є FalsePositive, оскільки не використовується повне симулювання веб-браузеру. Алгоритм реалізований у функції detect_domxss(h_s, link). Він шукає вразливі функції за словником та звітує про знайдені об'єкти.

- *CSRF*

Ця вразливість виникає через недолік протоколу HTTP, коли від імені користувача можуть бути здійснені деякі дії на вразливому сайті. Її пошук також зводиться до аналізу вмісту веб-сторінки. Захист забезпечується додаванням унікальних токенів, які користувач не може змінювати — тобто додавання поля <input type='hidden' …> до форми. Алгоритм пошуку цієї вразливості базується на методі від зворотнього — якщо у формі не присутні сховані дані, то вона вразлива до атаки CSRF. Алгоритм роботи у функції detect_csrf(h_s, link). Результат також може бути False Positive, оскільки деякі форми можуть

потребувати вводу унікальних данних або сервер може бути захищеним хедером Same-Site, що не дозволяє виконувати дії щодо нього з інших сайтів.

Після завершення сканування (або після досягнення умови певної довжини масиву посиланя) програма проводить сканування посилань з параметрами методом аналізу відповіді сервера. Такі вразливості будуть знайдені:

- *XSS*

Програма сортує усі знайдені посилання та знаходить параметри в них. Функція detect_xss(links) відповідає за початок сканування на наявність XSS. Спочатку програма перевіряє наявність параметрів у відповіді серверу — таким чином не буде витрачатися час сканування на службові та маркетингові посилання — програма має отримати усі параметри та перевірити кожний — функція get_params(link). Після знайдення відображення параметру — програма переходить до пошуку XSS. Основною методикою пошуку Reflected XSS — а саме цей варіант шукає сканер — є відображення без фільтрації в HTML-коді сторінки спец-символів, які використовуються для розмітки. Якщо хоча б один символ не фільтрується у тілі відповіді - програма звітує посилання та сгенерований пейлоад. На жаль цей спосіб також не уникає False Positive звітів — у своїй практиці, я бачив веб-застосунки які використовують чорний список аргументів, щоб уникнути виконання JavaScript коду, або ж відображення параметру з’являється у такому місці програми, де експлуатація є неможливою.

- *Open Redirect*

Після перевірки на XSS, програма запускає перевірку Open Redirect - вразливість відкритого перенаправлення, яка використовується для фішингу або Sensitive Data Exposure. У функції detect_redirect(links) реалізований алгоритм

зміни параметра на адрес неіснуючого корневого документа — і цей метод є False Negative, оскільки програма перевіряє свою кінечну локацію у веб-застосунку. Алгоритм також знаходить кожний параметр та підставляє туди згенерований пейлоад.

2.2.3 Archive finder

Однією з важливих складових сканування веб-застосунку, є розвідка даних, яка допомагає в ідентифікації вразливостей. Цей та інші модулі будуть виконувати цю розвідку, використовуючи метод перебору та аналізу надаї інформації.

У відкритому доступі знаходиться сервіс Web Archive, який зберігає сторінки, ідентифіковані пошуковими системами. Таким чином при аналізі веб-застосунку є можливість знайти видалені сторінки, які можуть містити:

- Секретну інформацію
- Схований функціонал, який не був видалений з веб-застосунку
- Лог файли з параметрами конфігурацій серверу
- Дані для входу до адмін панелі
- PHP та JavaScript файли, які не індексуються у сучасній структурі веб-застосунку

Для пошуку у Web Arhive в програмі реалізоване з’єднання з відкритим API, парсинг даних за необхідними параметрами та перевірка контенту на наявність вмісту. Код програми додається у Додатку Б. Спочатку програма робить запит з шаблоном http://web.archive.org/cdx/search/cdx?url=*.{{domain}}/*&output=txt&fl=original&collapse=urlkey для отримання звіту у текстовому стилі щодо сканованого хосту та його субдоменів, далі перевіряє наявність контенту у кожному файлі за шаблоном http://web.archive.org/cdx/search/cdx?url={{url_path}}.

2.2.3 Domains analize

При сканувані веб-застосунку необхідно також сканувати сервер та зовнішню мережу для ідентифікації працюючих в них сервісів. Це також являється частиною розвітки, з ухилом у досліджування мережі.

Для виканання цієї роботи був розроблений модуль — код у Додатку В — він знаходив субдомени веб-застосунку та виконував швидке сканування зі знаходженням відкритих портів.

Для пошуку субдоменів були використані open-source програми Subfinder та Sublist3r — вони виконують пошук доступних хостів як методом перебору, так і методом аналізу даних з відкритих джерел:

- Google
- Yahoo
- Bing
- Ask
- Virustotal
- SSL Certificates
- DNSdumpster
- PassiveDNS
- Netcraft

Дві програми були використані для охоплення більшої кількості джерел, таким чином була отримана більша кількість хостів.

Далі була використана також open-source інструмент nmap — шаблон команди: nmap {{domain}} -oG tmp.txt — ключ -oG був доданий для зручного парсингу данних типу XML отриманих портів. Після заврешення сканування програма звітує дані у вигляді таблиці.

Однак при використанні програми nmap виникла проблема — велика кількість часу для виконення сканування. Для формування звіту нам необхідна лише інформація по відкритим портам на працюючому хості, для

чого підійде будь-яка швидка програма без зайвого функціоналу. Тому була розроблена друга версія програми, яка виконує аналіз перших 10000 портів та звітує про відкриті.

Максимальна кількість портів обмежена 16-бітною адресацію, тому були обрані 10000 портів за таким принципом:

• 0-1023 — системні порти, на яких працюють службові програми (http, https, ssh, ftp)
• 1024-49151 — зареєстровані порти, на яких працюють нестандартні програми (mysql, proxy)
• 49152-65535 — динамічні порти.

Оскільки порт 8080 є популярним для розташування проксі, кількість портів для аналізу була розширена.

Для реалізації швидкого процесу сканування були використані Python-бібліотеки:

— socket

Доступ до інтерфейсу низкого рівня - socket BSD.

— threading

Додавання кілька потоків для виконання паралельних запитів

У Додатку В, V2 додається код. Функція scan_port(ip,port) з паузою 0.5 робить socket запит до сканового хоста на вказаний порт — при успішній операції строчка хост:порт додається до фінального звіту програми. Для збільшення швидкості у main() використовуються потоки — у даному випадку 10000 портів розділені на 2, однак при необхідності збільшити швидкість — можна додавати більше потоків. Програма використовує загальний(global) масив для зберігання, парсингу та звітування данних.

2.2.4 Cloud Storage Finder

Сучасні веб-застосунки для зменшення навантаженості серверу та економії плати за зберігання данних використовують Cloud Storage. Лідерами у цій сфері є Amazon, Microsoft та Google.

Для пошуку некоректних налаштувань прав була використана open-source програма, яка методом перебору проводить пошук. Ще одним способом проведення такого пошуку є звернення до сайтів, які індексують усі знайдені хмарні сховища з некоректними правами доступу — однак для цього необхідно здійснювати плату за використання сервісу.

Для комфорту використання був написаний модуль, який бере однією строкою усі ключові слова для пошуку, та окремо додає їх в ключах для виповнення програми — код наведений у Додатку Г.

Висновки до розділу 2

У даному розділі були продеменстровані обрані вектори атака та розвідки веб-застосунку. Написані модулі та вразливості, які вони можуть знаходити:

- Spider
  - Reflected XSS
  - DOM XSS
  - Open redirect
  - CSRF

  Збір посилань всередині веб-додатку та аналіз змісту документів.
- Archive finder
  - Sensitive Data Exposure
  - Security Misconfiguration
  - Insufficient Logging & Monitoring

  Пошук збережених документів веб-застосунку, які були видалені або сховані

- Domains analize
  - Using Components with Known Vulnerabilities

    Пошук субдоменів сервісу та аналіз мережі
- Cloud Storage finder
  - Security Misconfiguration

    Пошук некотректних налаштувань прав доступу на 3rd party service.

3 АНАЛІЗ РОБОТИ ТА ЗВІТІВ ПРОГРАМИ

- *Модуль Spider*

При виконані програми, вона звітує знайдені посилання та показує статус сканування. На Рисунку 3.1 можна побачити усі посилання, які проаналізувала програма.

```
olexandropanasiuk@mb-pro iSiN-diplom % python3 spider.py
Scanning http://localhost:8001
Scanning http://localhost:8001//csrf.html
Scanning http://localhost:8001//dom_xss.html
2
Scanning http://localhost:8001//index.php
Scanning http://localhost:8001//open_redirect.php
Scanning http://localhost:8001//sensitive.js
Scanning http://localhost:8001//xss.php
Scanning http://localhost:8001//resources/images/tracker.gif?searchTerms='+query+'
Scanning http://localhost:8001//open_redirect.php?url=/index.php
Scanning http://localhost:8001//xss.php?name=Kitty
olexandropanasiuk@mb-pro iSiN-diplom %
```

Рисунок 3.1 - Звітування програми щодо знайдених посилань

У папці проекту створився файл alerts.txt, у який програма звітувала знайдені вразливості. Також були створені файлі зі всіма посиланнями, які були знайдені при скануванні.

```
olexandropanasiuk@mb-pro test % cat alerts.txt
Possible CSRF: http://localhost:8001//csrf.html
Possible CSRF: http://localhost:8001//dom_xss.html
Possible DOM-XSS: http://localhost:8001//dom_xss.html
Possible XSS: http://localhost:8001//xss.php?name=Kittyend<new
Possible XSS: http://localhost:8001//xss.php?name=Kittyend>new
Possible XSS: http://localhost:8001//xss.php?name=Kittyend'new
Possible XSS: http://localhost:8001//xss.php?name=Kittyend"new
Possible Open redirect: http://localhost:8001//open_redirect.php?url=/randompage12
olexandropanasiuk@mb-pro test %
```

Рисунок 3.2 - Звітування програми щодо знайдених вразливостей

Ми бачимо False Positive звіт про CSRF на сторінці dom_xss.html — CSRF там існує, оскільки форма відправляється — однак ця форма лише для відворення введених слів на сторінці, тому загрози для користувача вона не несе.

Також у нас є можливість проаналізувати логи серверу та дізнатися кількість запитів для знаходження XSS. Програма проаналізувала усі посилання з параметрами, відібрала ті, в яких текст відображається у тексті документу, та підставила пейлоади, порівнюючи відповіді сервера. Таким чином лише два посилання були reflected, і кількість запитів модулю пошука XSS становила — 8.

```
olexandropanasiuk@mb-pro test-vuln-app % cat server-logs.txt | grep '?'
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62890 [404]: GET //resources/images/tracker.gif?searchTerms='+query+' - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62891 [302]: GET //open_redirect.php?url=/index.php
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62893 [200]: GET //xss.php?name=Kitty
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62894 [404]: GET //resources/images/tracker.gif?searchTerms=abcfgh778 - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62895 [404]: GET //resources/images/tracker.gif?searchTerms='+query+'end%3Cnew - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62896 [404]: GET //resources/images/tracker.gif?searchTerms='+query+'end%3Enew - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62897 [404]: GET //resources/images/tracker.gif?searchTerms='+query+'end'new - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62898 [404]: GET //resources/images/tracker.gif?searchTerms='+query+'end%22new - No such file or directory
[Tue Jun  1 00:56:42 2021] 127.0.0.1:62899 [302]: GET //open_redirect.php?url=abcfgh778
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62901 [200]: GET //xss.php?name=abcfgh778
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62902 [200]: GET //xss.php?name=Kittyend%3Cnew
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62903 [200]: GET //xss.php?name=Kittyend%3Enew
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62904 [200]: GET //xss.php?name=Kittyend'new
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62905 [200]: GET //xss.php?name=Kittyend%22new
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62906 [404]: GET //resources/images/tracker.gif?searchTerms=/randompage12 - No such file or directory
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62907 [302]: GET //open_redirect.php?url=/randompage12
[Tue Jun  1 00:56:43 2021] 127.0.0.1:62909 [200]: GET //xss.php?name=/randompage12
olexandropanasiuk@mb-pro test-vuln-app %
```

Рисунок 3.3 - Запити до сервера з параметрами

На Рисунку 3.3 зображені усі отримані запити - вони не містили шкідливих пейлоадів, через які сервер міг заблокувати наші запити, і не викликали великого навантаження.

Також були знайдені DOM-XSS та Open Redirect. На сторінці з DOM XSS програма розпізнала вразливу функцію (Рисунок 3.4)

```
<script>
    function trackSearch(query) {
        document.write('<img src="/resources/images/tracker.gif?searchTerms='+query+'">');
    }
    var query = (new URLSearchParams(window.location.search)).get('search');
    if(query) {
        trackSearch(query);
    }
</script>
```

Рисунок 3.4 - Вразливість DOM XSS

На сторінці open_redirect.php?url={{url}} відбувалося перенаправлення за вказаною адресою, як ми можемо бачити на Рисунку 3.5

```
olexandropanasiuk@mb-pro test % python3
Python 3.9.2 (default, Feb 24 2021, 13:26:09)
[Clang 12.0.0 (clang-1200.0.32.29)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import requests
>>> url = 'http://localhost:8001/open_redirect.php?url=//google.com'
>>> r = requests.get(url)
>>> r.status_code
200
>>> r.url
'http://www.google.com/'
>>>
```

Рисунок 3.5 - Вразливість Open Redirect

- *Модуль Archive finder*

На Рисунку 3.6 ми бачимо знайдені посилання, які наразі не є індексованими на сайті ipt.kpi.ua

```
http://ipt.kpi.ua:80/wp-content/uploads/2019/12/F_ZV_1920_1s_3.xls
http://ipt.kpi.ua/wp-content/uploads/2019/12/Gramota-215x300.jpg
http://ipt.kpi.ua:80/wp-content/uploads/2019/12/IPT_2019-2_Bonuses.pdf
http://ipt.kpi.ua/wp-content/uploads/2019/12/Ponoch-1-230x300.jpg
http://ipt.kpi.ua:80/wp-content/uploads/2019/12/Sessiya_ZIMA_1920_1.xls
http://ipt.kpi.ua/wp-content/uploads/2020/01/4nnbyru1-471x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/01/Indyvidualnyj-vybir-300x300.jpg
http://ipt.kpi.ua:80/wp-content/uploads/2020/01/IPT_2019-12_AverageScores.pdf
http://ipt.kpi.ua/wp-content/uploads/2020/01/isef.png
http://ipt.kpi.ua/wp-content/uploads/2020/01/isef2.png
http://ipt.kpi.ua/wp-content/uploads/2020/01/logo2-544x300.png
http://ipt.kpi.ua/wp-content/uploads/2020/02/01.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/1-453x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/2-453x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/3-454x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/4-453x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/5-453x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/IStyopochkina5-150x150.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/02/tkach-149x150.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/image001-150x150.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/image003-263x150.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/image004-267x150.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/Istoriya-1.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/Nakaz-pro-stvorennya-595x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/Programa-yuvileyu-142x300.png
http://ipt.kpi.ua/wp-content/uploads/2020/11/TV-kanal.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/11/YUvilej1-300x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/12/YUvilej-1-536x300.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/12/YUvilej-21.jpg
http://ipt.kpi.ua/wp-content/uploads/2020/12/YUvilej-3-534x300.png
http://ipt.kpi.ua/wp-includes/css/dashicons.css?ver=4.1
http://ipt.kpi.ua/wp-includes/css/dashicons.min.css?ver=4.1
http://ipt.kpi.ua/wp-includes/fonts/dashicons.eot
http://ipt.kpi.ua/wp-includes/fonts/dashicons.svg
http://ipt.kpi.ua/wp-includes/fonts/dashicons.ttf
http://ipt.kpi.ua/wp-includes/js/jquery/jquery-migrate.js?ver=1.2.1
http://ipt.kpi.ua/wp-includes/js/jquery/jquery-migrate.min.js?ver=1.2.1
http://ipt.kpi.ua/wp-includes/js/jquery/jquery.js?ver=1.10.2
http://ipt.kpi.ua/wp-includes/js/jquery/jquery.js?ver=1.11.1
http://ipt.kpi.ua/wp-includes/js/thickbox/loadingAnimation.gif
http://ipt.kpi.ua/wp-includes/js/thickbox/thickbox.css?ver=4.1
http://ipt.kpi.ua/wp-includes/js/thickbox/thickbox.js?ver=3.1-20121105
http://ipt.kpi.ua/wp-includes/wlwmanifest.xml
http://ipt.kpi.ua:80/wp-login.php
http://ipt.kpi.ua:80/wp-login.php?action=lostpassword
http://ipt.kpi.ua:80/wp-signup.php?
http://ipt.kpi.ua:80/wwwwwww
http://ipt.kpi.ua:80/wwwwwww/feed
http://ipt.kpi.ua:80/xmlrpc.php?rsd
http://ipt.kpi.ua:80/ya-postupiv-yaki-osoblivosti-navchalnogo-planu-spetsializatsiyi-matematichni-metodi-kiberbezpeki
http://ipt.kpi.ua:80/yaka-spetsialnist-krashha
http://ipt.kpi.ua:80/yaka-spetsialnist-krashha/feed
http://ipt.kpi.ua:80/yaki-osoblivosti-napryamu-prikladna-matematika-v-fiziko-tehnichnomu-instituti
http://ipt.kpi.ua:80/yaki-osoblivosti-napryamu-prikladna-matematika-v-fiziko-tehnichnomu-instituti/feed
http://ipt.kpi.ua:80/yasv
http://ipt.kpi.ua:80/yasv/feed
http://ipt.kpi.ua:80/yugakova
http://ipt.kpi.ua:80/yugakova/feed
http://ipt.kpi.ua:80/zagalna-informatsiya-kontakti
http://ipt.kpi.ua:80/zagalna-informatsiya-kontakti/feed
http://ipt.kpi.ua:80/zastosuvannya-matematichnih-metodiv-u-sporti
http://ipt.kpi.ua:80/zastosuvannya-matematichnih-metodiv-u-sporti/feed
http://ipt.kpi.ua:80/zavadska
http://ipt.kpi.ua:80/zavadska/feed
http://ipt.kpi.ua:80/zemliak
http://ipt.kpi.ua:80/zhdanov
http://ipt.kpi.ua:80/zhdanov/feed
http://ipt.kpi.ua:80/zoryane-nebo
http://ipt.kpi.ua:80/zoryane-nebo/feed
olexandropanasiuk@mb-pro test % cat ArchiveUrls.txt
```

Рисунок 3.6 - Результат роботи Archive finder на ipt.kpi.ua

- *Модуль Domains analize*

Хостом для тестування був взятий kpi.ua. Для зменшення навантаження мережі, зі знайдених 500+ були проаналізовані лише перші 5 субдоменів. На рисунку 3.7 можна побачити звіт, оформлений таблицею.

```
olexandropanasiuk@mb-pro test % cat subdomains.txt
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------+
|     Host    |                                                                  Ports                                                                  |
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------+
|  www.kpi.ua |                                                23, 25, 80, 81, 135, 139, 443, 445, 1688, 1783,                                          |
| 2016.kpi.ua |                                                                                                                                         |
|  3d.kpi.ua  |  21, 80, 443, 990, 50000, 50001, 50002, 50003, 50006, 50300, 50389, 50500, 50636, 50800, 51103, 51493, 52673, 52822, 52848, 52869,      |
+-------------+-----------------------------------------------------------------------------------------------------------------------------------------+
olexandropanasiuk@mb-pro test %
```

Рисунок 3.7 - Звіт роботи сервісів на субдоменах kpi.ua

- *Модуль Cloud Storage finder*

Були проаналізовані Cloud Storage місця сервісу Amazon, з kpi в назві. На рисунку 3.8 можна побачити звіт.

```
olexandropanasiuk@mb-pro test % cat cloudstorages.txt
Protected S3 Bucket: http://kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpi2.s3.amazonaws.com/
Protected S3 Bucket: http://kpi2018.s3.amazonaws.com/
Protected S3 Bucket: http://kpi2019.s3.amazonaws.com/
Protected S3 Bucket: http://kpi3.s3.amazonaws.com/
Protected S3 Bucket: http://kpi4.s3.amazonaws.com/
Protected S3 Bucket: http://kpi5.s3.amazonaws.com/
Protected S3 Bucket: http://amazon-kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpianalytics.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-app.s3.amazonaws.com/
Protected S3 Bucket: http://kpibackup.s3.amazonaws.com/
Protected S3 Bucket: http://kpi.backup.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-backup.s3.amazonaws.com/
Protected S3 Bucket: http://kpi.backups.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-backups.s3.amazonaws.com/
Protected S3 Bucket: http://kpibucket.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-bucket.s3.amazonaws.com/
OPEN S3 BUCKET: http://kpi-build.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-cloud.s3.amazonaws.com/
Protected S3 Bucket: http://common-kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpicontainer.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-data.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-db.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-dev.s3.amazonaws.com/
Protected S3 Bucket: http://dev-kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-development.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-files.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-images.s3.amazonaws.com/
Protected S3 Bucket: http://infrakpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-logs.s3.amazonaws.com/
Protected S3 Bucket: http://kpi.main.s3.amazonaws.com/
Protected S3 Bucket: http://media.kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpipictures.s3.amazonaws.com/
Protected S3 Bucket: http://prokpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-prod.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-reports.s3.amazonaws.com/
Protected S3 Bucket: http://reports-kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-services.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-stats.s3.amazonaws.com/
Protected S3 Bucket: http://kpistorage.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-storage.s3.amazonaws.com/
Protected S3 Bucket: http://kpistore.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-store.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-support.s3.amazonaws.com/
Protected S3 Bucket: http://kpitest.s3.amazonaws.com/
Protected S3 Bucket: http://kpi.test.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-test.s3.amazonaws.com/
Protected S3 Bucket: http://testkpi.s3.amazonaws.com/
Protected S3 Bucket: http://test-kpi.s3.amazonaws.com/
Protected S3 Bucket: http://kpitraining.s3.amazonaws.com/
Protected S3 Bucket: http://kpi-uploads.s3.amazonaws.com/
App Found: https://kpi-admin.awsapps.com
App Found: https://commonkpi.awsapps.com
App Found: https://testkpi.awsapps.com
App Found: https://test-kpi.awsapps.com
olexandropanasiuk@mb-pro test %
```

Рисунок 3.8 - Звіт пошуку Cloud Storage з kpi в назві

ВИСНОВКИ

При досліджені методів автоматизації пошуку вразливостей веб-застосунку були відокремлені дві різниці у реалізації: метод перебору та метод аналізу відповіді. Для демонстрації роботи методів був досліджений функціонал найкращих програм для сканування веб-застосунків, а також логи, залишені Burp Scanner, при тестувані вразливого веб-застосунку, написаного для демонстрації роботи сканера, який є практичною частиною написання дипломної роботи. Також були розглянуті методи збору та аналізу даних для пошуку секретної інформації, яка стосуються створення та підтримки працездатності веб-застосунку.

При створенні сканера були реалізовані модулі Spider (збирання посилань веб-застосунку, аналіз вразливостей: XSS, Open Redirect, CSRF), Archive finder (пошук індексованих пошуковими системами посилань), Domains analize (аналіз хостів на відкриті порти до 10000), Cloud Storage finder (пошук некоректних прав доступу на хмарних сервісах, де застосунок зберігає дані).

Для демонстрації працездатності створеного сканеру були протестовані сайти ipt.kpi.ua та kpi.ua, а також розроблений веб-застосунок з вразливостями.

Розроблена у результаті програма дає фундамент для створення нових модулів пошуку вразливстей та їх легкої інтеграції — таким чином алгоритм пошуку деякої вразивості може бути описаним окремою функцією та доданий до програми. Використання цього сканеру допоже покращити безпеку веб-застосунків.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. The OWASP foundation. OWASP TOP-10 [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://owasp.org/www-project-top-ten/

2. Project Discovery. Open list of created tools and methods[Електронний ресурс]. — 2021 — https://github.com/projectdiscovery/

3. PortSwigger. Research of new methods for detecting and exploiting. — 2021 — https://portswigger.net/research/

4. Peter Jaworowski. Bug Trap. A Field Guide to Web Hacking. — 2019

5. TomHunter. Чем искать уязвимости веб-приложений: сравниваем восемь популярных сканеров [Електронний ресурс]. — 2019. — https://habr.com/ru/company/tomhunter/blog/456892/

6. Xaker.ru. Хак в один клик. Сравниваем возможности автоматических сканеров уязвимостей [Електронний ресурс]. — 2020. — https://xakep.ru/2020/04/21/hack-scan/

7. Тимур Гильмуллин. Общие подходы и критерии тестирования сканеров безопасности web-приложений [Електронний ресурс]. — 2013. — https://forworktests.blogspot.com/2013/05/web.html

8. Larry Suto. Analyzing the Accuracy and Time Costs of Web Application Security Scanners [Електронний ресурс]. — 2010. — http://www.think-secure.nl/pdf/Accuracy_and_Time_Costs_of_Web_App_Scanners.pdf

9. OWASP. OWASP Top Ten [Електронний ресурс]. — 2017. — https://owasp.org/www-project-top-ten/

10. projectdiscovery. Nuclei [Open-source program]. — 2021. — https://github.com/projectdiscovery/nuclei

11. PortSwigger. Products, Research and Academy [Електронний ресурс]. — 2021. — https://portswigger.net/

12. Sullo. Nikto - web server scanner [Open-source program]. — 2021. — https://github.com/sullo/nikto

13. Acunetix. Acunetix Vulnerability Scanner [Електронний ресурс]. — 2021. — https://www.acunetix.com/vulnerability-scanner/

14. OWASP. OWASP Zed Attack Proxy (ZAP) [Електронний ресурс]. — 2021. — https://owasp.org/www-project-zap/

15. Ahmed Aboul-Ela. Sublist3r - subdomain enumerating tool [Open-source program]. — 2019. — https://github.com/aboul3la/Sublist3r

16. Archive.org. Wayback Machine APIs [Електронний ресурс]. — 2021. — https://archive.org/help/wayback_api.php

17. Arachni. Web application security scanner framework [Електронний ресурс]. — 2017. — https://www.arachni-scanner.com/

18. Vlada Korzun. Навыки OSINT(интернет-разведки) в кибербезопасности [Електронний ресурс]. — 2020. — https://proglib.io/p/navyki-osint-internet-razvedki-v-kiberbezopasnosti-2020-11-14

ДОДАТКИ

ДОДАТОК А

Модуль програми: spider.py

```
import requests, os, time
global p_name
f = open('current_project.txt','r')
p_name = f.read().replace('\n','')
f.close
def sort_links(links):
    global main_domain
    restrict_mas = ['.jpg', '.png', 'ico', 'jpeg', '.mp4', '.css', '.pdf', '.doc']
    new_links = []
    for i in links:
        link = i
        if '#' in link:                    # remove '#'
            tmp01 = link.find('#')
            link = link[:tmp01]
        if link == '' or link == '/':
            link = main_domain
            new_links.append(link)
            continue
        if len(link) < 2:
            link = main_domain + i
            new_links.append(link)
            continue
        if i[0] == '/' and i[1] != '/':     #'/abc'
            link = main_domain + i
        if i[0:1] == '//':                  #'//google.com'
            link = 'https://' + i
        if i[0:4] != 'http':                #'abc.txt'
            link = main_domain + '/' + i
        for j in restrict_mas:              #jpg etc
            if j in link:
                tmp01 = link.find('?')
                tmp02 = link.find(j)
                if tmp01 > tmp02 or tmp01 == -1:
                    link = link + '[nope_keks_8912]'
                    break
        if '[nope_keks_8912]' not in link:
            link.replace('://', ':/a/')
            link.replace('//', '/')
            link.replace(':/a/', '://')
            new_links.append(link)
```

```
    return new_links
def get_main_domain(start_url):
    tmp01 = start_url.find('://')
    tmp01 = start_url.find('/', tmp01+4)
    if tmp01 == -1:
        return start_url
    else:
        a = start_url[:tmp01]
        return a
def get_headers(): #+
    print('Opening headers.txt to take headers to insert. Confirm...')
    os.system('open ./tmp/headers.txt')
    tmp = input('Continue...')
    headers_data = {}
    filename = './tmp/headers.txt'
    f = open(filename, 'r')
    for line in f:
        line = line.replace('\n', '')
        tmp = line.find(':')
        a = line[:tmp]
        b = line[tmp+1:]
        if (b[0] == ' '):
            b = b[1:]
        headers_data[a] = b
    f.close
def report(links):
    global p_name
    filepath = './projects/' + p_name + '/'
    js_links = []
    param_links = []
    for link in links:
        if '.js' in link:
            js_links.append(link)
        if '?' in link:
            param_links.append(link)
    if js_links != []:
        filename = filepath + 'js_links.txt'
        com = 'touch ' + filename
        os.system(com)
        f = open(filename,'w')
        for i in js_links:
            f.write(i+'\n')
        f.close
    if param_links != []:
        filename = filepath + 'param_links.txt'
        com = 'touch ' + filename
```

```
        os.system(com)
        f = open(filename,'w')
        for i in param_links:
            f.write(i+'\n')
        f.close
    filename = filepath + 'all_links.txt'
    com = 'touch ' + filename
    os.system(com)
    f = open(filename,'w')
    for i in links:
        f.write(i+'\n')
    f.close
def get_links(h_s):
    links = []
    tmp1 = h_s.find('href=')
    while 1:
        if tmp1 == -1:
            break
        tmp1 = h_s.find('=',tmp1)
        to_find = h_s[tmp1+1]
        tmp01 = 0
        while to_find == ' ':
            tmp01 += 1
            to_find = h_s[tmp1+1+tmp01]
        tmp1 = h_s.find(to_find,tmp1)
        tmp2 = h_s.find(to_find,tmp1+1)
        link = h_s[tmp1+1:tmp2]
        if link != '':
            links.append(link)
        tmp1 = h_s.find('href=',tmp1+1)
        if tmp1 == -1:
            break
    tmp1 = h_s.find('src=')
    while 1:
        if tmp1 == -1:
            break
        tmp1 = h_s.find('=',tmp1)
        to_find = h_s[tmp1+1]
        tmp01 = 0
        while to_find == ' ':
            tmp01 += 1
            to_find = h_s[tmp1+1+tmp01]
        tmp1 = h_s.find(to_find,tmp1)
        tmp2 = h_s.find(to_find,tmp1+1)
        link = h_s[tmp1+1:tmp2]
        if link != '':
```

```
            links.append(link)
        tmp1 = h_s.find('src=',tmp1+1)
        if tmp1 == -1:
            break
    tmp1 = h_s.find('href =')
    while 1:
        if tmp1 == -1:
            break
        tmp1 = h_s.find('=',tmp1)
        to_find = h_s[tmp1+1]
        tmp01 = 0
        while to_find == ' ':
            tmp01 += 1
            to_find = h_s[tmp1+1+tmp01]
        tmp1 = h_s.find(to_find,tmp1)
        tmp2 = h_s.find(to_find,tmp1+1)
        link = h_s[tmp1+1:tmp2]
        if link != '':
            links.append(link)
        tmp1 = h_s.find('href =',tmp1+1)
        if tmp1 == -1:
            break
    tmp1 = h_s.find('src =')
    while 1:
        if tmp1 == -1:
            break
        tmp1 = h_s.find('=',tmp1)
        to_find = h_s[tmp1+1]
        tmp01 = 0
        while to_find == ' ':
            tmp01 += 1
            to_find = h_s[tmp1+1+tmp01]
        tmp1 = h_s.find(to_find,tmp1)
        tmp2 = h_s.find(to_find,tmp1+1)
        link = h_s[tmp1+1:tmp2]
        if link != '':
            links.append(link)
        tmp1 = h_s.find('src =',tmp1+1)
        if tmp1 == -1:
            break
    links = sort_links(links)
    return links
def detect_xss(links):
    for link in links:
        if '?' in link:
            check_reflected(link)
```

```
def get_params(link):
    params = []
    tmp01 = link.find('?') + 1
    fl = 0
    while fl != 1:
        tmp02 = link.find('&', tmp01 + 1)
        if tmp02 == -1:
            fl = 1
            tmp02 = len(link)
        tmp03 = link.find('=', tmp01, tmp02)
        if tmp03 == -1:
            params.append(link[tmp01:tmp02])
        else:
            params.append(link[tmp01:tmp03])
        tmp01 = tmp02 + 1
    return params
def check_reflected(link):
    global p_name
    params = get_params(link)
    payloads = []
    for param in params:
        tmp4 = '?' + param
        tmp1 = link.find(tmp4)
        if tmp1 == -1:
            tmp4 = '&' + param
            tmp1 = link.find(tmp4)
        tmp2 = link.find('&', tmp1+1)
        if tmp2 == -1:
            tmp2 = len(link)
        tmp3 = link.find('=', tmp1, tmp2)
        if tmp3 == -1:
            payload = link[:tmp2] + '=abcfgh778' + link[tmp2:]
        else:
            payload = link[:tmp3] + '=abcfgh778' + link[tmp2:]
        payloads.append(payload)
    for tmp_link in payloads:
        r = requests.get(tmp_link)
        tmp_hs = r.text
        if 'abcfgh778' in tmp_hs:
            tmp_0_mas = ['<','>','\"',"'"]
            for elem0 in tmp_0_mas:
                payload = link[:tmp2] + 'end' + elem0 + 'new' + link[tmp2:]
                r = requests.get(payload)
                if 'end' + elem0 + 'new' in r.text:
                    filename = './projects/' + p_name + '/alerts.txt'
```

```
                f = open(filename, 'a')
                f.write('Possible XSS: ' + payload + '\n')
                f.close
def detect_redirect(links):
    global p_name
    global main_domain
    for link in links:
        if '?' in link:
            payloads = []
            params = get_params(link)
            for param in params:
                tmp4 = '?' + param
                tmp1 = link.find(tmp4)
                if tmp1 == -1:
                    tmp4 = '&' + param
                    tmp1 = link.find(tmp4)
                tmp2 = link.find('&', tmp1+1)
                if tmp2 == -1:
                    tmp2 = len(link)
                tmp3 = link.find('=', tmp1, tmp2)
                if tmp3 == -1:
                    payload = link[:tmp2] + '=/randompage12' + link[tmp2:]
                else:
                    payload = link[:tmp3] + '=/randompage12' + link[tmp2:]
                payloads.append(payload)
            for payload in payloads:
                r = requests.get(payload)
                if r.url == main_domain + '/randompage12':
                    filename = './projects/' + p_name + '/alerts.txt'
                    f = open(filename, 'a')
                    f.write('Possible Open redirect: ' + payload + '\n')
                    f.close
def detect_csrf(h_s, link):
    if '<form ' in h_s:
        global p_name
        tmp1 = h_s.find('<form ')
        hidden_logic = False
        while tmp1 != -1:
            tmp2 = h_s.find('</form>',tmp1)
            tmp_hs = h_s[tmp1:tmp2]
            tmp3 = tmp_hs.find('<input')
            while tmp3 != -1:
                tmp4 = tmp_hs.find('>')
                if 'hidden' in tmp_hs[tmp3:tmp4]:
```

```
                hidden_logic = True
            tmp3 = tmp_hs.find('<input', tmp3 + 1)


        tmp1 = h_s.find('<form',tmp1+1)
        if hidden_logic == False:
            filename = './projects/' + p_name + '/alerts.txt'
            f = open(filename, 'a')
            f.write('Possible CSRF: ' + link + '\n')
            f.close
def detect_domxss(h_s, link):
    payloads_mas = ['document.write(', 'innerHTML', 'eval(']
    for payload in payloads_mas:
        tmp1 = 0
        for i in range(h_s.count(payload)):
            tmp1 = h_s.find(payload, tmp1+1)
            if '<script>' in h_s[:tmp1] and '</script>' in h_s[tmp1:]:
                print(2)
                filename = './projects/' + p_name + '/alerts.txt'
                f = open(filename, 'a')
                f.write('Possible DOM-XSS: ' + link + '\n')
                f.close
def main(links):
    global headers_data
    for link in links:
        print('Scanning ' + link)
        r = requests.get(link)#, headers=headers_data)
        new_links = get_links(r.text)
        detect_csrf(r.text, link)
        detect_domxss(r.text, link)
        for i in new_links:
            if i not in links:
                links.append(i)
        '''try:
            r = requests.get(link)#, headers=headers_data)
            new_links = get_links(r.text)
            detect_csrf(r.text, link)
            for i in new_links:
                if i not in links:
                    links.append(i)
        except:
            print('Some problems with link ' + link)'''
    report(links)
    detect_xss(links)
    detect_redirect(links)
#start_url = input('Enter url to start from: ')
```

```
start_url = 'http://localhost:8001'
global main_domain
main_domain = get_main_domain(start_url)
#global headers_data


#headers_data = get_headers()
links = []
links.append(start_url)
main(links)
```

ДОДАТОК Б

Модуль програми: archive.py

```
import requests, sys, os
domain = sys.argv[1]
def help():
    print('-c   -   include subdomains for search')
    print('-h   -   show help message')
url = 'http://web.archive.org/cdx/search/cdx?url=' + domain + '/
      *&output=txt&fl=original&collapse=urlkey'
if len(sys.argv) > 2:
    if sys.argv[2] == '-h':
        help()
    if sys.argv[2] == '-c':
        url = 'http://web.archive.org/cdx/search/cdx?url=*.' + domain + '/
        *&output=txt&fl=original&collapse=urlkey'
r = requests.get(url)
report_text = r.text
filename_report = './projects/test/ArchiveUrls.txt'
com = 'touch ' + filename_report
os.system(com)
f = open(filename_report, 'w')
f.write(report_text)
f.close
scan_filename = 'links.txt'
print('Getting list from ' + scan_filename + '. Confirm...')
tmpqaaa = input()
print('\n')
def progress(count, total, suffix=''):
    bar_len = 40
    filled_len = int(round(bar_len * count / float(total)))
```

```
    percents = round(100.0 * count / float(total), 1)
    bar = '=' * filled_len + '-' * (bar_len - filled_len)
    sys.stdout.write('[%s] %s%s ...%s\r' % (bar, percents, '%', suffix))
    sys.stdout.flush()
base_url = 'http://web.archive.org/cdx/search/cdx?url='
links = []
f = open(filename_report, 'r')
for line in f:
    links.append(line.replace('\n',''))
f.close
links_status = []
for link in links:
    url = base_url + link
    r = requests.get(url)
    tmp01 = r.text
    if ' 200 ' in tmp01:
        links_status.append('+')
    else:
        links_status.append('-')
    tmp_at = links.index(link) + 1
    tmp_from = len(links)
    progress(tmp_at, tmp_from, 'done')
print('\n')
for i in range(len(links)-1):
    if links_status[i] == '+':
        print(links[i])
```

ДОДАТОК В

Модуль програми: subdomains.py (V1)

```
import os
from prettytable import PrettyTable
domain = input('Enter domain to sublist: ')
report_filename = 'subdomains.txt'
com = 'python3 sublist3r.py -d ' + domain + ' -o tmp.txt'
os.system(com)
report_mas = []
f = open('tmp.txt', 'r')
for line in f:
    report_mas.append(line.replace('\n',''))
f.close
```

```
os.system('echo \'\' >  tmp.txt’)
os.system('rm tmp.txt')
com = 'subfinder -d ' + domain + ' -o tmp.txt'
os.system(com)
f = open('tmp.txt', 'r')
for line in f:
    tmp01 = line.replace('\n','')
    if tmp01 not in report_mas:
        report_mas.append(tmp01)
f.close
os.system('rm tmp.txt')
f = open(report_filename, 'w')
for tmp1 in report_mas:
    if tmp1 == report_mas[len(report_mas)-1]:
        f.write(tmp1)
    else:
        f.write(tmp1 + '\n')
f.close
x = PrettyTable(["Host", "Ports"])
host_mas = []
port_mas = []
i = 0
for domain in report_mas:
    host_mas.append(domain)
    com = 'nmap ' + domain + ' -oG tmp.txt'
    os.system(com)
    f = open('tmp.txt','r')
    ports = ''
    for line in f:
        line = line.replace('\n','')
        if 'Ports:' in line:
            tmp1 = line.find('Ports:')
            tmp1 = line.find(':',tmp1)
            for i in range(line.count(',')+1):
                tmp2 = line.find('/', tmp1)
                ports += line[tmp1+1:tmp2] + ','
                tmp1 = line.find(',',tmp1+1)
            break
    f.close
    port_mas.append(ports)
    i += 1
    if i == 20:
        print('Exit by owner...')
        break
for i in range(len(host_mas)):
    x.add_row([host_mas[i], port_mas[i]])
```

```
#project
filename = './projects/test/subdomains.txt'
com = 'touch ' + filename
f = open(filename, 'w')
f.write(str(x))
f.close
```

Модуль програми: subdomains.py (V2)

```
import socket, threading, os, sys
global open_ports
open_ports = []
subdomains_filename = sys.argv[1]
def scan_port(ip,port):
    global open_ports
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    sock.settimeout(0.5)
    try:
        connect = sock.connect((ip,port))
        rep_str = '|' + str(ip) + ':' + str(port)
        open_ports.append(rep_str)
        sock.close()
    except:
        pass
def save_report(report_mas):
    com = 'touch report.txt'
    os.system(com)
    f = open('report.txt', 'w')
    for i in report_mas:
        f.write(i + '\n')
    f.close
def generate_report(ips):
    report_mas = []
    global open_ports
    for ip in ips:
        report_str = ip + ': '
        for i in open_ports:
            if '|' + ip in i:
                report_str += i[i.find(':')+1:] + ' '
        report_mas.append(report_str)
    return report_mas
def get_domains():
    subdomains_filename = sys.argv[1]
```

```
    f = open(subdomains_filename)
    domains = []
    for line in f:
        domains.append(line.replace('\n',''))
    f.close
    return domains
def main():
    global open_ports
    #ips = ['novaposhta.ua', 'auth.novaposhta.ua', 'api.novaposhta.ua']
    ips = get_domains()
    for ip in ips:
        for i in range(5000):
            proc1 = threading.Thread(target=scan_port, args=(ip,i))
            proc1.start()
            proc2 = threading.Thread(target=scan_port, args=(ip,5000+i))
            proc2.start()
    report_mas = generate_report(ips)
    save_report(report_mas)
main()
```

ДОДАТОК Г

Модуль програми: storage_finder.py

```
import os
tmp_input = input('Enter keywords to start attack: ')
keywords = []
tmp1 = 0
if tmp_input.count(' ') == 0:
    keywords.append(tmp_input)
else:
    for i in range(tmp_input.count(' ')+1):
        tmp2 = tmp_input.find(' ',tmp1+1)
        if tmp2 == -1:
            tmp2 = len(tmp_input)+1
        keywords.append(tmp_input[tmp1+1:tmp2])
        tmp1=tmp2

com = 'python3 cloud_enum.py --disable-gcp '
for i in keywords:
    com += '-k \'' + i + '\' '
```

```
filename = './projects/test/cloudstorages.txt'
com += ' -l '
com += filename
os.system(com)
```