Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації
(повна назва)
Кафедра Комп'ютерної інженерії та систем технічного захисту інформації
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти магістерський
перший (бакалаврський) / другий (магістерський)

Аналіз вразливостей клієнтської частини вебзастосунків

(тема)

Виконав:
студент 2 курсу, групи СТЗІАм-22-1
Снєга М.М.
(прізвище, ініціали)
Спеціальність
125 Кібербезпека
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Системи технічного захисту інформації, автоматизація її обробки
(повна назва освітньої програми)
Керівник Доц. Ликов Ю.В
(посада, прізвище, ініціали)

Допускається до захисту
Завідувач кафедри _____ _____
(підпис) (прізвище, ініціали)

2024р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій та технічного захисту інформації

Кафедра Комп'ютерної радіоінженерії та систем технічного захисту інформації

Рівень вищої освіти другий (магістерський)

Спеціальність 125 Кібербезпека

Тип програми освітньо-професійна

Освітня програма Системи технічного захисту інформації, автоматизація її обробки

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2024 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Снєзі Максиму Миколайовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз вразливостей клієнтської частини вебзастосунків

затверджена наказом по університету від 03.11.2023 р. № 1281 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 січня 2024р.

3. Вихідні дані до роботи
В якості вебзастосунків використати сайти крупних організацій.
Самостійно обрати 10 сайтів та проаналізувати їх існуючими програмними інструментами аудиту безпеки.
Мінімальна кількість програмних інструментів аудиту безпеки -2.
Виявлені загрози класифікувати на 6 груп по степені критичності.

4. Перелік питань, що потрібно опрацювати в роботі Проаналізувати поточні тенденції і проблеми клієнтської частини вебзастосунків. Огляд сучасних протоколів взаємодії користувачів з клієнтською частиною вебзастосунків. Огляд найпоширеніших вразливостей вебзастосунків. Експериментально дослідити сайти на вразливості. Розробити рекомендації для захисту вебзастосунків на основі даних з дослідження. Побудувати гістограму найбільш поширених вразливостей вебзастосунків. Висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Актуальність

Протоколи взаємодії користувача з клієнтською частиною вебзастосувань

Класифікація вразливостей клієнтської частини вебзастосунків

Результати дослідження вразливостей вебзастосунків

Рекомендації по підвищенню захищеності клієнтської частини вебзастосунків

Висновки

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз літературних джерел.	1.11.23 – 8.11. 23	
2	Огляд методів виявлення вразливостей	9.11.23 – 21.11.23	
3	Аналіз типових вразливостей клієнтської частини вебзастосунків	22.11.23 – 5.12.23	
4	Проведення експериментального дослідження	6.12.23 – 10.12.23	
5	Аналіз результатів дослідження	11.12.23 – 20.12.23	
6	Розробка рекомендацій по усуненню вразливостей	21.12.23 – 31.12.23	
7	Оформлення пояснювальної записки	1.01.24 – 10.01.24	

Дата видачі завдання 1 листопада 2023 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Ликов Ю.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 103 с., 65 рис., 1 табл., 1 додаток, 11 джерел.

АНАЛІЗ ВРАЗЛИВОСТЕЙ КЛІЄНТСЬКОЇ ЧАСТИНИ, ВРАЗЛИВОСТІ ВЕБ ЗАСТОСУНКІВ, ВЗЛОМ, ВРАЗЛИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ВЕБ – ЗАСТОСУНКІВ, БЕЗПЕКА ВЕБКЛІЄНТА.

Об'єкт дослідження – клієнтська частина вебзастосунків, яка використовується для відображення інформації та надання послуг користувачам в Інтернеті.

Мета роботи – вивчення вразливостей клієнтської частини вебзастосунків та способів їх використання зловмисниками для атак.

У даній кваліфікаційній роботі було проведено:

– аналіз вразливостей клієнтської частини вебзастосунків, включаючи вразливості програмного забезпечення, проблеми безпеки та вразливості у відображенні;

– дослідження методів, якими зловмисники можуть використовувати вразливості клієнтської частини для атак, такі як використання шкідливого коду, атаки з використанням AJAX, а також Cross-Site Scripting (XSS) та інші;

– розробка рекомендацій щодо підвищення безпеки клієнтської частини вебзастосунків.

Результати дослідження свідчать про те, що клієнтська частина вебзастосунків також може бути вразливою до різноманітних атак. Ці атаки можуть призвести до порушення безпеки даних користувачів і інших проблем безпеки.

Запропоновані заходи допоможуть забезпечити безпеку клієнтської частини вебзастосунків та захистити користувачів від можливих загроз.

ABSTRACT

Explanatory note of the qualification work: 103 p., 65 figures, 1 tables, 1 appendice, 11 sources.

ANALYSIS OF CLIENT PART VULNERABILITIES, WEB APPLICATIONS VULNERABILITIES, HACKING, WEB APPLICATION SOFTWARE VULNERABILITIES, WEB CLIENT SECURITY.

The object of research is the client part of web applications, which is used to display information and provide services to users on the Internet.

The purpose of the work is to study the vulnerabilities of the client part of web applications and how they are used by attackers for attacks.

In this qualification work, the following was carried out:

– vulnerability analysis of client-side web applications, including software vulnerabilities, security issues, and display vulnerabilities;

– investigating methods by which attackers can use client-side vulnerabilities for attacks, such as using malicious code, AJAX attacks, as well as Cross-Site Scripting (XSS) and others;

– development of recommendations for improving the security of the client part of web applications.

The results of the study indicate that the client side of web applications can also be vulnerable to various attacks. These attacks can lead to breaches of user data and other security issues.

The proposed measures will help ensure the security of the client part of web applications and protect users from possible threats.

ЗМІСТ

ВСТУП 9
1 АНАЛІЗ ПОТОЧНИХ ТЕНДЕНЦІЙ І ПРОБЛЕМ 10
1.1 Поширення вразливостей 10
1.2 Складність виявлення вразливостей 11
1.3 Вартість усунення вразливостей 12
1.4 Поява нових вразливостей 12
2 ПРОТОКОЛИ ВЗАЄМОДІЇ КОРИСТУВАЧА З КЛІЄНТСЬКОЮ ЧАСТИНОЮ ВЕБЗАСТОСУВАНЬ 15
2.1 Протокол JWT 15
2.2 Протокол HTTP/HTTPS 16
2.3 Протокол WebSocket 17
2.4 Протокол WebRTC 18
3 КЛАСИФІКАЦІЯ ВРАЗЛИВОСТЕЙ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКІВ 19
3.1 Вразливість XSS 19
3.2 Вразливість SQL-ін'єкції 23
3.3 CSS Injection 25
3.4 Вразливі та застарілі компоненти 26
3.5 Вразливість CSRF 27
4 ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ 29
4.1 Методи виявлення та оцінка вразливостей вебзастосунків 29
4.2 Сайт nure.ua 32
4.3 Сайт drom.ru 38
4.4 Сайт gosuslugi.ru 44
4.5 Сайт rsl.ru 50
4.6 Сайт olx.ua 52
4.7 Сайт work.ua 54
4.8 Сайт 1win.org.ua 56
4.9 Сайт msk.rt.ru 58
4.10 Сайт rosneft.ru 60
4.11 Сайт autokraz.com.ua 62
4.12 Статистика знайдених вразливостей 66

5 РОЗРОБКА РЕКОМЕНДАЦІЙ ПО ПІДВИЩЕННЮ ЗАХИЩЕНОСТІ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКІВ 72
5.1 Використання сучасних технологій і практик розробки вебзастосунків 72
5.2 Встановлення актуальних версій програмного забезпечення..... 74
5.3 Використання інструментів для виявлення вразливостей 75
5.4 Сучасні нормативні документи в галузі кібербезпеки для створення вебресурсів..... 76
ВИСНОВКИ 80
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 82
Додаток А Комплект графічних матеріалів.....**Error! Bookmark not defined.**

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

API – набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення;

DOM – специфікація прикладного програмного інтерфейсу для роботи зі структурованими документами;

SSL/TLS – криптографічний протокол, що надає можливості безпечної передачі даних в інтернеті для навігації, отримання пошти, спілкування, обміну файлами, тощо;

JS – мова програмування Javascript;

HTML - стандартизована мова розмітки документів для перегляду вебсторінок у браузері;

CSS — це спеціальна мова стилю сторінок, що використовується для опису їхнього зовнішнього вигляду;

ПЗ – програмне забезпечення;

ІБ – інформаційна безпека.

ВСТУП

У нашому сучасному цифровому світі вебзастосунки стали не від'ємним аспектом повсякденного життя. Вони надають користувачам доступ до різноманітних послуг, інформації та розваг через мережу Інтернет. Проте, разом із зручністю використання вебзастосунків приходить і зростання потенційних загроз їхній безпеці.

Вразливості клієнтської частини виникають через помилки у коді веббраузерів, вебдодатків та інших програм, які ми використовуємо для взаємодії з вебзастосунками. Ці вразливості можуть бути використані зловмисниками для крадіжки особистих даних, грошей та навіть загрози нашому життю.

За даними компанії Verizon на 2022 рік, 80% атак на вебзастосунки були спрямовані на вразливості клієнтської частини. Це свідчить про те, що зловмисники все частіше використовують ці вразливості для отримання несанкціонованого доступу до інформації користувачів, включаючи паролі, кредитні картки та особисті дані.

Це дослідження присвячено аналізу вразливостей клієнтської частини вебзастосунків. Під "клієнтською частиною" розуміється код та функціональність, які виконуються на браузері користувача і відповідають за взаємодію з вебзастосунком. Наявність вразливостей у цій частині може призвести до серйозних проблем у сфері безпеки.

У роботі буде розглянуто різні аспекти вразливостей клієнтської частини вебзастосунків, включаючи "Вразливість XSS" (Cross-Site Scripting), "Вразливість SQL-ін'єкції," "CSS Injection," "Вразливі та застарілі компоненти" і "Вразливість CSRF" (Cross-Site Request Forgery). Кожна з цих вразливостей може бути використана зловмисниками для атаки на користувачів вебзастосунків або для отримання несанкціонованого доступу до їхніх даних.

Мета досліджень - виявити та оцінити вразливості вебзастосунків і проаналізувати деякі публічні вебсайти для виявлення можливих вразливостей.

Дослідження спрямовані на створення рекомендацій з підвищення рівня безпеки клієнтської частини вебзастосунків. Ці рекомендації включатимуть використання сучасних технологій та найкращих практик розробки, встановлення актуальних версій програмного забезпечення, використання спеціальних інструментів для виявлення вразливостей та дотримання сучасних стандартів кібербезпеки.

1 АНАЛІЗ ПОТОЧНИХ ТЕНДЕНЦІЙ І ПРОБЛЕМ

1.1 Поширення вразливостей

Поширення вразливостей в сучасних вебзастосунках є насущною проблемою в області кібербезпеки. Ця проблема стає особливо актуальною через швидкий розвиток і розповсюдження цифрових технологій.

Поширення вразливостей - це процес, коли конкретна вразливість, така як "Вразливість XSS" або "Вразливість SQL-ін'єкції," виявляється в одному вебзастосунку і, внаслідок недбалості або відсутності заходів безпеки, може поширюватися на інші вебзастосунки чи їхні компоненти, а також на користувачів, що взаємодіють із цими застосунками.

Сутність поширення вразливостей полягає в тому, що атаки або експлойти, спрямовані на вразливий вебзастосунок, можуть бути використані для атаки на інші вебзастосунки, з якими він взаємодіє, або на користувачів, що використовують ці застосунки. Одна вразливість може мати подвійний або навіть потрійний негативний ефект.

Поширення вразливостей стає особливо проблематичним через такі фактори:

— залежності між вебзастосунками: багато сучасних вебзастосунків побудовані з використанням сторонніх бібліотек, модулів і сервісів. Якщо хоча б один з цих компонентів містить вразливість, вона може поширитися на весь застосунок;

— збільшення кількості вебзастосунків: з ростом популярності і доступності веброзробки з'являється все більше вебзастосунків. Це створює більше можливостей для поширення вразливостей;

— недостатнє оновлення та виправлення помилок: власники вебзастосунків не завжди вчасно оновлюють і виправляють програмне забезпечення. Застарілі версії бібліотек і компонентів часто містять відомі вразливості, які можуть поширитися;

— спільнота користувачів: велика кількість користувачів вебзастосунків може стати жертвами атак, якщо їхні дані стають доступними для зловмисників через поширену вразливість.

Щоб запобігти поширенню вразливостей, важливо постійно відслідковувати стан безпеки вебзастосунків і їхніх компонентів, негайно виправляти виявлені вразливості та мати плани дій для реагування на можливі атаки. Захист від поширення вразливостей вимагає комплексного підходу, що включає технічні заходи безпеки, навчання користувачів та свідомості щодо кібербезпеки та оновлення програмного забезпечення та компонентів.

1.2 Складність виявлення вразливостей

Аналіз вразливостей у клієнтській частині вебзастосунків є складним завданням з-за кількох основних факторів:

1 Неконтрольований середовище: Код JavaScript виконується на стороні клієнта, і немає контролю над тим, як саме він виконується. Це означає, що можуть виникнути проблеми безпеки, які можуть бути важко помітити, так як вони не впливають на серверну частину вебзастосунку.

2 Складність аудиту коду: Великі вебзастосунки мають велику кількість клієнтського коду, що може бути розділеним на різні файли та бібліотеки. Аналізувати всі ці файли і знайти потенційні вразливості може бути часом та ресурсозатратним завданням.

3 Велика кількість відвідувачів і різні браузери: Вебзастосунки повинні працювати на різних браузерах і підтримувати велику кількість користувачів. Це може створити складності при виявленні і виправленні вразливостей, які можуть виникати тільки в певних браузерах або на певних платформах.

4 Асинхронні запити та обробка даних на клієнті: Багато сучасних вебзастосунків використовують асинхронні запити, які можуть робити взаємодію з сервером складною для аналізу. Також, обробка даних на клієнті може призвести до вразливостей, які важко виявити.

5 Взаємодія зі сторонніми сервісами та API: Багато вебзастосунків взаємодіють зі сторонніми сервісами і API, що може створювати потенційні шляхи для атак.

Для виявлення вразливостей у клієнтській частині вебзастосунків в основному використовуються інструменти та підходи, такі як статичний аналіз коду JavaScript, тестування на проникнення, використання браузерних розширень для аналізу запитів та відповідей, та ручний аналіз коду.

1 Статичний аналіз коду JavaScript - суть цього підхіду полягає в тому, щоб використовувати спеціалізовані інструменти для аналізу самого вихідного коду JavaScript на предмет потенційних вразливостей без його виконання.

2 Тестування на проникнення (Penetration Testing) – суть цього підходу в тому що це процес активного тестування вашого вебзастосунку на предмет вразливостей шляхом спроби використання можливих атак, які можуть використовувати зловмисники.

3 Використання браузерних розширень для аналізу запитів та відповідей – суть цього підходу в тому що деякі браузерні розширення дозволяють перехоплювати та аналізувати HTTP-запити та відповіді між клієнтом і сервером. Такий аналіз може допомогти виявити проблеми з безпекою.

4 Ручний аналіз коду – суть цього підходу в тому що цей підхід передбачає ретельний перегляд клієнтського коду JavaScript розробника або команди для виявлення потенційних вразливостей.

1.3 Вартість усунення вразливостей

Вартість усунення вразливостей в клієнтській частині вебзастосунків може значно варіюватися і залежить від декількох факторів:

1 Масштаб проекту: Вартість буде залежати від розміру вашого вебзастосунку. Якщо це невеликий односторінковий сайт, то вартість може бути відносно невеликою. Але для великих вебзастосунків з великою кількістю коду витрати можуть бути значно вищими.

2 Кількість вразливостей: Вартість залежить від кількості та серйозності виявлених вразливостей. Якщо вони незначні і легко усуваються, витрати будуть меншими, ніж у випадку серйозних вразливостей, які потребують великих зусиль для виправлення.

3 Складність виправлення: Вартість може зростати, якщо виправлення вразливостей потребують великої кількості часу та ресурсів. Наприклад, виправлення складних асинхронних атак або виправлення великої кількості файлів може бути дорогим.

4 Вимоги до безпеки: Якщо потрібно дотримуватися високих стандартів безпеки (наприклад, в індустрії фінансів або охорони здоров'я), це може призвести до додаткових витрат на аудит безпеки та відповідність стандартам.

5 Вид вебзастосунку: Витрати можуть різнитися в залежності від типу вебзастосунку. Наприклад, вартість виправлення вразливостей у важливій корпоративній системі може бути значно вищою, ніж у вебсайті з обмеженим функціоналом.

6 Методи виявлення вразливостей: Якщо використовувати високоякісні методи виявлення вразливостей, такі як статичний аналіз коду, тестування на проникнення та інші ефективні підходи, це може зекономити гроші у майбутньому.

7 Внутрішні або зовнішні ресурси: Вартість також залежить від того, чи використовуються власні розробники та інженери для виправлення вразливостей, чи наймаються зовнішні консультанти або фірми з безпеки.

1.4 Поява нових вразливостей

Поява нових вразливостей в клієнтській частині вебзастосунків є постійним процесом через різні фактори, такі як зміни в технологіях, розробці

програмного забезпечення, та еволюція загроз безпеці. Ось деякі типові фактори та приклади нових вразливостей:

1 Застарілі бібліотеки та фреймворки: Використання застарілих версій бібліотек та фреймворків у клієнтському коді може призвести до вразливостей, оскільки нові вразливості можуть бути виявлені і використані зловмисниками.

2 Кросс-сайтові атаки (XSS): Поява нових типів XSS-вразливостей викликана шляхом появи нових способів впровадження шкідливих скриптів на сторінку та недостатньої фільтрації введених даних користувачами.

3 Неправильна обробка введених даних: Нові способи обходу контролю введених даних і недостатня їх валідація можуть призвести до вразливостей, таких як SQL-ін'єкція або інші атаки на дані.

4 Хакерські атаки: Соціальна інженерія і атаки спрямовані на користувачів, такі як фішинг, можуть використовувати нові методи для отримання доступу до клієнтської частини вебзастосунку.

5 Нові браузери та технології: З появою нових версій браузерів та технологій, можуть з'явитися нові вразливості, а також можуть змінюватися вимоги до безпеки та підходи до захисту.

6 Нові стандарти та регуляції: Зміни в стандартах безпеки та регуляції можуть створювати нові вимоги та визначати, які види вразливостей слід враховувати при розробці вебзастосунків.

На рис. 1.1 показано статистику по відсоткам використання вразливостей знайдених за роком в атаках [1].

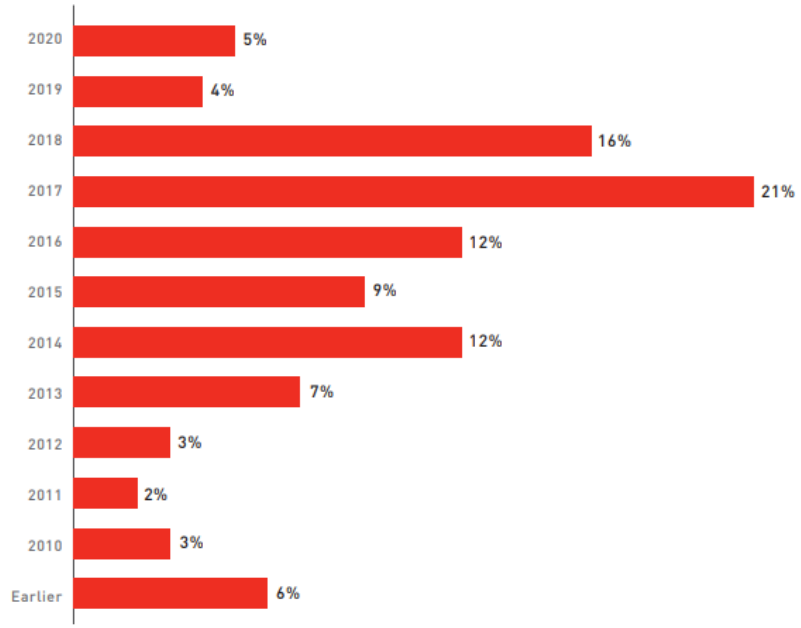


Figure 35: Percentage of Attacks Leveraging Vulnerabilities by Disclosure Year in 2020.

Рисунок 1.1 – Відсоток атак, які використовують вразливості, за роком розкриття інформації у 2020 році

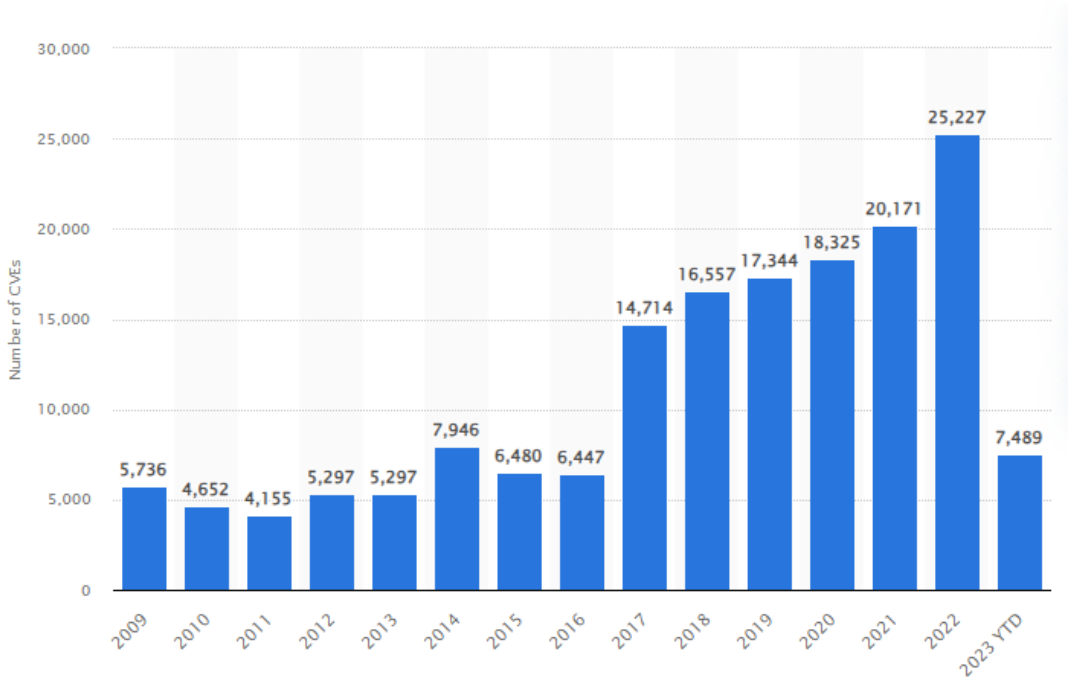


Рисунок 1.2 – Кількість поширених вразливостей у безпеці IT та ризиків (CVE) у всьому світі з 2009 по 2023 рік

На рис. 1.2 показано статистику знаходження вразливостей по рокам з 2009 по 2023 роки [2].

2 ПРОТОКОЛИ ВЗАЄМОДІЇ КОРИСТУВАЧА З КЛІЄНТСЬКОЮ ЧАСТИНОЮ ВЕБЗАСТОСУВАНЬ

2.1 Протокол JWT

Протокол JWT (JSON Web Token) - це відкритий стандарт (RFC 7519), який використовується для передачі інформації між двома сторонами у компактному і безпечному форматі. JWTs використовуються в основному для аутентифікації та авторизації вебдодатків та служб. Вони мають формат JSON і можуть бути підписані (зашифровані) для перевірки їх валідності та цілісності. JWT складається з трьох частин: заголовка (header), тіла (payload) і підпису (signature).

1 Заголовок (Header): Заголовок містить два основні поля: "alg" (алгоритм підпису) і "typ" (тип токена). Заголовок завжди представлений у форматі JSON і включений до JWT у кодировке Base64URL.

2 Тіло (Payload): тіло містить корисну інформацію, яка представлена у форматі JSON. В тілі можуть бути включені клейми (claims), які діляться на три основні категорії:

— registered Claims: стандартні клейми, які мають заздалегідь визначені значення. Наприклад, "iss" (видавець), "sub" (суб'єкт), "exp" (термін дії), і т. д.;

— public Claims: власні клейми, які може визначити користувач або розробник. Вони не повинні конфліктувати зі стандартними клеймами;

— private Claims: власні клейми, які використовуються для обміну інформацією між сторонами, які домовилися про їх значення.

3 Підпис (Signature): Підпис створюється шляхом підписування заголовка і тіла токена за допомогою секретного ключа або приватного ключа. Підпис дозволяє перевірити, чи була змінена інформація в JWT після підписування.

JWT використовується в багатьох сценаріях, включаючи:

— аутентифікація: вебдодатки можуть використовувати JWT для аутентифікації користувачів і надавання доступу до захищених ресурсів;

— одиночний вхід (Single Sign-On, SSO): JWT може використовуватися для SSO між різними додатками та службами;

— внутрішнє забезпечення даних: JWT може бути використаний для забезпечення даних, які передаються між мікросервісами у внутрішній мережі.

Однією з основних переваг JWT є його компактність та можливість передавати інформацію між сторонами без необхідності зберігання її на сервері. Однак важливо правильно використовувати JWT та надавати належну

увагу безпеці, оскільки некоректна настройка або недотримання безпечних практик може призвести до вразливостей.

2.2 Протокол HTTP/HTTPS

Протокол HTTP (Hypertext Transfer Protocol) та його захищена версія HTTPS (Hypertext Transfer Protocol Secure) є основними протоколами для обміну даними між вебклієнтами та вебсерверами.

HTTP - це простий протокол передачі даних, який використовується для отримання вебсторінок та інших ресурсів з вебсерверів.

Всі дані, передані за допомогою HTTP, передаються у текстовому форматі, що означає, що вони можуть бути з легкістю перехоплені та прочитані зловмисниками, якщо не використовується додаткове шифрування.

HTTPS - це розширення HTTP, яке використовує шифрування для забезпечення конфіденційності та цілісності передачі даних між вебклієнтом і вебсервером.

Шифрування забезпечується за допомогою протоколу TLS (Transport Layer Security) або його попередника SSL (Secure Sockets Layer). TLS шифрує дані, що передаються через мережу, і використовує сертифікати для перевірки автентичності сервера.

HTTPS вказується в URL за допомогою префіксу "https://" замість стандартного "http://".

Основні переваги HTTPS:

1 Захищена передача даних: Дані між вебклієнтом і сервером шифруються, що робить їх недоступними для зловмисників.

2 Перевірка автентичності сервера: Сертифікати SSL/TLS дозволяють перевірити, що ви спілкуєтеся з правильним сервером, а не зі зловмисником.

3 Підвищення довіри користувачів: Відображення зеленої адресної стрічки або індикації "безпеки" у браузері підвищує довіру користувачів.

4 Покращує рейтинг у пошукових системах: Багато пошукових систем, включаючи Google, враховують наявність HTTPS як фактор ранжування для вебсайтів.

5 Захищає від вразливості людина по середині на відкритих мережах: Якщо ви користуєтесь відкритими Wi-Fi мережами, HTTPS захищає ваші дані від вразливості людина по середині, які можуть намагатися перехопити вашу комунікацію.

HTTPS стає стандартом для більшості вебсайтів, особливо тих, де передаються конфіденційні дані, такі як дані користувачів або платежі. Вибір між HTTP і HTTPS визначається потребами та характером вебсайту, але для

забезпечення безпеки та конфіденційності даних користувачів рекомендується використовувати HTTPS [3].

2.3 Протокол WebSocket

Протокол WebSocket - це стандартний протокол комунікації між клієнтом і сервером через Інтернет, який дозволяє встановлювати постійне, повнодуплексне з'єднання для обміну даними в режимі реального часу. WebSocket відрізняється від традиційного HTTP протоколу, який ґрунтується на запит-відповідь і не забезпечує постійне з'єднання.

Основні характеристики та переваги протоколу WebSocket:

1 Постійне з'єднання: WebSocket дозволяє підтримувати відкрите з'єднання між клієнтом і сервером протягом тривалого часу, що дозволяє обмінюватися даними без необхідності встановлювати нові з'єднання для кожного запиту.

2 Реальний час: WebSocket підходить для роботи в режимі реального часу, такий як чат, онлайн-ігри, трансляція даних, споживання потокових медіаданих тощо.

3 Двосторонній обмін даними: WebSocket підтримує повноцінний двосторонній обмін даними, що дозволяє як клієнту, так і серверу ініціювати передачу даних.

4 Легкий протокол: WebSocket є легким протоколом з мінімальним накладанням на передачу даних, що робить його ефективним для великої кількості з'єднань.

5 Шифрування і безпека: WebSocket може використовувати шифрування, аби забезпечити конфіденційність передачі даних. Також можна встановити захист від підроблення підпису на рівні протоколу.

6 Спрощена робота з портами і проксі-серверами: WebSocket використовує стандартні порти HTTP (80 та 443) і добре працює через HTTP проксі-сервери.

7 Розширення і підтримка крос-платформ: WebSocket підтримує різноманітні розширення і багато мов та бібліотек мають підтримку для цього протоколу[4].

Приклади використання WebSocket включають вебчати, онлайн-гри, віддалену моніторинг та управління, споживання потокових медіаданих, а також роботу зі сенсорними пристроями в реальному часі. WebSocket робить взаємодію між вебдодатками більш динамічною та реактивною, дозволяючи створити багатоцільові додатки, які працюють в режимі реального часу.

2.4 Протокол WebRTC

Протокол WebRTC (Web Real-Time Communication) - це набір технологій та API, які дозволяють вебдодаткам встановлювати з'єднання в реальному часі для обміну аудіо-, відео- та даними між браузерами та іншими пристроями. WebRTC створений для реалізації веббазованих медіаприкладів, таких як відео-чати, аудіо-дзвінки, відео-конференції та інші додатки реального часу.

Основні компоненти та характеристики WebRTC:

1 Медіа-потоки: WebRTC дозволяє надсилати та отримувати аудіо- та відеопотоки в режимі реального часу між різними клієнтами. Це дозволяє створювати відео-чати та аудіо-дзвінки без необхідності встановлювати додаткові плагіни або додатки.

2 Peer-to-Peer з'єднання: WebRTC базується на концепції peer-to-peer (P2P) з'єднань, що дозволяє клієнтам підтримувати безпосереднє з'єднання один з одним, обходячи необхідність передавати трафік через центральний сервер. Це зменшує затримку та покращує якість взаємодії в реальному часі.

3 Шифрування даних: WebRTC забезпечує шифрування медіаданих за замовчуванням, що дозволяє забезпечити конфіденційність даних, які передаються між користувачами.

4 API для браузерів: WebRTC надає набір JavaScript API для веброзробників, які можуть використовувати їх для створення вебдодатків, що використовують медіапотоки.

5 Підтримка багатьох платформ: WebRTC підтримується багатьма веббраузерами, включаючи Google Chrome, Mozilla Firefox, Microsoft Edge та інші, що дозволяє створювати крос-платформені додатки.

6 Низька затримка: WebRTC оптимізований для реалізації медіаприкладів в реальному часі з низькою затримкою[5].

WebRTC широко використовується в різних додатках та сценаріях, таких як відео-конференції, відео-чати, віртуальні класи, онлайн-трансляції, технології віддаленого робочого місця та інше. Він відкриває нові можливості для веброзробників для створення додатків, які можуть спілкуватися в реальному часі, використовуючи медіа.

3 КЛАСИФІКАЦІЯ ВРАЗЛИВОСТЕЙ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКІВ

3.1 Вразливість XSS

XSS (Cross-site scripting) - це тип вразливості, що трапляється у web-додатках. XSS атаки дають змогу впровадити шкідливий скрипт (або, як його ще часто називають, експлойт) на сторінку застосунку, унаслідок чого у користувачів, які відвідують цю сторінку, можуть вкрасти дані різного ступеня чутливості: куки, сесійні токени, логіни з паролями і просто особисту інформацію про користувача.

Впровадити експлойт зловмисники можуть різними способами, наприклад залишити коментар під постом або товаром в онлайн-магазині, що містить скрипт. І, якщо розробники web-додатку не подбали про валідацію даних, то шкідливий скрипт запуститься у всіх користувачів, які відкрили коментарі на сторінці. Такий тип вразливості називається " збережений".

Також, напевно, популярніший спосіб, коли зловмисник передає шкідливий пейлоад прямо в посиланні на наш застосунок у параметрах запиту або в хеші, який читається в JS і може бути виконаний. Найчастіше це "відбиті" або "засновані на DOM" XSS атаки.

У сучасних застосунках, написаних на сучасних фреймворках (де здебільшого екрануються дані користувацького введення), стає дедалі менше і менше можливостей здійснити XSS атаку. Також і розробники браузерів не сидять на місці та покращують у них безпеку, наприклад за допомогою таких речей як:

1 SOP (Same-Origin Policy) - політика, що дає змогу визначити, які скрипти матимуть доступ до даних, а які ні, точніше навіть із яких ресурсів ми дозволяємо скриптам отримувати дані користувача. Простими словами допомагає заборонити скриптам з одного домену отримати доступ до даних користувача від іншого домену.

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

3 Валідація вхідних даних - сучасні браузери також самостійно валідують вхідні дані і не дають змогу нахабно прокидати скрипти, екрануючи їх.

Але попри все це XSS все ще можливий.

Існують три типи вразливостей XSS:

1 Stored (постійний або збережений) XSS: У цьому випадку шкідливий код зберігається на сервері та виводиться на сторінці, коли інший користувач переглядає цю сторінку. Наприклад, якщо зловмисник вставляє

шкідливий код у поле коментарів на блозі, інші користувачі, які читають цей коментар, можуть стати жертвами атаки.

Збережений XSS взаємодіє з жертвою через сервер застосунку. Найлегше пояснити принцип роботи за допомогою простого прикладу. Уявіть, що наш застосунок - це інтернет магазин, на якому розміщуються різні товари, у кожному товарі є розділ відгуків, у якому кожен користувач може залишити власний відгук. Якщо розробники не потурбувалися про безпеку цього розділу, а реалізовано його приблизно так:

Javascript код:

```
function addReviews() {
        var reviews = getReviews();
        reviews.forEach(function (review) {
                var reviewsList = document.getElementById("reviews");
          reviewsList.innerHTML += "<li>" + review + "</li>";
          })
}
addReviews()
```

HTML код:

```
<ul id=«reviews»></ul>
```

У такому разі зловмисник може додати свій відгук:

```
Гарний товар
<script>
   alert(document.cookie);
</script>
```

Далі в кожного, хто подивиться цей коментар, відобразиться тільки " Гарний товар", але крім цього вискочить alert з усіма його куками (замість цього зловмисник, найімовірніше, зробить запит, у тілі якого передасть усі видобуті дані про користувача до свого сервера)[8].

Така вразливість спрямована на велику кількість користувачів, тому що скрипт запуститься у всіх, хто відвідає сторінку. На відміну від "відбитого" XSS, для поширення якого часто потрібно застосовувати соціальну інженерію.

Приклад вище, звісно, максимально примітивний, і може здатися, що сьогодні таку вразливість зловити нереально, але хлопець знайшов вразливість XSS, що "збережена", у Microsoft Teams у 2021 році[11].

2 Reflected (відображений) XSS: Тут шкідливий код вставляється в URL-адресу і виконується при її обробці. Наприклад, зловмисник може створити посилання зі спеціальним параметром, який містить шкідливий код. Якщо користувач переходить за таким посиланням, то код виконується в його браузері.

У відображеному XSS реалізація доставки шкідливого скрипта виглядає інакше. Скрипт не повинен зберігатися на серверах додатка, він потрапляє

жертві через посилання. З цього випливає очевидний висновок, що кожному окремому користувачеві, на якого здійснюють атаку, потрібно дати таке посилання особисто, наприклад поштою або під час особистого листування, хоча посилання може бути розміщене на сайті зловмисника, на який у той чи інший спосіб потрапила жертва й тицьнула на нього. Але, знову ж таки, найімовірніше, на цей сайт ви потрапили за посиланням з email'a або з особистого листування.

Далі за цим посиланням (у якому в query параметрах зашитий скрипт) ми потрапляємо на сторінку, яку сформував сервер, відштовхуючись від змісту посилання і додаючи в сторінку всі ті параметри, що в ній є. Не важко здогадатися, що скрипт, який додав зловмисник до параметрів, теж потрапить у сформований HTML і благополучно запуститься у жертви. Тут уже зловмисник може відправити собі ваші куки або зібрати інші чутливі дані зі сторінки і теж відправити їх собі.

Звісно, скрипт не з будь-якого query параметра потрапить на сторінку і запуститься, у нас має бути ще й "особлива" реалізація роботи з цим параметром у застосунку.

Порівняно зі збереженим XSS, ця вразливість має менше охоплення, оскільки атаці піддається тільки той, хто перейшов за посиланням зі скриптом. У той час як збереженій XSS атаці піддається будь-хто, хто відвідав сторінку, на якій розмістили експлойт. Але й виявити таку вразливість складніше, оскільки її не вийде виявити за допомогою статичного аналізу.

З DOM-based (DOM-оснований) XSS: У цій вразливості шкідливий код впроваджується в об'єкт Document Object Model (DOM), який використовується для динамічної зміни вмісту сторінки на стороні клієнта. Вразливість виникає через недостатню обробку введених даних на клієнтській стороні, і код виконується в браузері користувача.

Такий тип XSS атак націлений безпосередньо на впровадження скрипта в DOM-дерево застосунку саме під час відпрацювання JS. Наприклад, як і у випадку з відбитим XSS, ми можемо прокинути шкідливий скрипт через query параметр. Але, на відміну від збереженого, наш застосунок не додасть цей скрипт у HTML і поверне користувачеві сторінку без експлойта.

Після того як клієнт завантажив сторінку і всі необхідні для її роботи файли, у клієнта запускається JS, який для тих чи інших цілей звертається до document.location, щоб дістати звідти дані з query параметра (в якому лежить наш шкідливий скрипт) і, наприклад, додати ці дані на сторінку користувача. Після цього скрипт запускається, маючи своєю чергою доступ до особистих даних користувача.

З огляду на те, що в сучасній веб індустрії більшість застосунків - це SPA, реалізовані на React, Angular, Vue та інших фреймворках, де більшу частину логіки перенесено на бік клієнта, а сторінки формують саме на основі

роботи JS у браузері, XSS вразливості на основі DOM трапляються найчастіше.

Наприклад ми працюємо над банківським вебдодатком, у якому є розділ відповідей на запитання, кожна з відповідей у ньому має вигляд окремої статті. Банк досить великий, відповідно питань до нього теж багато. В якийсь момент приймається рішення, що для зручності потрібно зробити пошук по розділу:

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

Але також потрібна можливість ділитися посиланням на результат пошуку. Наприклад, якщо клієнт банку звертається в службу підтримки, щоб співробітник міг відразу надати посилання зі знайденими відповідями по темі, що цікавить клієнта.

```
function updateSearchQueryParam(value) {
  const queryParams = new URLSearchParams(window.location.search);
  queryParams.set('search', value);
  const newPath = window.location.pathname + '?' + queryParams.toString();
  history.pushState(null, '', newPath);
}
function updateSearchSubtitle() {
  const searchFromQuery = newURLSearchParams(window.location.search)
  .get('search');
  const searchSubtitle = document.getElementById('searchSubtitle');
  if(searchFromQuery) {
    searchSubtitle.innerHTML = 'Результати пошуку за запитом' +
    searchFromQuery;
  }
}
```

Функцію updateSearchQueryParam ми викликаємо щоразу, коли здійснюємо пошук, щоб записати в query параметр те, що шукаємо. А функцію updateSearchSubtitle також викликаємо під час кожного пошуку, а також під час завантаження сторінки, щоб якщо в query параметрі щось було, ми це відобразили.

Ось у цей момент у нашу реалізацію пробралася вразливість. Побачивши параметр пошуку в посиланні і те, що його вміст потрапляє на сторінку, ми можемо спробувати передати скрипт з alert і побачити повідомлення на сторінці. Замість alert ми можемо зробити щось страшніше і, наприклад, відправити собі куки користувача.

Також є інші типи XSS атак але вони не такі поширені:

– mXSS або XSS з мутаціями — досить свіжий вид XSS атаки, про який вперше згадали в 2013 році. Для реалізації такої атаки потрібні глибокі знання в тому, як працюють браузери і які механізми вони використовують для боротьби з XSS. Цей вид атаки експлуатує механізм очищення та санітайзингу введення користувача браузером. Таким чином на вигляд неробочий скрипт, після проходження очищення браузером стає цілком валідним і може завдати шкоди клієнту, та й компанії, в цілому;

– blind XSS (Сліпа XSS) - це підмножина XSS, що зберігається, тільки запуститися експлойт може далеко не відразу і навіть не обов'язково в тому ж додатку. Наприклад, через форму зворотнього зв'язку, зловмисник відправляє відгук або питання, яке вбудовує скрипт. Далі співробітник служби підтримки відкриває це повідомлення, після чого і запускається скрипт. Це запросто може бути інша програма, який-небудь сервіс для адміністрування сайту;

– self XSS - вид атаки, при якому жертва самостійно запускає шкідливий скрипт на своєму пристрої, в основному, в консолі браузера. Виглядає це приблизно так: зловмисник робить розсилку в соцмережах, поштою або навіть в особистому повідомленні, де, з метою безпеки, або щоб допомогти зламати сайт (мовляв, зроби це і отримаєш доступ до обліку своєї колишньої), пропонує відкрити консоль і ввести код, який швидше за все надішле ваші персональні дані шахраям, ось власне і все. Таку вразливість по правді кажучи складно взагалі назвати вразливістю, оскільки це соціальна інженерія.

3.2 Вразливість SQL-ін'єкції

Вразливість SQL-ін'єкції (SQL Injection) - це тип атаки на безпеку вебдодатків, коли зловмисники використовують недостатньо оброблені введені дані для внесення та виконання SQL-запитів в базі даних. Якщо вебдодаток не належним чином валідує та екранує введені дані в запитах, то атакувач може вставити зловмисний SQL-код в запит, що може призвести до витоку, модифікації або видалення даних в базі даних, принцип роботи зображений на рис. 3.1. Це може мати серйозні наслідки, включаючи витік конфіденційної інформації, втрату даних та навіть може призвести до відмови в обслуговуванні (DoS) вебдодатку.

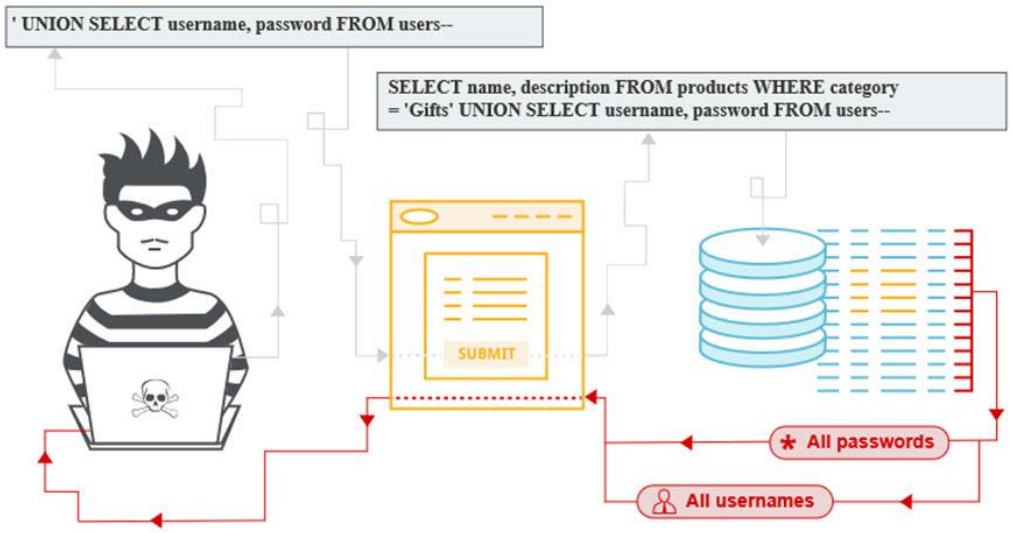


Рисунок 3.1 – Принцип використання SQL-ін'єкції

Основні причини вразливості SQL-ін'єкції включають:

1 Неналежна обробка введених даних: Введені дані користувача не належним чином валідуються та екрануються перед виконанням SQL-запитів.

2 Склеювання SQL-запитів: Введені дані додаються безпосередньо до SQL-запиту, що робить легкою атаку через введення спецсимволів.

3 Відсутність перевірки прав доступу: Користувачі можуть мати доступ до бази даних з привілеями, які не належать їм.

4 Помилки в обробці помилок: Вебдодаток може виводити повідомлення про помилку, які містять конфіденційну інформацію, таку як структура SQL-запиту.

Для захисту від SQL-ін'єкцій потрібно використовувати наступні практики:

1 Використовування параметризованих запитів (Prepared Statements): Замість склеювання SQL-запитів з введеними даними використовувати параметризовані запити, які автоматично екранують введені дані.

2 Валідація введених даних: Валідування введених даних на предмет відповідності очікуваному формату та обмеження доступу до даних.

3 Використовування ORM (Object-Relational Mapping): Використання ORM-систем дозволяє робити запити до бази даних без необхідності працювати зі сирим SQL.

4 Активація контролю доступу: Дотримування принципів "найменших привілеїв" і обмежування доступу користувачів до тільки необхідних даних та функціоналу.

5 Позбавлення від виведення помилок виробник: Відключення виведення докладних помилок виробника на продуктивному сервері.

3.3 CSS Injection

CSS Injection - це тип атаки на безпеку вебдодатків, при якій атакувач вводить вредонесний код у мові CSS (Cascading Style Sheets) в вебдодаток. CSS використовується для визначення вигляду та стилізації вебсторінок. Вразливість CSS Injection виникає, коли введені користувачем дані не належним чином валідуються або екрануються перед їх виведенням на сторінці, принцип роботи зображено на рис 3.2. В результаті атаки атакувач може змінювати відображення вебсторінки на стороні клієнта, призводячи до потенційно шкідливих наслідків.

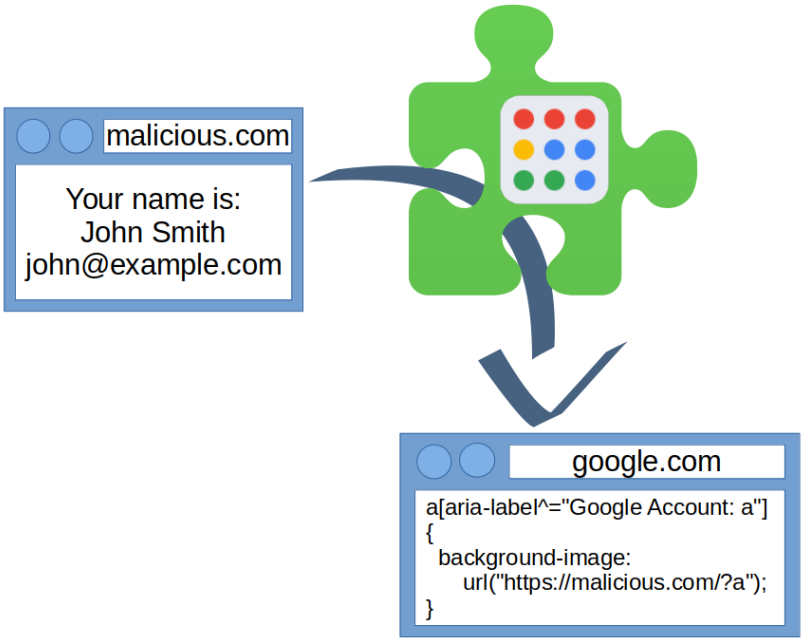


Рисунок 3.2 – Принцип атаки за допомогою CSS Injection

Основні характеристики та наслідки CSS Injection включають таке:

1 Зміна вигляду сторінки: Атакувач може вставляти CSS-код, який змінює колір, розмір шрифту, розташування або інші стилізаційні аспекти сторінки. Це може призвести до низького відображення вебсайту або виведення неавторизованих повідомлень.

2 Фішинг: Атакувач може використовувати CSS Injection для створення підроблених елементів інтерфейсу, які намагаються обдурити користувачів або надати фішинговий контент, що спрямований на виклик певних дій у користувачів.

3 Вразливості безпеки інших користувачів: Якщо вебдодаток дозволяє вставляти CSS-код в великій кількості місць, це може бути використано для вразливостей Cross-Site Scripting (XSS) або інших атак на безпеку.

4 Загроза конфіденційності даних: Атака CSS Injection може призвести до витоку конфіденційної інформації, так як атакувач може

змінювати відображення об'єктів, що мають конфіденційний характер.

Щоб захистити вебдодаток від CSS Injection, рекомендується дотримуватися таких практик безпеки:

1 Екранування введених даних: Всі дані, які вставляються в CSS, повинні бути безпечно екрановані, щоб уникнути внесення спецсимволів або вредонесного коду.

2 Використовуйте лише валідовані класи і ідентифікатори: Веброзробники повинні використовувати лише класи та ідентифікатори, які були попередньо валідовані та визначені в CSS.

3 Використовуйте заголовки безпеки: Забезпечте належні заголовки безпеки, такі як Content Security Policy (CSP), які можуть обмежити можливості вставки сторонніх ресурсів на сторінці.

4 Валідуйте та фільтруйте введені дані на сервері: Перевіряйте дані, які вставляються в CSS, на серверному рівні перед їх використанням.

Виконання цих практик безпеки допоможе захистити ваш вебдодаток від CSS Injection і зберегти правильну стилізацію та безпеку сторінки [7].

3.4 Вразливі та застарілі компоненти

Вразливі та застарілі компоненти - це важливий аспект безпеки в програмному забезпеченні, включаючи вебдодатки. Ці компоненти можуть бути сторонніми бібліотеками, фреймворками, плагінами або будь-якими складовими, які використовуються в вашому додатку для виконання певних функцій. Застарілі компоненти можуть містити вразливості безпеки, які можуть бути використані зловмисниками для атак на додаток.

Ось детальніше про вразливі та застарілі компоненти:

1. Вразливі компоненти:

– це компоненти, які мають відомі вразливості безпеки;

– вразливості можуть бути публічно відомими або виявленими при аудиті програмного забезпечення;

– вразливі компоненти можуть бути використані зловмисниками для внесення змін в роботу додатку, виконання коду або інших атак.

2. Застарілі компоненти:

– це компоненти, які використовують застарілі версії фреймворків, бібліотек або плагінів;

– застарілі компоненти можуть бути менш стійкими до вразливостей безпеки через відсутність оновлень та патчів виробника;

– застарілі компоненти можуть включати в себе сторонні бібліотеки, старі версії вебсерверів, баз даних або інші компоненти.

Наслідки використання вразливих та застарілих компонентів можуть включати в себе:

– витік конфіденційної інформації;
– виконання зловмисного коду;
– втрату доступу до системи або даних;
– порушення відповідності регулятивним вимогам;

Для запобігання цим проблемам, повинні виконуються наступні дії:

1 Моніторинг вразливостей: Використовуватися інструменти для виявлення вразливих та застарілих компонентів у програмному забезпеченні.

2 Оновлення та патчі: Регулярно оновлюватися компоненти до останніх безпечних версій та застосовуватися патчі безпеки.

3 Захисні шари: Додаватися захисні шари на рівні системи та додатку для запобігання атак на вразливі компоненти.

4 Аудит безпеки: Регулярно проводитися аудити безпеки для виявлення можливих вразливостей.

5 Служба підтримки від виробників: Встановлення зв'язку з виробниками компонентів і відслідковування оголошення про вразливості та оновлення.

6 Валідація вхідних даних: Всі введені дані, які передаються в компоненти, повинні бути валідовані і екрановані.

3.5 Вразливість CSRF

Вразливість Cross-Site Request Forgery (CSRF або XSRF) - це тип атаки на безпеку в вебдодатках, при якій зловмисний користувач (атакувач) змушує авторизованого користувача виконувати небажані дії на вебсайті або додатку без його на то наміру, принцип роботи зображений на рисунку 3.3. Атака CSRF може призвести до виконання небезпечних дій, таких як зміна паролю, створення небажаних дій, які потрібно авторизованому користувачеві.

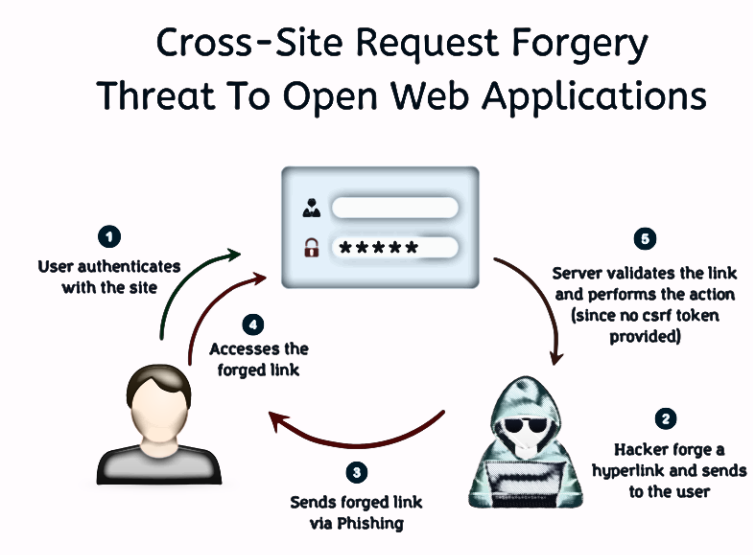


Рисунок 3.3 – Принцип CSRF атаки

Основні кроки, які призводять до атаки CSRF:

1 Злоумисник створює підроблену HTML-сторінку або email, яка містить код або посилання на виконання дій на цільовому вебсайті (наприклад, зміну паролю).

2 Злоумисник намагається надіслати цю сторінку або посилання жертві (авторизованому користувачу).

3 Якщо жертва відкриває сторінку або переходить за посиланням, її браузер надсилає HTTP-запит на виконання цієї дії на цільовому вебсайті.

4 Цільовий вебсайт не розрізняє, чи викликав цей запит жертва, чи атакувач, і виконує запит, таким чином, вразливість CSRF використовується для зміни налаштувань або виконання інших дій на вебсайті в ім'я авторизованого користувача.

Щоб захистити вебдодаток від атак CSRF, рекомендуються такі заходи:

1 Використовування випадковий токен: Вставлення випадкового токену в форми або посилання, які вимагають авторизації. Цей токен має бути унікальним для кожного користувача та запиту, і він перевіряється при обробці запиту.

2 Використовування методу POST: Використовування HTTP-методу POST замість GET для запитів, які здійснюють дії зі значущим впливом на вебсайт. GET-запити, як правило, не повинні виконувати зміни на сервері.

3 Валідування джерела запиту: Перевірення HTTP-заголовків та джерел запиту, щоб переконатися, що запити приходять від автентичних користувачів.

4 Підтвердження для дій зі значущим впливом: Вимагання додаткового підтвердження (наприклад, введення пароля або підтвердження електронною поштою) перед виконанням дій, які можуть мати великий вплив на вебсайт.

5 Використовування заголовків безпеки: Використання заголовків безпеки, таких як SameSite-кукі, які можуть обмежити використання кукі в небезпечних контекстах[9].

4 ДОСЛІДЖЕННЯ ВРАЗЛИВОСТЕЙ ВЕБЗАСТОСУНКІВ

4.1 Методи виявлення та оцінка вразливостей вебзастосунків

OWASP ZAP (OWASP Zed Attack Proxy) - це безкоштовний інструмент для перевірки безпеки вебзастосунків, який може виявляти широкий спектр вразливостей, включаючи XSS, SQL-ін'єкції, CSS-ін'єкції та застарілі компоненти.

Я вибрав OWASP ZAP для виявлення та оцінки вразливостей вебзастосунків з наступних причин:

— відкритий вихідний код. OWASP ZAP є відкритим вихідним кодом, що означає, що його можна безкоштовно використовувати та модифікувати;

— широкий спектр можливостей. OWASP ZAP може виявляти широкий спектр вразливостей, що робить його ефективним інструментом для повної перевірки безпеки вебзастосунків;

— простота використання. OWASP ZAP простий у використанні, що дозволяє навіть початківцям проводити аналіз безпеки вебзастосунків.

— можливість використання проксі для сайтів які заблоковані на території України.

В OWASP ZAP є свої групи вразливостей які будуть використовуватися для оцінювання вразливостей. В залежності від рівня ризику вразливості позначаються своєю колірною гамою. У OWASP ZAP існує 5 категорій, рис. 4.1:

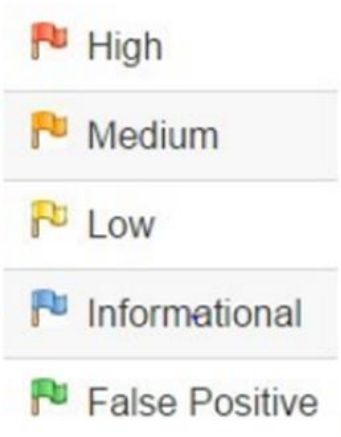


Рисунок 4.1 – Критерії оцінювання загроз OWASP ZAP

High – найвищий (критичний) рівень вразливостей. Сюди відносять SQL-ін'єкції, атаки типу path traversal, LDAP-ін'єкції, Spring4Shell та інші. Позначається червоним прапорцем.

Medium – середній рівень критичності знайдених вразливостей. Сюди

відносять Directory Browsing, HTTPOnly, XSLT-ін'єкції та інші. Позначається оранжевим прапорцем.

Low – низький рівень знайдених вразливостей. Сюди відносять знайдену інформацію, пов'язану з налагодженням, розкриття приватних IP-адрес (Private IP Disclosure) та інші. Позначається жовтим прапорцем.

Informational — інформаційний рівень, який повідомляє про те, що знайдено інформацію про компоненти вебсайту, що використовуються, а також про сервер. Сюди належать версії вебсервера, СУБД, бібліотек, фреймворків, а також версії мов програмування та операційних систем. Як правило, вразливості, що належать до цієї категорії, не можуть безпосередньо нашкодити системі, однак підвищують ризик збору інформації про сайт та його компоненти. Позначається синім прапорцем.

False Positive - помилкові спрацьовування. У поодиноких випадках ZAP може знайти вразливість, яка не є вразливістю. Якщо виникає сумнів, то програма позначає таку вразливість зеленим прапорцем. Однак користувач повинен сам переконатися та перевірити, чи є знайдена вразливість такою чи ні.

Для оцінки будуть використовуватися тільки вразливості критичного та середнього рівня.

На основі оцінки критичності вразливості будуть розроблені рекомендації щодо її усунення. Ці рекомендації повинні бути такими, щоб вразливість не могла бути використана зловмисниками.

Для більшої ефективності було додано плагін «DOM XSS Active scanner rule» щоб перевіряти на вразливість DOM-based XSS яка була описана в пункті 3.1.

OWASP ZAP є ефективним інструментом для виявлення та оцінки вразливостей вебзастосунків. Використовуючи цей інструмент є змога провести повний аналіз безпеки вебзастосунків і розробити рекомендації щодо усунення виявлених вразливостей.

Skipfish - це безкоштовний, відкритий вихідний код, активний інструмент сканування вебдодатків, який використовується для виявлення потенційних вразливостей у вебдодатках. Skipfish використовує рекурсивне сканування для виявлення всіх доступних сторінок і файлів вебдодатку, а потім використовує набори проб для виявлення потенційних вразливостей.

Skipfish використовує чотири рівні загроз для класифікації виявлених вразливостей:

— high: Ці вразливості можуть призвести до повної компрометації вебдодатку, включаючи крадіжку даних користувачів або виконання довільного коду;

— medium: Ці вразливості можуть призвести до обмеженої компрометації вебдодатку, наприклад, до крадіжки конфіденційної інформації

або можливості виконувати шкідливий код;

— low: Ці вразливості можуть призвести до незначної компрометації вебдодатку, наприклад, до можливості отримати доступ до неавторизованих ресурсів;

— info: Ці вразливості не є безпосередніми вразливостями, але можуть бути використані для виявлення інших вразливостей.

Ось деякі приклади вразливостей, які можуть бути класифіковані як High-ризик за допомогою Skipfish:

— sql-ін'єкція: Ця вразливість дозволяє зловмиснику ін'єктувати власний код у базу даних вебдодатку. Це може призвести до крадіжки даних користувачів або виконання довільного коду;

— cross-site scripting (XSS): Ця вразливість дозволяє зловмиснику вставити шкідливий код у сторінку вебдодатку. Це може призвести до крадіжки даних користувачів або виконання довільного коду;

— remote code execution (RCE): Ця вразливість дозволяє зловмиснику виконати довільний код на сервері вебдодатку. Це може призвести до повної компрометації вебдодатку.

Ось деякі приклади вразливостей, які можуть бути класифіковані як Medium-ризик за допомогою Skipfish:

— file inclusion: Ця вразливість дозволяє зловмиснику включати зовнішні файли у сторінку вебдодатку. Це може призвести до крадіжки даних користувачів або виконання шкідливого коду;

— insecure direct object references: Ця вразливість дозволяє зловмиснику отримати доступ до захищених ресурсів вебдодатку. Це може призвести до крадіжки даних користувачів або виконання довільного коду;

— session hijacking: Ця вразливість дозволяє зловмиснику перехопити сеанс користувача вебдодатку. Це може призвести до крадіжки даних користувачів або виконання довільного коду.

Ось деякі приклади вразливостей, які можуть бути класифіковані як Low-ризик за допомогою Skipfish:

— insecure CAPTCHA: Ця вразливість дозволяє зловмиснику легко пройти CAPTCHA, яка використовується для захисту вебдодатку. Це може призвести до доступу до захищених ресурсів вебдодатку;

— insecure cookie settings: Ця вразливість позволяет зловмиснику отримати доступ до куків користувача вебдодатку. Це може призвести до крадіжки даних користувачів;

— insecure file upload: Ця вразливість дозволяє зловмиснику завантажити шкідливі файли на сервер вебдодатку. Це може призвести до виконання шкідливого коду.

Ось деякі приклади вразливостей, які можуть бути класифіковані як Info-ризик за допомогою Skipfish:

— use of outdated software: Ця вразливість дозволяє зловмиснику використовувати відомі вразливості в програмному забезпеченні вебдодатку;

— insecure configuration: Ця вразливість дозволяє зловмиснику використовувати неправильні налаштування вебдодатку;

— insecure design: Ця вразливість дозволяє зловмиснику використовувати слабкий дизайн вебдодатку.

Skipfish - це потужний інструмент, який можна використовувати для виявлення потенційних вразливостей у вебдодатках. Важливо правильно використовувати Skipfish, щоб отримати точні результати. Ці два інструменти були обрані за їх параметри які зображені в таблиці 4.1.

Таблиця 4.1 – Порівняння OWASP ZAP та Skipfish

Параметр	OWASP ZAP	Skipfish
Тип сканування	Пасивне, активне, ручне	Динамічне
Широта покриття	Широке	Середнє
Глибина сканування	Середня	Глибока
Точність	Середня	Середня
Легкість використання	Середня	Середня
Вартість	Безкоштовний	Безкоштовний

4.2 Сайт nure.ua

За допомогою OWASP ZAP було проведено аналіз сайту, рис. 4.2: https://nure.ua/.

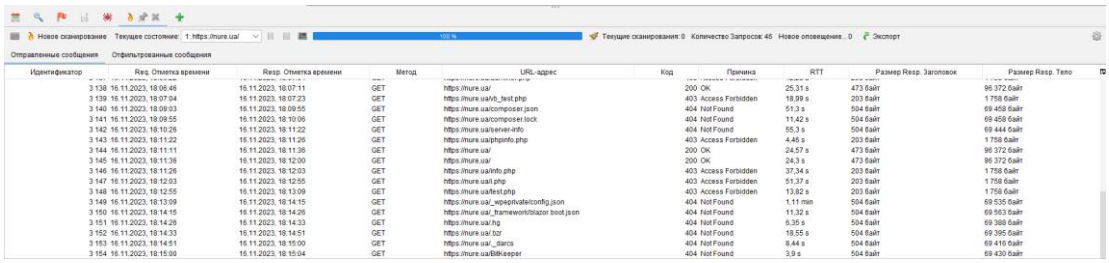


Рисунок 4.2 – Аналіз сайту https://nure.ua/

Було знайдено всього 12 вразливостей, рис. 4.3:

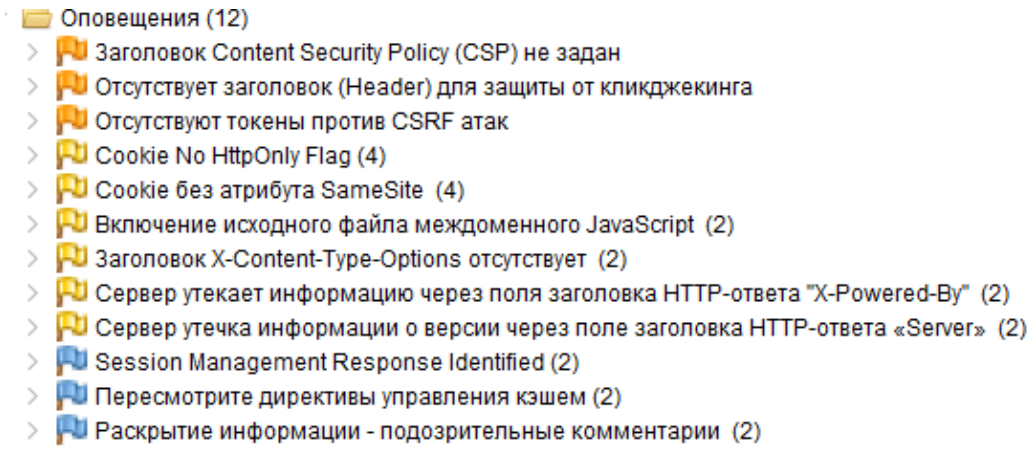


Рисунок 4.3 – Знайдені вразливості

Серед цих вразливостей 3 середнього рівня, 6 низького рівня та 3 інформаційного рівня.

1 Перша вразливість середнього рівня «Заголовок Content Security Police (CSP) не задано», рис. 4.5, вона є серйозною вразливістю безпеки вебзастосунків, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні.

Content Security Policy (CSP) - це механізм безпеки, який дозволяє веброзробникам контролювати, які ресурси можуть бути завантажені в вебзастосунок. CSP використовується для захисту від атак типу межсайтової скриптингу (XSS), атак на ін'єкцію, атак на відмову в обслуговуванні (DoS) та інших атак, принцип роботи зображений на рис. 4.4.

Якщо заголовок CSP не задано, вебзастосунок не буде захищатися від цих атак. Це означає, що зловмисник може завантажити шкідливий код на вебзастосунок, який може бути використаний для крадіжки даних користувачів, впровадження шкідливого програмного забезпечення або навіть для відключення вебзастосунку.

Рисунок 4.4 – Принцип роботи Content Security Police (CSP)

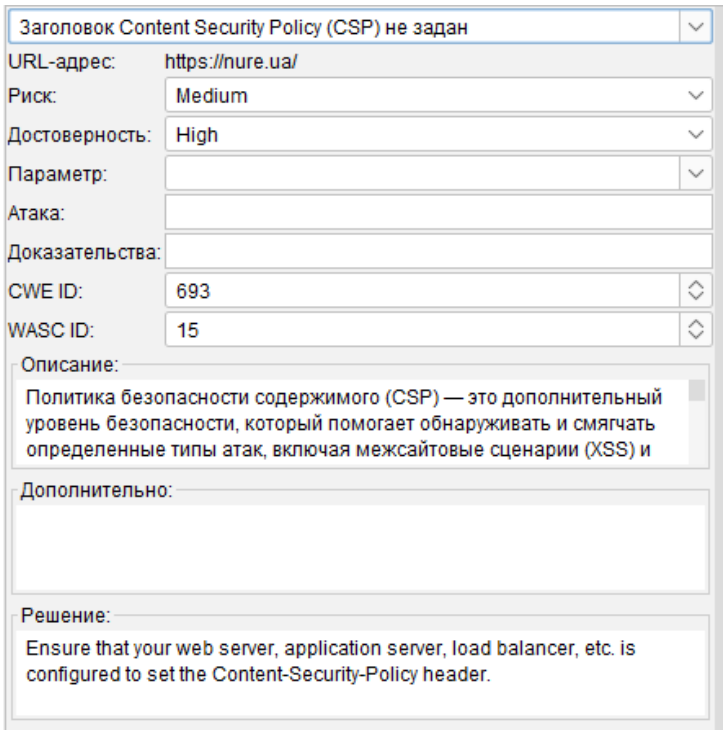


Рисунок 4.5 – Вразливість середнього рівня «Заголовок Content Security Police (CSP) не задано»

2 Наступна вразливість середнього рівня «Відсутній заголовок (Header) для захисту від клікджекінгу», рис. 4.7, є серйозною вразливістю безпеки вебзастосунків, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні.

Клікджекінг - це тип атаки, при якому зловмисник обманом перенаправляє користувачів на інший вебсайт, не повідомивши їм про це. Ця атака може бути використана для крадіжки даних користувачів, впровадження шкідливого програмного забезпечення або навіть для поширення дезінформації, принцип роботи бачимо на рис. 4.6.

Щоб захистити свій вебзастосунок від клікджекінгу, веброзробники повинні додавати заголовок Referrer-Policy до своїх вебзастосунків. Цей заголовок вказує веббраузеру, як він повинен передавати інформацію про джерело запиту.

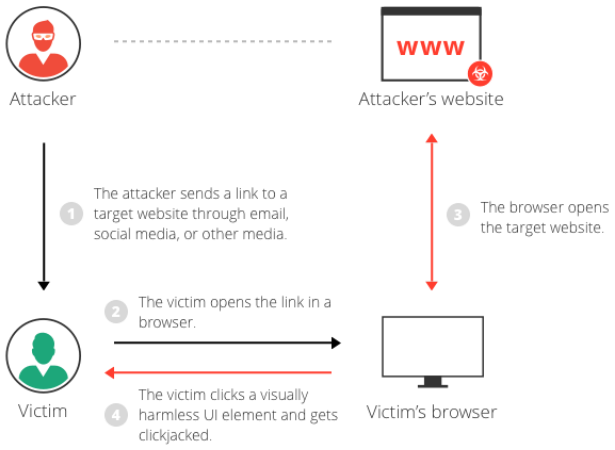


Рисунок 4.6 – Принцип роботи клікджекінгу

Якщо заголовок Referrer-Policy не задано, веббраузер буде передавати повну інформацію про джерело запиту. Це означає, що зловмисник може використовувати цю інформацію для перенаправлення користувачів на свій вебсайт.

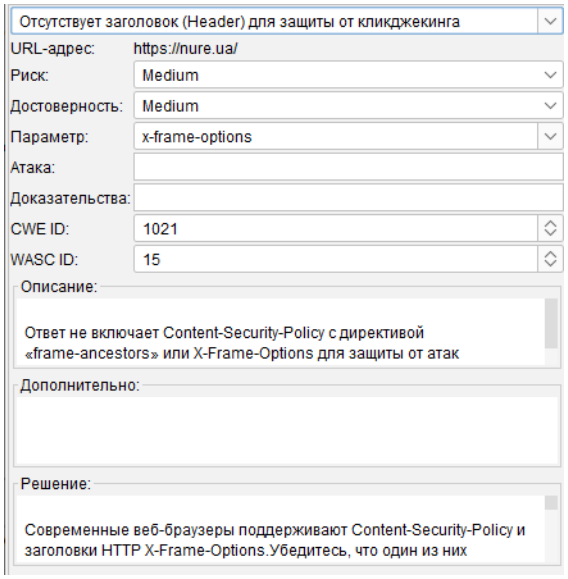


Рисунок 4.7 - Вразливість середнього рівня «Відсутній заголовок (Header) для захисту від клікджекінгу»

3 Остання вразливість середнього рівня - це вразливість «Відсутні токени проти CSRF атак» рис. 4.8, є серйозною вразливістю безпеки вебзастосунків, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні.

CSRF (Cross-Site Request Forgery) - це тип атаки, при якому зловмисник обманом змушує користувача виконати несанкціонований запит до вебзастосунку. Ця атака може бути використана для крадіжки даних користувачів, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні.

Щоб захистити свій вебзастосунок від CSRF атак, веброзробники

повинні використовувати токени CSRF. Токени CSRF - це унікальні значення, які генеруються сервером і передаються користувачеві. Коли користувач надсилає запит до вебзастосунку, він повинен також включити токен CSRF у запит. Сервер потім перевіряє токен CSRF, щоб переконатися, що запит був зроблений користувачем, який його авторизував.

Якщо токени CSRF не використовуються, вебзастосунок буде вразливий до CSRF атак. Це означає, що зловмисник може створити вебсторінку, яка містить шкідливий код. Коли користувач відвідає цю вебсторінку, шкідливий код буде автоматично виконаний на комп'ютері користувача. Шкідливий код може бути використаний для крадіжки даних користувача, впровадження шкідливого програмного забезпечення або навіть для відключення вебзастосунку.

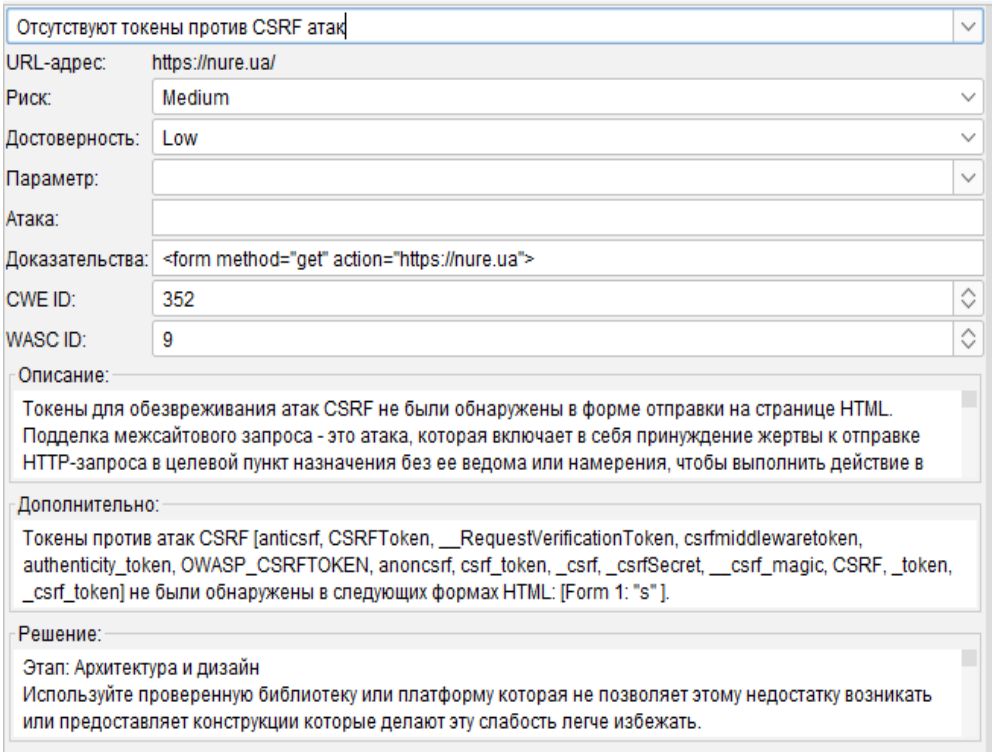


Рисунок 4.8 - Вразливість середнього рівня «Відсутній заголовок (Header) для захисту від клікджекінгу»

Загалом цей сайт можна вважати безпечним, тому що ці вразливості не з великим ризиком, а отже ризик їх використання зловмисником не є занадто великим.

За допомогою skipfish було проведено аналіз сайту: https://nure.ua/.

Було знайдено 10 вразливостей середнього рівня, 6 вразливостей низького рівня та 9 інформативного рівня, рис. 4.9.

Вразливість «External content embedded on a page» (вбудований зовнішній вміст на сторінці) - це тип вразливості безпеки вебдодатків, яка виникає, коли вебдодаток дозволяє користувачам включати зовнішній вміст у сторінки. Цей зовнішній вміст може бути у вигляді зображень, відео, скриптів

або інших файлів.

Ця вразливість може бути використана зловмисниками для кількох цілей, включаючи:

— крадіжку даних користувачів: Зловмисник може використовувати зовнішній вміст для крадіжки даних користувачів, таких як імена користувачів, паролі або інші конфіденційні дані. Наприклад, зловмисник може вставити шкідливий скрипт у зовнішній файл, який може викрасти дані користувачів, які вводять їх на вебсайті;

— виконання шкідливого коду: Зловмисник може використовувати зовнішній вміст для виконання шкідливого коду на комп'ютері користувача. Наприклад, зловмисник може вставити шкідливий скрипт у зовнішній файл, який може завантажити та запустити шкідливе програмне забезпечення на комп'ютері користувача;

— розсилка спаму або шкідливих посилань: Зловмисник може використовувати зовнішній вміст для розсилки спаму або шкідливих посилань користувачам. Наприклад, зловмисник може вставити шкідливий скрипт у зовнішній файл, який може автоматично додати користувача до списку розсилки спаму або перенаправити користувача на шкідливий вебсайт.

Рисунок 4.9 – Вразливості знайдені за допомогою skipfish

Наслідки вразливості «External content embedded on a page» можуть бути серйозними. Якщо ця вразливість не буде виправлена, зловмисник може використовувати її для крадіжки даних користувачів, виконання шкідливого коду або інших шкідливих дій.

Зі списку вразливостей skipship сайт можна вважати безпечним тому що загрози середнього рівня направлені на сайти з великим захистом, тому можна не хвилюватися.

4.3 Сайт drom.ru

За допомогою OWASP ZAP було проведено аналіз сайту, рис. 4.10: https://www.drom.ru/.

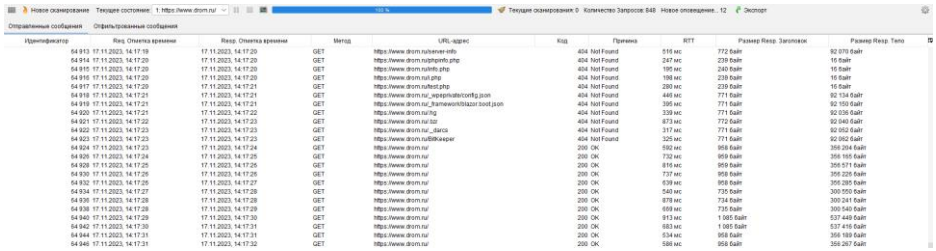


Рисунок 4.10 - Аналіз сайту https://www.drom.ru/

Було знайдено всього 19 вразливостей, рис. 4.11:

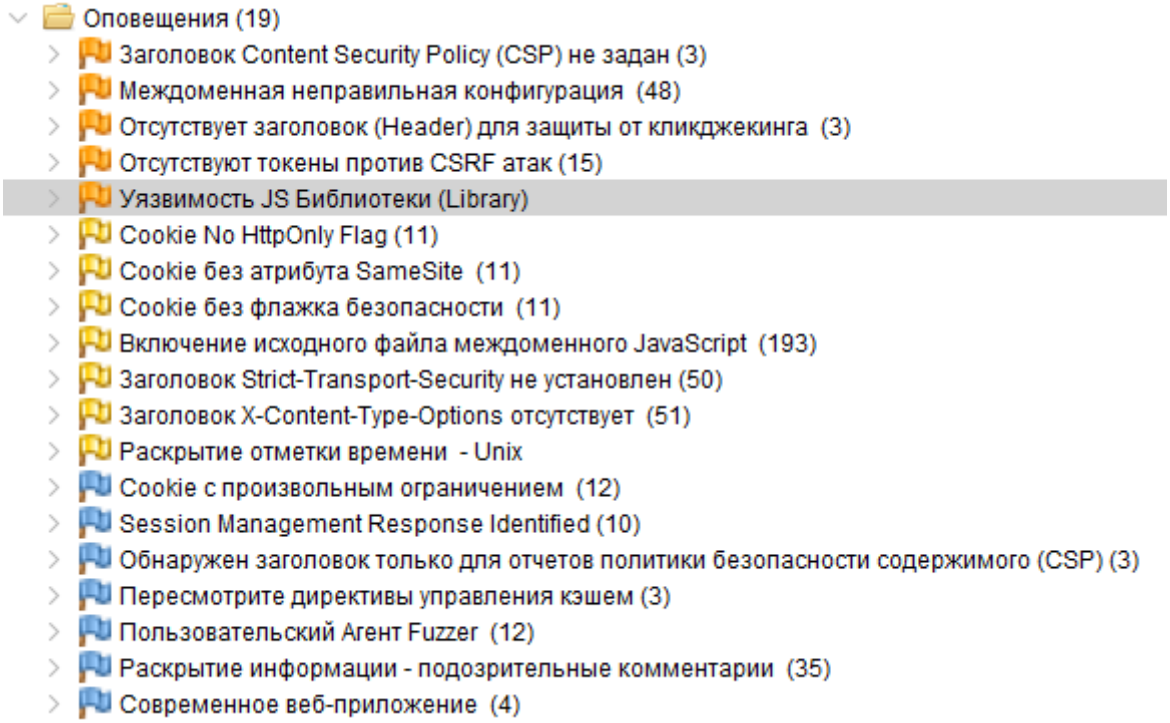


Рисунок 4.11 – Знайдені вразливості

Серед цих вразливостей 5 середнього рівня, 7 низького рівня та 7 інформаційного рівня.

Вразливості які були знайдені на попередньому сайті розбиратися знову не будуть, а саме: «Заголовок Content Security Police (CSP) не задано»,

«Відсутній заголовок (Header) для захисту від клікджекінгу», «Відсутні токени проти CSRF атак».

1 Перша вразливість середнього рівня «Неправильна міждоменна конфігурація», рис. 4.12. Вразливість «Неправильна міждоменна конфігурація» (Cross-Domain Misconfiguration) - це тип вразливості безпеки вебзастосунків, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні.

Неправильна міждоменна конфігурація виникає, коли вебзастосунок дозволяє запитувати ресурси з інших доменів без належного контролю. Це може бути зроблено випадково або через неправильні налаштування вебсервера або вебзастосунку.

Зловмисник може використовувати вразливість «Неправильна міждоменна конфігурація» для крадіжки даних користувачів, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні вебзастосунку.

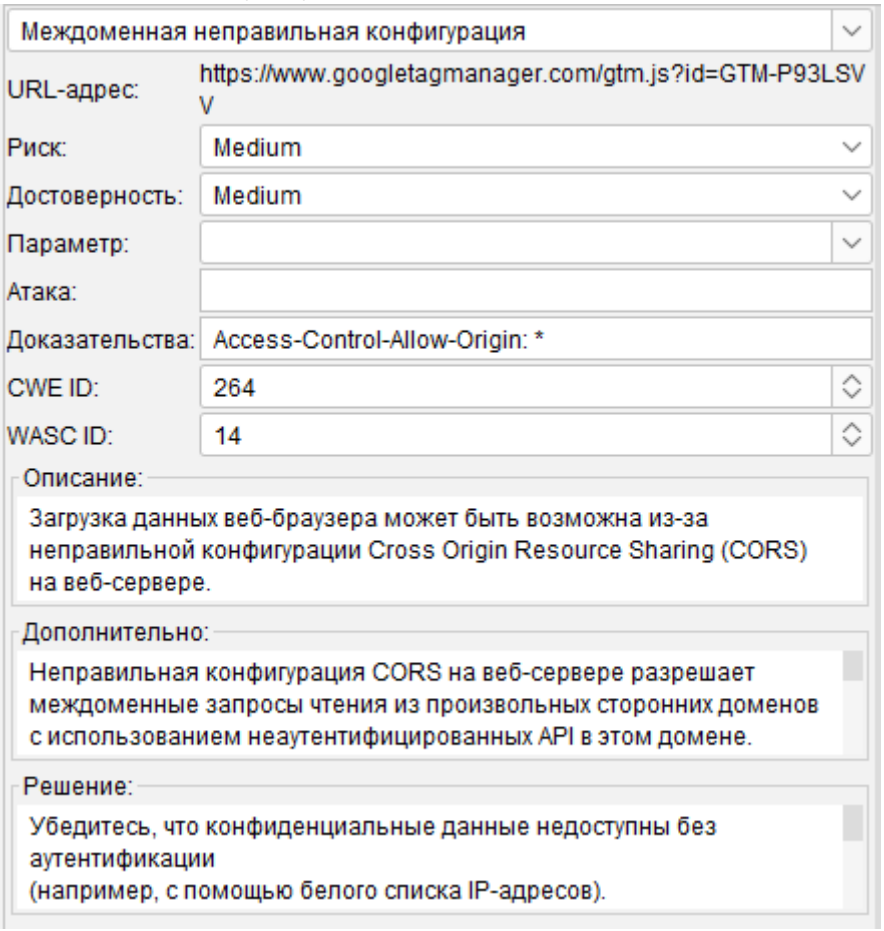


Рисунок 4.12 - Вразливість середнього рівня «Неправильна міждоменна конфігурація»

2 Друга і остання вразливість середнього рівня «Вразливість JS Бібліотеки (Library)», рис. 4.13. Вразливість «Вразливість JS Бібліотеки (Library)» - це тип вразливості безпеки вебзастосунків, яка виникає, коли

вебзастосунок використовує вразливу JavaScript-бібліотеку.

JavaScript-бібліотеки - це набори коду, які можна використовувати для виконання різних завдань, таких як обробка даних, інтерактивність та дизайн. Вони часто використовуються у вебзастосунка для покращення їх функціональності та візуального вигляду.

Вразливість JavaScript-бібліотеки може виникнути, якщо в бібліотеці виявлена проблема безпеки. Ця проблема може бути використана зловмисниками для крадіжки даних користувачів, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні вебзастосунку.

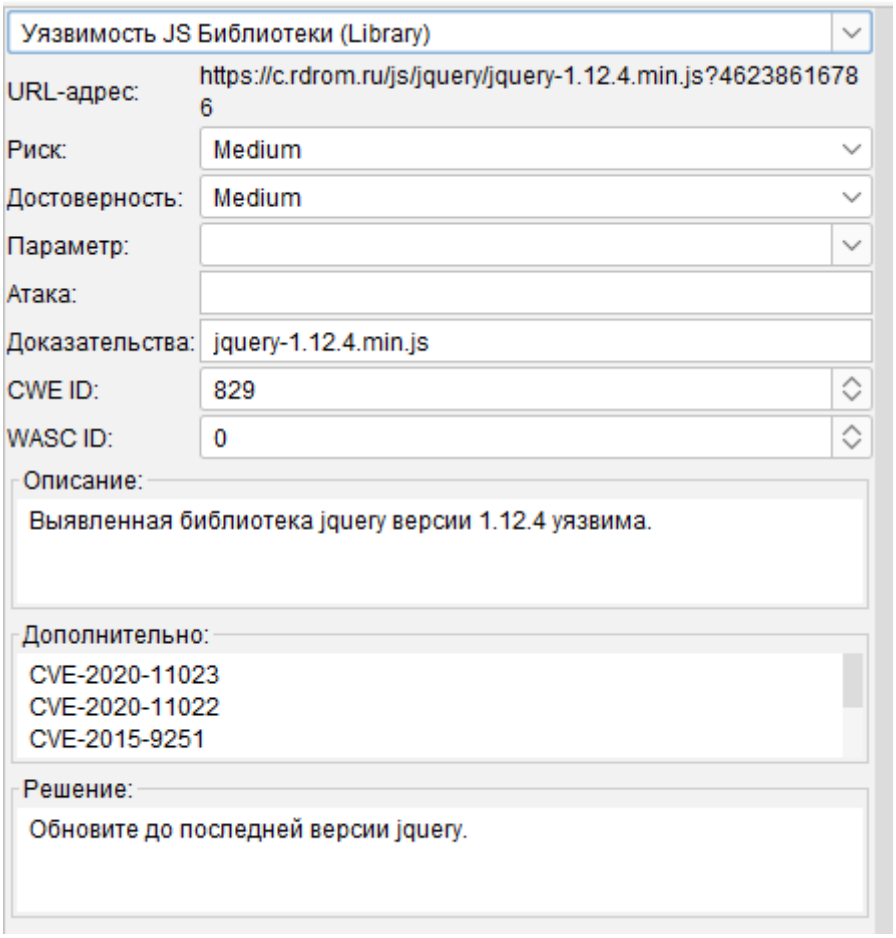


Рисунок 4.13 – Вразливість «JS Бібліотеки»

Цей сайт можна вважати небезпечним. Тому що серед цих вразливостей є «Вразливість JS Бібліотеки (Library)» яка є дуже небезпечною та легкою у використанні для зловмисників, а також «Неправильна міждоменна конфігурація» яка дозволяє зловмисникам відправляти через інші домени шкідливу інформацію.

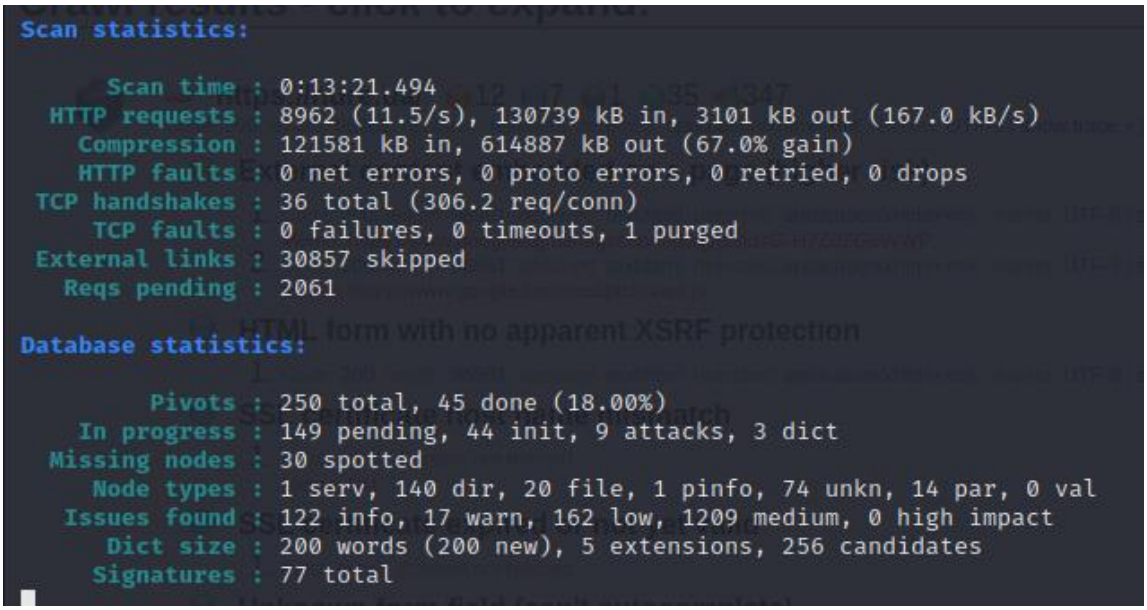


Рисунок 4.14 – Сканування сайту drom.ru

За допомогою skipfish було проведено аналіз сайту, рис. 4.14: drom.ru.

Як бачимо на рис. 4.15 було знайдено 1 вразливість високого рівня, 1489 вразливостей середнього рівня, 190 вразливостей низького рівня та 155 інформативного рівня і ще також показано 26 застережень.

- **Query injection vector** (1)
- **Incorrect or missing charset (higher risk)** (32)
- **External content embedded on a page (higher risk)** (1024)
- **HTML form with no apparent XSRF protection** (1)
- **External content embedded on a page (lower risk)** (189)
- **Node should be a directory, detection error?** (1)
- **Response varies randomly, skipping checks** (21)
- **Directory behavior checks failed (no brute force)** (1)
- **Resource fetch failed** (3)
- **Numerical filename - consider enumerating** (11)
- **Incorrect or missing charset (low risk)** (65)
- **Incorrect or missing MIME type (low risk)** (1)
- **HTML form (not classified otherwise)** (1)
- **Hidden files / directories** (20)
- **Resource not directly accessible** (18)
- **New 404 signature seen** (16)
- **New 'X-*' header value seen** (1)
- **New 'Server' header value seen** (1)
- **New HTTP cookie added** (40)
- **SSL certificate issuer information** (1)

Рисунок 4.15 - Вразливості знайдені за допомогою skipfish

Query injection vector (1)

1. https://www.drom.ru/commerce/new/sales_firm_model_year_9.sfish/"' [show trace +]
   Memo: response to "' different than to \'\"

Incorrect or missing charset (higher risk) (32)

1. https://www.drom.ru/404/-->'>'"<sfi000011v226276> [show trace +]
   Memo: windows-1251
2. https://www.drom.ru/404/.htaccess.aspx-->'>'"<sfi000025v226276> [show trace +]
   Memo: windows-1251
3. https://www.drom.ru/commerce/demands/.htaccess.aspx-->'>'"<sfi000051v226276> [show trace +]
   Memo: windows-1251
4. https://www.drom.ru/commerce/new/.htaccess.aspx-->'>'"<sfi000064v226276> [show trace +]
   Memo: windows-1251
5. https://www.drom.ru/commerce/new/907/.htaccess.aspx-->'>'"<sfi000207v226276> [show trace +]
   Memo: windows-1251
6. https://www.drom.ru/commerce/new/907/066.sfish [show trace +]
   Memo: windows-1251
7. https://www.drom.ru/commerce/new/907/135s.gif/-->'>'"<sfi000258v226276> [show trace +]
   Memo: windows-1251
8. https://www.drom.ru/commerce/new/907/594.html/.htaccess.aspx-->'>'"<sfi000285v226276> [show trace +]
   Memo: windows-1251
9. https://www.drom.ru/commerce/new/907/jaguar.gif/.htaccess.aspx-->'>'"<sfi000311v226276> [show trace +]
   Memo: windows-1251
10. https://www.drom.ru/commerce/new/907/moto.txt/.htaccess.aspx-->'>'"<sfi000324v226276> [show trace +]
    Memo: windows-1251
11. https://www.drom.ru/commerce/new/907/webkit.txt/.htaccess.aspx-->'>'"<sfi000272v226276> [show trace +]
    Memo: windows-1251
12. https://www.drom.ru/commerce/new/www.drom.ru/.htaccess.aspx-->'>'"<sfi000350v226276> [show trace +]
    Memo: windows-1251
13. https://www.drom.ru/commerce/new/.html/-->'>'"<sfi000375v226276> [show trace +]
    Memo: windows-1251
14. https://www.drom.ru/commerce/new/100.gif/-->'>'"<sfi000154v226276> [show trace +]
    Memo: windows-1251
15. https://www.drom.ru/commerce/new/12.html/.htaccess.aspx-->'>'"<sfi000077v226276> [show trace +]
    Memo: windows-1251
16. https://www.drom.ru/commerce/new/1439px.gif/.htaccess.aspx-->'>'"<sfi000168v226276> [show trace +]
    Memo: windows-1251
17. https://www.drom.ru/commerce/new/181.sfish [show trace +]
    Memo: windows-1251
18. https://www.drom.ru/commerce/new/1o6tcsu.sfish [show trace +]
    Memo: windows-1251
19. https://www.drom.ru/commerce/new/egapqn11.sfish [show trace +]
    Memo: windows-1251
20. https://www.drom.ru/commerce/new/flex.sfish [show trace +]
    Memo: windows-1251
21. https://www.drom.ru/commerce/new/makers.html/.htaccess.aspx-->'>'"<sfi000428v226276> [show trace +]

Рисунок 4.16 - Вразливості знайдені за допомогою skipfish

Вразливість «External content embedded on a page» була описана вище.

На рис. 4.16 та рис. 4.17 бачимо детальну інформацію по кожній вразливості.

1 Вразливість "Query injection vector" (вектор введення запиту) - це тип вразливості безпеки вебдодатків, яка виникає, коли вебдодаток дозволяє користувачам вводити дані в запити, які використовуються для доступу до бази даних.

Ця вразливість може бути використана зловмисниками для кількох цілей, включаючи:

– виконання довільного коду: зловмисник може використовувати вектор введення запиту для виконання довільного коду на сервері вебдодатку. Наприклад, зловмисник може ввести в запит шкідливий код, який може отримати доступ до конфіденційних даних або завантажити шкідливе програмне забезпечення на сервер;

– крадіж даних: зловмисник може використовувати вектор введення запиту для крадіжки даних з бази даних вебдодатку. Наприклад, зловмисник може ввести в запит запит на вибірку всіх даних користувачів;

– зміна даних: зловмисник може використовувати вектор введення запиту для зміни даних у базі даних вебдодатку. Наприклад, зловмисник може ввести в запит на зміну паролів користувачів.

Наслідки вразливості «Query injection vector» можуть бути серйозними. Якщо ця вразливість не буде виправлена, зловмисник може використовувати її для виконання шкідливого коду, крадіжки даних або зміни даних.

2 Вразливість "Incorrect or missing charset (windows-1251)" (неправильний або відсутній кодування (windows-1251)) - це тип вразливості безпеки вебдодатків, яка виникає, коли вебдодаток не правильно визначає кодування символів для вхідних даних.

Ця вразливість може бути використана зловмисниками для кількох цілей, включаючи:

– виконання довільного коду: зловмисник може використовувати неправильний кодування символів для вставки шкідливого коду в вхідні дані. Цей шкідливий код може бути виконаний на сервері вебдодатку, що може призвести до компрометації безпеки;

– крадіж даних: Зловмисник може використовувати неправильний кодування символів для крадіжки даних з бази даних вебдодатку. Наприклад, зловмисник може використовувати неправильний кодування символів для виводу даних з бази даних, які містять конфіденційну інформацію;

– зміна даних: Зловмисник може використовувати неправильний кодування символів для зміни даних у базі даних вебдодатку. Наприклад, зловмисник може використовувати неправильний кодування символів для зміни паролів користувачів.

Наслідки вразливості "Incorrect or missing charset (windows-1251)" можуть бути серйозними. Якщо ця вразливість не буде виправлена, зловмисник може використовувати її для виконання шкідливого коду, крадіжки даних або зміни даних.

Зі списку вразливостей skipship сайт можна вважати небезпечним тому що загрози високого та середнього рівня на цьому сайті дуже серйозні.

External content embedded on a page (higher risk) (1024)

1. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/25408.fb07639721ca8d545e49.js
2. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/82820.7e4688d721adcfa4247b.js
3. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/84249.968634b3bfc22a7c589e.js
4. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/60682.0d4e59d08ac90c9429c2.chunk.js
5. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/36566.45f8ad87beab0c828c4a.js
6. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/87633.c36db3f93546eb5c21e6.js
7. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/41736.aa8582bf4fc0369076c4.js
8. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/71962.e6e6928eca1a7464c89d.js
9. https://www.drom.ru/ [show trace +]
   Memo: https://c.rdrom.ru/js/bundles/80715.aa50ec7234eff06b5472.js
10. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/24835.a5960e47641404999061.chunk.js
11. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/69599.9b31d0321af8d0c8af90.chunk.js
12. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/31469.76ba042fa78e3bb4811e.chunk.js
13. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/75683.360e21f99fb21ede5fbe.chunk.js
14. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/92551.289f740c56bd4d545a3f.chunk.js
15. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/1761.32b28aaf18bba5e2f05d.chunk.js
16. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/95635.c6cb39424e9006bd06de.js
17. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/45972.ab14e40b123a01adc527.chunk.js
18. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/23314.ccfc125890acfa827a2d.chunk.js
19. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/40949.0f1f583011eb1e9b2cd7.chunk.js
20. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/home-layout-desktop.fcd4059594ab88c30523.chunk.js
21. https://www.drom.ru/ [show trace +]
    Memo: https://c.rdrom.ru/js/bundles/header-desktop-add-bull-button.3e856b902d334c688815.chunk.js

Рисунок 4.17 - Вразливості знайдені за допомогою skipfish

4.4 Сайт gosuslugi.ru

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.18: www.gosuslugi.ru.

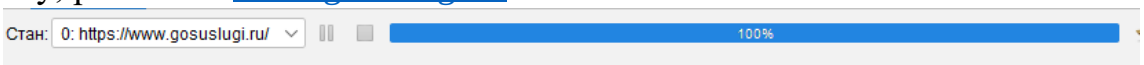


Рисунок 4.18 - Аналіз сайту www.gosuslugi.ru

Було знайдено всього 18 вразливостей, рис. 4.19:

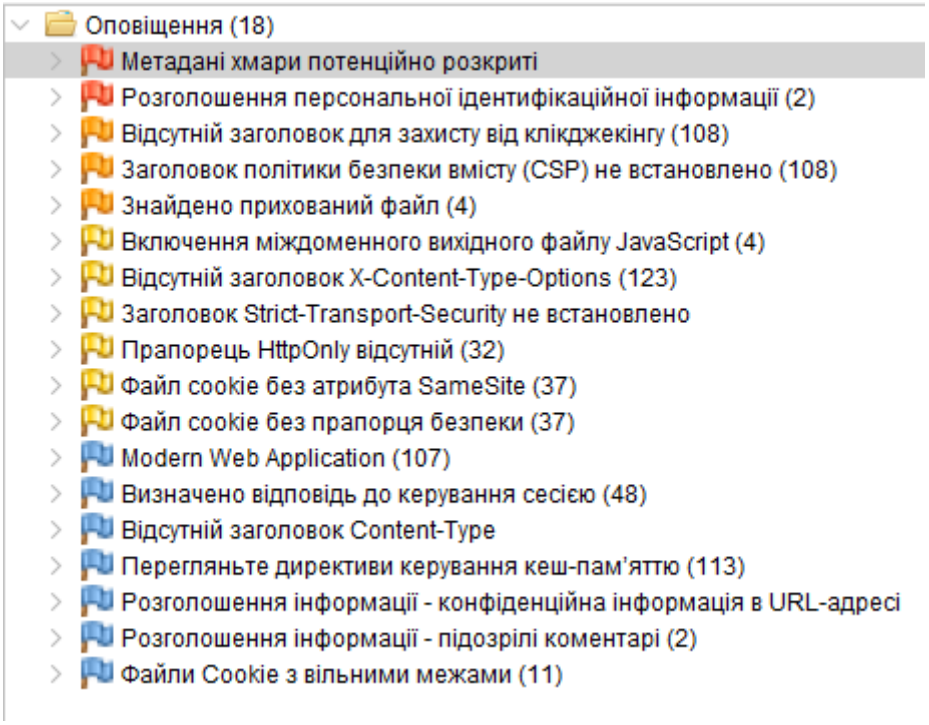


Рисунок 4.19 – Знайдені вразливості

Серед цих вразливостей 2 критичного рівня, 3 середнього рівня, 6 низького рівня та 7 інформаційного рівня.

Вразливості які були знайдені на попередньому сайті розбиратися знову не будуть, а саме: «Заголовок Content Security Police (CSP) не задано», «Відсутній заголовок (Header) для захисту від клікджекінгу», «Відсутні токени проти CSRF атак».

1 Вразливість «Метадані хмари потенційно розкриті» (Cloud Metadata Potentially Leaked) - це тип вразливості безпеки хмарних обчислень, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні, рис. 4.20.

Метадані хмари - це інформація про дані, які зберігаються в хмарі. Ця інформація може включати в себе такі дані, як імена користувачів, паролі, номери кредитних карток, IP-адреси, електронні адреси та інші конфіденційні дані.

Вразливість «Метадані хмари потенційно розкриті» виникає, коли метадані хмари стають доступними для несанкціонованих користувачів. Це може статися в результаті таких факторів, як:

– неправильні налаштування безпеки хмарного провайдера;
– помилка користувача;
– атака зловмисника.

Якщо метадані хмари стають доступними для несанкціонованих користувачів, зловмисники можуть використовувати цю інформацію для

крадіжки даних, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні.

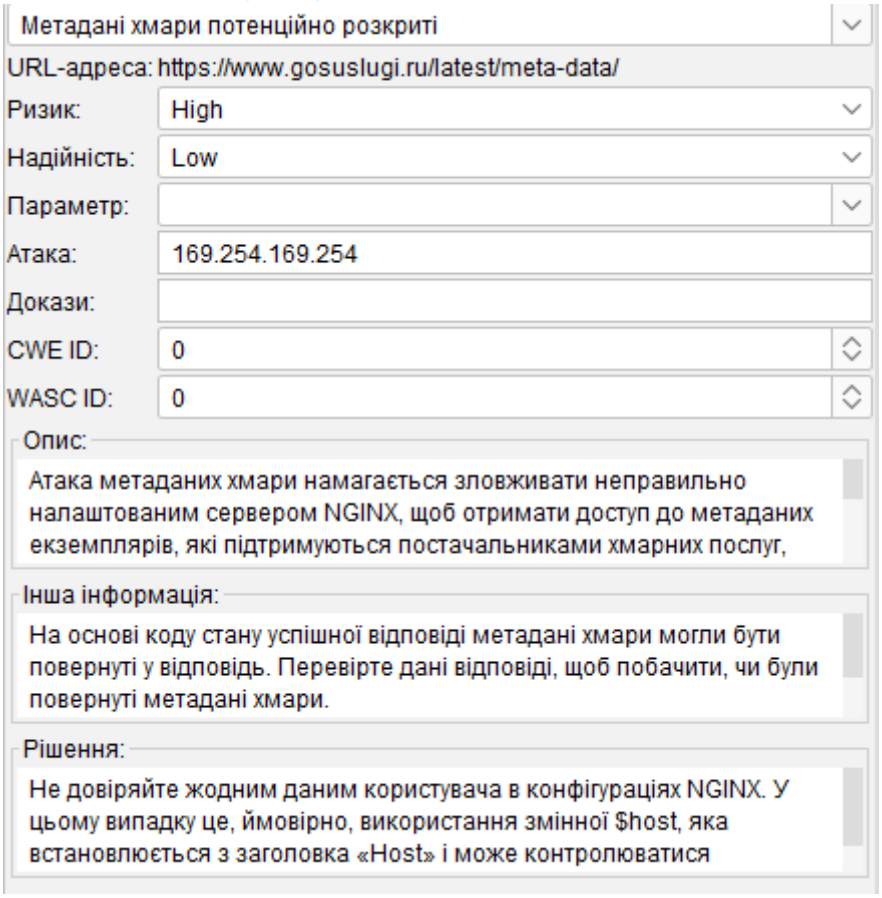


Рисунок 4.20 – Вразливість «Метадані хмари потенційно розкриті»

2 Вразливість «Розголошення персональної ідентифікаційної інформації» (Personal Identifiable Information Exposure) - це тип вразливості безпеки, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні, рис. 4.21.

Персональна ідентифікаційна інформація (PII) - це будь-яка інформація, яка може бути використана для ідентифікації конкретної людини. До PII належать такі дані, як імена, прізвища, номери соціального страхування, номери кредитних карток, адреси електронної пошти та інші конфіденційні дані.

Вразливість «Розголошення персональної ідентифікаційної інформації» виникає, коли PII стає доступною для несанкціонованих користувачів. Це може статися в результаті таких факторів, як:

– неправильні налаштування безпеки вебсайту або вебзастосунку;
– помилка користувача;
– атака зловмисника.

Якщо PII стає доступною для несанкціонованих користувачів, зловмисники можуть використовувати цю інформацію для крадіжки особистої інформації, фінансових коштів або навіть для здійснення шахрайських дій.

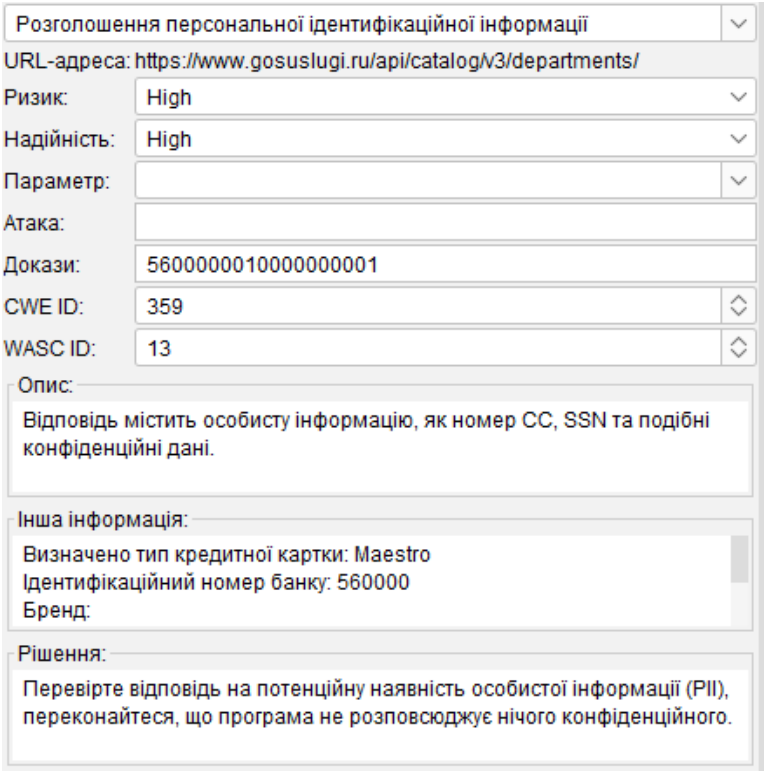


Рисунок 4.21 - Вразливість «Розголошення персональної ідентифікаційної інформації»

Ось кілька конкретних прикладів того, як зловмисник може використати вразливість «Розголошення персональної ідентифікаційної інформації»:

— крадіжка особистої інформації: зловмисник може використовувати PII для крадіжки імен, прізвищ, номерів соціального страхування, номерів кредитних карток або інших конфіденційних даних. Ці дані можна використовувати для крадіжки особистої інформації, фінансових коштів або навіть для здійснення шахрайських дій;

— фінансова шкода: зловмисник може використовувати PII для здійснення шахрайських дій, таких як відкриття кредитних карток або позичання грошей на ім'я іншої людини. Це може призвести до фінансової шкоди для жертви;

— особисті проблеми: зловмисник може використовувати PII для шантажу або залякування жертви. Це може призвести до особистих проблем для жертви.

3 Вразливість "Знайдено прихований файл" (Hidden File Found) - це тип вразливості безпеки, яка може призвести до крадіжки даних, порушення конфіденційності або навіть до відмови в обслуговуванні, рис. 4.22.

Прихований файл - це файл, який не відображається в звичайному файловому менеджері. Приховані файли часто використовуються для зберігання конфіденційної інформації, такої як паролі, ключі API або інші важливі дані.

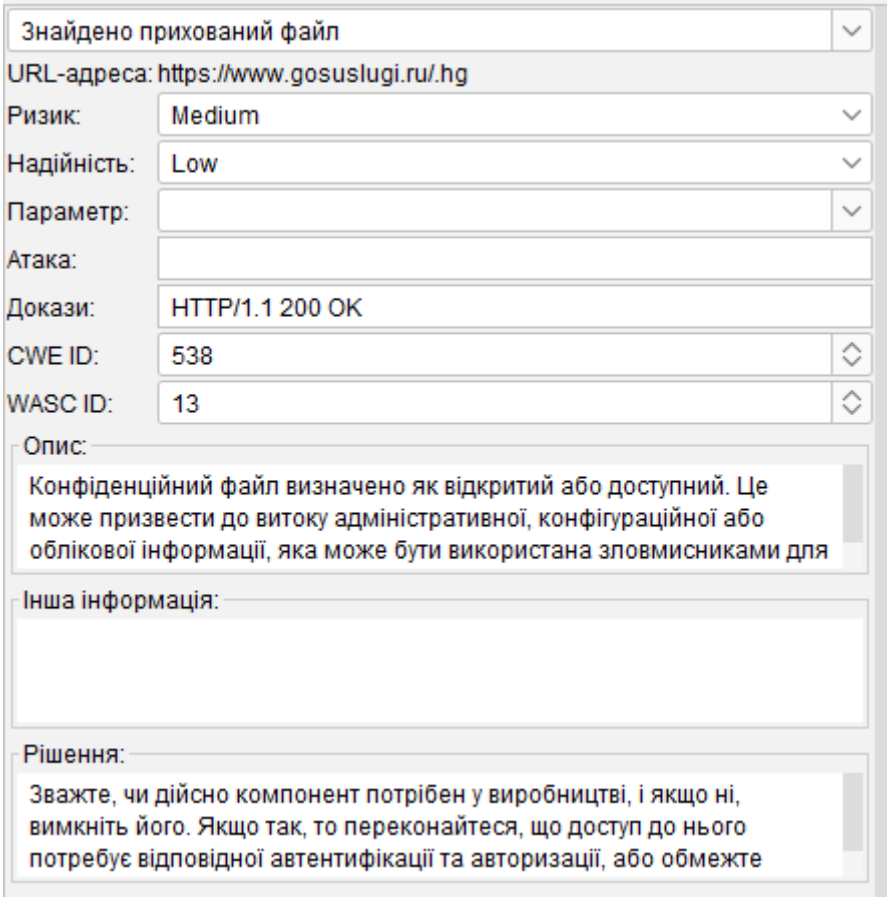


Рисунок 4.22 - Вразливість «Знайдено прихований файл»

Вразливість "Знайдено прихований файл" виникає, коли прихований файл стає доступним для несанкціонованих користувачів. Це може статися в результаті таких факторів, як:

— неправильні налаштування безпеки вебсайту або вебзастосунку;
— помилка користувача;
— атака зловмисника.

Якщо прихований файл стає доступним для несанкціонованих користувачів, зловмисники можуть використовувати цю інформацію для крадіжки даних, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні.

Загалом, цей сайт можна вважати небезпечним. Тому що серед вразливостей є відразу дві вразливості які можуть з великим ризиком надати особисті дані користувачів: «Метадані хмари потенційно розкриті» і «Розголошення персональної ідентифікаційної інформації».

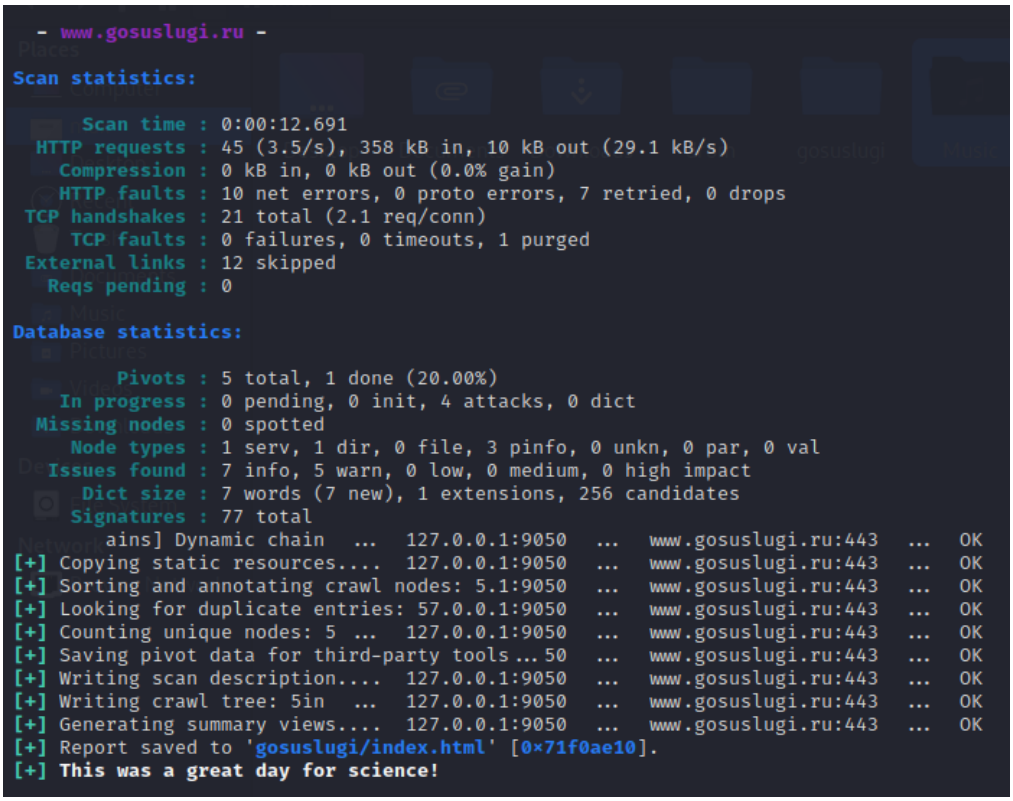


Рисунок 4.23 – Сканування сайту gosuslugi.ru

За допомогою skipfish було проведено аналіз сайту, рис. 4.23: gosuslugi.ru.

Оскільки skipfish не має функції використання проксі, тому сканування невдале але дещо він знайшов.

На рис. 4.24 бачимо що було знайдено 7 вразливостей інформативного рівня і 5 застережень з позначкою XSS яке було описане вище.

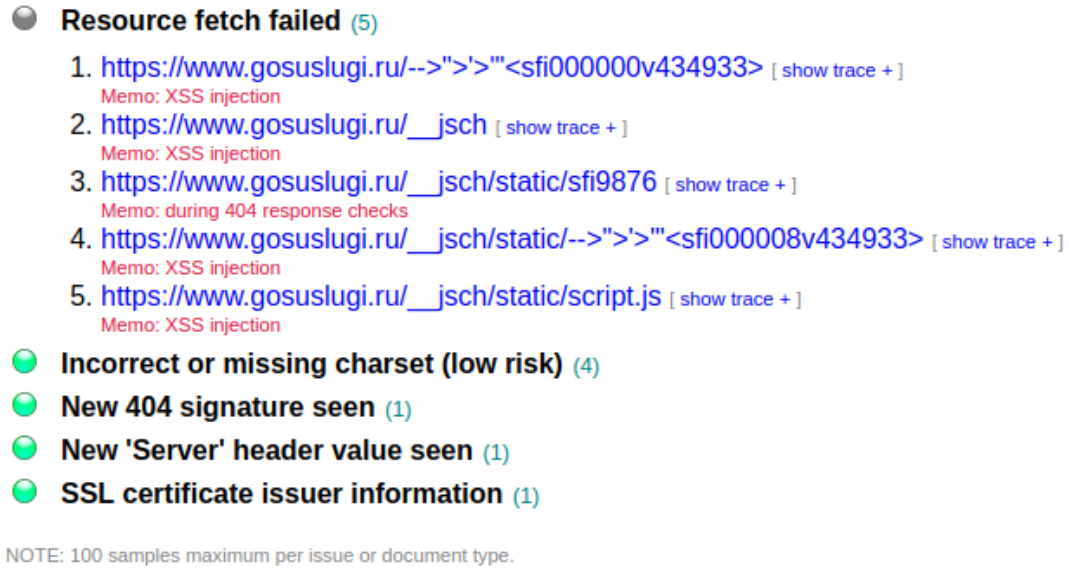


Рисунок 4.24 – Вразливості знайдені за допомогою skipfish

4.5 Сайт rsl.ru

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.25: www.rsl.ru

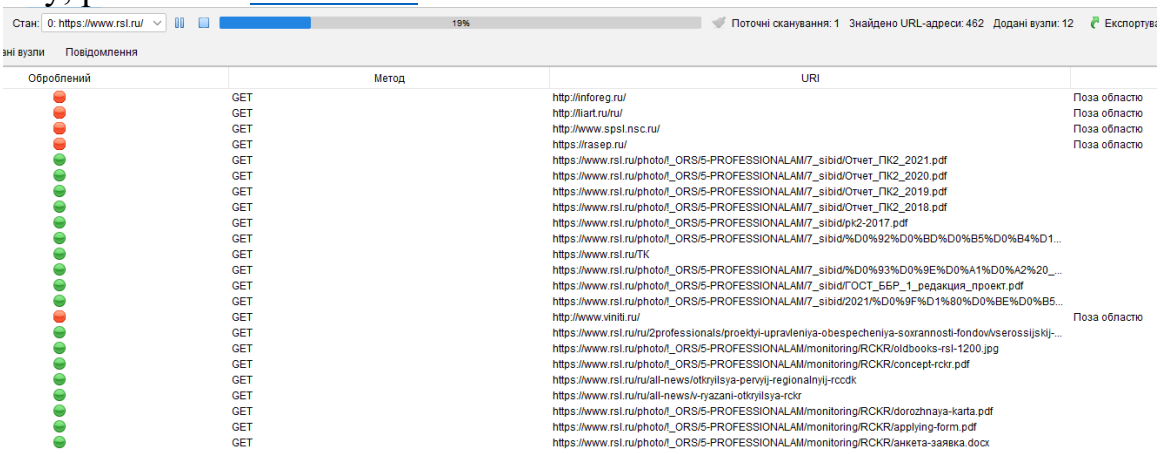


Рисунок 4.25 - Аналіз сайту www.rsl.ru

Було знайдено всього 16 вразливостей, рис. 4.26:

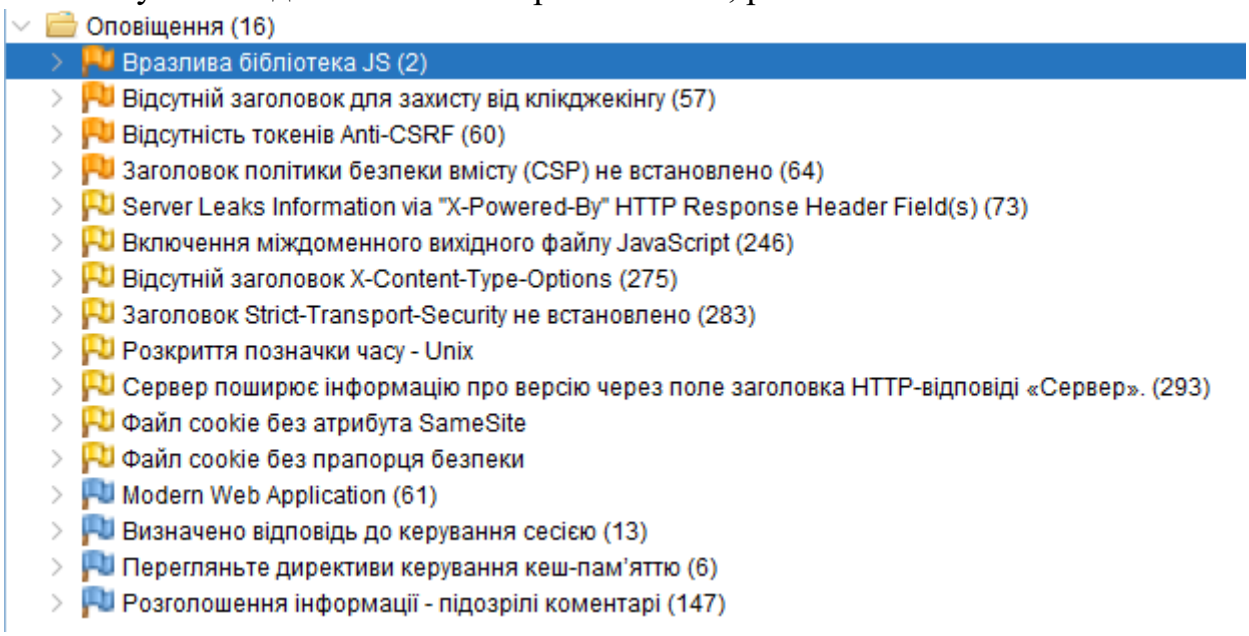


Рисунок 4.26 – Знайдені вразливості

Серед цих вразливостей 4 середнього рівня, 8 низького рівня та 4 інформаційного рівня.

Вразливості які були знайдені на попередніх сайтах розбиратися знову не будуть, а саме: «Заголовок Content Security Police (CSP) не задано», «Відсутній заголовок (Header) для захисту від клікджекінгу», «Відсутність токенів Anti-CSRF» та «Вразлива бібліотека JS».

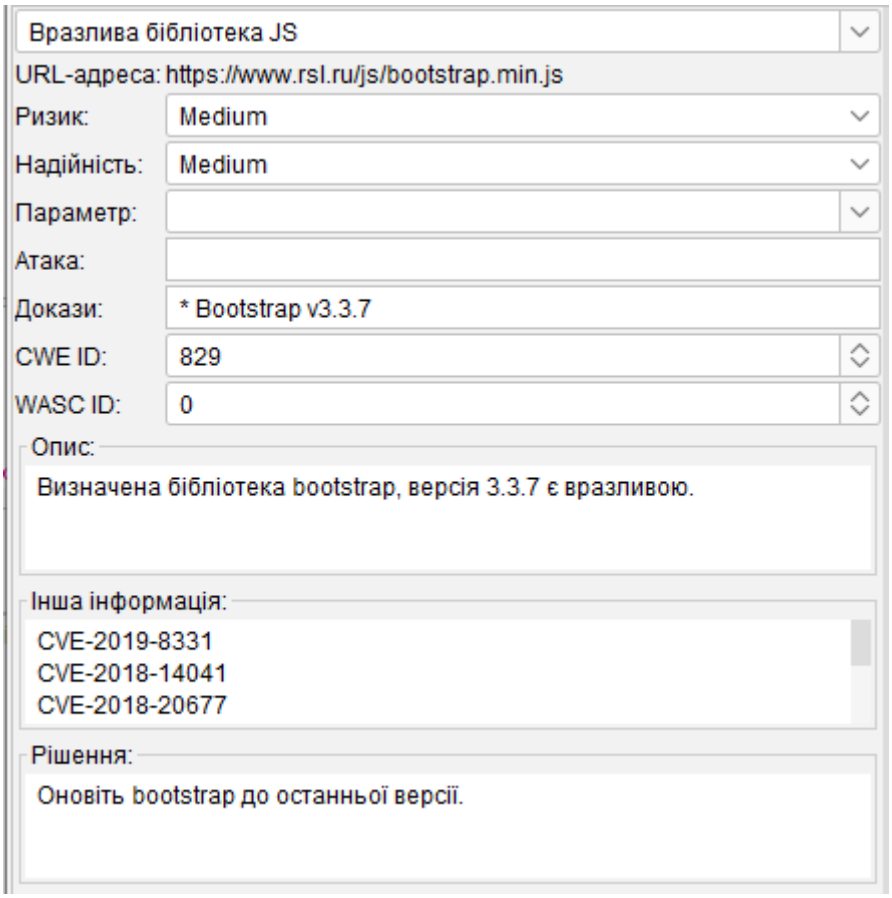


Рисунок 4.27 – Вразливість «Вразливість JS Бібліотеки (Library)»

Вразливість «Вразливість JS Бібліотеки (Library)» bootstrap біблотеки є небезпечною, оскільки вона може призвести до виконання довільного коду на комп'ютері користувача, рис. 4.27. Ця вразливість виникає через те, що bootstrap не належним чином обробляє вхідні дані, які надсилаються до веббраузера.

Зловмисник може використовувати цю вразливість, щоб вставити в вебсайт зловмисний код, який буде виконаний браузером користувача. Цей код може бути використаний для крадіжки даних користувача, таких як паролі або номери кредитних карток, або для зараження комп'ютера користувача шкідливим програмним забезпеченням.

Загалом, цей сайт можна вважати небезпечним. Тому що серед вразливостей є вразливість яка може бути з великим ризиком надати особисті дані користувачів: «Вразливість JS Бібліотеки (Library)».

Так як skipfish не може використовувати proxy тому він не зміг просканувати сайт, рис. 4.28.

Рисунок 4.28 - Помилка підключення до сайту olx.ua

4.6 Сайт olx.ua

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.29: www.olx.ua

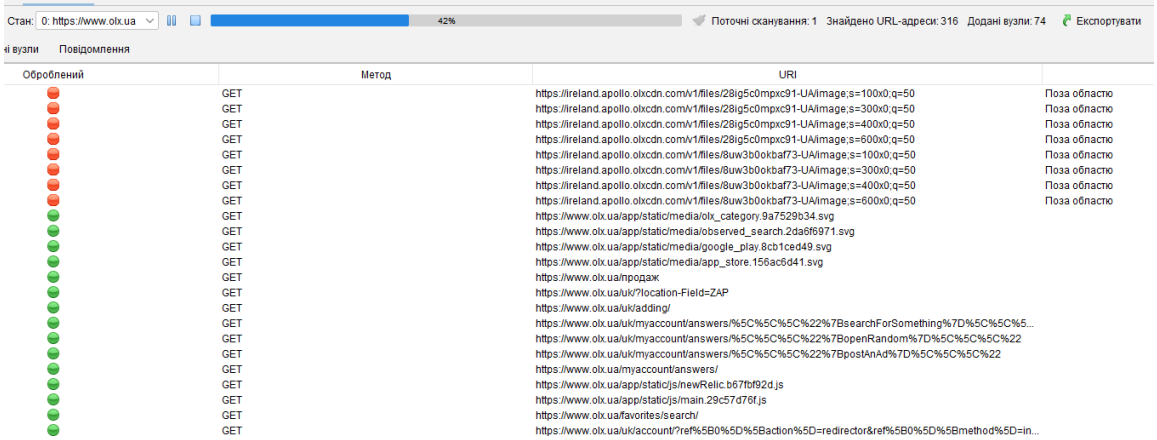


Рисунок 4.29 - Аналіз сайту www.olx.ua

Було знайдено всього 26 вразливостей, рис. 4.30:

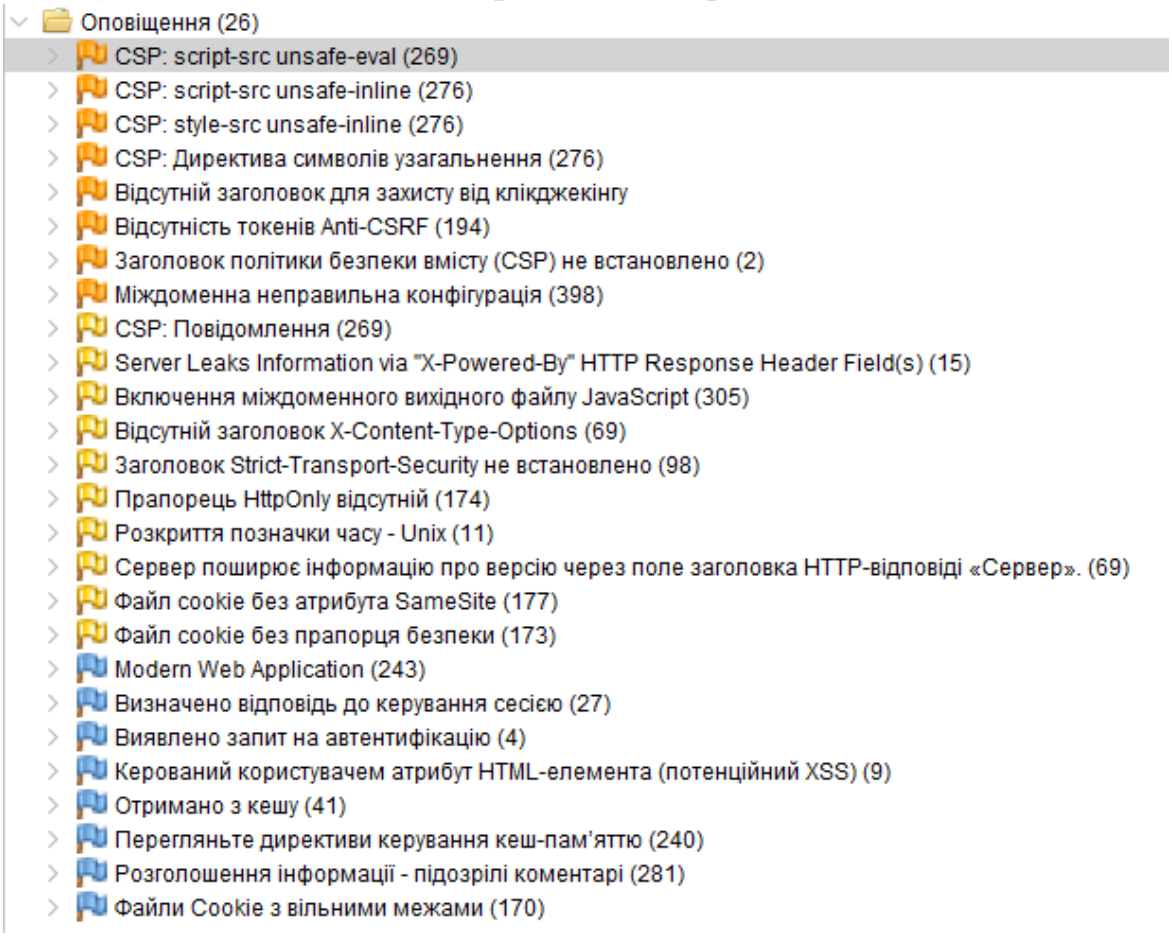


Рисунок 4.30 – Знайдені вразливості

Серед цих вразливостей 8 середнього рівня, 10 низького рівня та 8 інформаційного рівня.

Вразливості які були знайдені на попередніх сайтах розбиратися знову

не будуть, а саме: «Заголовок Content Security Police (CSP) не задано», «Відсутній заголовок (Header) для захисту від клікджекінгу», «Відсутність токенів Anti-CSRF» та «Вразлива бібліотека JS» та «Неправильна міждоменна конфігурація».

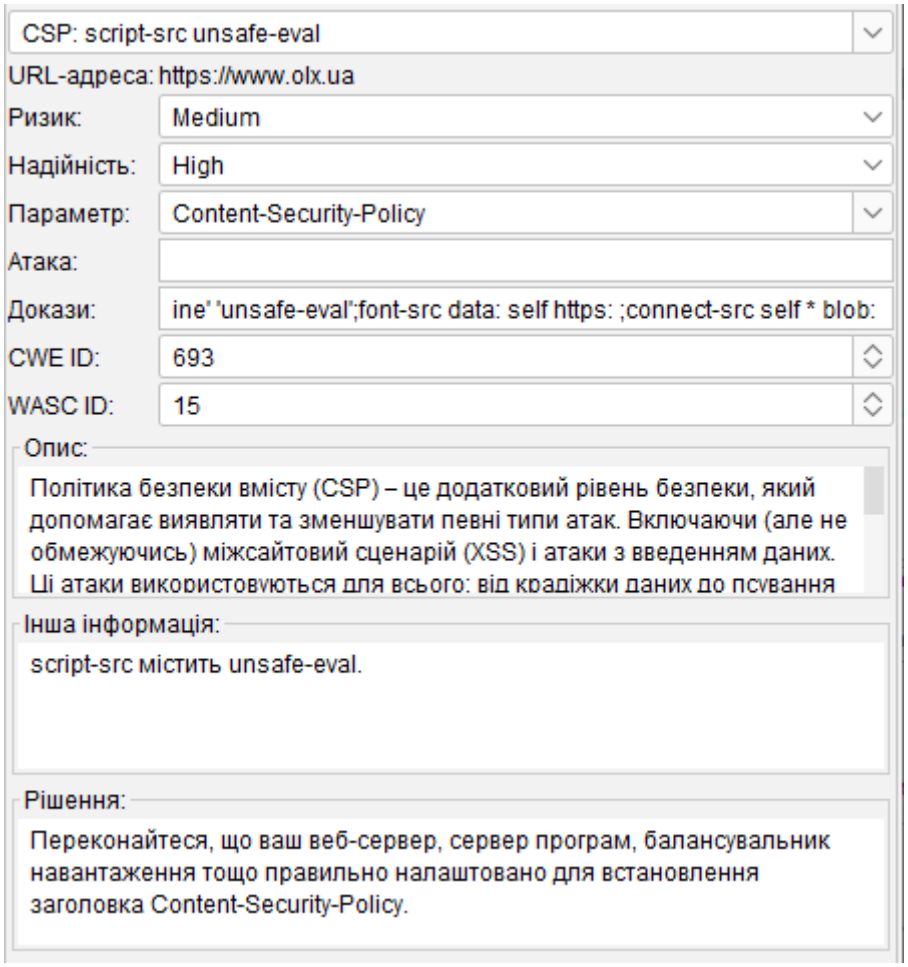


Рисунок 4.31 – Вразливість «CSP: script-src unsafe-eval»

Вразливість «CSP: script-src unsafe-eval» є небезпечною, оскільки вона дозволяє зловмисникам виконувати довільний код на вебсайті без необхідності попереднього доступу до нього, рис. 4.31. Ця вразливість виникає через те, що вебсайт дозволяє скриптам з неавторизованих джерел виконуватися в режимі eval().

Зловмисник може використовувати цю вразливість, щоб вставити в вебсайт зловмисний код, який буде виконаний браузером користувача. Цей код може бути використаний для крадіжки даних користувача, таких як паролі або номери кредитних карток, або для зараження комп'ютера користувача шкідливим програмним забезпеченням.

Загалом, цей сайт можна вважати безпечним. Тому що серед вразливостей немає вразливості яка має великого ризику для безпеки сайту.

За допомогою skipfish була спроба провести аналіз сайту: olx.ua, але показує помилку підключення, рис. 4.32.

.

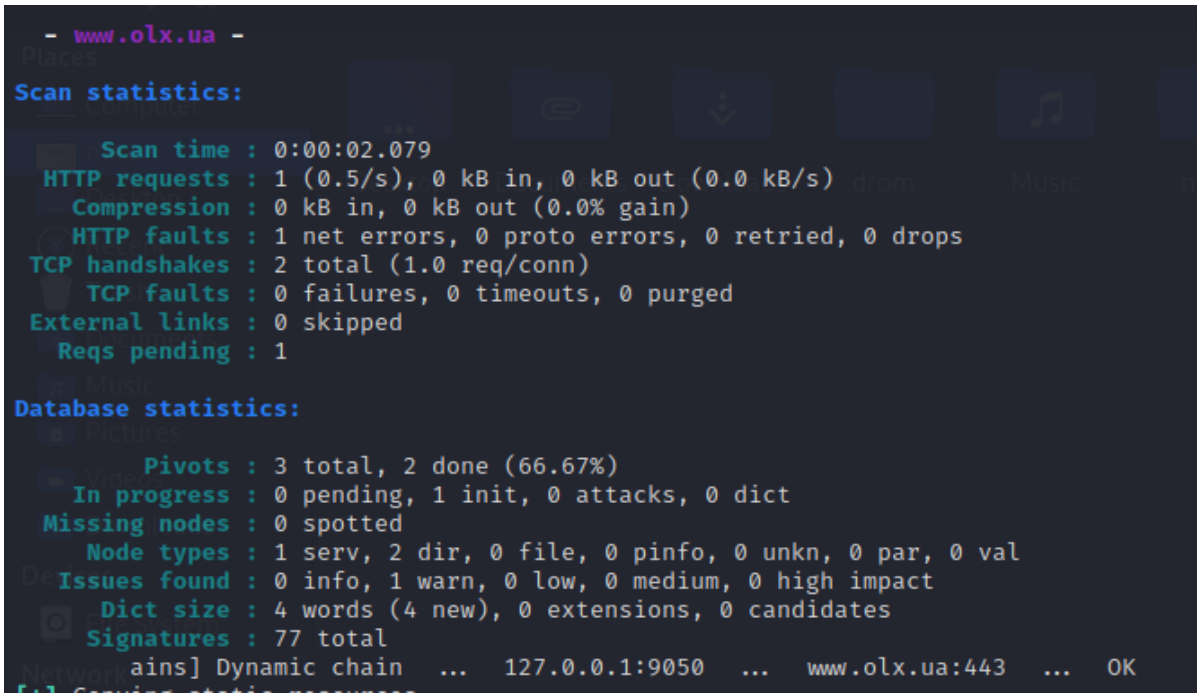


Рисунок 4.32 – Сканування сайту olx.ua

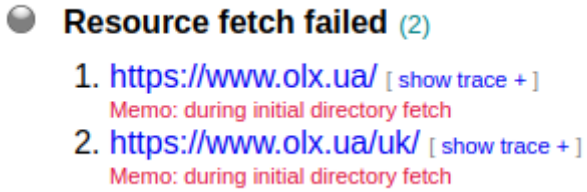


Рисунок 4.33 – Помилки підключення до сайту olx.ua

4.7 Сайт work.ua

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.34: www.work.ua

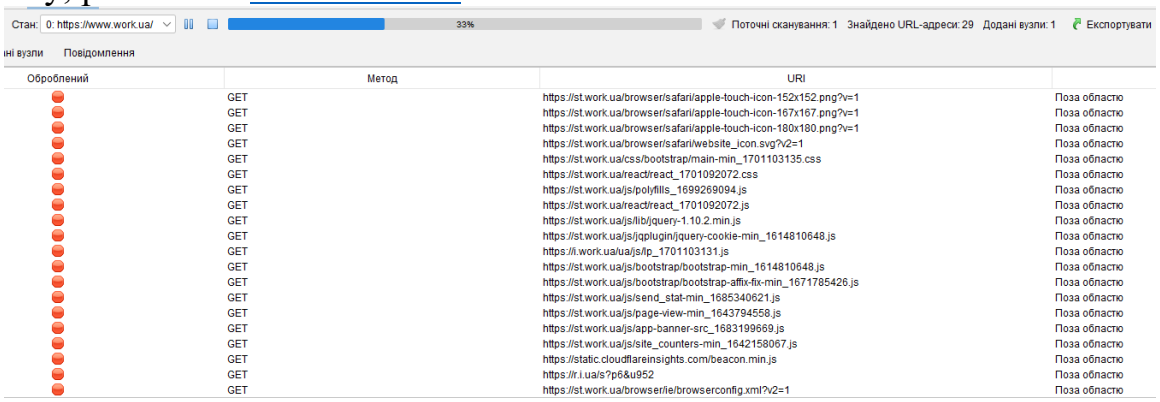


Рисунок 4.34 - Аналіз сайту work.ua

Було знайдено всього 22 вразливостей, рис. 4.35:

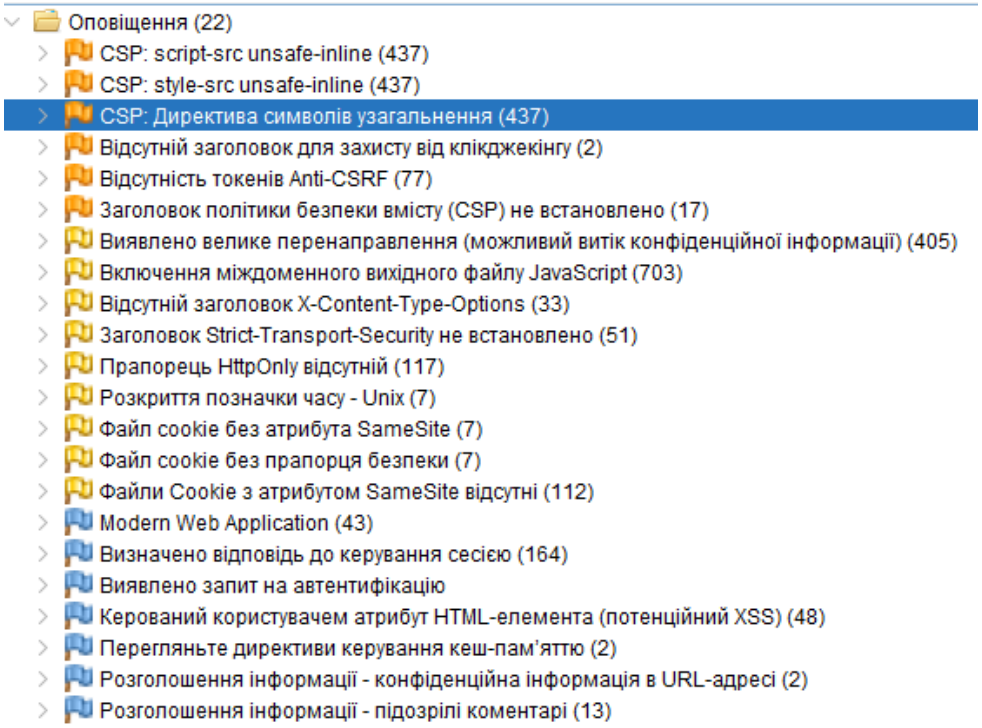


Рисунок 4.35 – Знайдені вразливості

Серед цих вразливостей 6 середнього рівня, 9 низького рівня та 7 інформаційного рівня.

Всі вразливості окрім «CSP: Директива символів узагальнення» були описанні вище, рис. 4.36.

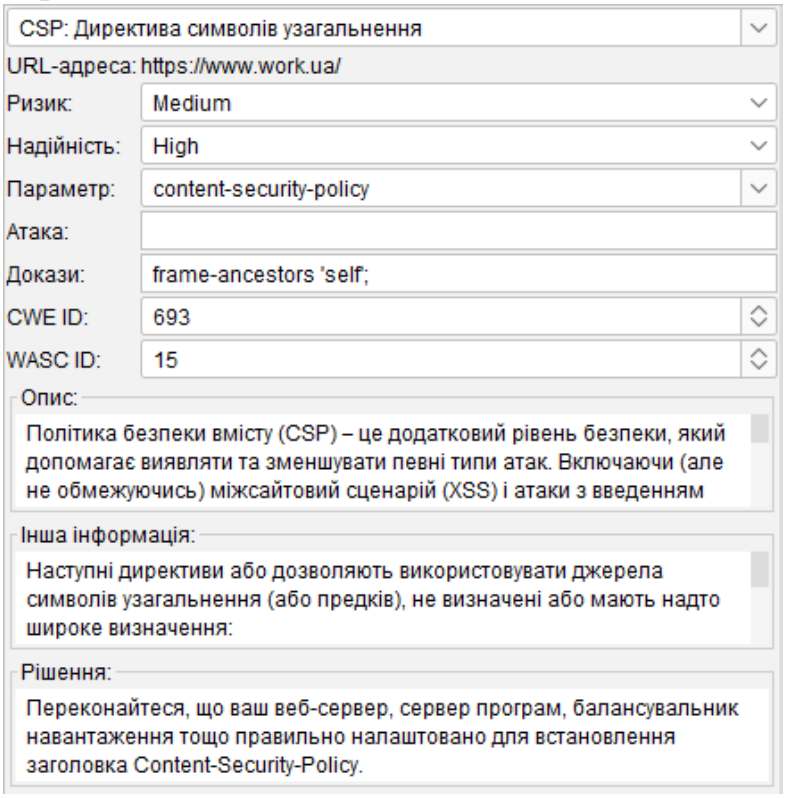


Рисунок 4.36 – Вразливість «CSP: Директива символів узагальнення»

Вразливість «CSP: Директива символів узагальнення» є небезпечною, оскільки вона дозволяє зловмисникам виконувати довільний код на вебсайті, використовуючи символи узагальнення в директивах CSP. Ця вразливість виникає через те, що вебсайти часто використовують символи узагальнення в директивах CSP, щоб дозволити скриптам завантажуватися з широкого кола доменів.

Зловмисник може використовувати цю вразливість, щоб вставити в вебсайт зловмисний код, який буде виконаний браузером користувача. Цей код може бути використаний для крадіжки даних користувача, таких як паролі або номери кредитних карток, або для зараження комп'ютера користувача шкідливим програмним забезпеченням.

Загалом, цей сайт можна вважати безпечним. Тому що серед вразливостей немає вразливості яка має великого ризику для безпеки сайту.

За допомогою skipfish була спроба провести аналіз сайту: work.ua, але показує помилку підключення, рис. 4.37.

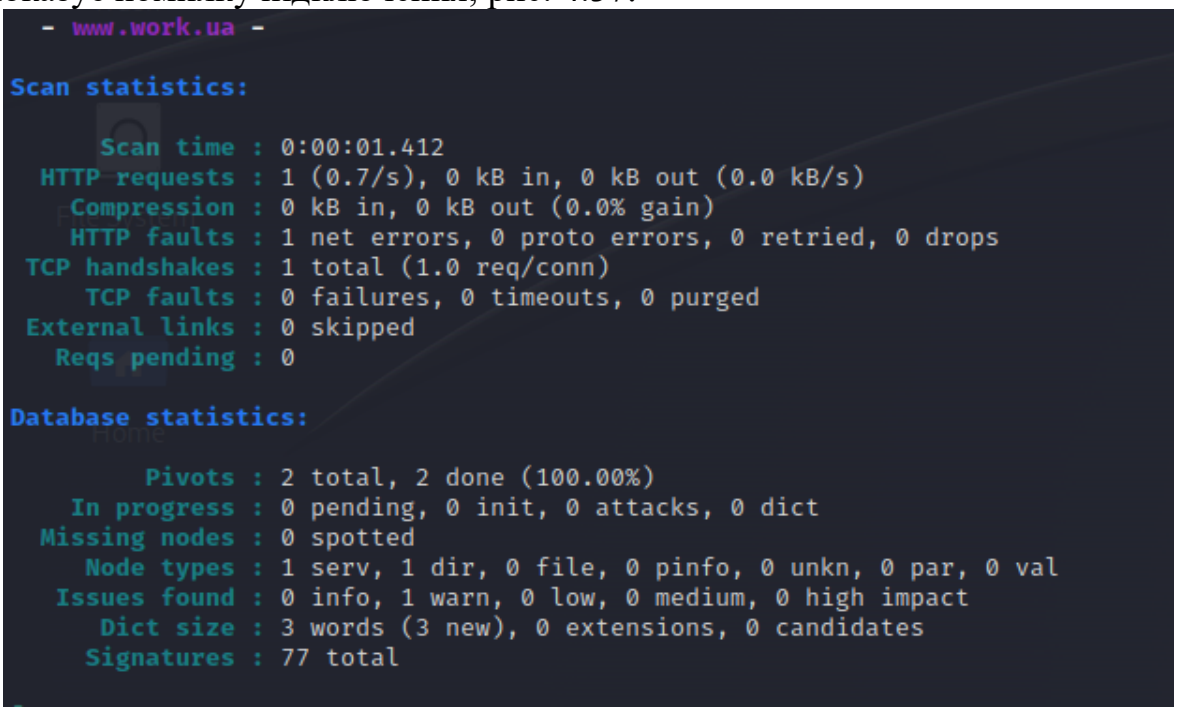


Рисунок 4.37 – Сканування сайту work.ua

Рисунок 4.38 – Помилка підключення до сайту work.ua

4.8 Сайт 1win.org.ua

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.39: www.1win.org.ua

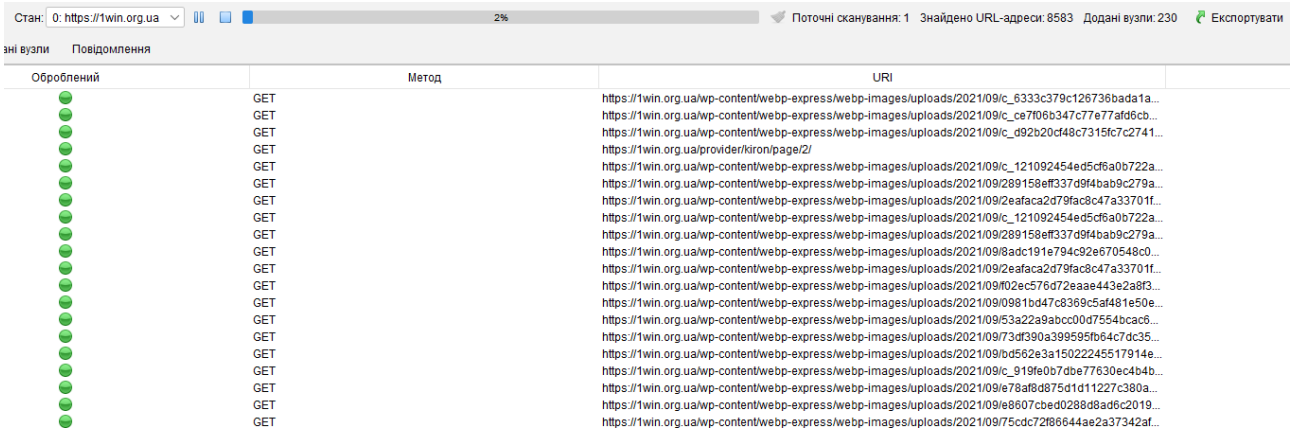


Рисунок 4.39 - Аналіз сайту 1win.org.ua

Було знайдено всього 11 вразливостей, рис. 4.40:

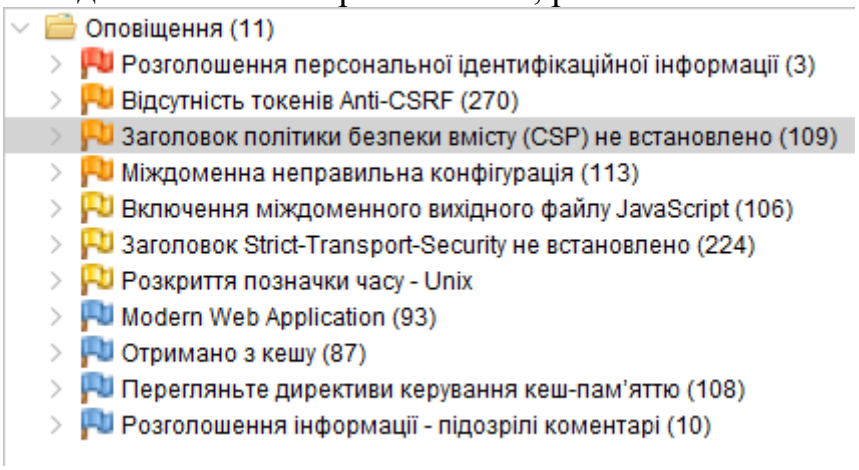


Рисунок 4.40 – Знайдені вразливості

Серед цих вразливостей 1 критичного рівня, 3 середнього рівня, 3 низького рівня та 4 інформаційного рівня.

Всі вразливості вже були описані вище.

Загалом, цей сайт можна вважати безпечним. Тому що серед вразливостей немає вразливості яка має великого ризику для безпеки сайту.

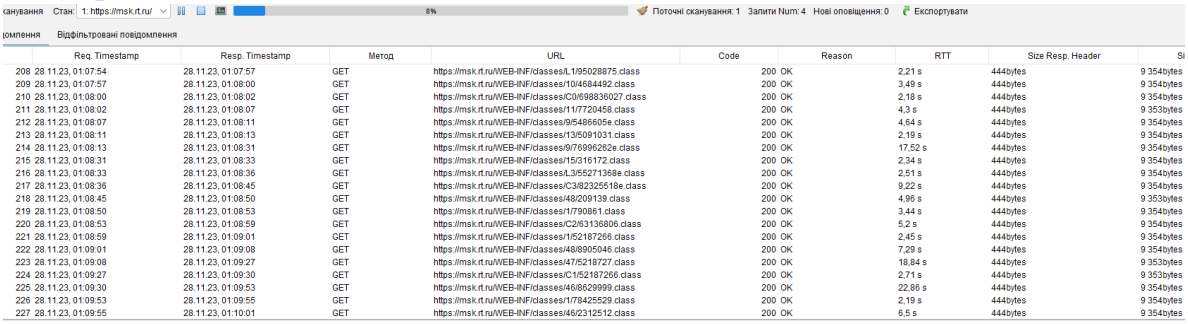
За допомогою skipfish була спроба провести аналіз сайту: 1win.org.ua, але показує помилку підключення, рис. 4.41.

Рисунок 4.41 – Сканування сайту 1win.org.ua

Рисунок 4.42 – Помилка підключення до сайту 1win.org.ua

4.9 Сайт msk.rt.ru

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.43: www. msk.rt.ru



	Req. Timestamp	Resp. Timestamp	Метод	URL	Code	Reason	RTT	Size Resp. Header	Si
208	28.11.23, 01:07:54	28.11.23, 01:07:57	GET	https://msk.rt.ru/WEB-INF/classes/L1/95028875.class	200	OK	2,21 s	444bytes	9 354bytes
209	28.11.23, 01:07:57	28.11.23, 01:08:00	GET	https://msk.rt.ru/WEB-INF/classes/10/4684492.class	200	OK	3,49 s	444bytes	9 354bytes
210	28.11.23, 01:08:00	28.11.23, 01:08:02	GET	https://msk.rt.ru/WEB-INF/classes/C0/698836027.class	200	OK	2,18 s	444bytes	9 354bytes
211	28.11.23, 01:08:02	28.11.23, 01:08:07	GET	https://msk.rt.ru/WEB-INF/classes/11/7720458.class	200	OK	4,3 s	444bytes	9 353bytes
212	28.11.23, 01:08:07	28.11.23, 01:08:11	GET	https://msk.rt.ru/WEB-INF/classes/9/5486605e.class	200	OK	4,64 s	444bytes	9 354bytes
213	28.11.23, 01:08:11	28.11.23, 01:08:13	GET	https://msk.rt.ru/WEB-INF/classes/13/5091031.class	200	OK	2,19 s	444bytes	9 354bytes
214	28.11.23, 01:08:13	28.11.23, 01:08:31	GET	https://msk.rt.ru/WEB-INF/classes/9/76996262e.class	200	OK	17,52 s	444bytes	9 354bytes
215	28.11.23, 01:08:31	28.11.23, 01:08:33	GET	https://msk.rt.ru/WEB-INF/classes/15/316172.class	200	OK	2,34 s	444bytes	9 354bytes
216	28.11.23, 01:08:33	28.11.23, 01:08:36	GET	https://msk.rt.ru/WEB-INF/classes/L3/55271368e.class	200	OK	2,51 s	444bytes	9 354bytes
217	28.11.23, 01:08:36	28.11.23, 01:08:45	GET	https://msk.rt.ru/WEB-INF/classes/C3/823255 18e.class	200	OK	9,22 s	444bytes	9 354bytes
218	28.11.23, 01:08:45	28.11.23, 01:08:50	GET	https://msk.rt.ru/WEB-INF/classes/48/209139.class	200	OK	4,96 s	444bytes	9 353bytes
219	28.11.23, 01:08:50	28.11.23, 01:08:53	GET	https://msk.rt.ru/WEB-INF/classes/1/790861.class	200	OK	3,44 s	444bytes	9 354bytes
220	28.11.23, 01:08:53	28.11.23, 01:08:59	GET	https://msk.rt.ru/WEB-INF/classes/C2/63136806.class	200	OK	5,2 s	444bytes	9 354bytes
221	28.11.23, 01:08:59	28.11.23, 01:09:01	GET	https://msk.rt.ru/WEB-INF/classes/1/52187266.class	200	OK	2,45 s	444bytes	9 354bytes
222	28.11.23, 01:09:01	28.11.23, 01:09:08	GET	https://msk.rt.ru/WEB-INF/classes/48/8905046.class	200	OK	7,29 s	444bytes	9 354bytes
223	28.11.23, 01:09:08	28.11.23, 01:09:27	GET	https://msk.rt.ru/WEB-INF/classes/47/5218727.class	200	OK	18,84 s	444bytes	9 353bytes
224	28.11.23, 01:09:27	28.11.23, 01:09:30	GET	https://msk.rt.ru/WEB-INF/classes/C1/52187266.class	200	OK	2,71 s	444bytes	9 353bytes
225	28.11.23, 01:09:30	28.11.23, 01:09:53	GET	https://msk.rt.ru/WEB-INF/classes/46/8629999.class	200	OK	22,86 s	444bytes	9 354bytes
226	28.11.23, 01:09:53	28.11.23, 01:09:55	GET	https://msk.rt.ru/WEB-INF/classes/1/78425529.class	200	OK	2,19 s	444bytes	9 354bytes
227	28.11.23, 01:09:55	28.11.23, 01:10:01	GET	https://msk.rt.ru/WEB-INF/classes/46/2312512.class	200	OK	6,5 s	444bytes	9 354bytes

Рисунок 4.43 - Аналіз сайту msk.rt.ru

Було знайдено всього 17 вразливостей, рис. 4.44:

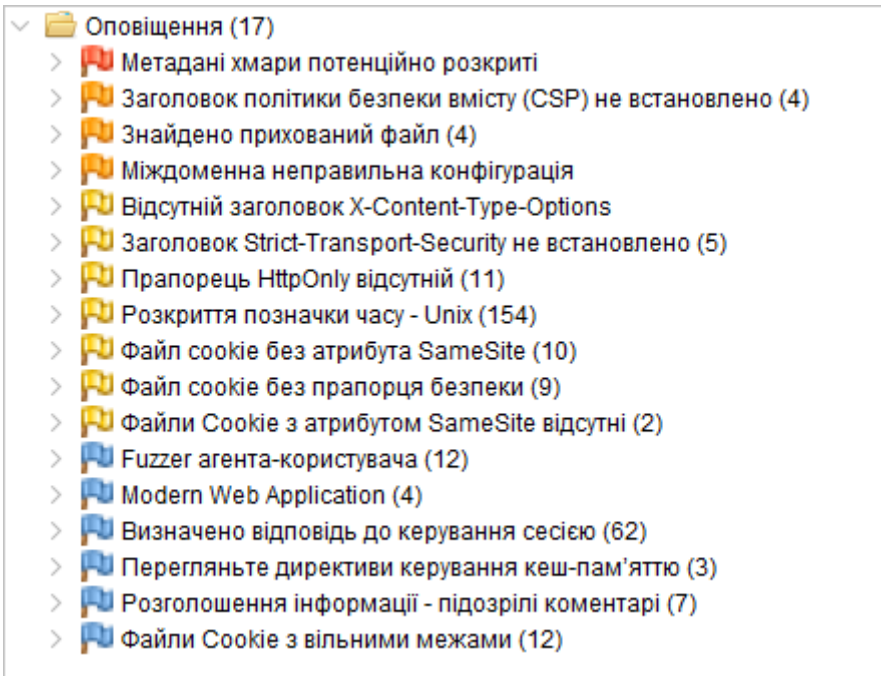


Рисунок 4.44 – Знайдені вразливості

Серед цих вразливостей 1 критичного рівня, 3 середнього рівня, 7 низького рівня та 6 інформаційного рівня.

Всі вразливості вже були описані вище. Загалом, цей сайт можна вважати безпечним. Тому що серед вразливостей немає вразливості яка має великого ризику для безпеки сайту.

Оскільки skipfish не може використовувати proxy тому він не зміг просканувати більшість сайту, рис.4.45.

Було знайдено 1 вразливість критичного рівня, 6 вразливостей інформативного рівня і 1 застережень з позначкою XSS яке було описане вище, рис. 4.46.

Вразливість "Put request accepted" (запит PUT прийнятий) - це тип вразливості безпеки вебдодатків, яка виникає, коли вебдодаток дозволяє користувачам виконувати запити PUT на будь-які ресурси.

Запит PUT використовується для внесення змін до ресурсу. Це означає, що зловмисник може використовувати вразливість "Put request accepted" для внесення змін до будь-яких ресурсів вебдодатку, включаючи:

— дані користувачів: зловмисник може використовувати вразливість для внесення змін до даних користувачів, таких як імена користувачів, паролі або інші конфіденційні дані;

— конфігурацію вебдодатку: зловмисник може використовувати вразливість для внесення змін до конфігурації вебдодатку, що може призвести до компрометації безпеки вебдодатку;

— файли вебдодатку: зловмисник може використовувати вразливість для завантаження шкідливих файлів на сервер вебдодатку.

Наслідки вразливості "Put request accepted" можуть бути серйозними.

Якщо ця вразливість не буде виправлена, зловмисник може використовувати її для крадіжки даних користувачів, компрометації безпеки вебдодатку або для інших шкідливих дій.

Рисунок 4.45 – Сканування сайту msk.rt.ru

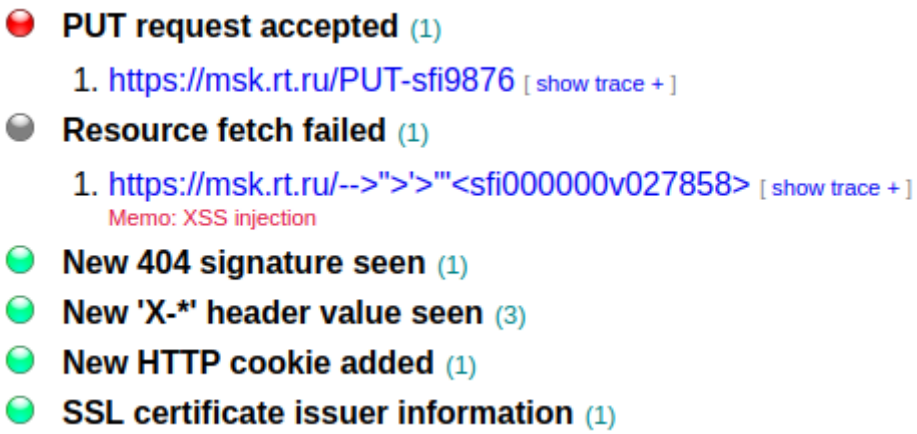


Рисунок 4.46 – Вразливості знайдені за допомогою skipfish

4.10 Сайт rosneft.ru

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.47: www. rosneft.ru

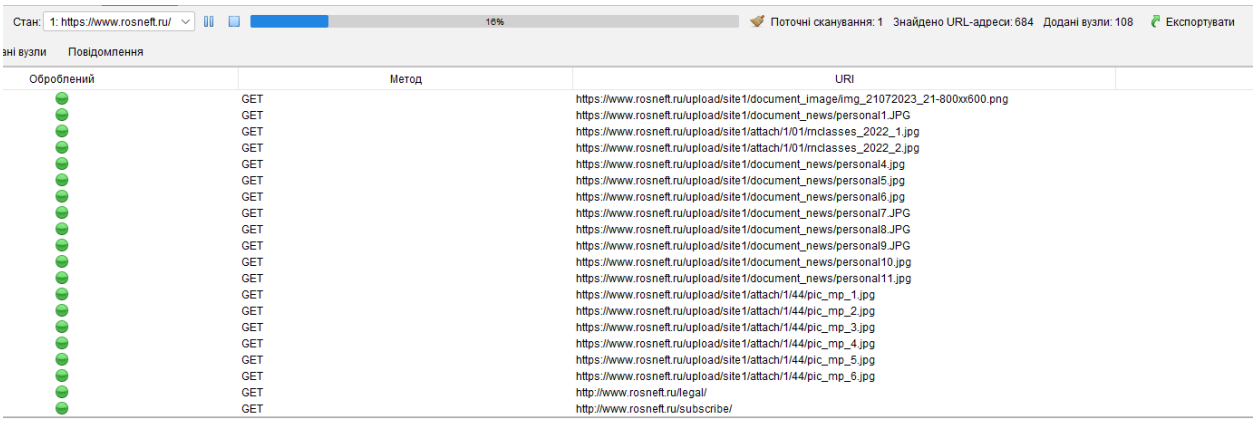


Рисунок 4.47 - Аналіз сайту rosneft.ru

Було знайдено всього 15 вразливостей, рис. 4.48:

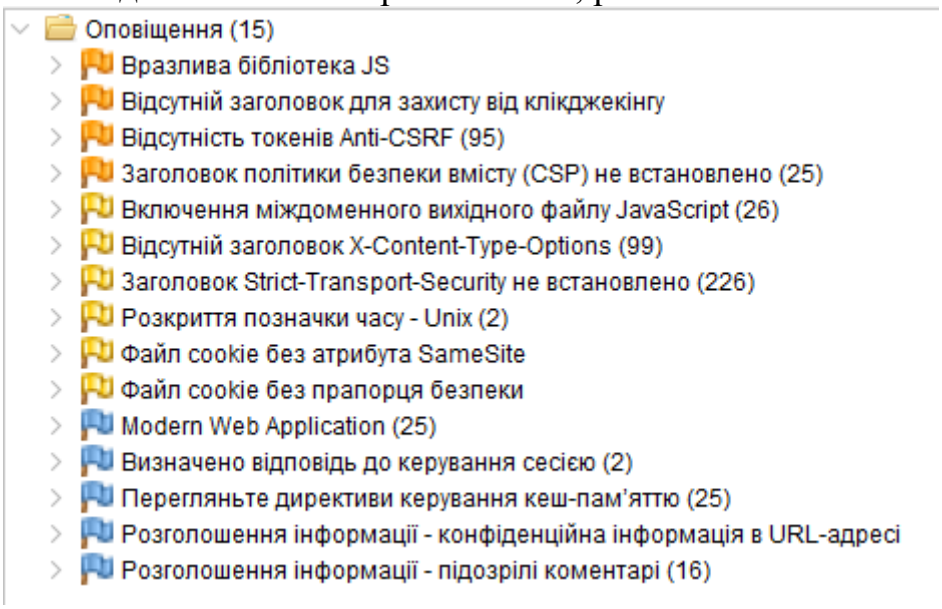


Рисунок 4.48 – Знайдені вразливості

Серед цих вразливостей 4 середнього рівня, 6 низького рівня та 5 інформаційного рівня.

Всі вразливості вже були описані вище.

Загалом, цей сайт можна вважати небезпечним. Тому що серед вразливостей є вразливість яка може бути з великим ризиком надати особисті дані користувачів: «Вразливість JS Бібліотеки (Library)».

Так як skipfish не може використовувати proxy тому він не зміг просканувати сайт, рис. 4.49 та рис. 4.50.

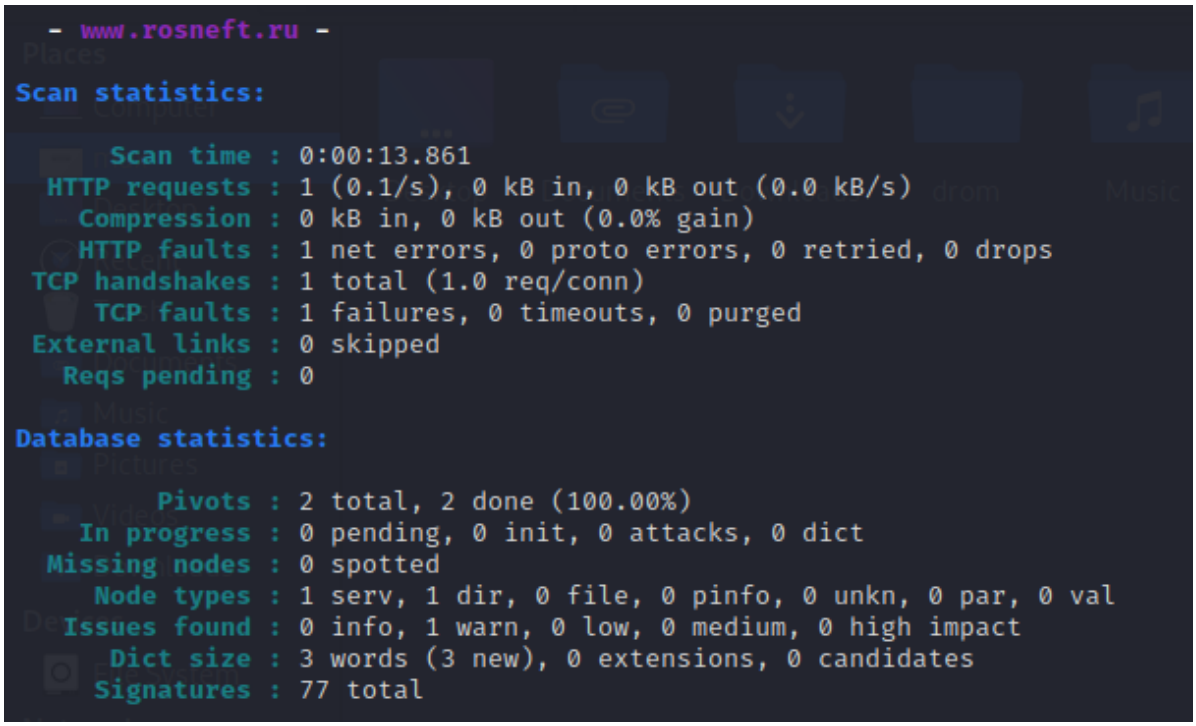


Рисунок 4.49 – Сканування сайту rosneft.ru

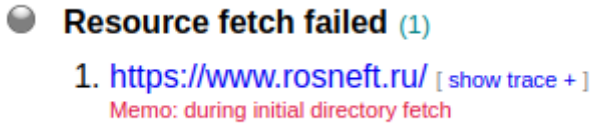


Рисунок 4.50 - Помилка підключення до сайту rosneft.ru

4.11 Сайт autokraz.com.ua

За допомогою OWASP ZAP та публічних проксі було проведено аналіз сайту, рис. 4.51: www.autokraz.com.ua.

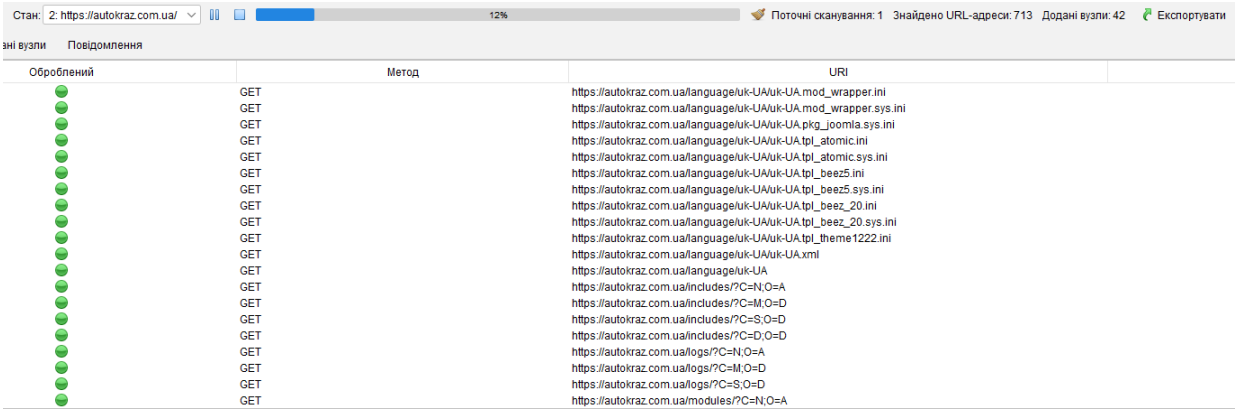


Рисунок 4.51 - Аналіз сайту autokraz.com.ua

Було знайдено всього 18 вразливостей, рис. 4.52:

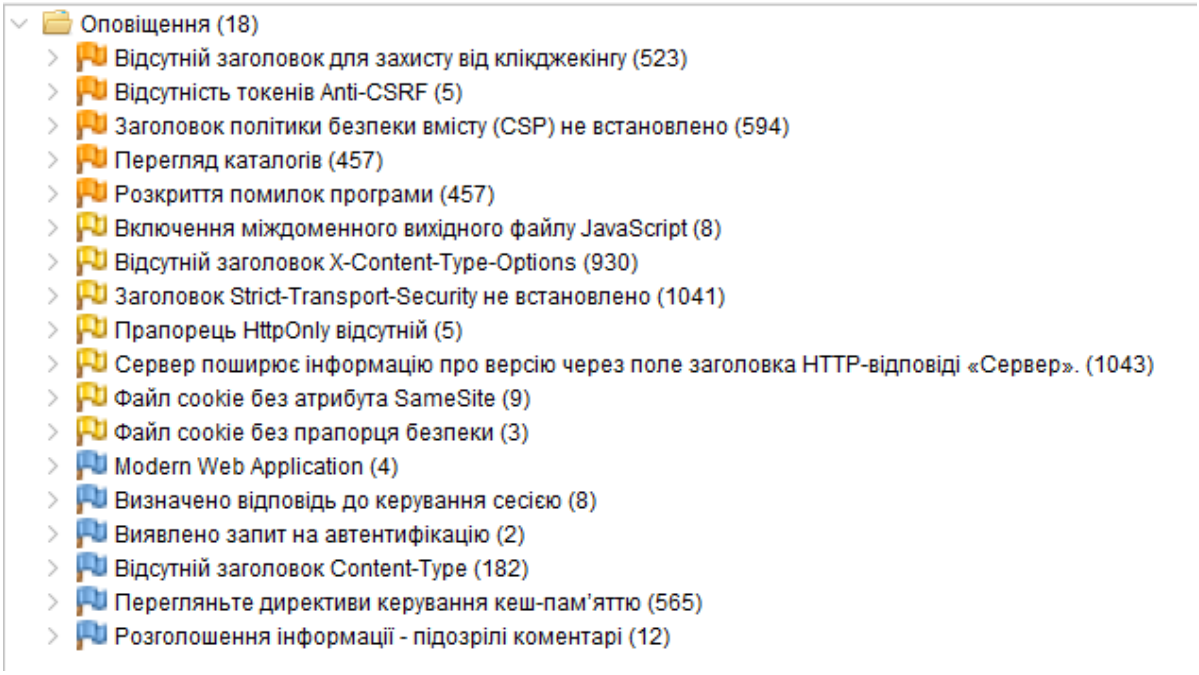


Рисунок 4.52 – Знайдені вразливості

Серед цих вразливостей 5 середнього рівня, 7 низького рівня та 6 інформаційного рівня.

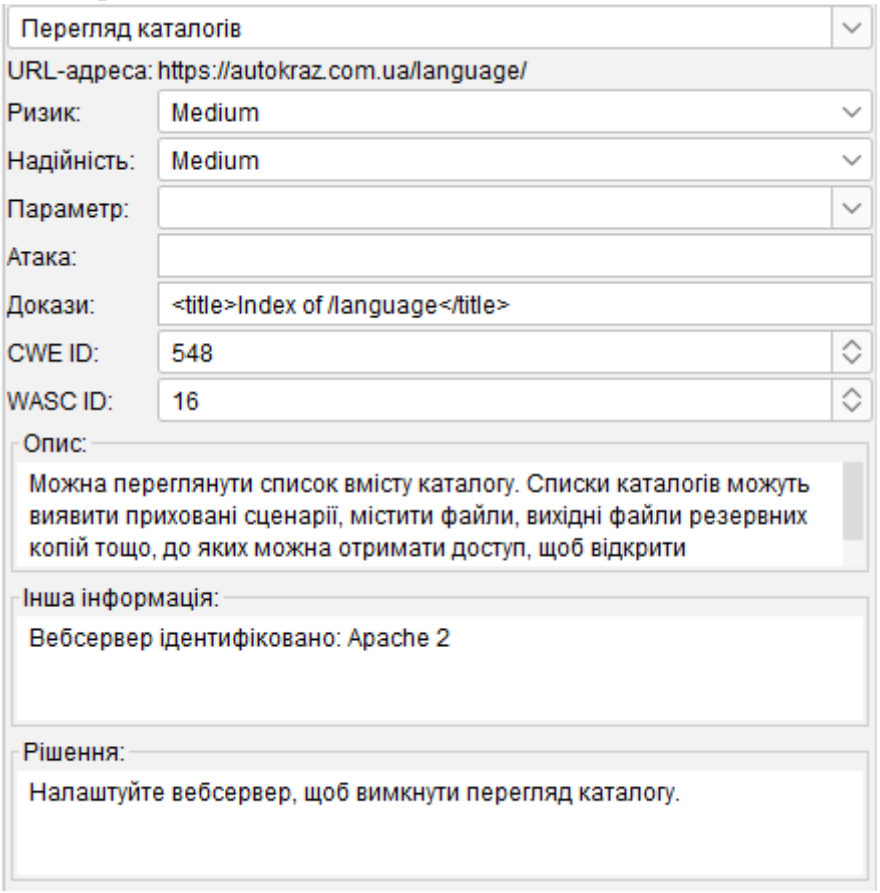


Рисунок 4.53 - Аналіз сайту autokraz.com.ua

1 Вразливість «Перегляд каталогів» є небезпечною, оскільки вона

дозволяє зловмисникам переглядати вміст вебкаталогів, які не повинні бути доступні для користувачів, рис. 4.53. Ця вразливість виникає через те, що вебсайти часто не належним чином обробляють вхідні дані, які надсилаються до вебсервера.

Зловмисник може використовувати цю вразливість, щоб отримати доступ до конфіденційної інформації, яка зберігається на вебсервері, наприклад, до імен користувачів, паролів або інших даних.

2 Вразливість «Розкриття помилок програми» є небезпечною, оскільки вона дозволяє зловмисникам отримувати конфіденційну інформацію про вебсайт, наприклад, про структуру бази даних, код вебсайту або інші дані, рис. 4.54. Ця вразливість виникає через те, що вебсайти часто не належним чином обробляють помилки, які виникають під час виконання коду.

Зловмисник може використовувати цю вразливість, щоб отримати доступ до конфіденційної інформації, яка може бути використана для проведення атак на вебсайт або для крадіжки даних користувачів.

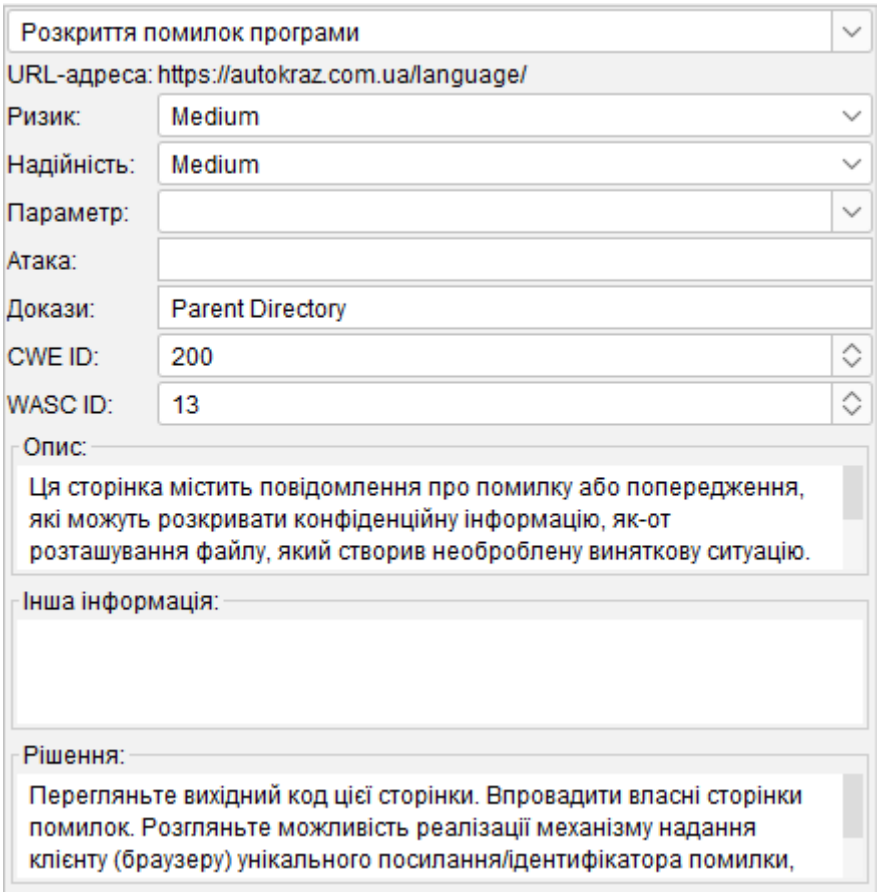


Рисунок 4.54 - Аналіз сайту autokraz.com.ua

Загалом, цей сайт можна вважати небезпечним. Тому що серед вразливостей є вразливість яка може бути з великим ризиком відключення сайту: «Розкриття помилок програми» та «Перегляд каталогів».

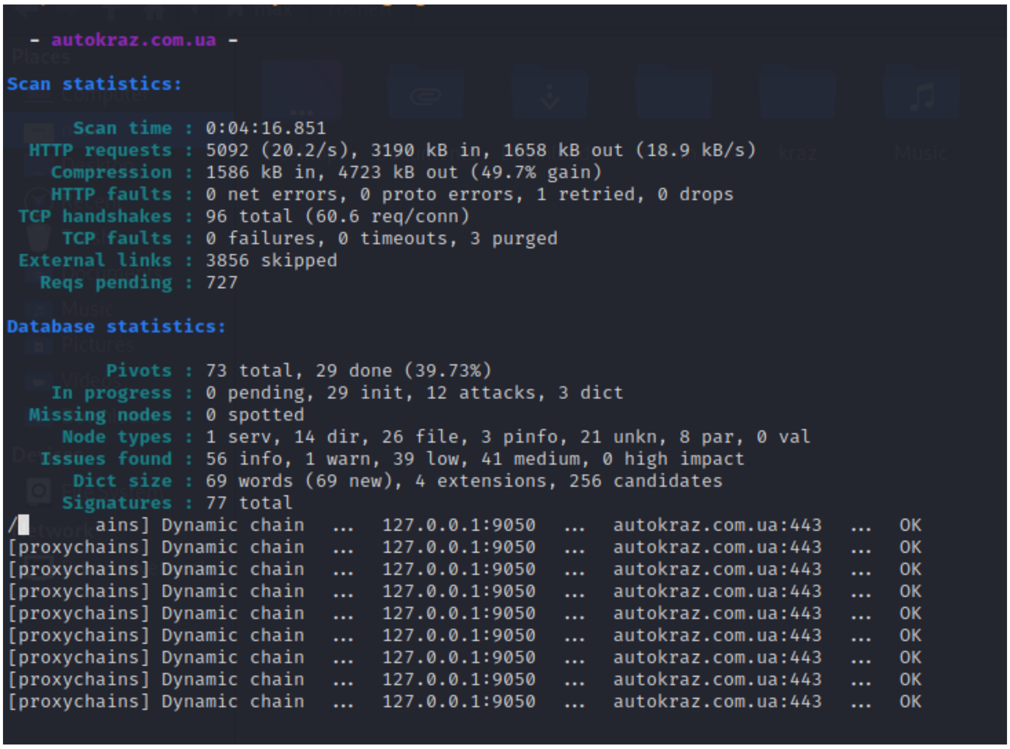


Рисунок 4.55 – Сканування сайту autokraz.com.ua

За допомогою skipfish було проведено аналіз сайту, рис. 4.55: autokraz.com.ua.

Було знайдено 41 вразливость середнього рівня, 40 низького рівня, 37 застережень і 758 інформаційного рівня, рис. 4.56

Вразливість «External content embedded on a page» була описана вище.

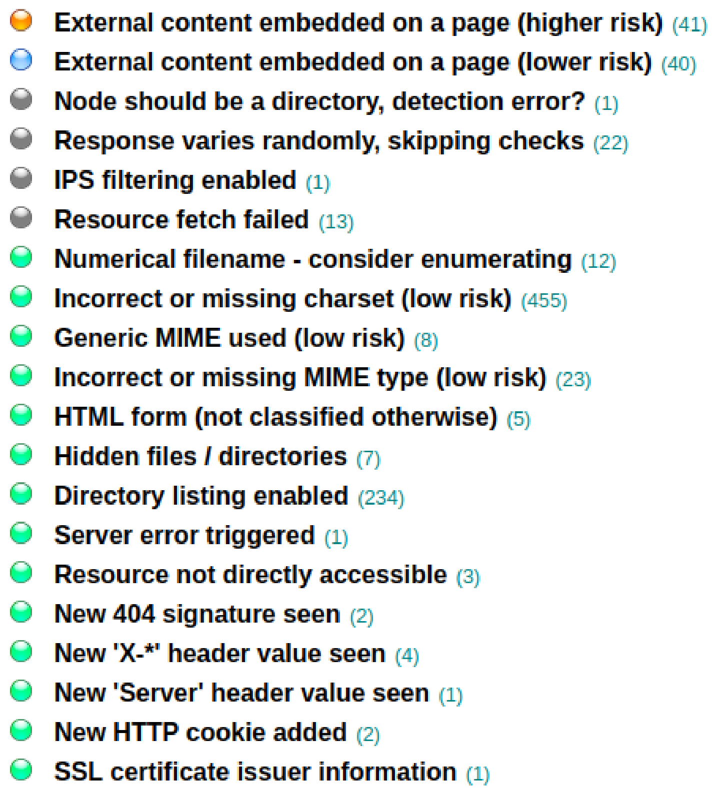


Рисунок 4.56 – Вразливості знайдені за допомогою skipfish

Взагалом цей сайт можна вважати безпечним тому що є незначні помилки але вони не занадто суттєві.

4.12 Статистика знайдених вразливостей

У процесі дослідження було проведено аналіз десяти вебсайтів.

За допомогою інструментів OWASP ZAP та Skipfish було виявлено всього 260 вразливості, з яких:

- 6 критичних (2,3 %)
- 53 середнього рівня (20,4%)
- 82 низького рівня (31,5%)
- 104 інформаційного рівня (40%)
- 15 застережень (5,8%)

Діаграма рівня вразливостей наведена на рис. 4.57 нижче.

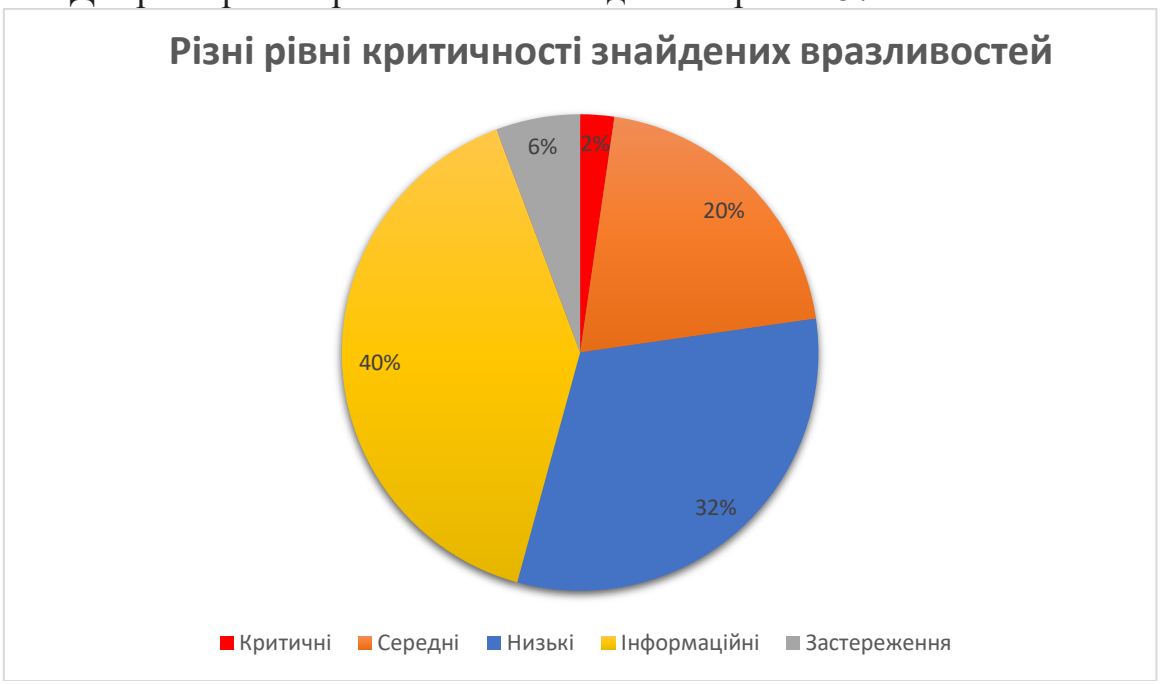


Рисунок 4.57 – Діаграма різних рівнів критичності знайдених вразливостей

З них:

1) OWASP ZAP (195):

Критичних (4):

- Метадані хмари потенційно розкриті – 2
- Розголошення персональної ідентифікаційної інформації – 2

Середніх (49):

- Заголовок Content Security Police (CSP) не задано – 10
- Знайдено прихований файл – 2
- Вразливість JS бібліотеки – 4
- Міждоменна неправильна конфігурація – 5
- Перегляд каталогів – 1

– Розкриття помилок програми – 1
– Заголовок політики безпеки вмісту CSP не встановлено – 1
– Відсутній заголовок (Header) для захисту від клікджекінгу – 9
– Відсутні токени проти CSRF атак – 9
– CSP: script-src unsafe-eval – 1
– CSP: script-scr unsafe-inline – 2
– CSP: style-scr unsafe-inline – 2
– CSP: Директива символів узагальнення – 2

Низьких (76):

– Cookie No HttpOnly Flag – 8
– Cookie без атрибута SameSite – 10
– Включення вихідного файлу міждоменного JavaScript – 10
– Заголовок X-Content-Type-Options відсутній – 10
– Сервер утікає інформацію через поля заголовка HTTP-відповіді «X-Powered-By» - 2
– Сервер утікає інформацію про версію через поле заголовка HTTP-відповіді «Server» - 6
– Cookie без прапору безпеки – 9
– Заголовок Strict-Transport-Security не встановлено – 10
– Розкриття відмітки часу – 7
– Виявлено велике перенаправлення (можливий витік конфіденційної інформації) – 1
– Server Leaks information via «X-Powered-By» HTTP Response Header Field – 2
– CSP: Повідомлення – 1

Інформаційних (66):

– Ідентифіковано відповідь на управління сеансами – 5
– Перегляньте директиви управління кешем – 2
– Розкриття інформації підозрілі коментарі – 10
– Cookie з довільним обмеженням – 5
– Перегляньте директиви управління кешем – 10
– Агент користувача Fuzzer – 3
– Сучасний вебдодаток – 10
– Виявлено заголовок лише для звітів політики безпеки вмісту – 1
– Визначено відповідь до керування сессією – 8
– Відсутній заголовок Content-Type – 2
– Розголошення інформації – конфіденційна інформація в URL-адресі – 3
– Виявлено запит на автентифікацію – 3
– Керований користувачем атрибут HTML-елемента (потенційний XSS) – 2
– Отримано з кешу – 2

2) Skipfish (65):
Критичних (2):
- Query injection vector – 1
- PUT request accepted – 1

Середніх (4):
- External content embedded on a page (higher risk) – 3
- Incorrect or missing charset (higher risk) – 1

Низьких (6):
- External content embedded on a page (lower risk) - 2
- HTML form with no apparent XSRF protection – 2
- SSL certificate host name mismatch – 1
- SSL certificate expired or not yet valid – 1

Інформаційних (38):
- Numerical filename - consider enumerating – 2
- Unknown form field (can't autocomplete) – 1
- New 404 signature seen – 5
- New 'X-*' header value seen – 4
- New HTTP cookie added – 4
- SSL certificate issuer information – 4
- Incorrect or missing charset (low risk) – 3
- Incorrect or missing MIME type (low risk) – 2
- HTML form (not classified otherwise) – 2
- Hidden files / directories – 2
- Resource not directly accessible – 2
- New 'Server' header value seen – 4
- Generic MIMO used (low risk) – 1
- Directory listing enabled – 1
- Server error triggered – 1

Застережень (15):
- Node should be a directory, detection error – 2
- Response varies randomly, skipping checks – 2
- Directory behavior checks failed (no brute force) – 1
- Resource fetch failed – 9
- IPS filtering enabled – 1

Статистика порівняння знайдених вразливостей за програмами OWASP ZAP та Skipfish, рис. 4.58, рис. 4.60:

Рисунок 4.58 – Діаграма порівняння аудиторів загроз

За гістограмою порівняння програм для кожного сайту видно що OWASP ZAP є більш ефективною в порівнянні з Skipfish, показано на рис. 4.59 нижче та 4.61 і 4.62.

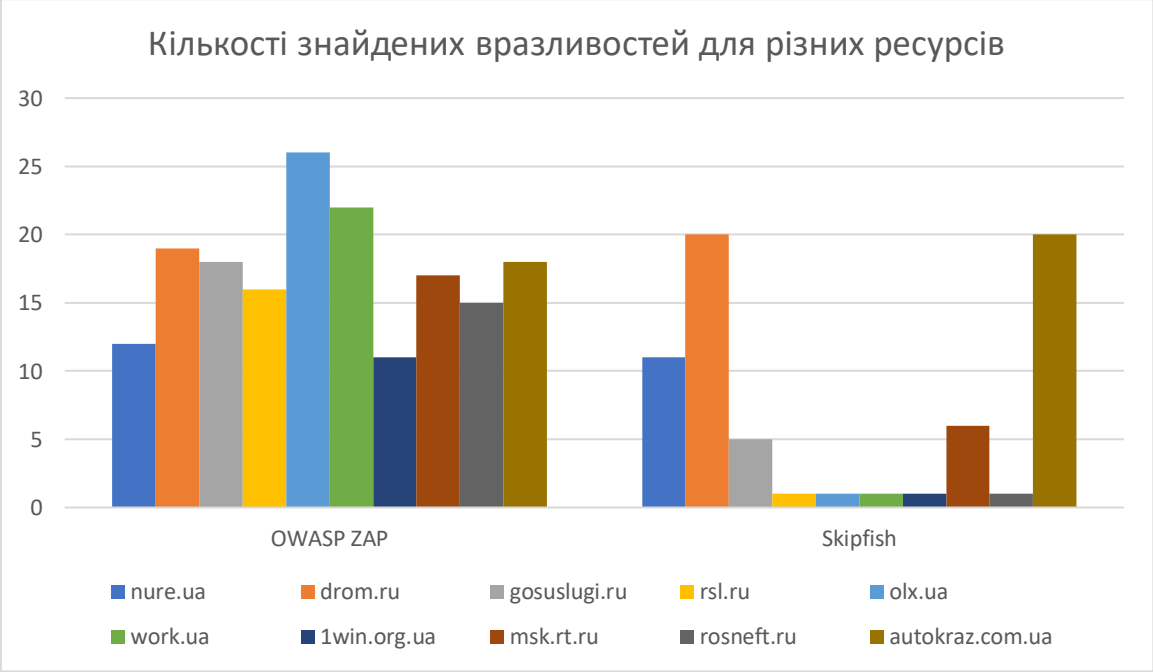


Рисунок 4.59 – Гістограма Кількості знайдених вразливостей для різних ресурсів

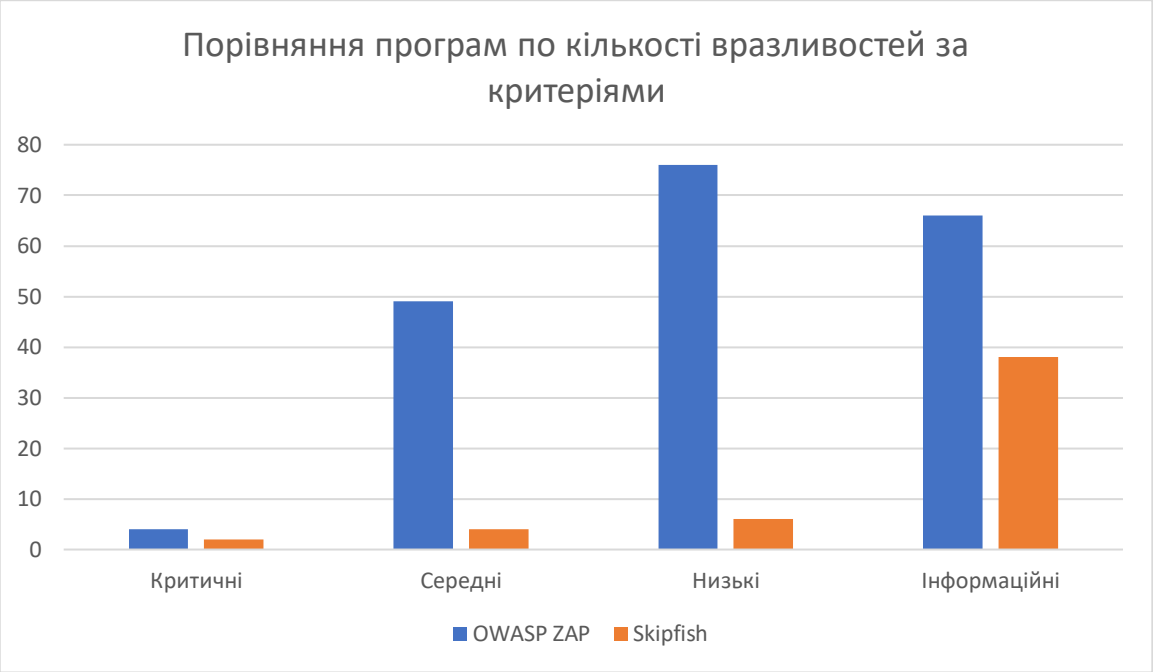


Рисунок 4.60 – Гістограма критерію вразливостей для OWASP ZAP

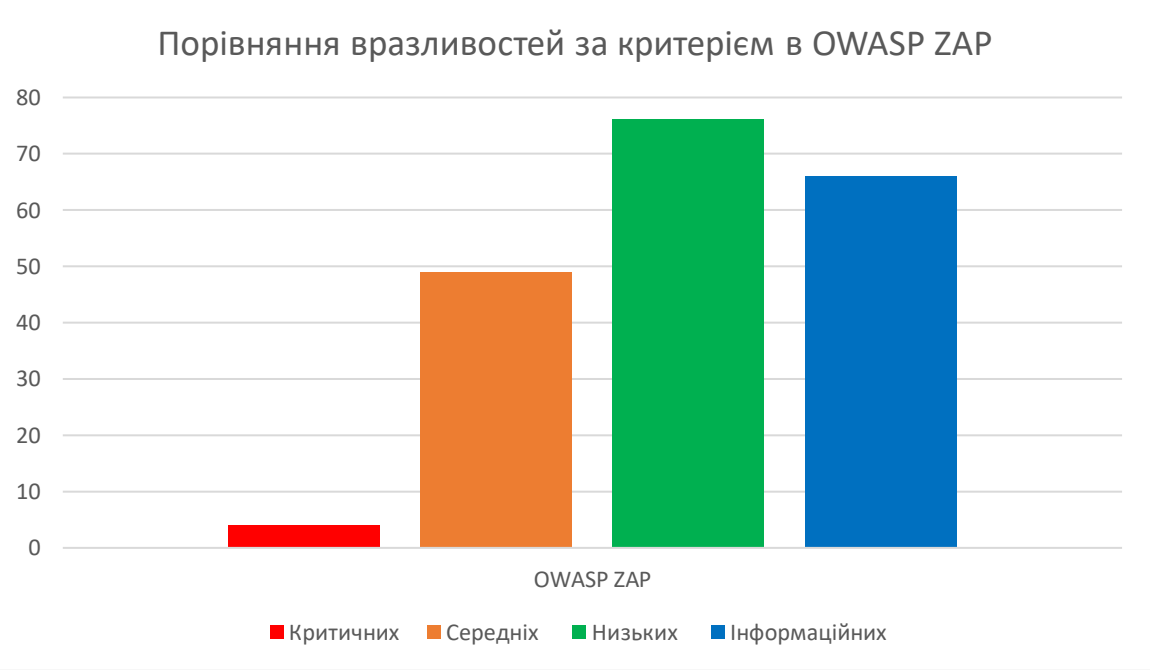


Рисунок 4.61 – Гістограма кількості вразливостей за критерієм для OWASP ZAP

Рисунок 4.62 – Гістограма кількості вразливостей за критерієм для Skipfish

5 РОЗРОБКА РЕКОМЕНДАЦІЙ ПО ПІДВИЩЕННЮ ЗАХИЩЕНОСТІ КЛІЄНТСЬКОЇ ЧАСТИНИ ВЕБЗАСТОСУНКІВ

5.1 Використання сучасних технологій і практик розробки вебзастосунків

Вразливості безпеки можуть бути виявлені в будь-якому програмному забезпеченні, включаючи клієнтську частину вебзастосунків. Зловмисники можуть використовувати ці вразливості для крадіжки даних, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні.

Щоб захистити свій вебзастосунок від вразливостей, слід регулярно перевіряти його за допомогою інструментів для сканування безпеки.

Важливо використовувати сучасні технології безпеки, щоб захистити клієнтську частину вебзастосунку від атак зловмисників. Ось кілька конкретних рекомендацій:

— використання HTTPS для всіх запитів до вебзастосунку. HTTPS шифрує всі дані, що передаються між веббраузером і вебсервером, що робить їх недоступними для перехоплення зловмисниками;

— впровадження аутентифікації та авторизації для всіх користувачів вебзастосунку. Це допоможе захистити вебзастосунок від несанкціонованого доступу;

— шифрування всіх конфіденційних даних, які зберігаються або передаються через вебзастосунок. Це допоможе захистити ці дані від крадіжки або розголошення.

Сучасні практики розробки вебзастосунків можуть допомогти підвищити безпеку клієнтської частини вебзастосунку. Ось кілька конкретних рекомендацій:

— розробляйте клієнтську частину вебзастосунку на основі компонентів. Це допоможе полегшити тестування і обслуговування безпеки вебзастосунку;

— використовуйте модульну розробку для клієнтської частини вебзастосунку. Це також допоможе полегшити тестування і обслуговування безпеки вебзастосунку.

Ось кілька конкретних прикладів того, як можна використовувати ці рекомендації:

— щоб реалізувати HTTPS, можна використовувати фреймворк, який підтримує HTTPS за замовчуванням;

— щоб реалізувати аутентифікацію та авторизацію, можна використовувати фреймворк, який надає готові рішення для цих задач;

— щоб реалізувати шифрування даних, можна використовувати

бібліотеки або фреймворки, які надають готові рішення для цього завдання;

— щоб регулярно перевіряти клієнтську частину вебзастосунку на наявність вразливостей, можна використовувати інструменти для сканування безпеки, які дозволяють автоматизувати цей процес;

— щоб розробляти клієнтську частину вебзастосунку на основі компонентів, можна використовувати фреймворк, який підтримує цей підхід;

— щоб розробляти клієнтську частину вебзастосунку з використанням модульної розробки, можна використовувати фреймворк, який підтримує цей підхід.

За даними OWASP, Бібліотека ReactJS (або React) завдяки своїй гнучкості є найбільш поширеним рішенням для створення клієнтської частини вебзастосунків (завдяки JSX (JavaScript XML) синтаксису). У зв'язку зі зростанням складності розроблюваних застосунків і збільшенням обсягів даних, проблеми безпеки таких продуктів невпинно зростають. Так, серед найпоширеніших загроз для вебзастосунків, що створені із застосуванням ReactJS слід виділити:

— вразливість Zip Slip, є критично важливою і дозволяє маніпулювати застосунком за рахунок вилучення архіву та інтегрування будь-яких файлів у систему;

— XSS (або Cross-Site Scripting), що передбачає можливість виконання шкідливого програмного коду, котрий приймається за справжній і виконується за стосунком;

— XXE (XML External Entity) атаки на XML парсери у випадках, коли ті некорректно опрацьовують посилання на деякі зовнішні сутності;

— arbitrary Code Execution, яка працює за принципом впровадження зловмисником експлойту, що забезпечує віддалене ініціювання виконання нештатного програмний коду та/або маніпулювання діями скомпрометованого додатку та/або апаратного засобу;

— вразливість Broken Authentication здійснюється завдяки існування передумов компрометації діючих процедур авторизації (тобто недосконаліть автентифікації);

— атаки типу SQL Injection, де зловмисник може відправляти шкідливі SQLзапити до бази даних та маніпулювати даними бази даних задля отримання своїх цілей[10].

Серед способів покращення ReactJS від OWASP , що обумовлюють показники безпеки вебзастосунків, можна виділити наступні:

— захист від XSS, за допомогою використання технології прив'язки даних;

— поточний моніторинг URL посилань та контроль впровадження стороннього шкідливого ПЗ, котре може здійснюватися через URL;

— своєчасне оновлення використовуваних бібліотек. Актуальні версії React позбавлені від багатьох вразливостей минулих релізів;

— при розміщенні коду до вузлів об’єктної моделі документа (DOM – Document Object Model), слід не допускати прямого відкритого доступу;

— забезпечити протидію вразливостям, що діють за принципом впровадження даних JSON (JavaScript Object Notation). Зазвичай ці дані відображаються на боці сервера і передаються разом зі сторінками React у форматі обміну даними JSON;

— при розгортанні HTML коду, його слід розміщувати відразу у DOM вузли, що потребує ретельної перевірки всіх процедур;

— забезпечити регулярність перевірок поточного стану вразливостей (інструмент OWASP Dependency-Check), що надає змогу виявляти відомі різновиди вразливостей, які розташовуються у залежностях проекту, перш ніж додавати їх до проекту;

— виключити застосування сумнівного коду із бібліотек та використовувати ПЗ, яке оптимізує створюваний код.

Впроваджувати корисні функції React для серверного відображення даних на клієнті, що надасть змогу екранувати дані при їх візуалізації на клієнтській частині застосунку у автоматичному режимі.

Виконання цих рекомендацій може допомогти підвищити безпеку клієнтської частини вебзастосунків і захистити дані користувачів.

5.2 Встановлення актуальних версій програмного забезпечення

Встановлення актуальних версій програмного забезпечення для клієнтської частини вебзастосунку важливо з кількох причин:

— безпека: актуальні версії програмного забезпечення часто містять виправлення вразливостей, які можуть бути використані зловмисниками для крадіжки даних, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні;

— функціональність: актуальні версії програмного забезпечення часто містять нові функції та поліпшення, які можуть зробити використання вебзастосунку більш зручним і ефективним;

— стабільність: актуальні версії програмного забезпечення часто більш стабільні, ніж старі версії, і менш схильні до збоїв.

Ось кілька конкретних прикладів того, як зловмисники можуть використовувати вразливості в старому програмному забезпеченні:

— крадіжка даних: зловмисник може використовувати вразливість у веббраузері для крадіжки імен користувачів, паролів або інших конфіденційних даних, які вводяться користувачем на вебсайті;

— впровадження шкідливого програмного забезпечення: зловмисник

може використовувати вразливість у веббраузері для впровадження шкідливого програмного забезпечення на комп'ютер користувача. Це шкідливе програмне забезпечення може використовуватися для крадіжки даних, пошкодження системи або навіть для контролю над комп'ютером;

— відмова в обслуговуванні: зловмисник може використовувати вразливість у веббраузері для відмови в обслуговуванні вебсайту. Це може призвести до того, що вебсайт стане недоступним для користувачів, що може мати серйозні наслідки для бізнесу.

Щоб захистити свої дані та вебзастосунок від атак зловмисників, важливо регулярно встановлювати актуальні версії програмного забезпечення. Більшість веббраузерів і інших програм для клієнтської частини вебзастосунків автоматично перевіряють наявність оновлень і пропонують їх встановити. Користувачі також можуть самостійно перевіряти наявність оновлень на вебсайтах розробників програмного забезпечення.

5.3 Використання інструментів для виявлення вразливостей

Інструменти для виявлення вразливостей в програмному коді - це програмне забезпечення, яке використовується для пошуку вразливостей у програмному коді.

Вразливості безпеки - це помилки в програмному коді, які можуть бути використані зловмисниками для крадіжки даних, впровадження шкідливого програмного забезпечення або навіть для відмови в обслуговуванні.

Існує багато різних типів інструментів для виявлення вразливостей в програмному коді. Деякі з них використовують статичний аналіз коду, тобто вони аналізують код без його виконання. Інші використовують динамічний аналіз коду, тобто вони аналізують код під час його виконання.

Найбільш популярним інструментом для виявлення вразливостей в програмному коді є Snyk. Snyk - це інструмент для виявлення вразливостей в програмному коді, який використовує статичний аналіз коду. Snyk підтримує широкий спектр мов програмування, включаючи JavaScript, Python, Java, C/C++, PHP та інші.

Статичні аналізатори коду працюють, скануючи код на наявність певних шаблонів, які можуть вказувати на вразливості. Наприклад, статичний аналізатор може сканувати код на наявність використання ненадійних бібліотек або ненадійних методів.

Динамічне аналізатори коду працюють, запускаючи код в контрольованому середовищі і спостерігаючи за його поведінкою. Наприклад, динамічний аналізатор може намагатися викликати вразливість, щоб побачити, чи зможе зловмисник її використовувати.

Інструменти для виявлення вразливостей в програмному коді можна використовувати для різних цілей, таких як:

— тестування безпеки програмного коду: інструменти для виявлення вразливостей в програмному коді можна використовувати для тестування безпеки програмного коду, щоб виявити вразливості, які можуть бути використані зловмисниками;

— автоматизація процесу виявлення вразливостей: інструменти для виявлення вразливостей в програмному коді можна використовувати для автоматизації процесу виявлення вразливостей, щоб значно прискорити цей процес;

— пошук вразливостей в відкритому коді: інструменти для виявлення вразливостей в програмному коді можна використовувати для пошуку вразливостей в відкритому коді, щоб захистити користувачів від атак зловмисників.

Інструменти для виявлення вразливостей в програмному коді можуть бути ефективним інструментом для підвищення безпеки програмного коду. Однак важливо пам'ятати, що ці інструменти не є бездоганними. Вони можуть пропускати деякі вразливості, і вони можуть помилково генерувати помилкові позитивні результати.

Щоб отримати максимальну користь від інструментів для виявлення вразливостей в програмному коді, важливо використовувати їх у поєднанні з іншими методами виявлення вразливостей, такими як статичний аналіз коду, динамічний аналіз коду і аналіз коду експертами.

5.4 Сучасні нормативні документи в галузі кібербезпеки для створення вебресурсів

Сучасні нормативні документи в галузі кібербезпеки для створення веб ресурсів спрямовані на забезпечення безпеки вебзастосунків та захист даних користувачів. Ці документи встановлюють вимоги до безпеки вебзастосунків, а також надають рекомендації щодо їх реалізації.

Ось деякі з найважливіших нормативних документів в галузі кібербезпеки для створення веб ресурсів:

— ISO/IEC 27001:2013 - Стандарт системи управління інформаційною безпекою (ISMS);

— ISO/IEC 27701:2019 - Доповнення до ISO/IEC 27001:2013 для захисту персональних даних;

— OWASP Top 10: Список найбільш поширених вразливостей вебзастосунків.

OWASP являє собою головний ресурс з інформацією про стандарти, відомі методи захисту та інструменти безпеки. Через велику базу користувачів

та відкриту спільноту, OWASP роками залишається своєрідною бібліотекою, в котрій містяться різноманітні документи, що допомагають вирішити проблеми безпеки при розробці продуктів.

Головною метою проекту OWASP Top Ten є формулювання основних загроз ІБ, визначення найважливіших та найбільш критичних недоліків захищеності вебзастосунків. Цей список створено за сприяння об'єднаної спільноти проекту. На початку існування ціллю проекту було покращити знання розробників, щодо безпеки, але згодом проект став справжнім стандартом безпеки вебзастосунків. Наразі актуальною є версія 2021 року, котра вийшла відносно недавно й включає найбільш актуальний список загроз та вразливостей.

Останню версію OWASP складено на базі опитувань у відповідній галузі та більш ніж 40 відгуків від компаній, що займаються забезпечення безпеки. Також, була зібрана інформація з більш ніж ста тисяч застосунків та API, що дозволило створити переконливу інформаційну базу задля подальшого аналізу даних.

Останнім часом, вебінструменти та технології зазнали суттєвих змін. Так, мову програмування JavaScript та її фреймворки AngularJS та ReactJS, можливо впевнено назвати своєрідним фундаментом або основою мережі.

Завдяки цим фреймворкам деяка «відповідальність» серверів була перенесена на сторону браузера. В той же час величезну популярність набрала серверна технологія розробки Node.js, котра працює на базу подій. На рис. 5.1 можна бачити зміни у списку OWASP Top Ten [6].

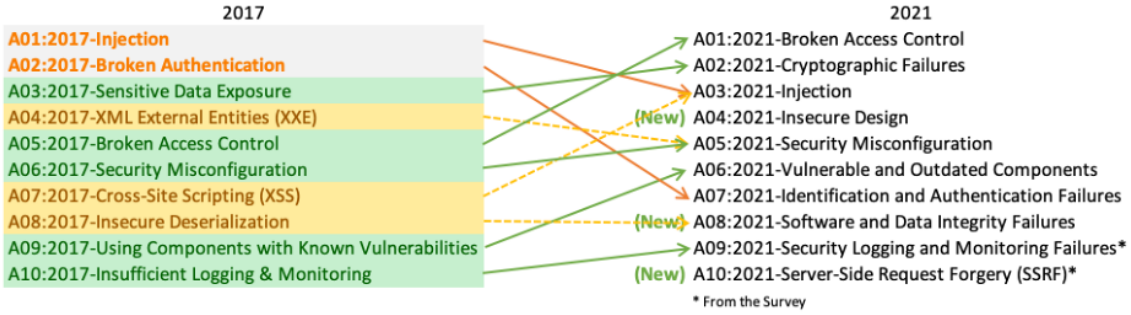


Рисунок 5.1 – OWASP Top Ten вразливостей за 4 роки

Як можна бачити, за цей термін, перелік OWASP зазнав великих змін.

Так, Broken Access Control - піднялася в списку й опинилася на 1-й позиції. В реалізованих вебзастосунках має бути реалізовано доступ до інформації згідно з привілегіями користувачів. У випадку ігнорування безпеки доступу до інформації, це може дозволити користувачам отримати доступ до чутливої інформації без наданих на це повноважень, що може призвести до незворотних наслідків та/або втрати критично важливих даних.

Cryptographic Failures (раніше Sensitive Data Exposure) - піднялася на 2-ге місце. Ця вразливість пов'язана з втратою конфіденційних даних чи зломом

системи, якщо використовувані криптографічні методи не здатні забезпечити достатній рівень ІБ: наприклад, неактуальні криптографічні шифри та протоколи.

Injection (до цієї категорії було додано і XSS (Cross Site Scripting)) - міжсайтове виконання сценаріїв наразі стало невід'ємною частиною цієї категорії (опустилася на 3 місце).

Дана вразливість надає злочинцям змогу впроваджувати у вебзастосунок потрібні їм недекларовані дані та/або інструкції, що забезпечує отримання несанкціонованого доступу до цільових даних. Також, ін'єкції здатні власноруч змінювати вебзастосунок.

Insecure Design - поєднує у собі ризики безпеки, котрі пов'язані з недоліками дизайну (тобто загальної структури і окремих алгоритмічних конструкції додатків). Тому, дизайн має охоплювати шаблони проектування й безпечну еталонну архітектуру програмного забезпечення (ПЗ). Являє нову категорію у релізі загроз 2021 року (4-те місце).

Security Misconfiguration – ця категорія піднялася вгору (на 5-ту позицію), через розширення кількості використань ПЗ з широкими можливостями їх налаштування, що обумовлює потенційну можливість нелегітимного доступу до інформації та маніпулювання даними, у тому числі, модифікацію і зміну даних тощо.

Vulnerable and Outdated Components (раніше, Using Components with Known Vulnerabilities) – є єдиною категорією, яка не має жодних CVE (Common Vulnerabilities and Exposures), зіставлених із включеними CWE (Common Weakness Enumeration), піднялася на 6 місце.

Охоплює всі застарілі складові ПЗ, експлуатація складає критичні ризики ІБ системи. Потребує регулярне сканування системи, що дозволяє завчасно виявляти наявні проблеми ІБ (патчі безпеки, використання експлойт-паків).

Identification and Authentication Failures (раніше, Broken Authentication) – ця категорія, відтепер, включає CWE, котрі більшою мірою відносяться до помилок ідентифікації, опустилася на 7 місце. Категорія охоплює вразливості, що включають в себе фальсифікацію облікових даних та "brute force" атаки. Вразливості цієї категорії є похідними недоліків застосованих механізмів автентифікації та перевірки сеансу користувача.

Software and Data Integrity Failures - нова категорія (8 місце рейтингу), яка фокусується на ідеях оновлень ПЗ, критичних даних і конвеєрів CI/CD (Continuous Integration/Continuous Delivery) без перевірки на цілісність, що забезпечує злочинців можливостями використовувати дані, котрі надходять до серверу, для забезпечення можливостей проведення атак.

Security Logging and Monitoring Failures (раніше, Insufficient Logging & Monitoring) – яка займає 9 місце, була розширена задля включення більшій

кількості типів вразливостей, що пов’язані з незадовільним поточним контролем і фіксуванням подій ІБ. Наявність цих вразливостей обумовлює труднощі зі своєчасністю реагування на збої у роботі застосунків.

Server-Side Request Forgery - нова категорія, яка була додана з опитування галузевих фахівців. Її наявність надає зловмиснику можливість реалізувати недекларованe надсилання (сервером додатку) запитів до обраного зловмисником домену. Ця вразливість може виникнути у разі, якщо вебзастосунок отримує дані без перевірки адреси, яку надає користувач.

Ці документи містять вимоги до безпеки вебзастосунків, такі як:

1 Захист даних користувачів:

— захист конфіденційності: захист даних користувачів від несанкціонованого доступу, використання або розголошення;

— захист цілісності: захист даних користувачів від несанкціонованих змін;

— захист доступу: забезпечення доступу до даних користувачів лише уповноваженим користувачам.

2 Захист від атак:

— захист від вразливостей: захист від атак, які використовують вразливості в вебзастосунку;

— захист від злому: захист від атак, які використовують недоліки в реалізації вебзастосунку;

— захист від кібератак: захист від атак, які використовують зовнішні фактори, такі як соціальна інженерія або фішинг.

Реалізація вимог цих документів допоможе підвищити безпеку вебзастосунків та захистити дані користувачів від атак зловмисників.

ВИСНОВКИ

Інтернет став невід'ємною частиною нашого життя. Ми використовуємо його для спілкування, навчання, роботи, розваг та покупок.

В даній магістерській роботі було розглянуто такі протоколи взаємодії користувача з клієнтською частиною вебзастосувань:

— протокол JWT;

— протокол HTTP/HTTPS;

— протокол WebSocket;

— протокол WebRTC.

Зроблено дослідження класифікації вразливостей клієнтської частини вебзастосунків:

— вразливість XSS;

— вразливість SQL-ін’єкції;

— CSS Injection;

— вразливі та застарілі компоненти;

— вразливість CSRF.

У ході дослідження вразливостей клієнтської частини вебзастосунків було виявлено 260 вразливості, які можуть бути використані зловмисниками для атаки на вебзастосунки і викрадення даних користувачів, а також створено статистику вразливостей клієнтської частини вебзастосунків.

Відповідно до результатів дослідження, вразливості клієнтської частини вебзастосунків можна сказати що OWASP ZAP і Skipfish - це два популярні інструменти для пошуку вразливостей вебсайтів, OWASP ZAP має широкий набір функцій, включаючи пасивне, активне та ручне сканування. Він підтримує широкий спектр вразливостей, включаючи SQL-ін'єкцію, XSS та порушення між доменів. OWASP ZAP також має простий у використанні інтерфейс, він знайшов 195 вразливостей.

Однак OWASP ZAP може бути менш ефективним у виявленні деяких вразливостей, ніж Skipfish. Він також може бути менш ефективним у скануванні складних вебсайтів.

Skipfish - це інструмент динамічного сканування, який генерує випадкові запити до вебсайту, щоб виявити вразливості. Він може бути ефективним у виявленні деяких вразливостей, які можуть бути пропущені іншими інструментами. Skipfish також добре підходить для сканування складних вебсайтів, він знайшов 65 вразливостей.

Skipfish має менший набір функцій, ніж OWASP ZAP, і може бути менш ефективним у виявленні поширених вразливостей.

В ході OWASP ZAP є більш ефективнішим порівняно з Skipfish.

Для підвищення безпеки клієнтської частини вебзастосунків розробники повинні дотримуватися таких рекомендацій:

— використовувати сучасні технології та практики розробки вебзастосунків, які враховують вимоги безпеки;

— регулярно перевіряти вебзастосунок на наявність вразливостей, використовуючи інструменти для виявлення вразливостей такі як Snyk;

— впровадити систему управління інформаційною безпекою (ISMS), щоб забезпечити всебічний захист вебзастосунку.

Виконання цих рекомендацій допоможе розробникам підвищити безпеку клієнтської частини вебзастосунків і захистити дані користувачів від атак зловмисників.

Магістерська робота виконана в повному обсязі і відповідає вимогам технічного завдання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Check Point Cyber Security Report 2021.2021. P. 55. URL: https://www.checkpoint.com/downloads/resources/cyber-security-report-2021.pdf (дата звернення 23.12.2023).

2. Number of common IT security vulnerabilities and exposures (CVEs) worldwide from 2009 to 2023 YTD. URL: https://www.statista.com/statistics/500755/worldwide-common-vulnerabilities-and-exposures/ (дата звернення 13.12.2023).

3. Що таке HTTPs: у чому відмінність від http, навіщо потрібний?, 2022, URL: https://hyperhost.ua/info/uk/shho-take-https-u-comu-vidminnist-vid-http-navishho-potribnii (дата звернення 12.12.2023).

4. WebSockets: навіщо потрібні та як з ними працювати, 2023, URL: https://proit.org.ua/websockets-navishcho-potribni-ta-iak-z-nimi-pratsiuvati-2/ (дата звернення 17.11.2023).

5. WebRTC та Розробка ВебДодатків для Реального Часу Комунікації, 2023, URL: https://it-rating.ua/webrtc-ta-rozrobka-veb-dodatkiv-dlya-realnogo-chasu-komunikatsii (дата звернення 03.10.2023).

6. Сучасні загрози та способи забезпечення безпеки вебзастосунків, 2022, URL: https://periodicals.karazin.ua/cscs/article/view/21038 (дата звернення 19.11.2023).

7. Фантастичні вебвразливості і де вони мешкають, 2022, URL: https://habr.com/ru/companies/simbirsoft/articles/659847/ (дата звернення 10.12.2023).

8. Що таке XSS-вразливість і як тестувальнику не пропустити її, 2020, URL: https://habr.com/ru/articles/511318/ (дата звернення 12.12.2023).

9. Лікнеп за поширеними Client-Side вразливістю, 2023, URL: https://habr.com/ru/companies/bastion/articles/757590/ (дата звернення 20.12.2023).

10. How to Secure Your React.js Application, 2021, URL: https://www.freecodecamp.org/news/best-practices-for-security-of-your-react-js-application/ (дата звернення 22.12.2023).

11. XSS атакує! Короткий огляд XSS вразливостей, 2023, URL: https://habr.com/ru/companies/alfa/articles/717896/ (дата звернення 26.12.2023).