ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти Бакалавр

на тему: «Аналіз вразливостей вебдодатків на основі стандарту OWASP»

Виконав: здобувач вищої освіти
за освітньо-професійною
програмою Інформаційні
управляючі системи спеціальності
126 Інформаційні системи та
технології
ступеня вищої освіти Бакалавр
групи 126ІСТбд41
Крикливець В.О.
Керівник: Копішинська О.П.
Рецензент: Брикун О.М.

Полтава – 2024 року

ВСТУП

У сфері продуктового IT напрямок розробки найрізноманітніших вебдодатків став дійсно масовим і популярним. Це обумовлено інформатизацією всіх сфер діяльності людини, проведенням величезної кількості та видів операцій в мережевих застосунках, переведенням на вебплатформи не лише традиційних комерційних операцій, але й державних послуг, навчання, комунікаційних процесів тощо. Створити функціональний вебдодаток можна за допомогою розвинутих технологій, серед яких сучасні мови програмування, спеціалізовані платформи для розробки вебдодатків та спеціальні фреймворки, які надають вже готові компоненти оформлення загального функціоналу. Ці продукти дозволяють швидко отримати готові робочі проєкти. При цьому значна увага може приділятися питанням SEO-оптимізації вебсайтів, видимості для пошукових систем і доступності. У той же час, розробникам часто не вистачає технічного розуміння особливостей їх функціонування та можливих потенційних ризиків у сфері інформаційної безпеки. Однак, існує суттєва різниця між кодом, який є функціональним, і кодом, який є інформаційно захищеним [1].

Актуальність кваліфікаційної роботи обумовлена тим, що, хоча, проблема безпеки вебдодатків стоїть дуже гостро, спеціалізованих засобів прогнозування потенційних загроз і захисту досить мало, і здебільшого подібні завдання покладають на розробників. Тому аналіз можливостей, які надає спільнота Open Web Application Security Project (OWASP), дозволяє отримати та структурувати механізми діагностики і захисту будь-яких вебдодатків на етапі розроблення і використання. Ресурс, який надає OWASP, дозволяє розробникам знайти і використати ключові принципи безпеки, дізнатися про найпоширеніші загрози на сьогоднішній день та методи їх усунення. Також OWASP створила стандарт, що дозволяє проводити аналіз вебдодатків на наявність типових вразливостей, розглянути різні види можливих вебатак.

Метою кваліфікаційної роботи є ознайомлення з існуючими категоріями вразливостей і методами їх виявлення, а також підвищення інформаційної безпеки вебдодатків.

Завданнями кваліфікаційної роботи є: проведення теоретичних і практичних досліджень вразливостей вебдодатків; формування переліку вимог щодо підвищення інформаційної безпеки вебдодатку; визначення стеку технологій та інструментів розробки, їх відносну продуктивність, ступінь інтеграції та зручність використання; практична реалізація захисту вебдодатку на основі стандарту OWASP.

Об'єктом дослідження є процеси виявлення і розробки програмної реалізації усунення вразливості вебдодатку.

Предметом дослідження є методи виявлення вразливостей вебдодатку за допомогою автоматичного тестування та засоби їх усунення.

Методами дослідження є: інформаційно-пошуковий, аналітико-синтетичний, методи тестування вебдодатків, співставлення та порівняння, методи оцінювання економічної ефективності тощо.

Інформаційна база роботи сформована на основі наукових статей, публікацій популярних видань, платформ розроблення програмного забезпечення, а також даних, отриманих під час виробничих практик.

Практична значущість роботи: проведено аналіз існуючих вразливостей та методів тестування вебдодатків. Результати роботи можуть бути використані в системах, для яких необхідно створити процес автоматизованого тестування вебдодатків, а також в навчальному процесі.

Структура і обсяг кваліфікаційної роботи: пояснювальна записка викладена на 52 сторінках і складається зі змісту, вступу, трьох розділів, а також списку використаних джерел та додатків. Робота містить 20 рисунків, лістинги програмних кодів.

РОЗДІЛ 1
ОСОБЛИВОСТІ ВИКОРИСТАННЯ ТА ПРИНЦИПИ БЕЗПЕКИ ВЕБДОДАТКІВ

1.1 Загальні відомості про функції вебдодатків

При зародженні сучасного інтернету на початку 90-х років XX ст. всесвітня павутина складалась лише з вебсайтів. По суті, це були просто сховища інформації, які містили статичні документи, приклад якого наведено на рис. 1.1. Браузери були створені як необхідний засіб пошуку та відображення інформації. Потік інформації був одностороннім, від сервера до браузера. Більшість сайтів не проводили автентифікацію користувачів: не було потреби. Всі користувачі були рівноправними, тобто мали доступ до однієї ж інформації. Будь-які загрози безпеці, які виникали при розміщенні вебсайту, значною мірою були пов'язані із вразливістю програмного забезпечення вебсервера. Якщо зловмисник скомпроментував вебсервер, він не отримав би доступ до будь-якої конфіденційної інформації, оскільки інформація, що розміщена на сервері, відкрита для загального перегляду. Нападник змінював файли на сервері, щоб змінити вміст вебсайту або використовував пропускну здатність сервера для розповсюдження нелегального програмного забезпечення.

Рисунок 1.1 – Традиційний вебсайт, що містить статичну інформацію

Сучасна всесвітня павутина майже нічим не схожа на її попередню версію. Більшість сайтів в інтернеті насправді є високофункціональними додатками, які покладаються на двосторонній потік інформації між сервером та браузером [2]. Вони підтримують реєстрацію та вхід в систему, фінансові транзакції, пошук і створення вмісту додатку користувачем. Інформація, що представлена для користувачів, генерується динамічно і часто підлаштовується під конкретного користувача (рис. 1.2).

Значна частина інформації є конфіденційною та високочутливою. Отже, безпека є важливим питанням. Ніхто не буде користуватись вебдодатком, якщо він не впевнений, що введена ним інформація не буде розкрита стороннім особам.

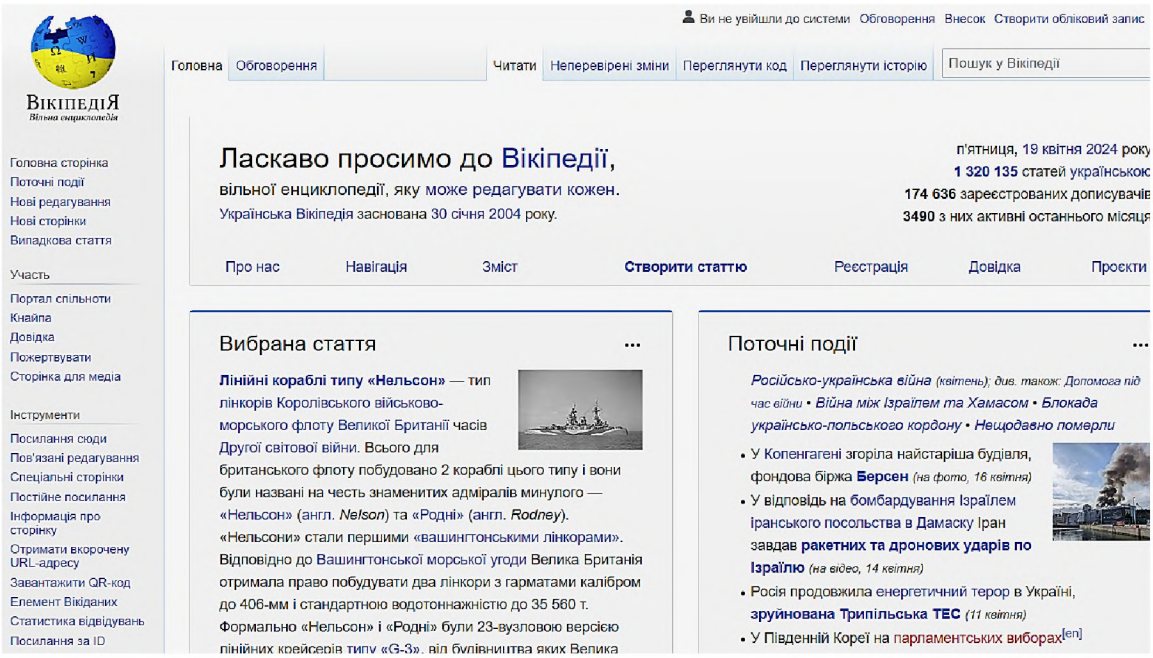


Рисунок 1.2 – Типовий вебдодаток

Нові розроблені вебдодатки приносять із собою нові та значні загрози безпеці. Кожен додаток унікальний і може містити різні вразливості. Більшість додатків розробляються власноруч, проте багато розробників мають лише часткове розуміння проблем безпеки, що можуть виникати в коді, який вони пишуть [3].

Щоб забезпечити основний функціонал, вебдодатки зазвичай вимагають підключення до внутрішніх комп'ютерних систем, які містять високочутливі дані та можуть виконувати багато потужних функцій [3]. Наприклад, 25 років тому для переказу коштів або інших фінансових олперацій потрібно було відвідати банк. Сьогодні ж потрібно просто зайти у вебдодаток та здійснити переказ самостійно. Зловмисник, що атакує такий додаток, може викрасти особисту інформацію, здійснити фінансове шахрайство та здійснити будь-які інші шкідливі дії проти користувача.

Вебдодатки створені для виконання практично будь-якої корисної функції, яку можна реалізувати в мережі інтернет. Ось деякі функції вебдодатків, що мають активне поширення за останні роки [4]:

- покупки;
- соціальні мережі;
- банківські справи;
- пошук в інтернеті;
- аукціони;
- вебжурнали;
- вебпошти;
- інтерактивна інформація.

Додатки, доступ до яких здійснюється за допомогою комп'ютерного браузера, все більше перетинається з мобільними додатками, доступ до яких здійснюється за допомогою смартфона або планшета. Більшість мобільних додатків використовують або браузер, або індивідуальний клієнт, який використовує API (application program interface – інтерфейс додатку) на основі протоколу HTTP (Hyper Text Transfer Protocol – протокол передачі гіпертекстової інформації) для спілкування з сервером. Функції додатку та дані, як правило, поділяються між різними інтерфейсами, які застосовує додаток на різних користувацьких платформах.

Окрім публічного інтернету, вебдодатки широко розповсюджені всередині організацій для підтримки роботи бізнесу. Багато з них надають доступ до високочутливих даних та таких функціональних можливостей [5].

1. HR-програми, що дозволяють користувачам отримувати доступ до інформації про оплату праці, надавати та отримувати зворотній зв'язок щодо ефективності, а також керувати набором кадрів та дисциплінарними процедури.

2. Адміністративні інтерфейси до ключової інфраструктури, такі як Інтернет, поштові сервери, робочі станції користувачів та адміністрування віртуальних машин.

3. Програмне забезпечення для співпраці, що використовується для обміну документами, управління робочим процесом і проектами, а також відстеження проблем. Ці типи функціональності часто пов'язані з критичними питаннями безпеки та управління, а також організації часто повністю покладатися на елементи управління, вбудовані в їх веб-додатки.

4. Бізнес-додатки, такі як програмне забезпечення для планування і управління ресурсами підприємства (ERP – Enterprise Resource Planning), до якого раніше можна було отримати доступ лише за допомогою власного додатка, тепер можна отримати доступ за допомогою браузера і хмарних технологій.

5. Програмні послуги, такі як електронна пошта, спочатку вимагали окремого клієнта на робочій станції, тепер доступні через вебінтерфейси, наприклад Outlook.

6. Традиційні настільні офісні програми, такі як текстові процесори та електронні таблиці, були переміщені до веб-додатків через такі сервіси, як Google Apps і Microsoft Office Live.

У всіх цих прикладах додатки, що сприймаються як внутрішні, все частіше розміщуються зовні, коли організації переходять до сторонніх постачальників послуг, щоб скоротити витрати. У цих так званих хмарних рішеннях функціональність і дані відкриті для широкого кола потенційних зловмисників і

організації все більше покладаються на цілісність засобів захисту, які знаходяться поза межами їх контролю.

Наближається час, коли єдиним клієнтським програмним забезпеченням, яке знадобиться для більшості користувачів комп'ютерів, буде веббраузер. Різноманітний спектр функцій буде реалізовано за допомогою спільного набору протоколів та технологій що буде ускладнювати характерний спектр загальних уразливих місць безпеки.

1.2 Технічні переваги і засоби використання вебдодатків

Неважко зрозуміти, чому вебдодатки так широко використовуються. Кілька технічних факторів, що відображують очевидні комерційні стимули в революції використання інтернету, наведено в наступному переліку [6].

1. HTTP – основний протокол зв'язку, який використовується для доступу до всесвітньої павутини, легкий і не потребує підключення. Даний протокол забезпечує стійкість зв'язку та уникає необхідності сервера тримати відкрите мережеве з'єднання для кожного користувача, так як це було в застарілих клієнт-серверних програмах. HTTP також може бути проксі-сервером і тунельним над іншими протоколами, що забезпечує зв'язок у будь-якій мережній конфігурації.

2. Кожен користувач інтернету вже має встановлений браузер на своєму комп'ютері та мобільному пристрої. Вебдодатки розгортають свій користувацький інтерфейс динамічно, уникаючи необхідності поширювати та керувати окремими клієнтськими додатками, як це було у випадку з сайтами, що передували веб-додаткам. Зміни в інтерфейсі потрібно реалізувати лише один раз, на сервері, в цьому випадку і зміни починають діяти негайно.

3. На сьогоднішній день браузери вирізняються високою функціональністю, що задовольняють користувацькі бажання та вимоги. Веб-

інтерфейси використовують стандартні навігаційні та вхідні елементи управління, які вже знайомі користувачам, уникаючи необхідності вивчати функціонал окремої програми. Скрипти зі сторони клієнта дозволяють програмам перенести частину їхніх процесів на клієнтську сторону і тим самим розширити можливості браузера використовуючи розширення, де необхідно [7].

4. Основні технології та мови розробки вебдодатків є відносно простими. Широкий спектр платформ та інструментів для розробки доступні для того, щоб полегшити розробку потужних додатків початківцям, а також велика кількість відкритого коду та інших ресурсів доступні для включення у власні програми.

Вебдодатки використовують безліч технологій для реалізації своєї функціональності. У даному підрозділі описані ключові технології, що пов'язані з різноманітними атаками на вебдодатки. Особливу увагу приділено HTTP-протоколу, а також технологіям, які зазвичай використовуються на боці серверу або клієнтів.

1. Протокол HTTP. Протокол передачі гіпертексту (HTTP) – це основний протокол доступу до всесвітньої мережі інтернет, який використовується всіма вебдодатками. Спочатку даний протокол був розроблений для отримання статичних текстових ресурсів. З часом протокол розширювався та наразі використовується для забезпечення підтримки складних сучасних розподілених додатків.

HTTP використовує модель на основі повідомлень, в якій клієнт надсилає повідомлення із запитом, а сервер повертає відповідь. Даний протокол є по суті протоколом без підключення: хоча HTTP використовує протокол TCP в якості транспортного механізму, кожен обмін запита і відповіді є автономним і може використовувати інше TCP-з'єднання.

2. HTTP-запити. Усі повідомлення HTTP (запити та відповіді) складаються з одного або декількох заголовків, кожен в окремому рядку, після чого

обов'язковий порожній рядок, а потім необов'язковий текст повідомлення. На рис. 1.3 зображено типовий запит.

```
GET /auth/488/YourDetails.ashx?uid=129 HTTP/1.1
Accept: application/x-ms-application, image/jpeg, application/xaml+xml,
image/gif, image/pjpeg, application/x-ms-xbap, application/x-shockwave-
flash, */*
Referer: https://mdsec.net/auth/488/Home.ashx
Accept-Language: en-GB
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64;
Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR
3.0.30729; .NET4.0C; InfoPath.3; .NET4.0E; FDM; .NET CLR 1.1.4322)
Accept-Encoding: gzip, deflate
Host: mdsec.net
Connection: Keep-Alive
Cookie: SessionId=5B70C71F3FD4968935CDB6682E545476
```

Рисунок 1.3 – Код типового HTTP-запиту

Перший рядок кожного запиту HTTP (див. рис. 1.3)складається з трьох елементів, розділених пропусками:

– Дієслово із зазначенням методу. Найпоширеніший метод це GET, функцією якого є отримання даних з вебсервера.

– Потрібна URL-адреса. URL-адреса, як правило, функціонує як назва ресурсу, а також містить необов'язковий рядок запиту, що містить параметри.

– Версія HTTP, що використовується. Більшість браузерів використовують версію 1.1 за замовчуванням.

3. HTTP-відповіді. На рис. 1.4 зображено типову HTTP-відповідь:

```
HTTP/1.1 200 OK
Date: Tue, 19 Apr 2011 09:23:32 GMT
Server: Microsoft-IIS/6.0
X-Powered-By: ASP.NET
Set-Cookie: tracking=tI8rk7joMx44S2Uu85nSWc
X-AspNet-Version: 2.0.50727
Cache-Control: no-cache
Pragma: no-cache
Expires: Thu, 01 Jan 1970 00:00:00 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1067

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"><html xmlns="http://
www.w3.org/1999/xhtml" ><head><title>Your details</title>
```

Рисунок 1.4 – Код типової HTTP-відповіді

Перша лінія кожної HTTP-відповіді складається з трьох частин, розділених пропуском (відступом):

– Версія HTTP.

– Числовий код статусу, що вказує на результат запиту. Найчастіше це 200, що означає, що запит був успішним і запитувана інформація повертається.

– Текстова фраза, що описує стан відповіді. Може набувати будь-якого значення і не використовується браузерами.

4. HTTP методи. Метод GET призначений для отримання ресурсів. Його можна використовувати для надсилання параметрів на потрібний ресурс у рядку запиту URL. Це дає змогу користувачам запам'ятовувати URL-адресу для динамічного ресурсу, який вони зможуть використати повторно. Також інші користувачі можуть перейти по посиланню в майбутньому (якщо запит був доданий до закладок) [8].

5. URL-адреси відображаються на екрані та реєструються в різних місцях, таких як історія браузера або журнали доступу вебсервера. Вони передаються в заголовку в разі переходу по зовнішнім посиланням, тому рядок запитів не повинен використовуватись для передачі будь-якої конфіденційної інформації.

Метод POST призначений для виконання дій. За допомогою цього методу параметри запиту можуть надсилатись як у рядку запиту, так і в тілі повідомлення. Незважаючи на те, що URL-адресу можна додати до закладок, будь-які параметри, надіслані в тілі повідомлення, будуть виключені із закладок. Ці параметри будуть також виключені з інших місць, в яких підтримуються журнали URL-адрес [8].

Оскільки метод POST призначений для виконання дій, якщо користувач натискає кнопку «Назад» в браузері, щоб повернутись на попередню сторінку, доступ до якої використовується цим методом, браузер не повторить запит. Натомість він просто попереджає користувача про те, що збирається зробити, як показано на рис. 1.5.

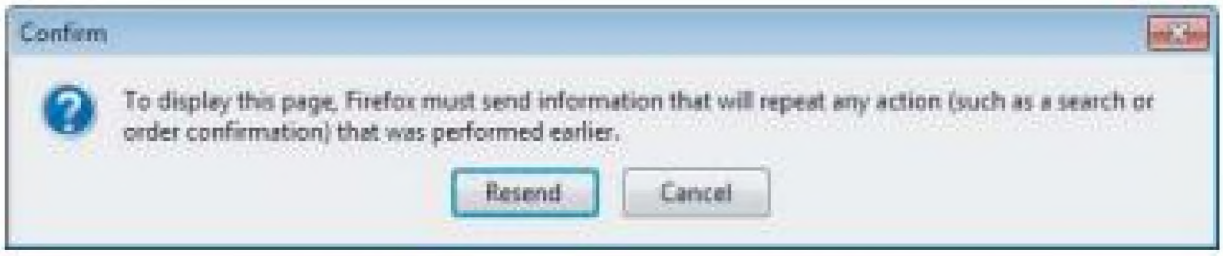


Рисунок 1.5 – Вікно-попередження браузера

6. Протокол HTTPS. Протокол HTTP використовує звичайний TCP як транспортний механізм, який є незашифрованим і тому може бути перехопленим зловмисником. HTTPS (HyperText Transfer Protocol Secure) – це по суті той самий протокол рівня додатків як HTTP, але налаштований через захищений транспортний механізм – SSL [8]. Це захищає конфіденційність та цілісність даних, що передаються через мережу, зменшуючи можливість перехоплення. HTTP запити та відповіді функціонують так само, незалежно від того використовується SSL чи ні.

1.3 Технології забезпечення вебфункціональності та безпеки

Крім основного протоколу зв'язку, який використовується для обміну повідомленнями між клієнтом і сервером, веб-додатки використовують численні технології для забезпечення їх функціональності. Будь-яка широко функціональна програма може використовувати десятки різних технології в межах серверного і клієнтського компонентів. Перш ніж здійснити серйозну атаку на веб-додаток, потрібно зрозуміти, як реалізовується його функціональність, які технології використовуються, де можуть знаходитись слабкі місця.

1. Функціональність на стороні сервера. Рання всесвітня павутина містила повністю статичний вміст. Вебсайти складались із різних ресурсів, таких як

HTML-сторінки та зображення, які просто завантажувались на вебсервер та доставлялись користувачеві, які робили до них запит, в однаковій формі [9]. Сьогодні вебдодатки використовують достатню кількість динамічних даних: користувач робить запит на динамічний ресурс, відповідь сервера створюється миттєво, і користувач отримує унікально налаштований для нього вміст. Динамічний контент генерується за допомогою скриптів або іншого коду, що виконується на сервері. Про суті, користувач подає різні параметри разом із запитом. Саме ці параметри дозволяють серверному додатку генерувати індивідуальний вміст [9]. Вебдодатки використовують широкий спектр технологій на боці сервера, щоб забезпечити їх функціональність:

– скриптові мови такі як PHP, VBScript, Perl;
– платформи для веб-додатків, такі як ASP.NET, JAVA;
– веб-сервери, такі як Apache, IIS, Netscape Enterprise;
– бази даних: MS-SQL, Oracle та MySql;
– інші бек-енд компоненти такі як різноманітні файлові системи, веб-сервери, що базуються на SOAP [10].

2. Кодування URL-адрес. В URL-адресах дозволяється розміщення тільки символів з US-ASCII символьного набору – тобто тільки ті символи, чий код в ASCII знаходиться в діапазоні від 0x20 до ox7e включно. Крім того, кількість символів в цьому діапазоні обмежена, оскільки вони мають особливе значення в межах самої схеми URL-адреси або всередині протоколу HTML. Схема кодування URL використовується для кодування будь-яких символів в розширеному наборі ASCII, щоб їх можна було безпечно транспортувати по HTTP. Форма будь-якого символу, кодована в межах URL-адреси, є префіксом %, за яким йде двозначний код символу в ASCII, виражений в шістнадцятірковій системі числення.

3. Кодування Unicode. Unicode – стандарт кодування символів, розроблений для підтримки всіх світових систем письма. В ньому

використовуються різні схеми кодування, деякі з них можна використовувати для представлення незвичайних символів у веб-додатках [11].

4. HTML кодування. HTML-кодування використовується для представлення проблемних символів, щоб їх можна було безпечно включити в документ HTML. Різні символи мають особливе значення і використовуються для визначення структури документа, а не його вмісту. Щоб безпечно використовувати ці символи як частину вмісту документа, необхідно кодувати їх використовуючи HTML-кодування.

5. Base64 кодування. Кодування base64 дозволяє безпечно представляти будь-які двійкові дані, використовуючи лише друковані символи ASCII. Зазвичай дане кодування використовується для кодування вкладень в електронній пошті для безпечної передачі через SMTP, а також для кодування облікових даних користувачів в базовій автентифікації HTTP [12]. Кодування base64 обробляє вхідні дані в блоки, які складаються з трьох байт. Кожен з цих блоків розділений на чотири фрагменти по шість біт кожен. Шість бітів дозволяють отримати 64 різні можливі перестановки, тому кожен фрагмент може бути представлений за допомогою набору з 64 символів.

6. Кодування HEX. У багатьох додатках використовується пряме шістнадцятирічне кодування при передачі бінарних даних, використовуючи символи ASCII для представлення шістнадцяткового блоку.

Щодо питань безпеки, то користувачам часто рекомендують перевірити сертифікат сайту, звернути увагу на передові криптографічні протоколи, що використовуються [13]. Розробники більшості програм запевняють, що їхні додатки захищені, оскільки вони, наприклад, використовують SSL протокол (Secure Sockets Layer – рівень захищеності сокетів).

SSL –технологія, яка захищає конфіденційність та цілісність даних, які перебувають у режимі переходу між браузером користувача та веб-сервером. Це допомагає захищатися проти підслухувачів, і може гарантувати користувачеві

ідентифікацію вебсервера, з яким він має справу. Але це не зупиняє атак, які безпосередньо націлені на серверні або клієнтські компоненти програм, як це робиться в більшості успішних атак.

Організації також цитують свою відповідність до стандарту Галузевої Персональної Картки (PCI), щоб переконати користувачів у їх захищеності: «Ми ставимось до безпеки дуже серйозно. Наш вебсайт щодня сканується, щоб переконатися, що ми залишаємося сумісними з PCI і захищені від хакерів». Однак, більшість вебдодатків, незважаючи на використання протоколу SSL та регулярного сканування PCI, небезпечні. Протягом останніх років було перевірено сотні додатків. На рис. 1.6 показано відсоток протестованих додатків протягом 2016-2021 рр., на які впливають деякі категорії вразливостей [13].

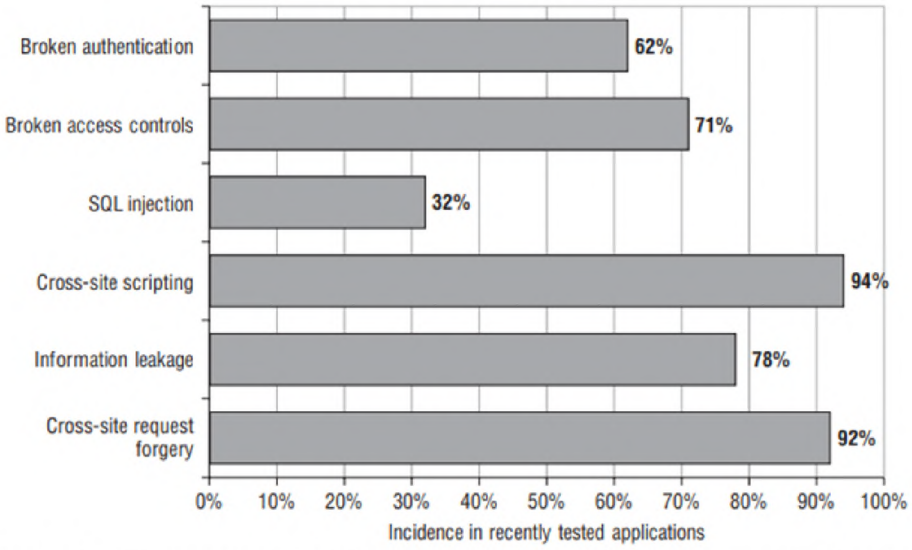


Рисунок 1.6 – Категорії найбільш поширених вразливостей у вебдодатках

Види і зміст цих вразливостей наведено нижче.

1. Порушена автентифікація (Broken Authentication) – включає різні дефекти в механізмі входу до програми, за якими зловмисник може відгадати слабкі паролі, запустити процес підбору пароля або обійти сам процес входу.

2. Порушений контроль входу (Broken Access Controls) стосується випадків, коли додатку не вдається належним чином захистити доступ до своїх

даних та функціональних властивостей, що дозволяє зловмиснику переглядати дані інших користувачів або виконувати певні привілейовані дії.

3. SQL-ін'єкція (SQL injection) – ця вразливість дозволяє зловмисникові виконати вхід для втручання в роботу баз даних додатку. В такому випадку зловмисник має змогу отримати довільні дані з програми, втручатись в логіку або виконувати певні команди в базі даних самого сервера.

4. Міжсайтовий скрипт (Cross-Site Scripting) – ця вразливість дозволяє зловмисникові націлитись на інших користувачів програми, отримавши доступ до їхніх даних і виконуючи від їхнього імені несанкціоновані атаки проти них.

5. Витік інформації (Information Leakage) – це стосується випадків, коли додаток оприлюднює конфіденційну інформацію, яка корисна для нападника.

6. Підробка заявок (Cross-Site Request Forgery): означає, що заявка користувачів може спонукати виконувати ненавмисні дії над програмою в межах їхнього контексту та рівня привілеїв. Ця вразливість дозволяє зловмиснику виконувати протиправні дії в вебдодатку, що відвідує користувач.

Більшість атак на вебдодатки пов'язані з надсиланням даних на сервер, щоб викликати якусь подію, яка не була передбачена розробниками додатку [14]. Кілька прикладів надсилання введених даних для досягнення мети [15]:

– зміна ціни товару, що передається у прихованому полі форми HTML, щоб обманним шляхом придбати товар за більш дешеву суму;

– зміна маркера сеансу, переданого у файлі cookie HTTP, щоб викрасти сеанс іншого автентифікованого користувача;

– видалення певних параметрів, які зазвичай подаються для використання недоліків логіки в процесах додатків;

– зміна деяких вхідних даних, що будуть оброблені за допомогою бази даних, щоб ввести запит та отримати доступ до конфіденційних даних [16].

Таким чином, питання забезпечення безпечності за рахунок зниження вразливостей вебдодатків є надзвичайно поширеними.

Сьогодні навіть такі прості вебайти, як особисті блоги, мають багато залежностей. Ми всі можемо погодитися, що неможливість оновити кожну частину програмного забезпечення на серверній і зовнішній частині вебсайту, без сумніву, швидше за все створить серйозні ризики для безпеки. Наприклад, у 2019 році 56% усіх додатків CMS були застарілими на момент зараження.

Питання в тому, чому ми не оновлюємо програмне забезпечення вчасно? Чому це все ще є такою великою проблемою сьогодні?

Є деякі варіанти причин, наприклад:

– веброзробники не встигають за темпами оновлень; зрештою, належне оновлення вимагає часу;

– застарілий код не працюватиме на новіших версіях його залежностей;

– вебмайстри бояться, що на їх сайті щось зламається;

– вебмайстри не мають досвіду, щоб правильно застосувати оновлення.

Якою б не була причина запуску застарілого програмного забезпечення у вашій вебпрограмі, ви не можете залишити її без захисту. І Sucuri, і OWASP рекомендують віртуальне виправлення у випадках, коли неможливо інакше.

Віртуальні виправлення дають змогу захистити вебсайти, які є застарілими (або мають відомі вразливості), від атак, запобігаючи використанню цих уразливостей на льоту. Зазвичай це робиться за допомогою брандмауера та системи виявлення вторгнень.

РОЗДІЛ 2
ХАРАКТЕРИСТИКА ТА АНАЛІЗ ВРАЗЛИВОСТЕЙ НА ОСНОВІ САТНДАРТУ OWASP TOP 10

2.1 Сутність проєкту захисту онлайн-спільноти OWASP

Проєкт захисту відкритих вебдодатків OWASP (the Open Web Application Security Project) – це відкрита спільнота, присвячена наданню організаціям можливості розробляти, купувати та підтримувати такі вебдодатки, програми та API, яким можна довіряти [17].

У цьому проекті можливо знайти:

- інструменти та стандарти безпеки програм;
- книги про тестування безпеки додатків, розробки захищеного коду та безпечний огляд коду;
- презентації та відео;
- шпаргалки на багато поширених тем;
- стандартні засоби безпеки та бібліотеки;
- прогресивні дослідження;
- конференції;
- підписку на розсилку.

Всі інструменти, документи, відео та презентації OWASP вільні та відкриті для всіх, хто зацікавлений у вдосконаленні безпеки додатків [18].

OWASP виступає за підхід до безпеки додатків, як до технічної проблеми, оскільки найефективніші підходи до захисту додатків потребують вдосконалення в цих областях. Свобода від комерційного тиску дозволяє надавати організації об’єктивну, практичну та рентабельну інформацію про безпеку додатків [19]. OWASP виробляє багато матеріалів спільним, прозорим та відкритим способом.

OWASP Top 10 – це документ, який представляє собою комплекс матеріалів щодо найважливіших і часто розповсюджених ризиків безпеки веб-додатків. Хоча початковою метою проекту OWASP Top 10 було просто підвищити рівень обізнаності серед розробників і менеджерів, фактично він став стандартом безпеки веб-додатків. OWASP Top 10 ґрунтується на інформації, яка надана фірмами, що спеціалізуються на безпеці додатків, а також на галузевому опитуванні, яке пройшло більше 500 людей. Ці дані охоплюють уразливості, зібрані з сотень організацій та понад 100 000 реальних додатків та API. У Top 10 вразливості обираються та визначаються відповідно до проаналізованих даних про поширеність у поєднанні з оцінкою експлуатаційності. Основною метою OWASP Top 10 є навчити розробників, дизайнерів, менеджерів та організації про наслідки найпоширеніших та найважливіших недоліків безпеки вебдодатку [20]. На рис. 2.1 зображено статистику OWASP Top 10 з початку 2017 р., а також на 21021 р.

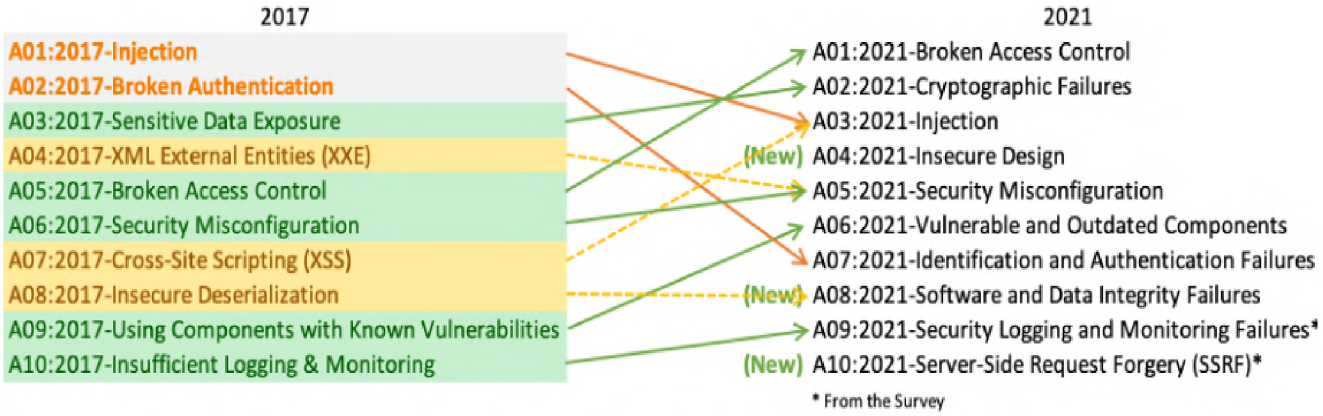


Рисунок 2.1 – Рейтинг частоти вразливостей OWASP Top 10

Як видно з рис. 2.1 є три нові категорії, чотири категорії зі змінами в назві та охопленні, а також деяка консолідація в Топ-10 за 2021 рік. Були змінені назви, коли це було необхідно, щоб зосередитися на першопричині, а не на симптомі.

Основна інформація про види вразливостей з OWASP Top 10 та їхня характеристика, причини появи наведено в додатку А (табл. А.1). Розглянемо детальніше можливості і недоліки окремих з них.

1. Broken Access Control. Експлуатація контролю доступу є основним навиком зловмисників. Інструменти SAST і DAST можуть виявити відсутність контролю доступу, але не можуть перевірити, чи функціональний він, коли він присутній. Ручне тестування – найкращий спосіб виявити відсутні або неефективні засоби контролю доступу, включаючи метод HTTP (GET vs PUT тощо), контролер, прямі посилання на об'єкти тощо [22].

За статистикою (див. рис. 2.1) піднімається з п'ятої позиції в категорію з найсерйознішим ризиком безпеки вебдодатків; надані дані вказують на те, що в середньому 3,81% протестованих програм мали один або кілька загальних перерахувань слабких місць (CWE) із понад 318 тисячами випадків CWE у цій категорії ризику. 34 CWE, віднесені до Broken Access Control, мали більше випадків у програмах, ніж будь-яка інша категорія. Контроль доступу застосовує політику таким чином, що користувачі не можуть діяти за межами передбачених дозволів. Збої, як правило, призводять до несанкціонованого розкриття інформації, зміни або знищення всіх даних або виконання дій функції поза межами повноважень користувача.

2. Криптографічні збої переміщуються на одну позицію вгору до №2, раніше відомі як A3:2017-Sensitive Data Exposure, що було загальним симптомом, а не основною причиною. Оновлена назва зосереджена на збоях, пов'язаних із криптографією, як це неявно було раніше. Ця категорія часто призводить до розкриття конфіденційних даних або компрометації системи. Замість того, щоб безпосередньо атакувати зашифровані дані, зловмисники викрадають ключі, використовуючи ручні атаки, або просто текстові дані з сервера під час передачі, або з користувацького інтерфейсу, наприклад браузера [21]. Для цього виду атак необхідний ручний, а не автоматизований підхід. Раніше отримані паролі баз

даних можуть бути пробрутфорсені, використовуючи користувацький інтерфейс. За останні декілька років ця атака стала найчастіше вражаючою. Найпоширеніший недолік – це просто відсутність шифрування конфіденційних даних. При використанні зашифрованих даних, слабке генерування ключів та управління, використання слабого алгоритму та використання протоколів та шифрів є загальним, особливо для більшості методів зберігання паролів. Для даних, що надсилаються, слабкі сторони сервера виявляються в основному легко, але важко переносять дані під час зберігання.

Будь-яка помилка часто може скомпрометувати всі дані, які повинні бути захищені. Зазвичай така інформація включає в себе конфіденційні особисті дані, такі як записи про стан здоров'я, облікові дані, персональні дані та дані кредитних карток, які часто потребують захисту, визначеного законами чи нормативними актами, такими як EU GDPR або місцевими законами про конфіденційність.

3. SQL Injection. Практично будь-яке джерело даних може бути ін'єкційним вектором для змінного середовища, параметрів, зовнішніх та внутрішніх веб-служб та для всіх типів користувачів. Ін'єкції є дуже поширеними, особливо у застарілому коді. Ін'єкційні вразливості часто зустрічаються в запитах SQL, LDAP, XPath або NoSQL, командах ОС, XML-аналізаторах, заголовках SMTP, запитах ORM.

Ін'єкції можуть призвести до втрати даних, розкриття конфіденційної інформації стороннім особам, втрати підзвітності або навіть до заборони доступу.

4. Insecure design. Незахищений дизайн – це нова категорія 2021 року, яка зосереджена на ризиках, пов'язаних із недоліками дизайну. Якщо ми справді хочемо «рухатися ліворуч» як галузь, нам потрібно більше моделювання загроз, безпечних шаблонів і принципів проектування та еталонних архітектур. Незахищений дизайн не можна виправити ідеальною реалізацією, оскільки за

визначенням необхідні засоби контролю безпеки ніколи не створювалися для захисту від конкретних атак.

5. Security Misconfiguration (XML External Entities). Неправильна конфігурація безпеки переміщається з №6 у попередньому виданні; 90% додатків було перевірено на певну форму неправильної конфігурації із середнім рівнем захворюваності 4,5%, і понад 208 тисяч випадків CWE віднесено до цієї категорії ризику. Зі збільшенням кількості переходів до програмного забезпечення з широкими можливостями налаштування не дивно, що ця категорія просувається вгору. Попередня категорія для A4:2017-XML External Entities (XXE) тепер є частиною цієї категорії ризику.

Зловмисники можуть використовувати вразливості XML, якщо вони можуть завантажувати XML або включати ворожий вміст у XML-документ, використовуючи вразливий код, залежності чи інтеграцію. За замовчуванням чимало старих процесорів XML дозволяють специфікувати зовнішню сутність, URI, що перезаписується та оцінюється під час обробки XML. Інструменти SAST можуть виявити цю проблему, перевіривши залежності та конфігурацію. Інструменти DAST вимагають додаткових кроків для виявлення та використання цієї проблеми. Ці недоліки можуть бути використані для вилучення даних, виконання віддаленого запиту з сервера, сканування внутрішніх систем, виконання атаки відмови в обслуговуванні, а також виконання інших атак.

6. Vulnerable and Outdated Components (Broken Authentication).Хоча і легко знайти вже написані експлоїти для багатьох відомих вразливих місць, інші вразливості потребують зосереджених зусиль для розробки спеціального експлоїта [23]. Ця категорія піднялася з №9 у 2017 році, і це відома проблема, яку ми намагаємося перевірити та оцінити ризик. Це єдина категорія, яка не має загальних уразливостей і ризиків (CVE), зіставлених із включеними CWE, тому в їхніх оцінках враховуються експлойт і ваги впливу за замовчуванням 5,0. Зловмисники мають доступ до сотень мільйонів дійсних комбінацій імен

користувачів та паролів для заповнення облікових даних, списків адміністративних облікових записів за замовчуванням, автоматизованих брутфорсів та засобів атаки словником.

Поширеність зламаної автентифікації є широко розповсюдженою завдяки розробці та впровадженню більшості засобів контролю ідентифікації та контролю доступу. Керування сесіями є основою автентифікації та контролю доступу, і є присутнім у всіх стаціонарних програмах. Зловмисники можуть виявляти порушену автентифікацію за допомогою ручних засобів та використовувати її за допомогою автоматизованих інструментів зі списками паролів та атак словниками.

Зловмисникам достатньо отримати доступ лише до кількох облікових записів або навіть до одного облікового запису адміністратора, щоб скомпрометувати всю систему. Залежно від сфери додатку, це може дозволити відмивати гроші, проводити шахрайство з соціальною безпекою та крадіжку особистих даних, або навіть розкривати юридично захищену конфіденційну інформацію. Підтвердження особистості користувача, автентифікації та управління сеансом є критично важливим для захисту від атак, зв'язаних з зламаною автентифікацією.

7. Security Misconfiguration. Зловмисники часто намагаються використовувати невстановлені недоліки або отримати доступ до дефолтних облікових записів, невикористаних сторінок, незахищених файлів і каталогів тощо, щоб отримати несанкціонований доступ до системи [22].

Неправильне налаштування безпеки може статися на будь-якому рівні стеку додатків, включаючи мережеві послуги, платформу, веб-сервер, сервер додатків, базу даних, фреймворки, спеціальний код та попередньо встановлені віртуальні машини, контейнери або сховище. Автоматизовані сканери корисні для виявлення неправильних конфігурацій, використання облікових записів або конфігурацій за замовчуванням, непотрібних служб, застарілих параметрів тощо.

8. Cross-Site Scripting. Автоматизовані інструменти можуть виявляти та використовувати всі три форми XSS, і є вільно доступні серед фреймворків експлуатації. Автоматизовані інструменти можуть автоматично знайти деякі проблеми з XSS, особливо в зрілих технологіях, таких як PHP, J2EE / JSP та ASP.NET. Вплив XSS є помірним для відображення та DOM XSS, а також серйозним для зберігання XSS, з віддаленим виконанням коду у браузері жертви, наприклад, крадіжка облікових даних, сеанси чи доставка зловмисного програмного забезпечення жертві.

9. Insufficient Logging & Monitoring. Як відомо, використання зловмисниками недостатньої реалізації процесу моніторингу та ведення журналів подій є основою майже всіх основних інцидентів. Однією із стратегій по визначенню того, чи достатній рівень моніторингу реалізовано є перевірка журналів реєстрації подій після проведення тестування на проникнення. Дії розробників та тестувальників повинні реєструватися таким чином, щоб можна було зробити висновок, які збитки можуть нанести дії потенційних зловмисників. Згідно досліджень, у 2017 році час на виявлення вторгнень складав 191 день.

10. Server-Side Request Forgery додано з опитування 10 найкращих спільнот (№1). Дані свідчать про відносно низький рівень захворюваності з охопленням тестування вище середнього, а також вищими за середні оцінками потенціалу використання та впливу. Ця категорія представляє сценарій, коли учасники спільноти безпеки повідомляють нам, що це важливо, навіть якщо на даний момент це не показано в даних.

Таким чином, розглянувши особливості різних видів вразливостей додатків, необхідно визначити, яке програмне забезпечення є ефективним при виявленні та усуненні атак.

2.2 Аналіз методів визначення відомих вразливостей та їх усунення

Описані в п. 2.1 види вразливостей проявляються у вебдодатках за специфічними ознаками. Для ефективної протидії необхідно застосовувати відповідні критерії виявлення кожного з видів та вживати спеціальних заходів щодо їхнього усунення, використовуючи відповідні методики. Розглянемо більш детально на прикладах відносно видів уражень способи їх ліквідації.

1. Ін'єкції (SQL Injection). Ознаки, які дозволяють визначити, що додаток вразливий, наведені далі:

– дані, що надаються користувачем, не перевіряються, не фільтруються та не санітуються додатком.

– динамічні запити використовуються безпосередньо в інтерпретаторі.

– дані використовуються в параметрах пошуку об'єктного реляційного відображення (ORM) для отримання додаткових, чутливих записів.

– дані безпосередньо використовуються або об'єднуються, таким чином, що SQL або команда містить як структурні, так і ворожі дані в динамічних запитах, командах або збережених процедурах.

Концепція ін'єкцій однакова в усіх інтерпретаторах. Огляд вихідного коду – найкращий спосіб виявити, чи є додатки вразливими до ін'єкцій, після чого слід автоматизувати тестування всіх параметрів, заголовків, URL-адрес, файлів cookie, входів даних JSON, SOAP та XML. Організації можуть включати інструменти статичного джерела (SAST) та інструменти динамічного тестування додатків (DAST) у комунікаційні лінії CI / CD для виявлення нещодавно введених ін'єкцій.

Для запобігання ін'єкції потрібно зберігати дані окремо від команд та запитів. Кращим варіантом є використання безпечного API, який дозволяє уникнути використання інтерпретатора повністю або надає параметризований інтерфейс, або використовує інструменти реляційного відображення об'єктів (ORM) [21]. Для будь-яких динамічних запитів слід уникати спеціальних символів, використовуючи специфічний синтаксис.

2. Broken Authentication. Недоліки автентифікації можуть бути виявлені, якщо додаток:

– Дозволяє автоматизовані атаки, такі як заповнення облікових даних, де зловмисник має список дійсних імен користувачів та паролів.

– Дозволяє проводити брутфорс або інші автоматизовані атаки.

– Дозволяє встановлювати за замовчуванням слабкі або добре відомі паролі.

– Використовує слабкі або неефективні процеси відновлення облікових даних та забутих паролів, наприклад "відповіді, засновані на знаннях", які не можна зробити безпечними.

– Використовує звичайний текст, зашифровані або слабо хешовані паролі.

– Має відсутню або неефективну багатофакторну автентифікацію.

– Розкриває ідентифікатори сесії в URL-адресі (наприклад, перезапис URL-адреси).

– Неправильно скасовує ідентифікатори сесії. Сеанси користувача або токени автентифікації (зокрема, маркерів для одиночного входу (SSO)) недійсні під час виходу або періоду бездіяльності.

– Не повертає ідентифікатори сесії після успішного входу.

Щоб запобігти атакам з використанням зламаної автентифікації, потрібно виконувати такі завдання:

– Де можливо, потрібно застосовувати багатофакторну автентифікацію, щоб запобігти автоматизованим заповненням облікових даних, брутфорсу та атакам з використанням повторного введення облікових даних.

– Потрібно здійснювати перевірку паролів, таку як тестування нових паролів та змінення паролів, які є в списку 10000 найслабших паролів.

– Адміністраторам ресурсу потрібно змінювати політику щодо довжини, складності пароля згідно з вказівками NIST 800-63.

– Не можна надсилати будь-які облікові дані, зокрема дані адміністратора.

– Потрібно бути впевненим, що реєстрація, а також відновлення облікових даних змінюються і зміцнюються проти атак щодо зламаної автентифікації.

– Обов'язково використовувати захищений вбудований менеджер сеансів на стороні сервера, який створює новий ідентифікатор випадкових сеансів з високою ентропією входу.

Ідентифікатори сеансу не повинні містити URL-адреси, повинні надійно зберігатися і визначатись недійсними після виходу з режиму очікування, або витоком часу.

3. Sensitive Data Exposure. Для того, щоб бути впевненим, що особисті дані в безпеці, потрібно точно знати відповіді на такі запитання [24]:

– Чи передаються введені дані просто тестом чи зашифровані? Це стосується протоколів наприклад HTTP, SMTP та FTP. Зовнішній Інтернет-трафік є особливо небезпечним. Потрібно перевірити весь внутрішній трафік, наприклад, баланс між завантаженням, веб-серверами або бек-енд-системами.

– Чи зберігаються дані в чистому чи зашифрованому вигляді, включаючи резервні копії.

– Чи використовуються старі або слабкі криптографічні алгоритми за замовчуванням.

– Чи використовуюся крипто-ключі за замовчуванням, чи генеруються крипто-ключі кожен раз чи вони використовуються повторно.

Щоб запобігти атакам, бажано виконувати наступне:

– Класифікувати дані, що обробляються, зберігаються або передаються додатком. Визначити, які дані є чутливими відповідно до законів про конфіденційність, нормативних вимог чи потреб бізнесу.

– Застосовувати елементи керування відповідно до класифікації.

– Не зберігати конфіденційні дані без потреби. Відмовитись, на скільки можливо або використовувати токенізацію або навіть усікання, сумісні з PCI DSS. Дані, які не зберігаються, не можуть бути викрадені.

– Не забувати шифрувати чутливі дані на час зберігання.

– Забезпечити наявність сучасних та чітких стандартних алгоритмів, протоколів та ключів; використовувати належне управління ключами.

– Зашифрувати всі дані під час пересилки за допомогою захищених протоколів, таких як TLS (Transport Layer Security – захист на транспортному рівні) з ідеальними шифрами прямої секретності (PFS), визначення пріоритетності шифру на сервері та захищених параметрів. Наприклад, застосовувати шифрування за допомогою таких директив, як HTTP Strict Transport Security (HSTS – механізм, що примусово активує захищене з'єднання).

– Вимкнути кешування відповідей, що містять конфіденційні дані.

– Зберігати паролі, використовуючи сильні адаптивні та сильні хеш-функції з фактором роботи (коефіцієнтом затримки), такими як Argon2, scrypt, bcrypt або PBKDF2.

– Самостійно перевірити ефективність конфігурації та налаштування.

4. XML External Entities. Програми та, зокрема, веб-сервіси на основі XML або інтеграції, можуть бути вразливими для нападу, якщо [25]:

– Додаток приймає XML безпосередньо або завантажує XML, особливо з ненадійних джерел, або вставляє недовірені дані в документи XML, які потім аналізують процесор.

– Будь-який з процесорів XML в додатку чи веб-службах на базі SOAP увімкнено визначення типу документа (DTD – document type definition).

– Якщо у додатку використовується SAML (Security Assertion Markup Language – мова розмітки декларації безпеки) для обробки ідентифікації в рамках об'єднаної безпеки або єдиного входу (SSO). SAML використовує XML для підтвердження ідентичності та може бути вразливим.

– Якщо програма використовує SOAP до версії 1.2, вона, ймовірно, піддається атакам XML, якщо об'єкти XML передаються в структуру SOAP.

– Бути вразливим до атак XML, означає, що додаток уразливий для відмови в обслуговуванні, включаючи атаку Billion Laughs.

Навчання розробників має важливе значення для виявлення та пом'якшення XML атак. Крім того, запобігання XML вимагає:

– По можливості використовувати менш складні формати даних, такі як JSON, і уникати серіалізації конфіденційних даних.

– Оновлювати всі процесори та бібліотеки XML, до яких звертається додаток.

– Вимкнути зовнішню сутність XML та обробку DTD у всіх XML-аналізаторах програми.

– Реалізувати позитивне ("біле") серверне підтвердження введення чи фільтрування для запобігання ворожих даних у XML-документах, заголовках чи вузлах.

– Переконатись, що функціональність завантаження файлів XML або XSL (Extensible Stylesheet Language – мова для опису перетворень XML-документів) підтверджує вхідний XML за допомогою перевірки XSD (XML Schema) або подібного.

– Інструменти SAST можуть допомогти виявити XML у вихідному коді, хоча ручний огляд коду є найкращою альтернативою для великих, складних програм із багатьма інтеграціями.

Існує два способи тестування захищеності вебдодатків: ручний і автоматизований. Все частіше пентестери користуються автоматизованими способами тестування, оскільки це має суттєві переваги:

1. Швидкість: в програмних засобах для тестування захищеності вже є готові і прописані скрипти, за якими перевіряються додатки. Вручну це б зайняло набагато більше часу.

2. Точність: менша ймовірність виникнення помилки під час тестування, так як в додатку прописаний певний скрипт, який виконується крок за кроком, і ніякі пункти тестування не пропускаються.

3. Дешевизна: ручне тестування в декілька раз дорожче, ніж автоматизоване, тому компанії все частіше і частіше використовують саме автоматизоване тестування.

Є велика кількість таких програмних засобів, які відрізняються своїми функціями, сценаріями і також ціною використання. Для виконання практичної частини роботи використано такі безкоштовні програмні засоби:

– OWASP Zed Attack Proxy (ZAP);
– Nmap;
– Burp Suite;
– Sqlmap.

Функціональні особливості зазначених вище програм мають характристики, які наведено далі.

1. OWASP Zed Attack Proxy (ZAP) – один із найпопулярніших у світі безкоштовних інструментів безпеки, і його активно підтримують сотні міжнародних волонтерів. Даний інструмент може допомогти автоматично знайти вразливості безпеки в веб-додатках під час розробки й тестування (рис. 2.2) [25].

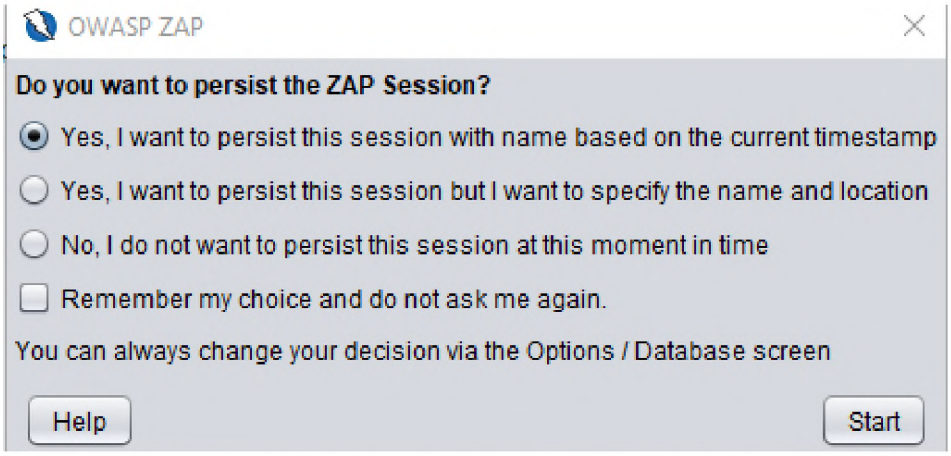


Рисунок 2.2 – Стартове вікно OWASP Zed Attack Proxy

ZAP створює проксі-сервер і змушує трафік вашого вебсайту проходити через цей сервер. Він складається з автоматичних сканерів, які допоможуть вам перехопити вразливості вашого веб-сайту.

2. Nmap – дуже популярний сканер мережі з відкритим вихідним кодом, який може використовуватись як у Windows, так і в Linux. Програма Nmap або Network Mapper була розроблена Гордоном Луоном і в даний момент використовується спеціалістами по безпеці та системними адміністратори по всьому світу (рис. 2.3). Ця програма допомагає системним адміністраторам дуже швидко зрозуміти які комп'ютери підключені до мережі, дізнатися їхні імена, а також подивитися яке програмне забезпечення на них встановлено, яка операційна система і які типи фільтрів застосовуються.

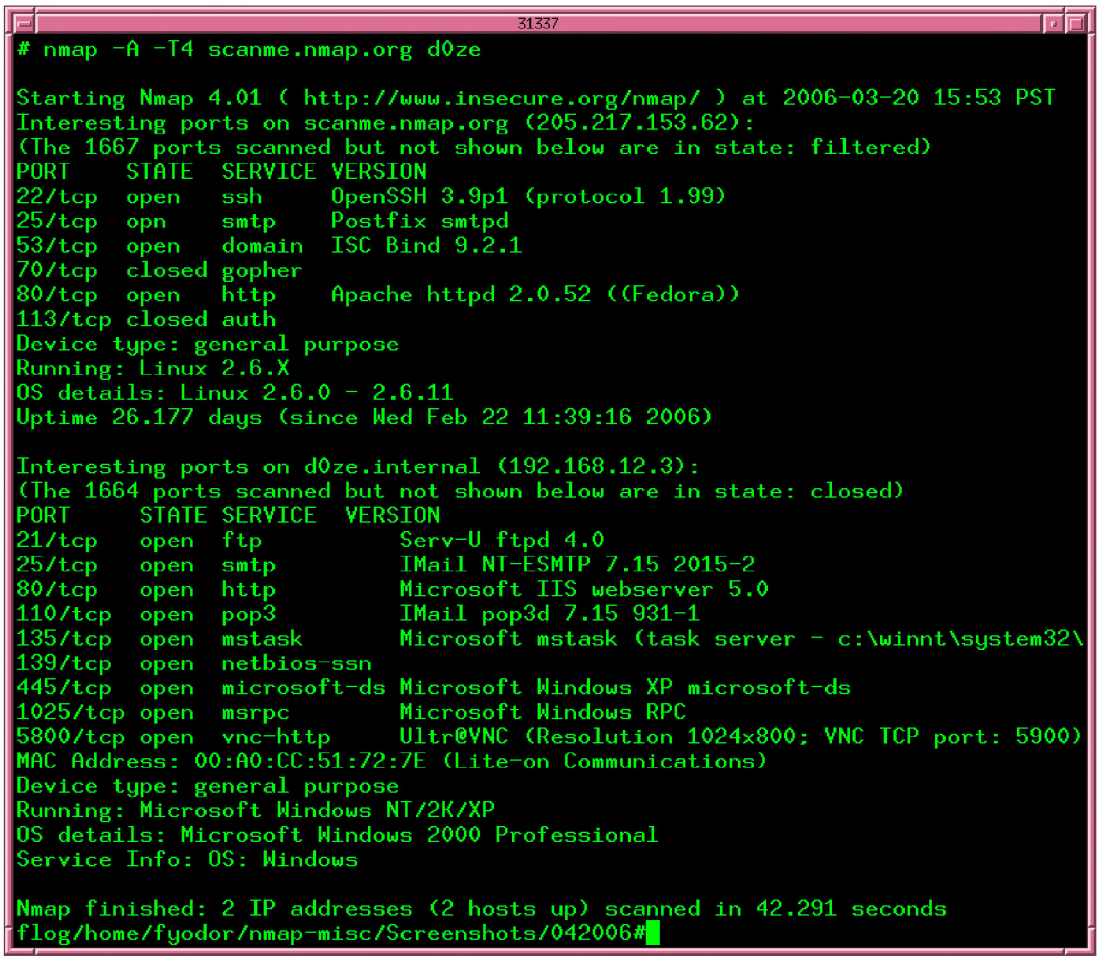


Рисунок 2.3 – Консольна версія Nmap

Функціональність програми може бути розширена за рахунок власного скриптової мови, який дозволяє адміністраторам автоматизувати багато дій. Наприклад, за допомогою скриптів можна автоматично виявляти нові вразливості безпеки в вашої мережі. Однак, при використанні потрібно діяти обережно, щоб не використовувати Nmap проти закону.

3. Burp or Burp Suite – це графічний інструмент для перевірки безпеки веб-додатків (рис. 2.4). Інструмент, написаний на Java і розроблений компанією PortSwigger Web Security. Інструмент має три видання: Community Edition, яке можна безкоштовно завантажити, Professional Edition та Enterprise Edition, які можна придбати після пробного періоду.

Community edition має значно знижену функціональність. Вона може забезпечити комплексне рішення для перевірок безпеки веб-додатків. Окрім основних функціональних можливостей, таких як проксі-сервер, сканер та зловмисник, інструмент містить також більш вдосконалені параметри, такі як павук, ретранслятор, декодер, компаратор, розширювач та секвенсор.

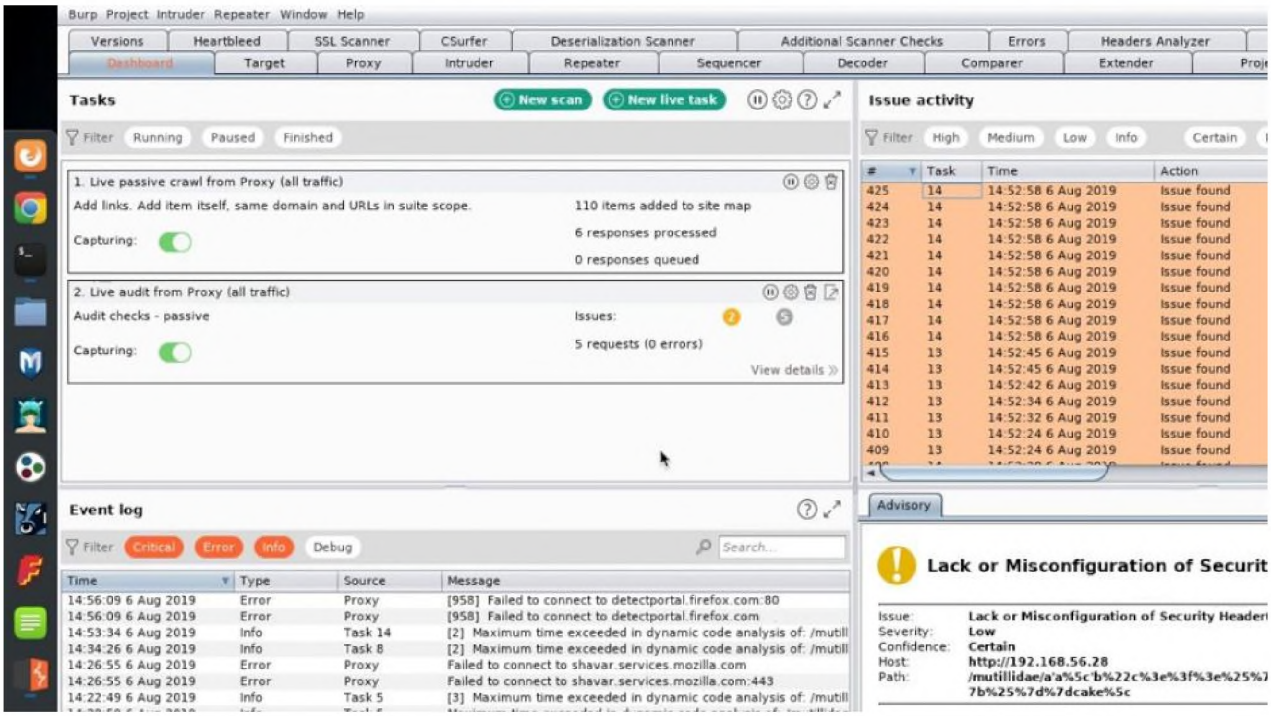


Рисунок 2.4 – Фрагмент інтерфейсу програми Burp Suite

Доступні інструменти [26]:

– HTTP-проксі – працює як веб-проксі-сервер і працює як посередник між браузером та веб-серверами призначення. Це дозволяє перехоплювати, перевіряти та змінювати необроблений трафік, що проходить в обох напрямках.

– Сканер – сканер безпеки веб-додатків, який використовується для автоматизованого сканування вразливості веб-додатків.

– Інструмент зловмисника – інструмент може виконувати автоматизовані атаки на веб-додатки. Інструмент пропонує налаштований алгоритм, який може генерувати шкідливі запити HTTP. Інструмент зловмисника може перевіряти та виявляти SQL ін'єкції, міжсайтові сценарії, маніпуляції параметрами та вразливості, сприйнятливі до грубої атаки.

– Spider – інструмент для автоматичного сканування веб-додатків. Він може використовуватися разом із ручними методами картографування для прискорення процесу відображення вмісту та функціональності програми.

– Повторювач – простий інструмент, який можна використовувати для тестування програми вручну. З його допомогою можна змінювати запити на сервер, надсилати їх і спостерігати за результатами.

– Декодер – засіб для перетворення кодованих даних у канонічну форму або для перетворення необроблених даних у різні закодовані та хешовані форми. Він здатний інтелектуально розпізнавати кілька форматів кодування за допомогою евристичних прийомів.

– Порівняльник – інструмент для порівняння (візуальної "відмінності") між будь-якими двома елементами даних.

– Extender – дозволяє тестеру безпеки завантажувати розширення Burp, розширювати функціональність Burp, використовуючи власний або сторонній код тестерів безпеки (BAppStore).

– Sequencer – інструмент для аналізу якості випадковості у вибірці елементів даних. Його можна використовувати для тестування маркерів сеансу

програми або інших важливих елементів даних, які мають бути непередбачуваними, наприклад, маркери анти-CSRF, символи скидання пароля.

4. Sqlmap – це інструмент тестування з відкритим кодом на проникнення, який автоматизує процес виявлення та використання недоліків введення SQL та захоплення серверів баз даних. Він оснащений потужним механізмом виявлення, багатьма нішевими функціями для остаточного тестера проникнення та широким діапазоном комутаторів, що тривають від дактилоскопічного відбитка даних, над отриманням даних із бази даних, до доступу до базової файлової системи та виконання команд в операційній системі через діапазонних з'єднань (рис. 2.5).

Рисунок 2.5 – Консоль Sqlmap

Особливості інструменту Sqlmap:

– Повна підтримка MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB, Informix, HSQLDB і H2-систем управління базами даних.

– Повна підтримка шести методів ін'єкцій SQL: логічний на основі булів, часовий блайнд, заснований на помилках, UNION на основі запитів, складені запити та позадіапазонні.

– Підтримка безпосередньо підключення до бази даних, не проходячи через ін'єкцію SQL, шляхом надання облікових даних СУБД, IP-адреси, порту та імені бази даних.

– Підтримка перерахування користувачів, хешей паролів, привілеїв, ролей, баз даних, таблиць і стовпців.

– Автоматичне розпізнавання форматів хешу для паролів та підтримка їх злому за допомогою словника-атаки.

– Підтримка повного скидання таблиць баз даних, набір записів або конкретних стовпців відповідно до вибору користувача.

– Підтримка пошуку конкретних імен баз даних, конкретних таблиць у всіх базах даних або конкретних стовпців у всіх таблицях баз даних.

– Підтримка для завантаження та вигрузки будь-якого файлу з сервера бази даних, що лежить в основі файлової системи, коли програмне забезпечення баз даних є MySQL, PostgreSQL або Microsoft SQL Server.

– Підтримка виконання довільних команд та отримання їх стандартного виводу на сервері баз даних, що лежить в основі операційної системи, коли програмним забезпеченням бази даних є MySQL, PostgreSQL або Microsoft SQL Server.

– Підтримка встановлення позадіапазонного стану TCP-зв'язку між машиною зловмисника та сервером бази даних, що лежить в основі операційної системи. Цей канал може бути інтерактивним командним рядком, сеансом метрпретера або сеансом графічного користувальницького інтерфейсу (VNC) за вибором користувача.

– Підтримка ескалації привілеїв користувачів баз даних за допомогою команди Metasploit Meterpreter «getystem».

Як бачимо, існує надзвичайно широкий спектр інструментарію для виявлення і обробки різних уразливостей, навіть можливості з їхнього створення. У практичній частині роботі показано використання одного з них.

РОЗДІЛ 3
РЕЗУЛЬТАТИ ТЕСТУВАННЯ ВЕБДОДАТКУ НА ОСНОВІ СТАНДАРТУ OWASP TOP-10

3.2 Процес тестування додатку програмним забезпеченням для автоматизованого пошуку вразливостей

Для практичної реалізації завдань кваліфікаційної роботи проведено тестування вебдодатку за допомогою OWASP Zed Attack Proxy (ZAP). Особливості використання самої програми виявляються в ході тестування.

Завантажити дане програмне забезпечення можна за посиланням https://github.com/zaproxy [27]. Встановлення легке та інтуїтивне. Після встановлення та відкриття програми відкривається стартове вікно (рис.3.1).

Рисунок 3.1 – Стартове вікно ZAP (верхня частина)

У програмі можливо тестувати наявність вразливостей вручну або автоматично. Але навіть автоматично програма тестує додаток на загрози з списку OWASP Top 10, а також на інші поширені вразливості.

Для перевірки вебдодатку зі списку режимів обираємо стандартний режим (рис. 3.2).

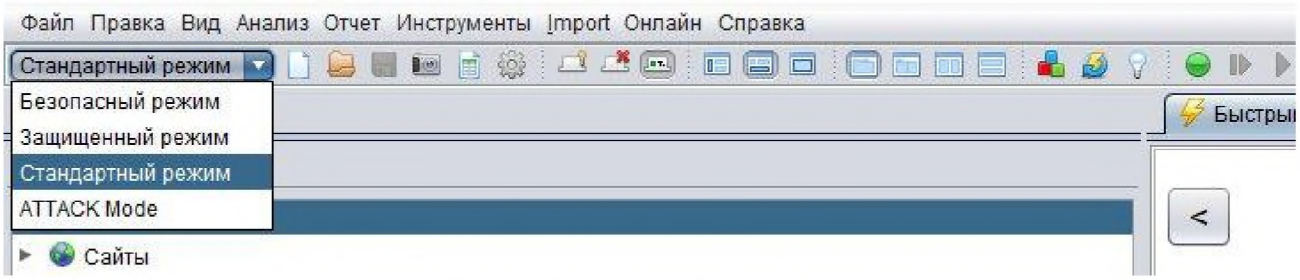


Рисунок 3.2 – Вибір режиму перевірки додатку

В адресному рядку «URL to attack» вписуємо посилання на сайт, який будемо тестувати і натискаємо кнопку «Атака» (рис. 3.3).

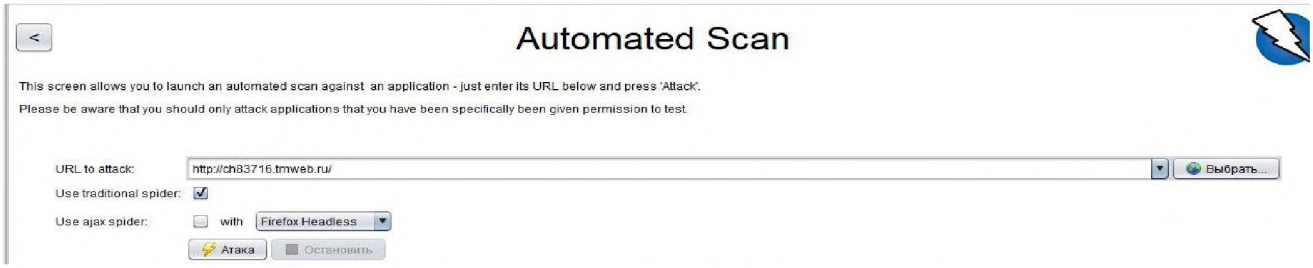
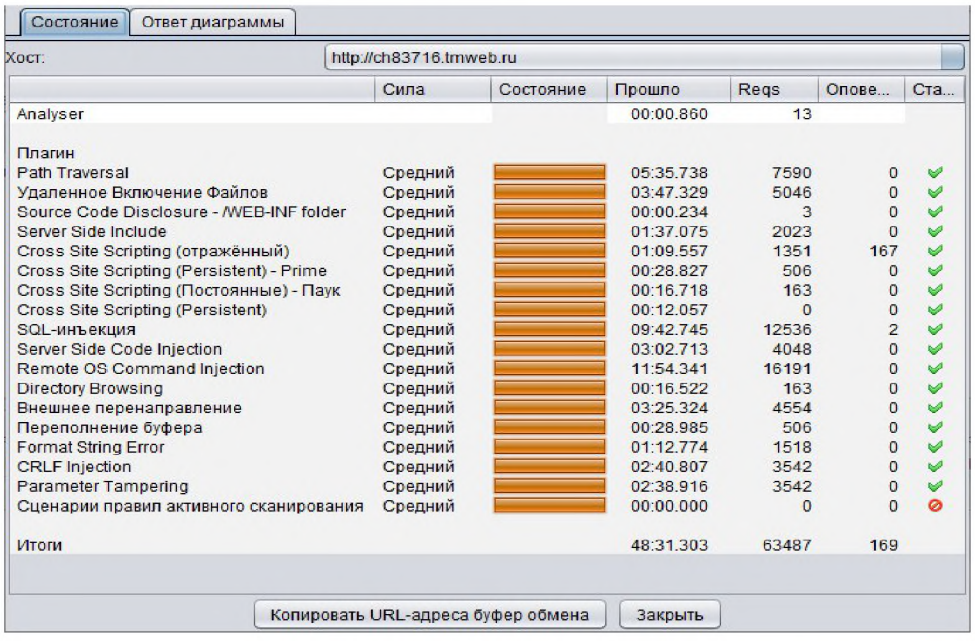


Рисунок 3.3 – Запуск сканування додатку

Процес сканування триває порівняно швидко, враховуючи детальність роботи: зайняв трохи менше години (рис. 3.4).



	Сила	Состояние	Прошло	Reqs	Опове...	Ста...
Analyser			00:00.860	13		
Плагин						
Path Traversal	Средний		05:35.738	7590	0	
Удаленное Включение Файлов	Средний		03:47.329	5046	0	
Source Code Disclosure - /WEB-INF folder	Средний		00:00.234	3	0	
Server Side Include	Средний		01:37.075	2023	0	
Cross Site Scripting (отражённый)	Средний		01:09.557	1351	167	
Cross Site Scripting (Persistent) - Prime	Средний		00:28.827	506	0	
Cross Site Scripting (Постоянные) - Паук	Средний		00:16.718	163	0	
Cross Site Scripting (Persistent)	Средний		00:12.057	0	0	
SQL-инъекция	Средний		09:42.745	12536	2	
Server Side Code Injection	Средний		03:02.713	4048	0	
Remote OS Command Injection	Средний		11:54.341	16191	0	
Directory Browsing	Средний		00:16.522	163	0	
Внешнее перенаправление	Средний		03:25.324	4554	0	
Переполнение буфера	Средний		00:28.985	506	0	
Format String Error	Средний		01:12.774	1518	0	
CRLF Injection	Средний		02:40.807	3542	0	
Parameter Tampering	Средний		02:38.916	3542	0	
Сценарии правил активного сканирования	Средний		00:00.000	0	0	
Итоги			48:31.303	63487	169	

Рисунок 3.4 – Процес сканування

За цей час програма просканувала додаток на всі відомі їй вразливості. В процесі сканування при знаходженні вразливості, програма додає дану вразливість і місце, де вона була знайдена, в вкладку «Сповіщення». На рис. 3.5 зображені всі знайдені вразливості після сканування програмою ZAP.

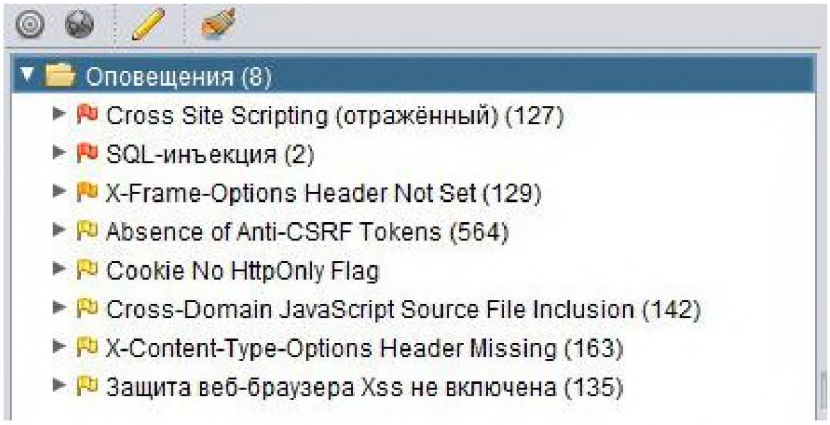


Рисунок 3.5 – Вкладка «Сповіщення» зі знайденими вразливостями

Для кожної знайденої вразливості можна переглянути деталі, такі як місце, де вразливість була знайдена, наприклад. рис. 3.6.

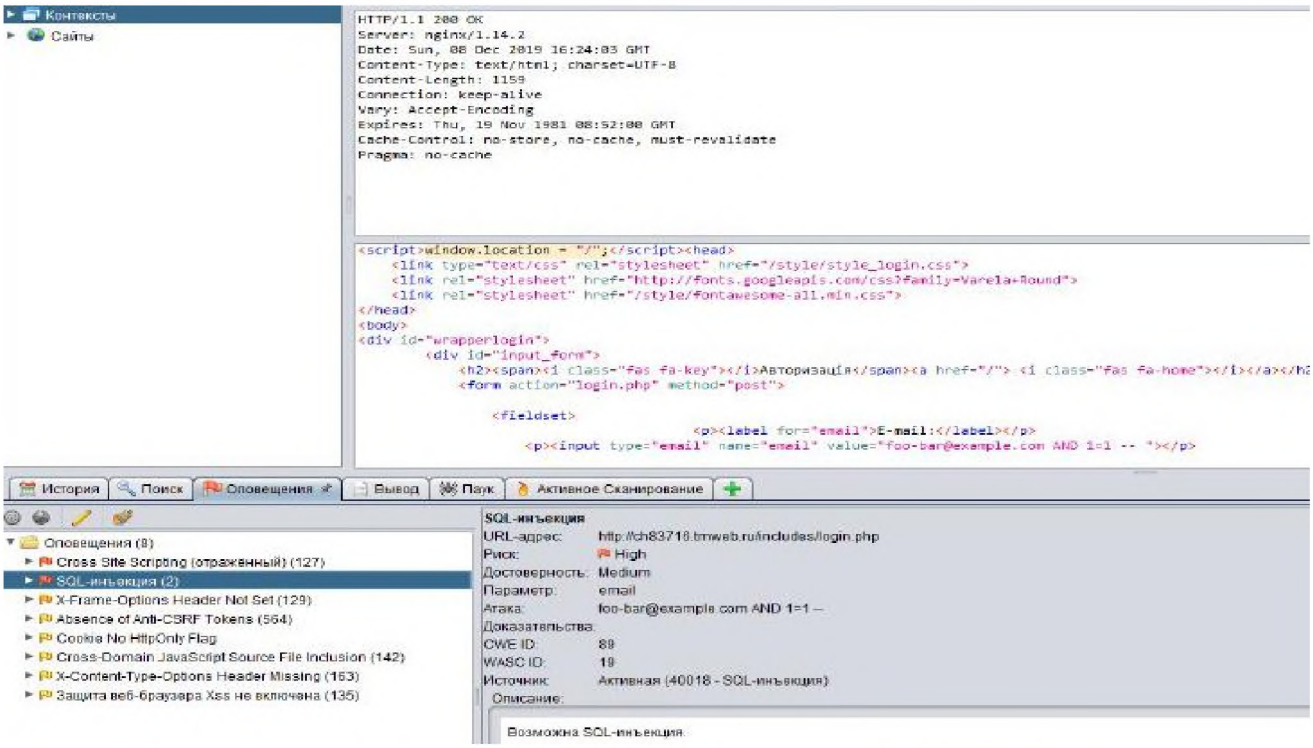


Рисунок 3.6 – Знайдена вразливість SQL-injection

Окрім критичних вразливостей зі списку OWASP TOP-10, були знайдені також інші некритичні вразливості. Для більшої деталізації звіти по різним видам вразливостей наведено в додатку Б. Всього знайдено 8 типів різних уразливостей розглянутого вебдодатку.

Щоб визначити тип SQL-injection була обрана програма Sqlmap. Програма має консольний вигляд (див. рис. 2.5), найпростіше її встановити та користуватись нею через операційну систему Ubuntu.

Завантажити її можна по посиланню https://github.com/sqlmapproject/sqlmap. Також на ресурсі доступні всі команди для встановлення та запуску Sqlmap. На рис. 3.7 зображена стартова консоль.

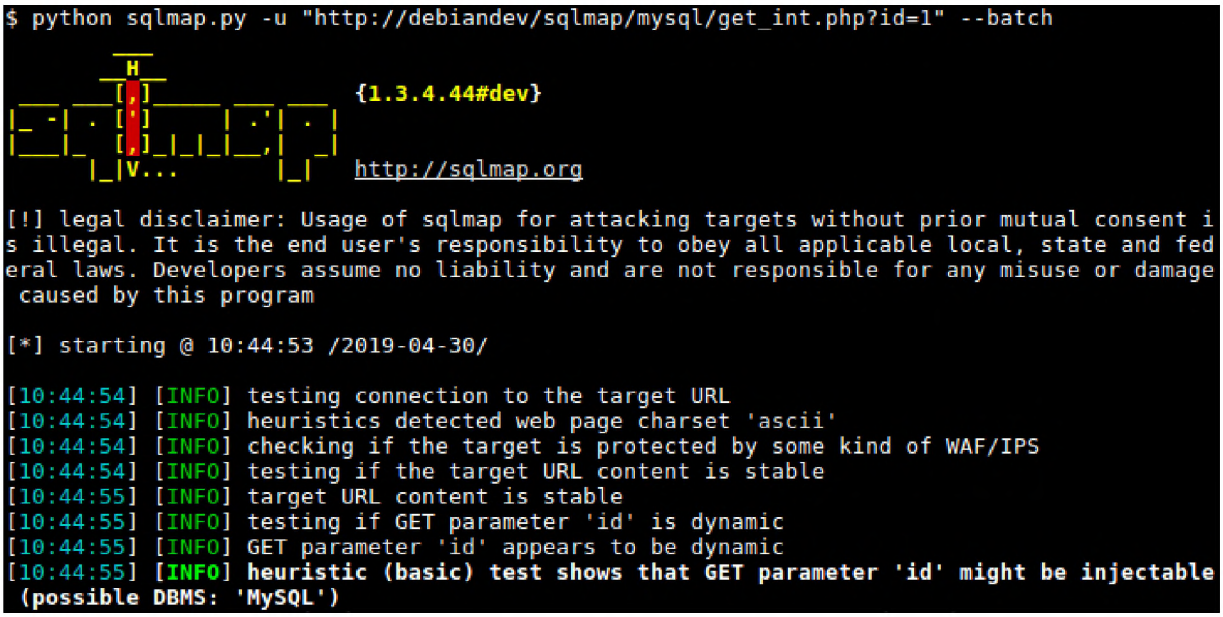


Рисунок 3.7 – Стартове зображення консолі Sqlmap

Для того щоб перевірити наявність вразливості SQL-injection та її тип, запускаємо свій веб-додаток в приватній мережі. Далі вводимо команду:

```
./sqlmap.py --url=”http://192.168.152.129/dvwa/vulnerabilities/sqli/?id=1&Submit=Submit#” --cookie="security=low; PHPSESSID=e8495b455c5ef26c415ab480425135ee
```

Після перевірки виводиться результат, показаний на рис. 3.8.

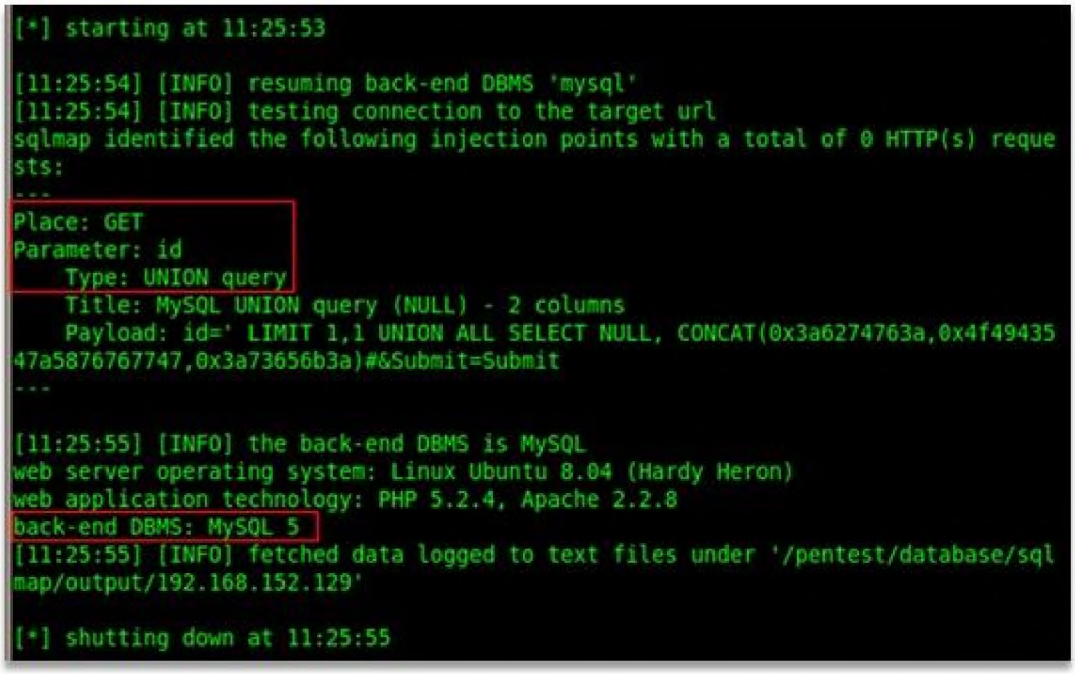


Рисунок 3.8 – Результат тестування із позначенням типу вразливості

Червоною рамкою (див. рис. 3.8) виділено знайдену вразливість, місце, де вона міститься – Get, параметр – id та її тип – UNION query. Також в результатах сканування видно версію MySQL – 5.

Після проведеного сканування і знаходження вразливостей слідує підбір методів їхнього усунення (ліквідації).

3.2 Методи усунення виявлених вразливостей

У ході реалізації практичного завдання роботи під час сканування додатку було знайдено 8 вразливостей (див. додаток Б):

- Cross Site Scripting;
- SQL-injection;
- X-Frame-Options Header Not Set;
- Absence of Anti-CSRF Tokens;

- Cookie No HttpOnly Flag;
- Cross-Domain JavaScript Source File Inclusion;
- X-Content-Type-Options Header Missing;
- Защита веб-браузера XSS не включена.

Не дивлячись, що деякі з них і є критичними їх усунення досить не складне. До виявлених проблем існують відповідні методи, які застосовуються на певних етапах розроблення самих вебдодатків.

Етап: архітектура і дизайн.

Слід використовувати перевірену бібліотеку або інфраструктуру, яка допускає виникнення цієї уразливості або надає конструкції, що полегшують її усунення. Приклади бібліотек і структур, що полегшують створення правильно закодованого виведення, включають бібліотеку Microsoft Anti-XSS, модуль кодування OWASP ESAPI і Apache Wicket [28].

Для будь-яких перевірок безпеки, які виконуються на боці клієнта, переконайтеся, що ці перевірки дубльовані на боці сервера, щоб уникнутиCWE-602. Зловмисники можуть обійти перевірки на стороні клієнта, змінивши значення після виконання перевірок або змінивши клієнта, щоб повністю видалити перевірки на стороні клієнта. Потім ці змінені значення будуть відправлені на сервер. По можливості використовувати структуровані механізми, які автоматично забезпечують поділ даних і коду. Ці механізми можуть автоматично надавати релевантне цитування, кодування і перевірку, замість того щоб покладатися на те, що розробник надасть цю можливість в кожній точці, де генерується висновок.

Етапи: реалізація; архітектура і дизайн. Важливо розуміти контекст, в якому будуть використовуватися свої дані і очікуване кодування. Це особливо важливо при передачі даних між різними компонентами або при створенні вихідних даних, які можуть одночасно містити кілька кодувань, таких як вебсторінки або поштові повідомлення, що складаються з декількох частин. Для визначення необхідної

стратегії кодування вивчіть всі очікувані протоколи зв'язку і представлення даних. Для будь-яких даних, які будуть виводитися на іншу вебсторінку, особливо для отриманих із зовнішніх входів, використовується кодування для всіх не буквено-цифрових символів, використвуються рекомендації XSS Prevention для отримання докладної інформації про необхідні типи кодування та екранування.

Етап: Реалізація. Для кожної згенерованої вебсторінки вказується кодування символів: ISO-8859-1 або UTF-8. Якщо кодування не вказано, ве браузер може вибрати інше кодування за замовчуванням або випадково. Це може привести до того, що браузер буде обробляти певні послідовності як особливі, відкриваючи клієнт для прихованих XSS-атак. Щоб допомогти пом'якшити атаки XSS проти файлу cookie сеансу користувача, встановіть для файлу cookie сеансу значення HttpOnly. У браузерах, які підтримують функцію HttpOnly (наприклад, в більш пізніх версіях Internet Explorer і Firefox), цей атрибут може перешкоджати доступу файлу cookie призначеного для користувача сеансу для шкідливих сценаріїв на стороні клієнта, використовують document.cookie. Це не повне рішення, оскільки HttpOnly підтримується не всіма браузерами. Що ще більш важливо, XMLHTTPRequest та інші потужні технології браузера забезпечують доступ для читання до заголовкам HTTP, включаючи заголовок Set-Cookie, в якому встановлений прапор HttpOnly [28]. Припустимо, що всі ввідні дані є шкідливими. Використовуйте стратегію перевірки вхідних даних «приймайте свідомо хороше», тобто використовуйте білий список допустимих вхідних даних, які строго відповідають специфікаціям. Відхилити будь введення, який не строго відповідає специфікаціям, або перетворити його у щось, що відповідає. Не покладайтеся виключно на пошук зловмисних або перекручених даних (тобто не покладайтеся на чорний список). Проте, чорні списки можуть бути корисні для виявлення потенційних атак або визначення того, які вхідні дані спотворені їх слід відхилити відразу. При виконанні перевірки введення враховуйте всі потенційно релевантні властивості, включаючи довжину, тип введення, повний

діапазон допустимих значень, пропущені або додаткові вхідні дані, синтаксис, узгодженість по зв'язаних полях і відповідність бізнес-правилам. Як приклад логіки бізнес-правил «човен» може бути синтаксично допустимої, оскільки вона містить тільки букви і цифри символи, але вона неприпустима, якщо ви очікуєте кольору, такі як «червоний» або «синій». Переконайтеся, що ви виконуєте перевірку вхідних даних в чітко визначених інтерфейсах додатка. Це допоможе захистити додаток, навіть якщо компонент використовується повторно або переміщений в інше місце [29].

. Не довіряйте введення даних на стороні клієнта, навіть якщо там є перевірка. Як правило, вводьте перевірку всіх даних на стороні сервера. Якщо додаток використовує JDBC, використовуйте PreparedStatement або CallableStatement з параметрами, переданими через «?». Якщо додаток використовує ASP, використовуйте об'єкти команд ADO з суворою перевіркою типів і параметризовані запити. Якщо можна використовувати збережені процедури бази даних, використовуйте їх. 'Not' об'єднуйте рядки в запити в збереженій процедурі, або використовуйте 'exec', 'exec immediate' або еквівалентну функціональність. Не створюйте динамічні запити SQL, використовуючи просту конкатенацію рядків [28].

Скинути всі дані, отримані від клієнта. Застосуйте «білий список» дозволених символів або «чорний список» заборонених символів при введенні користувачем. Застосуйте принцип найменших привілеїв, використовуючи користувача бази даних з найменшими привілеями. Зокрема, уникайте використання користувачів бази даних 'sa' або 'db-owner'. Це не усуває впровадження SQL, але зводить до мінімуму його вплив. Надайте мінімальний доступ до бази даних, необхідний для додатку.

Більшість сучасних веб-браузерів підтримують HTTP-заголовок X-Frame-Options. Переконайтеся, що він встановлений на всіх веб-сторінках, повертаються вашим сайтом [29].

. Етап: архітектура і дизайн. Використовуйте перевірену бібліотеку або інфраструктуру, яка допускає виникнення цієї уразливості або надає конструкції, полегшують її усунення. Наприклад, використовуйте пакети анти-CSRF, такі як

Етап: Реалізація. Переконайтеся, що у вашому додатку немає проблем з міжсайтинговим скриптингом, тому що більшість захистів CSRF можна обійти, використовуючи скрипт,контрольований зловмисником.

Етап: архітектура і дизайн. Створіть унікальний одноразовий номер для кожної форми, помістіть одноразовий номер в форму і перевірте одноразовий номер при отриманні форми. Переконайтеся, що одноразовий номер не передбачуваний (CWE-330), це можна обійти й за допомогою XSS. Визначте особливо небезпечні операції. Коли користувач виконує небезпечну операцію, відправте окремий запит підтвердження, щоб переконатися, що користувач мав намір виконати цю операцію: але це можна обійти за допомогою XSS. Використовуйте елемент управління ESAPI Session Management, який включає в себе компонент для CSRF. Не використовуйте метод GET для будь-якого запиту, який викликає зміну стану.

Етап: Реалізація. Перевірте заголовок HTTP Referer, щоб побачити, стався чи запит з очікуваної сторінки. Це може порушити законну функціональність, оскільки користувачі або проксі-сервери можуть не надсилати Referer з міркувань конфіденційності. Решта виявлені вразливості розглядати не має сенсу, так як вони вимагають більш високого рівня підготовки атакуючого і не такі небезпечні.

Кожного разу, коли файл cookie містить конфіденційну інформацію або є токеном сеансу, він завжди повинен передаватися з використанням зашифрованого каналу. Переконайтеся, що для файлів cookie, які містять таку конфіденційну інформацію, встановлений прапор secure.

. Переконайтеся, що вихідні файли JavaScript завантажуються лише з надійних джерел, і кінцеві користувачі програми не можуть контролювати джерела.

Переконайтеся, що додаток / веб-сервер встановлює Контент-Тип заголовка належним чином, і що він встановлює х-Контент-Тип заголовка параметром 'nosniff' для всіх веб-сторінок. Якщо можливо, переконайтеся, що кінцевий користувач використовує відповідно до стандартів і сучасними веб-браузер, який не виконує пробну перевірку MIME взагалі, або які можуть бути спрямовані на веб-додатки / веб-сервера, виконують пробну перевірку MIME [30].

Защита веб-браузера XSS не включена. Увімкнути захист веб-браузера XSS.

Дотримання нескладних рекомендацій, наведених вище, дозволить значно зменшити вразливість будь-якого вебдодатку, зберегти тим самим інформацію та посилити безпеку.

3.3 Оцінювання ефективності прграмного моніторингу уразливостей вебдодатків

Проаналізувавши дані, отримані на основі проведеного дослідження, можна зробити попередні оцінки щодо ефективності та доцільності використання програм тестування вебдодатків на предмет уразливостей та запобігання їх появи.

Перший висновок полягає у тому, що біьшість уразливостей закладається на етапах проєктування архітектури і дизайну вебдодатків. Тестування є можливість проводити на етапі розробки, або безпосередньо після її завершення.

Якщо в стандарті OWASP TOP-10 детально зібрано і описано причини появи 10 найбільших видів уражень вебдодатків, то за допомогою спеціального програмного забезпечення OWASP Zed Attack Proxy (ZAP) було виявлено 8 з 10 відомих видів. Це дає підстави вважати, що ефективність даної програми складає

80-100 % (або інших видів у тестовому додатку не було або ж вони були не помічені). Однак, це досить високий рівень ефективності.

На етапі розроблення проєктувальники виконують вебдодаток як продукт, найчастіше комерційного характеру. Тобто, виконується оплачувана робота. Якщо замовник заощаджує кошти на проведенні попереднього тестування для запобігання уразливостей, то воно не виконується, так само, як і попередній SEO-аналіз.

Порівняємо вартість проведення тестування V_t і визначимо частку його вартості k_t від вартості вебсайту V_s за формулою:

$$k$$

Для проведення розрахунків за формулою (3.1) необхідно отримати дані про вартість відповідних робіт. Для цього можна використати дані з вебсайтів офіційних компаній, які надають послуги з розроблення вебдодатків та їхнього тестування. Тестування, безумовно, потребують вебдодатки, які мтимуть інтерактивний функціонал, виконують складні операції, у т.ч. фінансові, проводять авторизацію тощо.

Наприклад, розроблення корпоративного вебсайту в сегменті «базовий» коштує від $450 $620 в еквіваленті [31]. За курсом НБУ $1=29,8 грн, відповідно інтервал вартості послуги 13500-18500 грн.

Робота із тестування готового вебдодатку відбувається протягом одного дня (1 год.). Усунення недоліків, редизайн та технічне коригування може зайняти до 2-х робочих днів. При заробітній платі тестувальника $450 (близько 14 000 грн/місяць) вартість роботи складє: 13500/22*2=1230 грн.

Зрештою, частка роботи з тестування в сумі замовлення вебдодатку складає:

k_t= 1230/13500=0,09.

Таким чином, лише 9 % вартості розробки всього вебдодатку можуть забезпечити максимально високий рівень його безпеки, зберегти важливу інформацію та в рази зменшити уразливість.

З наведених вище міркувань, рекомендовано:

1. Розробникам інформувати замовника про можливість проведення тестування на уразливість на етапах розробки вебдодатків.
2. Включати в пакетні пропозиції операції тестування.
3. Фірмам, що займаються вебдизайном та проєктуванням вебдодатків рекомендовано або ж тримати в штаті тестувальника. Або звертаися за послугами до спеціальних компаній.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було з'ясовано теоретичні основи появи вразливостей вебдодатків, описано та проаналізовано 10 типів уразливостей зі списку OWASP TOP-10 – переліку найбільш популярних вразливостей за оцінкою волонтерів, що разом розвивали даний проект та надавали інформацію про виявлені вразливості у вебдодатках їх компаній.

1. Під час виконання теоретичної частини кваліфікаційної роботи було проаналізовано велику кількість літератури на предмет історії, розвитку та безпеки вебдодатків.

2. В аналітичній частині детально описано вразливості зі списку OWASP TOP-10, а саме:

- ін'єкції;
- недоліки автентифікації;
- розголошення конфіденційних даних;
- зовнішні сутності xml;
- недоліки контролю доступу;
- некоректне налаштування параметрів безпеки;
- міжсайтове виконання скриптів;
- небезпечна дисеріалізація;
- використання компонентів з відомими вразливостями;
- недоліки моніторингу.

Було описано їх характеристику, як і де можуть виникати, як з ними боротися.

3. У третьому розділі описані програмні продукти, використані при виконанні практичної частини кваліфікаційної роботи, що дозволяють автоматизувати виявлення вразливостей. Серед них було обрано: OWASP Zed Attack Proxy (ZAP); Nmap; Burp Suite; Sqlmap.

4. У практичній частині роботи показані та проаналізовані результати виконання тесту на проникнення та створення звіту по цьому тесту, а також наведені можливі рішення по усуненню знайдених вразливостей. Зокрема, у результаті тестування веб-додатку було виявлено такі типи вразливостей:

- Cross Site Scripting;
- SQL-injection;
- X-Frame-Options Header Not Set;
- Absence of Anti-CSRF Tokens;
- Cookie No HttpOnly Flag;
- Cross-Domain JavaScript Source File Inclusion;
- X-Content-Type-Options Header Missing;
- Защита веб-браузера XSS не включена.

5. Проведено оцінювання ефективності застосованого програмного забезпечення для виявлення вразливостей та отримано підтвердження на рівні не менше 80 %. Розраховано частку вартості тестування вебдодатків на наявність уразливостей на стадії завершення розробки. Вартість склала 9% від загальної вартості роботи з проєктування.