МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет прикладної математики та інформатики
(повне найменування назва факультету)

кібербезпеки
(повна назва кафедри)

Дипломна робота

ДОСЛІДЖЕННЯ СИСТЕМИ ЗАХИСТУ ІНФОРМАЦІЇ ВЕБ-ЗАСТОСУНКУ

Виконав: студент групи ПМК-41с
спеціальності
125 «Кібербезпеки»
(шифр і назва спеціальності)

(підпис) Мегель Д.О (прізвище та ініціали)

Керівник (підпис) Комар К.В (прізвище та ініціали)

Науковий консультант (підпис) Вайганг Г.О (прізвище та ініціали)

Рецензент (підпис) Гордеєв О.О. (прізвище та ініціали)

2023

ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА

Факультет Прикладної математики та інформатики
Кафедра Кібербезпеки
Спеціальність 125 «Кібербезпека»
(шифр і назва)

«ЗАТВЕРДЖУЮ»
Завідувач кафедри _____

"31 _"серпня 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мегеля Дениса Олександровича
(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження системи захисту інформації веб-застосунку

керівник роботи Асистент кафедри Комар Катерина Вячеславівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені Вченою радою факультету від **" 13" вересня 2022 року № 15**

2. Строк подання студентом роботи **13.06.2023р.**

3. Вихідні дані до роботи: тестування та аналіз системи захисту інформаціїї веб-застосунку з метою виявлення потенційних вразливостей. Проведення тестів на проникнення, перевірка наявності вразливостей, оцінку ризиків.

4. Зміст дипломної роботи (перелік питань, які потрібно розробити)
1. Оцінка законодавчого регулювання з питань захисту веб-застосунків
2. Аналіз вразливостей веб-застосунків
3. Тестування веб-застосунку на вразливості

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація доповіді, виконана в Microsoft PowerPoint
4 додатки зі скріншотами результатів виконання тестування веб-застосунку на вразливості

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Вступ	Кандидат технічних наук Вайганг Г.О	22.03	15.05
Розділ 2	Кандидат технічних наук Вайганг Г.О	22.03	15.05
Розділ 3	Кандидат технічних наук Вайганг Г.О	22.03	15.05
Розділ 4	Кандидат технічних наук Вайганг Г.О	22.03	15.05

7. Дата видачі завдання **15 березня 2023 р.**

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Уточнення постановки завдання	22.03.2023 р.	
2	Аналіз літератури	29.03.2023 р	
3	Обґрунтування вибору рішення	02.04.2023 р	
4	Збір даних	07.04.2023 р	
5	Оцінка законодавчого регулювання з питань захисту веб-застосунків	17.04.2023 р	
6	Аналіз вразливостей веб-застосунків	30.04.2023 р	
7	Тестування веб-застосунку на вразливості	10.05.2023 р	
8	Оформлення та друк пояснювальної записки	21.05.2023 р	
9	Оформлення презентацій	03.06.2023 р	
10	Отримання рецензій	05.06.2023 р	
11	Захист в ЕК	15.06.2023 р	

Студент _____ Мегель Д.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Комар К.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка дипломного проекту складається зі вступу, трьох розділів, що містять 28 рисунків, висновків та списку використаних джерел з 67 найменувань. Загальний обсяг роботи становить 86 сторінок.

Об'єкт дослідження: веб-застосунок, вразливості веб-застосунку, вразливості мережі, вразливості SSL/TLS, вразливості XSS атак.

Метою даної роботи є аналіз і оцінка системи захисту інформації веб-застосунку та виявлення потенційних уразливостей.

У першому розділі аналізується оцінка законодавчого та нормативного регулювання з питань захисту веб-застосунків. Аналіз регулювання на законодавчому рівні в Україні, країнах Європейського Союзу та в Сполучених Штатах Америки.

У другому розділі розглядається оцінка ефективності сучасних методів захисту веб-застосунків. Зокрема про захист на рівні мережі, серверу, додатку та бази даних.

У третьому розділі розглядається реалізація та тестування методів захисту веб-застосунків. Вибір технологій для веб-застосунку, опис архітектури та компонентів цього застосунку. Описується процес тестування додатку на вразливості, перевірка вразливостей SSL/TLS, XXS атаки та вразливостей мережі.

Галузь застосування. Матеріали роботи можуть бути використані організаціями, що розробляють або використовують веб-застосунки для подальшого вдосконалення методів тестування та захисту веб-застосунків.

Ключові слова: ВЕБ-ЗАСТОСУНОК, ВРАЗЛИВІСТІ, МЕРЕЖА, XSS АТАКА, МЕТОДИ ЗАХИСТУ ВЕБ-ЗАСТОСУНКУ, ТЕХНОЛОГІЇ.

ABSTRACT

The explanatory note of the diploma project consists of an introduction, three chapters containing 28 figures, conclusions and a list of used sources from 67 titles. The total volume of work is 86 pages.

Object of research: web application, web application vulnerabilities, network vulnerabilities, SSL/TLS vulnerabilities, XSS attack vulnerabilities

The purpose of the diploma project there is an analysis and evaluation of the information protection system of the web application and the identification of potential vulnerabilities..

The first section analyzes trends, devices, basic technologies of the Internet of Things. The ecosystem, possibilities of connections are considered. General cybersecurity and popular IoT threats are also studied.

The second section describes the main objectives, methods and means of IoT protection and the basic concepts of protection. A deeper analysis of IoT security, threats and vulnerabilities is being conducted, and current protection options are possible. Basic data protection protocols and Internet of Things protocols are studied.

The third section defines the functional requirements and structure for the IoT information security system. The algorithm of the system is developed and designed for the selected field. Performance analysis and performance evaluation are performed.

Field of application. The materials of the work can be used in the development of IoT technology, in the analysis of the security of the Internet of Things and in the construction of information security protection.

Keywords: IOT, CYBER SECURITY, THREAT, PROTOCOL, PROTECTION OF INFORMATION, INFORMATION SECURITY, VULNERABILITIES, TECHNOLOGIES.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

WEB	World Wide Web
SQL	Structured Query Language
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
XML	Extensible Markup Language
NoSQL	Not only SQL
OWASP	Open Web Application Security Project
SSL	Secure Sockets Layer
TLS	Transport Layer Security
VPN	Virtual Private Network
SSH	Secure Shell
FTP	File Transfer Protocol
Telnet	Telecommunication Network
DOM	Document Object Model
URL	Uniform Resource Locator
ORM	Object–Relational Mapping
API	Application Programming Interface

ВСТУП

На сьогоднішній день інформаційні технології займають значну частину, як робочого, так і особистого життя. Ми слухаємо музику, викликаємо таксі, робимо покупки, читаємо електронні книги, дивимось фільми, спілкуємося з друзями, навчаємось, працюємо, тощо. Усі ці послуги так чи інакше доступні через веб–застосунки в мережі Інтернет.

Проте зі збільшенням використання веб–застосунків зростає кількість чутливої інформації (“sensitive data”), яку користувачі вносять в різні застосунки. До прикладу: номери телефонів, електронні адреси, адреси проживання, банківські реквізити, данні документів, тощо.

Також, окрім кінцевих веб–застосунків, яким користуються звичайні користувачі існують також застосунки Enterprise спрямування. Їхнє призначення в автоматизації процесів для різноманітних компаній. Такі застосунки можуть містити чутливу інформацію як для не великих компаній, так і для великих корпорацій.

Кількість зломів веб–застосунків щороку зростає, що є великою загрозою для безпеки даних інтернет–користувачів та підприємств. За даними звіту "***2021 Cost of a Data Breach Report***" компанії ***IBM Security***, середня вартість даних порушень у світі залишилася високою – близько $ 4,24 мільйона на порушення, а число реєстрації випадків було виявлено понад 500 у різних країнах.

За даними ***OWASP Top Ten Project***, уразливості у веб–застосунках, такі як відкриті точки входу або відкриті точки виводу даних, залишаються найбільш поширеними проблемами у сфері веб–безпеки.

За даними дослідження компанії ***Verizon*** за 2021 рік, понад 80% випадків кібератак націлені на веб–застосунки, що робить їх одними з найбільш вразливих місць для атак. Дослідження також показало, що серед найбільш поширених типів атак на веб–застосунки входять: SQL–ін'єкції,

крадіжки ідентифікаторів сесій, крос–сайтові скрипти, використання підроблених файлів cookie.

Інші дослідження також вказують на те, що кількість кібератак на веб–застосунки продовжує зростати з року в рік, і що вони можуть бути дуже коштовними для компаній у вигляді витрат на відновлення систем та втрати даних. Тому дуже важливо розуміти загрози для веб–застосунків та вживати необхідні заходи для захисту їх від кібератак.

Злам веб–застосунків є серйозною загрозою для безпеки інформації та приватності користувачів, а також може призвести до значних фінансових втрат для організацій, що розробляють та використовують веб–застосунки. Причинами таких зломів можуть бути вразливості в програмному забезпеченні, слабкі паролі, недостатня охорона даних, а також соціальна інженерія та інші атаки, які спрямовані на отримання доступу до конфіденційної інформації. Організації, що розробляють та використовують веб–застосунки, повинні ретельно перевіряти свої системи захисту, надавати належну увагу забезпеченню безпеки та приватності користувачів, а також підвищувати рівень своєї кібербезпеки за допомогою застосування найкращих практик та технологій захисту.

Актуальність дослідження системи захисту інформації веб–застосунку полягає у тому, що зростаюча кількість веб–застосунків, які зберігають та обробляють чутливу інформацію, спричиняє зростання загроз з боку кіберзлочинців. Відсутність адекватного захисту може призвести до втрати даних, порушення конфіденційності, цілісності та доступності даних, що може призвести до серйозних наслідків для користувачів та власників веб–застосунків. Отже, дослідження системи захисту інформації веб–застосунку є актуальним для забезпечення високого рівня безпеки та захисту веб–застосунків від зловмисних атак.

Об'єктом дослідження системи захисту інформації веб–застосунку є комплекс заходів та технологій, що застосовуються з метою забезпечення безпеки веб–застосунку та його користувачів.

Предметом дослідження є комплекс системних заходів та процесів, спрямованих на захист інформації, яка обробляється веб–застосунком, від можливих загроз та вразливостей, що можуть призвести до порушення конфіденційності, цілісності та доступності інформації.

Метою дослідження системи захисту інформації веб–застосунку є аналіз та оцінка потенційних загроз, вразливостей та ризиків для безпеки веб–застосунку, а також розроблення та впровадження ефективних методів та засобів захисту, які забезпечать конфіденційність, цілісність та доступність інформації, що обробляється веб–застосунком.

Якщо дослідження системи захисту інформації веб–застосунку базується на наукових публікаціях, теоретичних аспектах та використовує методи наукового аналізу та порівняння, то можна стверджувати, що дослідження має високий науковий рівень розробленості. Однак, якщо дослідження базується на практичних аспектах та використовує методи тестування та оцінки ефективності захисних засобів, то можна говорити про середній науковий рівень розробленості.

РОЗДІЛ 1. АНАЛІЗ ПОТЕНЦІЙНИХ ЗАГРОЗ ТА ВРАЗЛИВОСТЕЙ ВЕБ–ЗАСТОСУНКІВ

Сутність дослідження системи захисту інформації веб–застосунку полягає у проведенні аналізу та оцінці ефективності захисту веб–застосунку від можливих загроз безпеці, виявленні потенційних вразливостей, їх описі та класифікації для запобігання порушення захисту додатків: вилучення чутливих даних користувача/компанії, зміна контенту веб додатку.

Існує декілька способів дослідження вразливостей веб–застосунків:

1. **Тестування на проникнення** (*Penetration testing*) – це процес тестування безпеки веб–застосунків шляхом спроб вторгнення в систему атакуючого зловмисника [1]. Цей підхід дозволяє ідентифікувати реальні вразливості в системі та оцінити їхню потенційну шкідливість.

2. **Аудит безпеки** – це дослідження системи з метою виявлення вразливостей та оцінки ризику їх експлуатації [2]. Аудит безпеки може включати ретельний аналіз коду веб–застосунку, аналіз конфігурації сервера, перевірку правильності налаштування прав доступу, перевірку системи відстеження журналів і т. д.

3. **Використання автоматизованих інструментів** – це метод дослідження вразливостей веб–застосунків з використанням спеціальних програм, що сканують код веб–застосунку на предмет вразливостей [3]. Цей підхід може значно скоротити час, необхідний для виявлення вразливостей, проте може бути менш ефективним у порівнянні з іншими методами.

4. **Тестування відмов**: цей метод використовується для виявлення вразливостей, пов'язаних з некоректним поведінкою додатку при невірних вхідних даних або під час навантаження на систему [4].

5. **Візуальний аналіз**: цей метод використовується для виявлення вразливостей, які можуть бути виявлені шляхом візуального аналізу веб–застосунку, таких як підозрілі кнопки або поля вводу, або зміни у вигляді веб–сторінок [5].

Після проведення аналізу, розробляються заходи для запобігання вразливостей та захисту веб–застосунку від можливих атак, забезпечуючи надійність та безпеку даних користувачів.

Таким чином, дослідження системи захисту інформації веб–застосунку допомагає забезпечити безпеку веб–застосунків та користувачів, зменшує ризики від вразливостей та загроз безпеці, а також підвищує довіру до веб–застосунків.

Проведення дослідження системи захисту інформації веб–застосунку є дуже важливим, оскільки забезпечення безпеки веб–застосунків є критично важливим для бізнесу, організацій та індивідуальних користувачів. Якщо веб–застосунок стає жертвою атаки, то це може призвести до втрати конфіденційної інформації, злому системи, втрати репутації та інших негативних наслідків.

Дослідження системи захисту інформації веб–застосунку дозволяє виявити потенційні вразливості та недоліки системи захисту, що допоможе розробити стратегії захисту, а також вжити відповідних заходів, щоб попередити можливі атаки.

Додатково, дослідження системи захисту інформації веб–застосунку дозволяє виявити можливі проблеми з безпекою, що можуть знаходитися під поверхнею, і забезпечити їх вирішення раніше, ніж вони можуть бути використані хакерами. Також, дослідження допомагає зрозуміти, як веб–застосунок працює з точки зору безпеки, і забезпечити ефективну захист від потенційних загроз.

Мета аналізу потенційних загроз та вразливостей веб–застосунків полягає у виявленні можливих шляхів атак на веб–додаток та виявленні

вразливостей, які можуть бути використані для злому системи. Цей аналіз дозволяє розробникам та адміністраторам забезпечити належний рівень захисту веб–застосунку та запобігти можливим кібератакам, які можуть призвести до втрати даних, порушення конфіденційності користувачів та інших проблем. Крім того, аналіз потенційних загроз може допомогти забезпечити відповідність веб–додатку нормам безпеки та захисту даних, таким як GDPR або HIPAA.

1.1 Види ін’єкційних загроз

Ін’єкційні загрози (Injection flaws) – клас вразливостей веб–застосунків, які дозволяють зловмисникам впроваджувати вразливий код у вхідні дані додатку [6]. Далі цей код може бути виконаний на стороні сервера, що дозволяє зловмисникам виконувати різні дії, такі як отримання конфіденційної інформації, віддалене керування сервером або зловживання повноваженнями користувача.

До Injection flaws відносяться такі загрози:

SQL Injection – це техніка злому веб–застосунків, яка полягає в введенні шкідливого коду SQL в веб–форму, що призначена для взаємодії з базою даних [7]. Ця техніка дозволяє зловмисникам отримувати несанкціонований доступ до бази даних, отримувати конфіденційну інформацію та виконувати різні операції з базою даних, такі як додавання, видалення і зміна даних (рис.1.1).

SQL Injection є однією з найбільш поширених технік злому веб–застосунків, і її можна використовувати для атак на будь–які веб–сайти, які використовують бази даних для зберігання інформації. Це може стати серйозною загрозою для безпеки веб–додатків і даних користувачів.

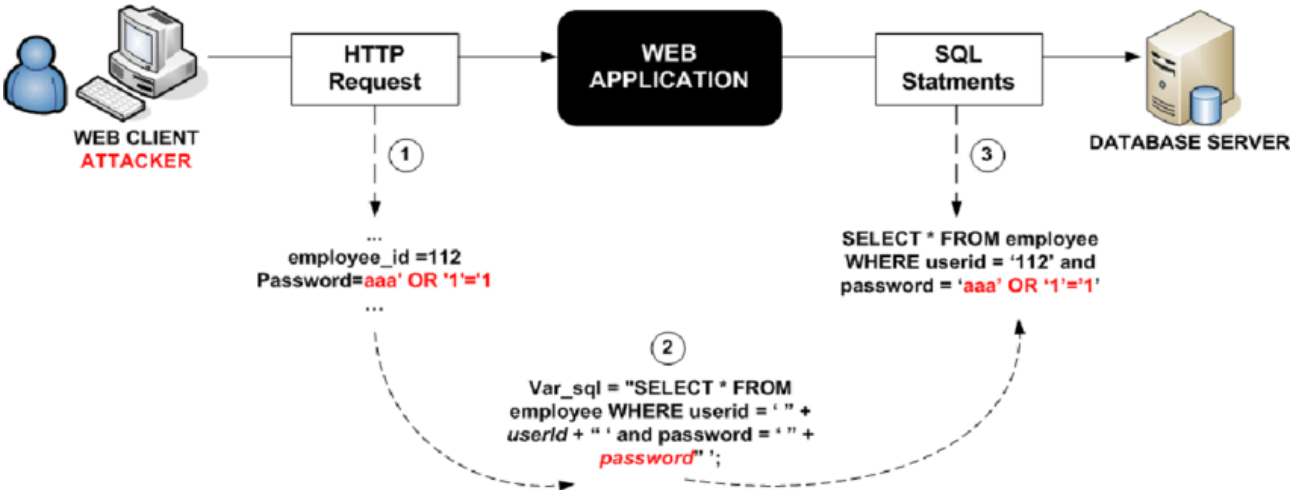


Рисунок 1.1 – Візуалізація SQL ін'єкції

NoSQL Injection – це вразливість, яка дозволяє зловмисникам виконувати зловживання в NoSQL базах даних, подібно до SQL Injection для реляційних баз даних [8]. Бази даних зберігають дані у різних форматах, таких як документи, ключ–значення, графи тощо. В залежності від конкретної реалізації бази даних, можуть бути використані різні механізми запитів і обробки даних. Однак, багато NoSQL баз даних використовують спеціальну мову запитів, яка називається Query Language (наприклад, MongoDB використовує мову запитів MongoDB Query Language, а CouchDB використовує мову запитів JavaScript).

Атака NoSQL Injection полягає в тому, що зловмисник вставляє шкідливий код в запит до бази даних, використовуючи механізми вводу, які не враховують правильність введеного коду. Як результат, шкідливий код виконується в контексті запиту до бази даних, що дозволяє зловмиснику отримувати незаконний доступ до даних в базі даних, модифікувати їх або видаляти (рис.1.2).

```
db.users.find({ username: { $ne: "" }, password: { $ne: "" } })
```

Рисунок 1.2. – Приклад шкідливого запиту до бази даних MongoDB

Зловмисник може вставити такий шкідливий код у рядок запиту для отримання доступу до всіх даних в базі даних, незалежно від того, який логін і пароль будуть введені (1.3).

```
db.users.find({ $where: "this.username.length >= 0 && this.password.length >= 0" })
```

Рисунок 1.3. – Приклад шкідливого запиту до бази даних MongoDB

Command Injection – це тип атаки на веб–додаток, при якому зловмисник може виконати злоякісну команду на системному рівні [9]. Це може статися через недостатньо валідації вхідних даних, що передаються в системні команди. Наприклад, якщо додаток використовує функцію командного рядка для виконання деякої дії, то зловмисник може ввести спеціально створену команду, яка виконає додаткові дії на системному рівні.

Ця атака може мати серйозні наслідки, такі як видалення файлів, встановлення шкідливого програмного забезпечення або навіть повний контроль над системою.

Наведемо декілька прикладів, де можуть виникнути проблеми з командним виконанням (Command Injection) (рис.1.4):

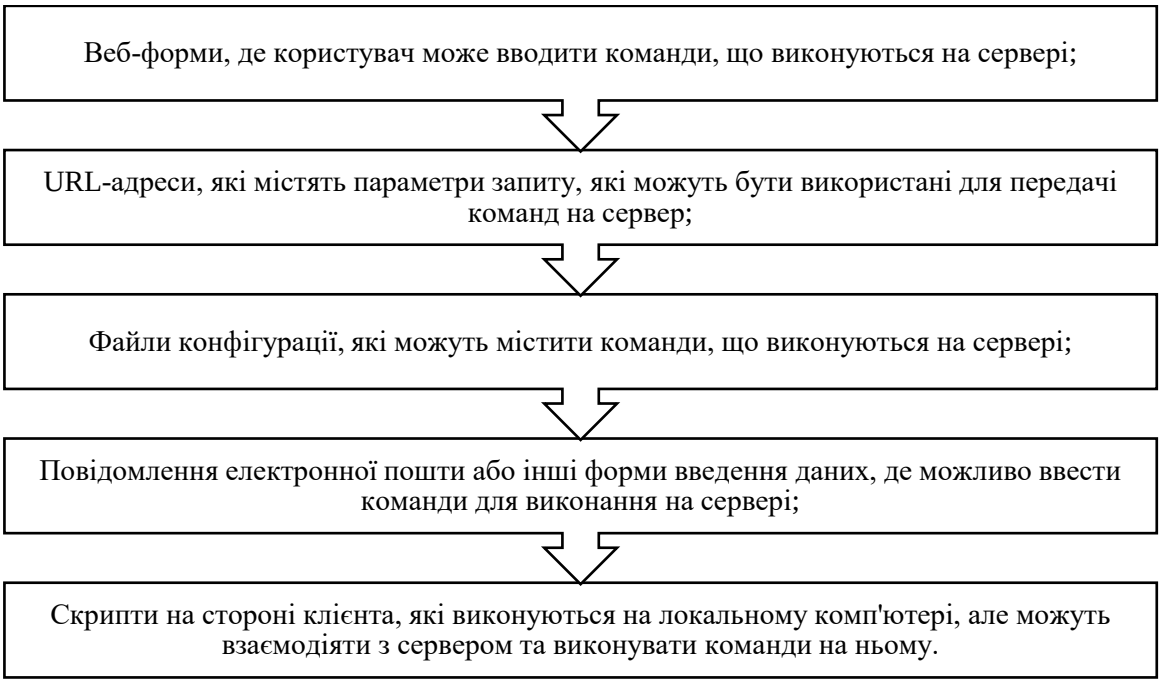


Рисунок 1.4. – Приклад некоректного виконання Command Injection

У випадку, якщо зловмисник виконує команди на сервері, він може мати доступ до конфіденційної інформації, змінювати конфігурації системи, видаляти файли або навіть отримувати повний контроль над сервером. Тому важливо забезпечити належний рівень захисту від подібних атак.

LDAP Injection – це вразливість в додатках, які використовують Lightweight Directory Access Protocol (LDAP) для доступу до каталогів. Ця вразливість дозволяє зловмисникам виконувати шкідливі LDAP запити, які можуть призвести до отримання неправомірного доступу до даних в каталозі або виконання інших шкідливих дій на сервері [10].

Приклади LDAP Injection включають в себе:

1. Шукання з використанням недійсного фільтра: Ця атака використовується для знаходження інформації, яка не повинна бути доступна користувачу. Наприклад, зловмисник може ввести недійсний фільтр під час пошуку користувачів в каталозі, що дозволяє йому отримати доступ до конфіденційної інформації, такої як паролі.

2. Видалення обмежень: Ця атака полягає в тому, щоб змінити LDAP запит, щоб отримати доступ до обмеженого доступу або забороненого контенту.

3. Додавання користувача: Зловмисник може використовувати LDAP Injection для створення нового користувача з підвищеними привілеями і отримання доступу до конфіденційної інформації.

4. Підміна параметрів: Зловмисник може змінити параметри LDAP запиту, щоб отримати доступ до неправомірної інформації або виконати шкідливі дії на сервері.

5. Використання підрядків: Зловмисник може використовувати підрядки в запиті LDAP, щоб знайти чутливу інформацію, яка може бути використана для виконання подальших атак.

Деякі приклади LDAP Injection (рис. 1.5):

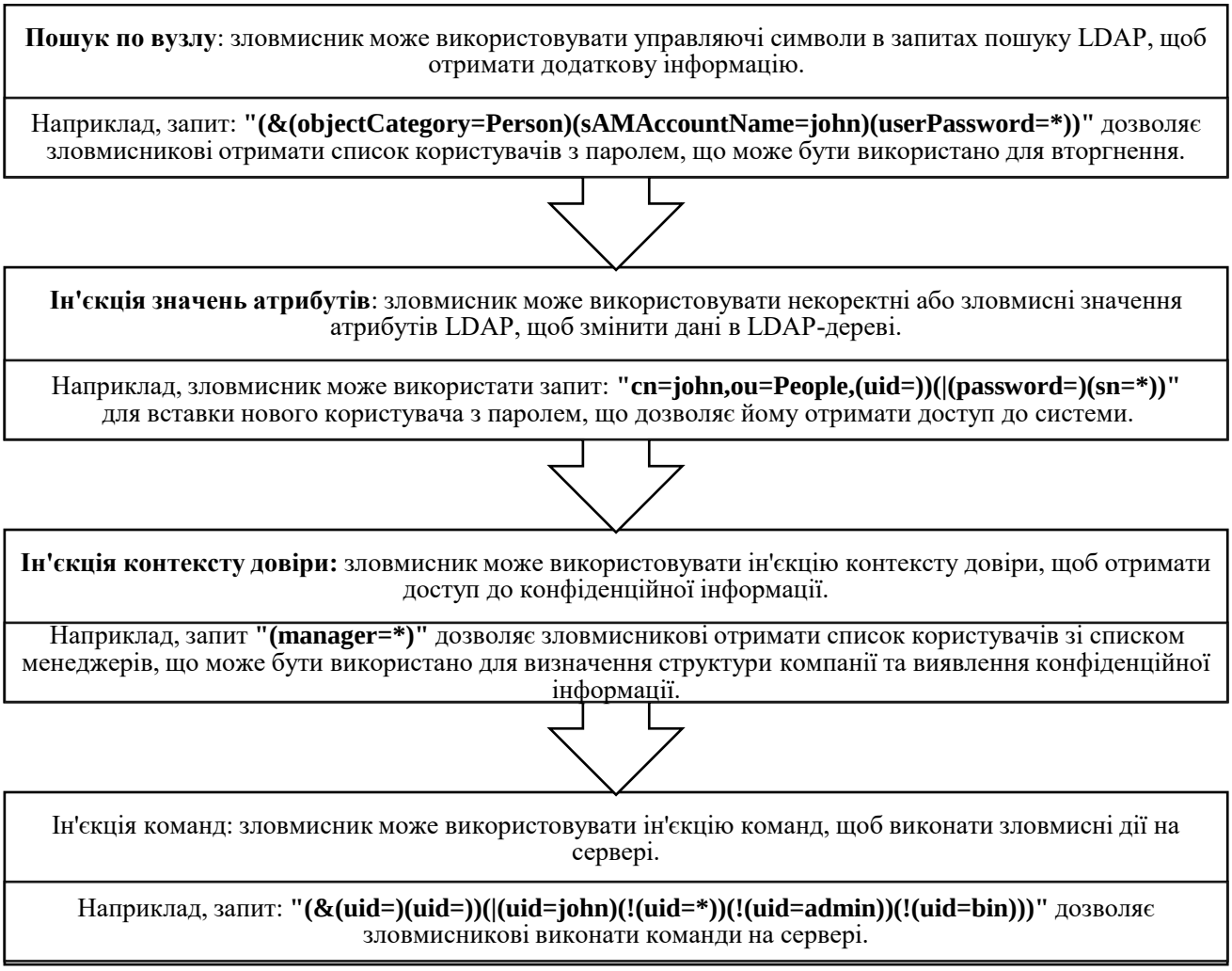


Рисунок 1.5. – Приклад виконання LDAP Injection

XPath Injection – це вразливість, яка дозволяє зловмисникам виконувати XPath–запити з метою отримання неправомірного доступу до даних або отримання конфіденційної інформації. Ця вразливість виникає, коли вхідні дані не достатньо перевіряються перед тим, як вони будуть використані для створення XPath–запитів [11].

XPath (XML Path Language) – мова запитів до документів у форматі XML. Вона дозволяє вибирати із документу певні його елементи, атрибути, текст та інші дані за допомогою логічних виразів, які називаються XPath–запитами.

Вони використовуються у програмуванні та розробці веб–застосунків для отримання даних з XML–документів. Вони можуть використовуватись

для пошуку певних елементів в документі, фільтрації даних за певними умовами, вибору конкретних атрибутів та інших операцій.

Дані запити можуть бути складними та містити різні функції та умови, що дозволяє отримати потрібні дані з XML–документу у зручному форматі для подальшої обробки.

Нижче наведемо деякі приклади XPath Injection (1.6):

Рисунок 1.6. – Приклад виконання XPath Injection

Наприклад, якщо веб–застосунок використовує запит XPath для пошуку даних у базі даних XML, зловмисник може ввести такі дані, як ' or 1=1 or "=', що призведе до вибору всіх записів у базі даних, оскільки умова буде завжди вірна.

Іншим прикладом може бути введення у запит некоректного символу, який призведе до помилки. Зловмисник може скористатися цим для отримання детальної інформації про сервер та його налаштування.

У результаті XPath Injection може дозволити зловмиснику отримати доступ до конфіденційної інформації, виконати дії від імені адміністратора, внести зміни у базу даних та інше.

На сьогоднішній день такий формат даних вважається застарілим і не використовується у більшості додатків, так як більшість сучасних систем розробляється на базі обміну даними JSON формату. Проте така концепція все ще присутня у застарілих банківських системах, CRM, ERP, тощо.

OS Commanding (або OS Command Injection) є вразливістю, яка дозволяє зловмиснику виконувати команди на операційній системі, яка виконується на сервері. Це досягається шляхом вставки шкідливого коду в вхідні дані, які після обробки сервером передаються як команди до операційної системи (1.7) [12].

Наприклад, якщо сервер використовує команду оператора системи `"ls"` для виведення списку файлів, зловмисник може вставити додатковий код в параметри запиту для виконання будь–якої іншої команди, наприклад, **`"rm"`** для видалення файлів або **`"cat"`** для читання вмісту файлів на сервері.

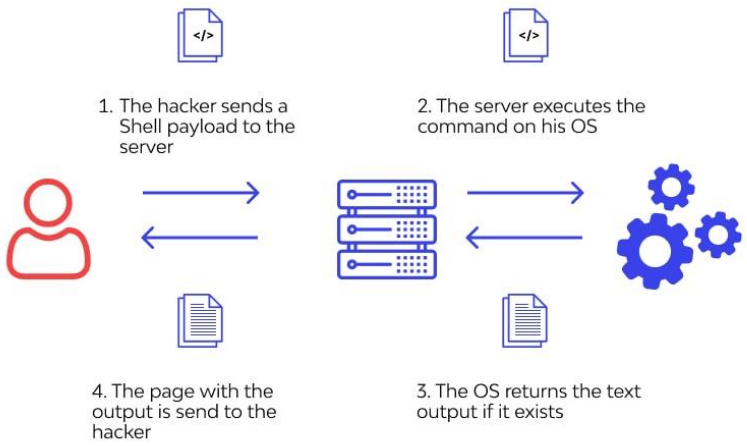


Рисунок 1.7 – Зображення OS Command Injection

OS Command Injection може бути використаний для виконання різних атак, таких як отримання конфіденційної інформації, внесення змін до файлів, завантаження інших шкідливих програм на сервер та інших дій, що можуть призвести до порушення безпеки системи.

XXS атака (англ. **Cross–site scripting**) – це тип вразливості веб–додатка, при якому зловмисник вставляє у веб–сторінку зловмисний код, який потім виконується в браузері користувача [13]. Це може призвести до крадіжки даних користувача, обмеження доступу до сайту, виконання шкідливого коду на стороні користувача та інших негативних наслідків.

Атака XXS може бути здійснена в результаті недостатньої перевірки введених даних, включення користувацьких даних в вихідну HTML–сторінку без очищення або інших помилок в програмному коді веб–додатка. Існують два типи XXS атак: збочений (англ. stored) та зображений (англ. reflected). У збоченому XXS зловмисник вставляє зловмисний код в базу даних, який потім зображається на веб–сторінках. У зображеному XXS зловмисник використовує посилання або форму на веб–сторінці, щоб передати зловмисний код користувачеві (рис. 1.8).

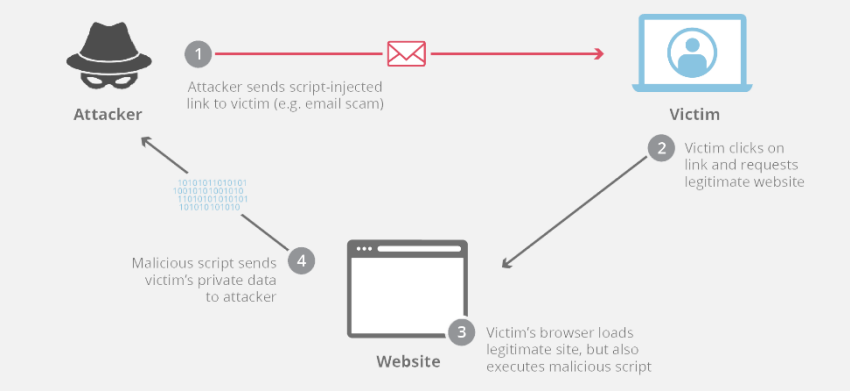


Рисунок 1.8 – Схема XXS атаки

Для захисту від атак XXS рекомендується використовувати перевірку та очищення введених даних, обмеження користувацького вводу та використання безпечних методів зображення даних на веб–сторінках. Також можна використовувати відповідні настройки безпеки веб–сервера і використання спеціальних інструментів.

Існують три типи XXS атак:

1. **Stored XSS** – уразливість, при якій зловмисник вбудовує скрипти на сторінки веб–додатка, які зберігаються на сервері та зображаються

відвідувачам сайту. Таким чином, кожен, хто відвідає сторінку, піддається атаці;

2. **Reflected XSS** – уразливість, при якій зловмисник використовує спеціальні посилання або форми, щоб вбудувати скрипти на сторінки веб–додатка, які будуть відображені користувачам, які перейдуть за цими посиланнями або відправлять форму;

3. **DOM–based XSS** – уразливість, при якій зловмисник використовує JavaScript код для вбудовування скриптів в сторінки веб–додатка. Ці скрипти виконуються в браузері користувача при зображенні сторінки, що може призвести до виконання шкідливого коду.тів для виявлення інших вразливостей у веб–додатку.

1.2 Порушення аутентифікації та авторизації

Broken authentication and session management – це вразливість, яка дозволяє зловмисникам отримати доступ до облікових записів користувачів без необхідності знати їхні паролі або інші конфіденційні дані [14]. Ця вразливість може виникнути внаслідок недостатньої перевірки аутентифікаційних даних, використання слабких сесійних ключів або інших проблем, пов'язаних з керуванням сесіями.

Наприклад, зловмисник може використовувати вразливість автентифікації для входу до облікового запису користувача без необхідності вводити правильний пароль. Якщо зловмисник здобув доступ до сесійного ключа, він може використовувати його для підробки сесії та отримання доступу до даних користувача.

У класифікації вразливостей Broken authentication and session management можна виділити наступні загрози:

1. Перехоплення сесії: зловмисники можуть перехопити і використовувати сесійний ідентифікатор, щоб отримати доступ до системи без авторизації;

2. Session fixation: це атака, в ході якої зловмисники вимушують жертву використовувати певний сесійний ідентифікатор, який вони контролюють;

3. Брутфорс авторизації: зловмисники можуть намагатися перебрати різні комбінації логінів та паролів, щоб отримати несанкціонований доступ;

4. Недостатня автентифікація: слабкі паролі, недостатній контроль доступу та інші проблеми можуть дозволити зловмисникам отримати доступ до системи без дійсної авторизації;

5. Недостатнє управління сесією: недостатній час життя сесій, неналежне знищення сесій після виходу користувача з системи, недостатній контроль за сесіями інших користувачів можуть дозволити зловмисникам отримати доступ до системи без авторизації.

Наведемо декілька прикладів вразливостей, які можуть бути пов'язані з "Broken authentication and session management":

1. Ін'єкції авторизації: коли зловмисник може ввести спеціальні символи або код, щоб обійти механізми авторизації;

2. Погано налаштований час життя сесії: якщо час життя сесії завеликий або не належним чином налаштований, то зловмисники можуть зловити дійсну сесію та отримати доступ до ресурсів;

3. Слабка генерація токенів сесії: якщо токени сесії генеруються ненадійно або недостатньо складно, то зловмисники можуть легко їх перехопити та використовувати для зловживання правами;

4. Атаки на міжсайтову сесію (session hijacking): коли зловмисники перехоплюють дійсну сесію, здебільшого через незахищені канали зв'язку, та використовують її для доступу до ресурсів;

5. Атаки на міжсайтовий скриптинг (cross–site scripting): коли зловмисники вставляють скрипти в надіслані форми або запити, щоб отримати доступ до сесій інших користувачів;

6. Використання стандартних паролів: якщо система дозволяє використання стандартних або слабких паролів, то зловмисники можуть легко проникнути в систему, використовуючи перелік найбільш поширених паролів.

1.3 Загрози мережевої безпеки

Загрози мережевої безпеки – це потенційні небезпеки для інформаційної системи, які виникають через її підключення до мережі, та можуть призвести до несанкціонованого доступу до інформації, її пошкодження або втрати.

До загроз мережевої безпеки відносяться різноманітні атаки, такі як **DDoS** атаки, **фішинг**, мережеві вразливості, атаки на протоколи мережі, атаки на середовище мережі та інші.

Мережева безпека – це галузь інформаційної безпеки, що стосується захисту мережевих систем, комп'ютерів, пристроїв та даних від несанкціонованого доступу, викрадення, пошкодження або знищення. Вона забезпечує захист мережевої інфраструктури та даних від шкідливих загроз, таких як зловмисні програми, хакерської атаки, витоки даних та інші загрози. Мережева безпека має велике значення для бізнесу, урядових організацій та індивідуальних користувачів, оскільки захист мережевих систем є важливою складовою інформаційної безпеки в цілому [15]. Існує багато різних загроз мережевої безпеки, ось кілька прикладів (рис.1.9).

Детально розглянемо найбільш небезпечні на даний момент загрози.

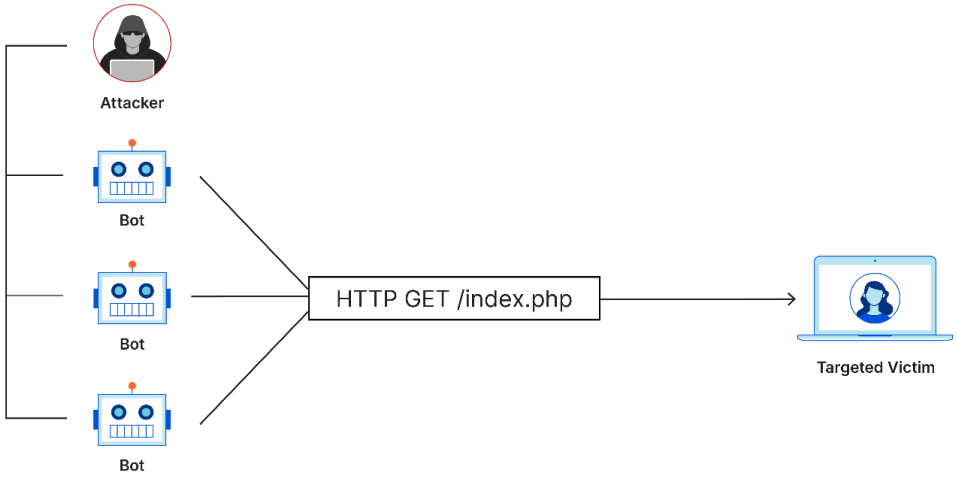
Рисунок 1.9 – Загроз мережевої безпеки

DDoS (Distributed Denial of Service) – це тип атак на комп'ютерну мережу, який спрямований на перевантаження ресурсів комп'ютерів або серверів, з метою позбавити їх користувачів доступу до сервісу або ресурсу, що забезпечується цими комп'ютерами або серверами.

Атака DDoS зазвичай здійснюється шляхом надсилання великої кількості запитів на сервер або комп'ютер, що призводить до перевантаження системи. Надходження великої кількості запитів з різних джерел одночасно робить їх неможливими для обробки, що може призвести до зупинки роботи сервера або комп'ютера.

DDoS–атаки можуть бути запущені з використанням ботнетів (масштабних мереж комп'ютерів, які були заражені вірусами), що дозволяє

зловмисникам координувати та керувати атакою з декількох джерел. Інші методи атак включають в себе використання низькоресурсних пристроїв, включаючи IoT–пристрої, що вимагають меншої потужності, але також

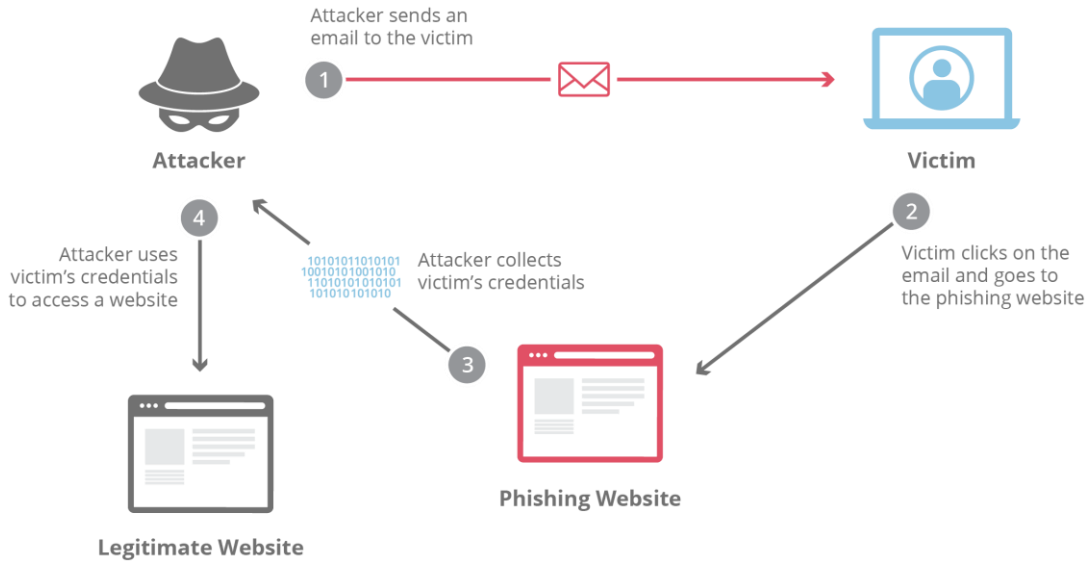


можуть бути використані для атак.

При успішному виконанні атаки DDoS, комп'ютер або сервер стає недоступним для користувачів, що може призвести до серйозних наслідків, особливо для бізнесів, які залежать від доступності своїх ресурсів в Інтернеті (рис. 1.10).

Рисунок 1.10 – Діаграма DDoS атаки

Фішинг та соціальний інжиніринг є одними з найбільш поширених методів злочинців для здійснення атак на користувачів інформаційних систем. Ці методи ґрунтуються на обмані людей з метою отримання доступу до їхньої конфіденційної інформації, такої як логіни, паролі, фінансові дані



тощо (рис. 1.11).

Рисунок 1.11 – Принцип дії фішингу

Фішинг – це вид атаки, при якій злочинці намагаються видатися за довірену особу або організацію, щоб викликати довіру у потенційної жертви [16]. Наприклад, злочинці можуть відправляти листи на ім'я користувачів, що містять посилання на фальшиві веб–сайти, які виглядають як веб–сайти реальних організацій. Коли користувачі вводять свої логіни та паролі на цих фальшивих веб–сайтах, злочинці можуть використати цю інформацію для вторгнення в реальний обліковий запис користувача та здійснення інших видів атак.

Соціальний інжиніринг – це також метод атаки, який базується на обмані людей. В цьому випадку злочинці не використовують технічні аспекти для злому інформаційної системи, а намагаються отримати доступ до конфіденційної інформації, переконавши людину розкрити її [17]. Це може бути здійснено через відсилання електронної пошти або повідомлень у соціальних мережах, приховуючи свою справжню ідентичність та створюючи ситуації, які спонукають людину до виконання певних дій.

Перехоплення даних (англ. data interception) – це процес збору, перехоплення і записування даних, що передаються між двома точками у мережі, з метою зловживання цими даними [18].

Ця загроза полягає у тому, що зловмисники можуть отримати доступ до конфіденційної інформації, такої як паролі, кредитні картки, персональні дані тощо. Інтернет–злочинці можуть використовувати різні методи перехоплення даних, такі як використання шпигунського програмного забезпечення, внесення змін до настройок мережі або використання методів перехоплення пакетів (рис. 1.12)

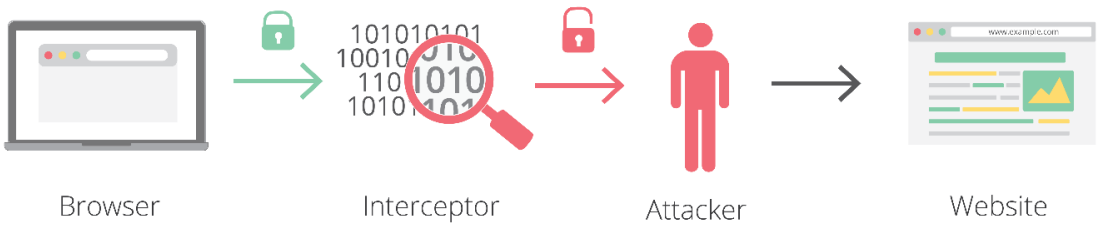


Рисунок 1.12 – Процес перехоплення даних

Щоб захистити себе від перехоплення даних, можна використовувати різноманітні криптографічні протоколи та технології, такі як SSL / TLS, VPN, SSH тощо. Важливо також дотримуватися загальних правил безпеки, таких як встановлення сильних паролів, оновлення програмного забезпечення та використання антивірусного програмного забезпечення.

Мережа Інтернет може піддається різноманітним методам перехоплення даних. Ось деякі причини можливого перехоплення даних в мережі Інтернет:

1. **Незахищені мережеві з'єднання**: Якщо мережеве з'єднання між двома пристроями не захищене, то зловмисник може легко перехопити дані, які передаються між ними.

2. **Відсутність шифрування**: Якщо дані передаються по мережі без шифрування, зловмисник може перехопити ці дані та прочитати їх.

3. **Відсутність аутентифікації**: Якщо пристрої, які здійснюють обмін даними, не проходять аутентифікацію, то зловмисник може проникнути в мережу та перехопити дані.

4. **Використання незахищених протоколів**: Деякі протоколи передачі даних, такі як FTP (File Transfer Protocol) та Telnet, не шифрують дані, що передаються. Якщо зловмисник перехопить ці дані, то він може легко отримати доступ до конфіденційної інформації.

5. **Віруси та шкідливі програми**: Шкідливі програми можуть перехоплювати дані, що передаються через мережу, та відправляти їх до зловмисника.

6. **Недостатня захищеність мережевого обладнання**: Якщо мережеве обладнання (наприклад, маршрутизатори, комутатори) не має належної захисту, то зловмисник може використовувати це обладнання для перехоплення даних.

7. Таким чином, загрози типу Injection Flaw, Broken Authentication and Session Management та мережевої безпеки є серйозними проблемами

безпеки, які можуть призвести до витоку чутливої інформації, порушення конфіденційності, цілісності і доступності системи.

Загрози типу Injection Flaw, такі як SQL Injection, NoSQL Injection, LDAP Injection і XPath Injection є серйозними уразливостями, які можуть бути використані для отримання несанкціонованого доступу до баз даних і отримання конфіденційної інформації. Для запобігання таким загрозам рекомендується використовувати параметризовані запити та валідацію введених даних, а також забезпечувати правильні права доступу до баз даних для різних користувачів.

Також важливо пам'ятати, що використання одного виду Injection Flaw може призвести до появи інших типів таких уразливостей. Наприклад, використання SQL Injection може призвести до використання OS Commanding або зламу пароля.

Загрози типу Broken authentication and session management можуть бути серйозними для системи безпеки веб–додатків та можуть призвести до компрометації конфіденційної інформації користувачів та витоку даних. Для забезпечення безпеки веб–додатків слід використовувати ефективні механізми автентифікації та керування сесіями, щоб унеможливити зловмисникам використовувати відкриті сесії ідентифікації користувача для доступу до конфіденційної інформації.

Загрози мережевої безпеки є дуже серйозною проблемою для будь–якої організації. Зловмисники можуть використовувати різні методи атак, такі як DDoS, перехоплення даних, фішинг, соціальний інжиніринг тощо, для того щоб отримати несанкціонований доступ до конфіденційної інформації, зламати систему або завдати іншої шкоди.

Важливо розуміти, що захист від загроз мережевої безпеки – це постійний процес, оскільки зловмисники постійно шукають нові шляхи атаки та вразливості. Організації повинні регулярно аудитувати свою мережеву

інфраструктуру та вживати необхідні заходи для забезпечення безпеки та захисту від зловмисних атак.

РОЗДІЛ 2. ОЦІНКА ЗАКОНОДАВЧОГО ТА НОРМАТИВНОГО РЕГУЛЮВАННЯ З ПИТАНЬ ЗАХИСТУ ВЕБ–ЗАСТОСУНКІВ

Оцінка законодавчого та нормативного регулювання з питань захисту веб–застосунків відіграє важливу роль в забезпеченні безпеки в Інтернеті. Чим більш розвинена законодавча база в питаннях захисту веб–додатків, тим ефективнішими будуть засоби захисту веб–застосунків в кінцевому підсумку.

Одним з ключових законів, пов'язаних з захистом веб–застосунків, є Регламент про захист персональних даних (GDPR), що вступив в дію в квітні 2018 року. Цей нормативний акт встановлює вимоги щодо збору, зберігання та обробки персональних даних, які повинні дотримуватися всіма організаціями, що здійснюють операції з персональними даними в Європейському Союзі [19].

Крім того, в кожній країні можуть бути національні закони, що регулюють питання захисту веб–додатків. Наприклад, в США діє закон про комп'ютерну злочинність (Computer Fraud and Abuse Act), який передбачає кримінальну відповідальність за злам комп'ютерних систем та веб–застосунків [20].

Крім законодавства, існують також нормативні акти, які визначають вимоги щодо безпеки веб–додатків. Наприклад, OWASP Top 10 є стандартом для виявлення та усунення найбільш поширених вразливостей веб–додатків. Цей стандарт охоплює такі вразливості, як SQL–ін'єкції, кросс–сайт скриптінг та інші.

Законодавство та нормативне регулювання в галузі захисту веб–застосунків можуть відрізнятися в різних країнах і регіонах. Проте, деякі загальні положення і стандарти визначені на міжнародному рівні.

Одним з найважливіших міжнародних стандартів в області захисту веб–додатків є OWASP (**Open Web Application Security Project**). OWASP – це некомерційна організація, яка складається з добровольців з усього світу,

які працюють над покращенням безпеки веб–додатків. OWASP публікує різноманітні матеріали, такі як стандарти та інструменти, які допомагають розробникам, тестувальникам та адміністраторам захищати веб–застосунки від різних загроз [21].

Іншим важливим міжнародним стандартом є ISO 27001 – стандарт з інформаційної безпеки, який визначає вимоги до системи управління інформаційною безпекою (ISMS). Цей стандарт може бути застосований для захисту веб–застосунків, оскільки він встановлює процеси та підходи до ідентифікації, оцінки та зменшення ризиків інформаційної безпеки [22].

Крім того, існують міжнародні стандарти з захисту інформації, такі як ISO 27001, які містять рекомендації з захисту даних та веб–додатків.

У багатьох країнах, включаючи Європейський Союз, існують закони та нормативні акти, які визначають вимоги до захисту особистої інформації, включаючи інформацію, яка обробляється веб–застосунками. Наприклад, у ЄС існує Загальний регламент про захист даних (**GDPR**), який встановлює правила щодо обробки, зберігання та передачі персональних даних.

2.1 Регулювання на законодавчому рівні в Україні

Україна має законодавче регулювання з питань захисту персональних даних та кібербезпеки в цілому. Основним законом є Закон України "Про захист персональних даних", який був прийнятий у 2010 році та відповідає вимогам Європейської конвенції про захист персональних даних [23].

Крім того, у 2018 році було прийнято Закон України "Про основні засади (стратегію) державної кібербезпеки України", який визначає стратегію державної політики в галузі кібербезпеки та встановлює вимоги до забезпечення кібербезпеки в Україні [24].

Закон України "Про електронні довірчі послуги" 2017 р. також встановлює вимоги до забезпечення безпеки та конфіденційності електронних документів та електронних підписів [25].

Крім законодавства, українська влада приймає різноманітні нормативні акти та рекомендації щодо захисту інформації та кібербезпеки. Наприклад, у 2018 році Національний банк України видав "Інструкції щодо захисту інформації в банківській сфері", яка містить вимоги щодо захисту персональних даних та іншої конфіденційної інформації в банківській галузі.

Однак, незважаючи на наявність законодавчого та нормативного регулювання, в Україні відбувається значна кількість кібератак на веб–застосунки, що свідчить про необхідність посилення заходів з кібербезпеки та вдосконалення законодавчої бази у цьому напрямі.

2.2 Регулювання на законодавчому рівні в ЄС

Загальний регламент про захист персональних даних (**General Data Protection Regulation, GDPR**) є законодавчим актом Європейського Союзу, прийнятим у 2016 році, що набув чинності з 25 травня 2018 року. Цей регламент замінив Директиву 95/46 / ЄС і встановлює нові правила для збору, зберігання та обробки персональних даних громадян ЄС.

Основні принципи GDPR:

1. Чітке та зрозуміле повідомлення про збір та обробку даних;
2. Згода з обробкою персональних даних повинна бути вільно надана, конкретна, інформована та визначена дія;
3. Право на заборону обробки та видалення даних;
4. Право на доступ до власних персональних даних;
5. Повідомлення про порушення безпеки даних протягом 72 годин після виявлення.

Для підприємств та організацій, які збирають та обробляють персональні дані, GDPR встановлює вимоги щодо збору та обробки даних, включаючи:

1. Захист даних від несанкціонованого доступу, втрати або крадіжки;
2. Захист даних шляхом шифрування та інших заходів безпеки;
3. Підтримка політики доступу до даних та контролю над доступом до них;
4. Повідомлення про порушення безпеки даних;
5. Проведення оцінки впливу на захист персональних даних;
6. Розробка та введення політики зберігання даних та контролю їх збереження.

У випадку порушення GDPR може бути накладено адміністративні штрафи до 20 млн євро або до 4% річного глобального обороту підприємства за попередній рік, в залежності від тяжкості порушення та його наслідків.

2.3 Регулювання на законодавчому рівні в США

У США це закон про захист персональних даних (HIPAA) для охорони медичної інформації, а також федеральний закон США про захист даних про платіжні картки (PCI DSS).

HIPAA (Health Insurance Portability and Accountability Act) – це американський федеральний закон, прийнятий у 1996 році, який регулює передачу, зберігання та обробку медичної інформації (зокрема, здоров'я та медичних записів пацієнтів) в США [26].

Головною метою HIPAA є забезпечення конфіденційності, цілісності та доступності медичної інформації пацієнтів, зокрема шляхом встановлення стандартів щодо зберігання, передачі та обробки цієї інформації. Згідно з HIPAA, організації, які займаються зберіганням, передачею або обробкою

медичної інформації, повинні приймати відповідні технічні та організаційні заходи для захисту цієї інформації від несанкціонованого доступу, втрати, зміни чи пошкодження.

HIPAA визначає стандарти захисту медичної інформації, а також вимагає від організацій, що обробляють цю інформацію, виконання певних вимог щодо обробки даних, аудиту та звітності, а також установлення практик забезпечення конфіденційності, цілісності та доступності цієї інформації.

За порушення стандартів HIPAA можуть бути нараховані штрафи, які залежать від серйозності порушення та можуть досягати великих сум.

Таким чином, законодавче та нормативне регулювання з питань захисту веб–застосунків в різних країнах може суттєво вплинути на рівень захисту веб–додатків та на підходи до нього.

Наприклад, в ЄС діє загальний регламент про захист персональних даних (GDPR), який вимагає від підприємств захищати персональні дані своїх клієнтів, включаючи дані, що зберігаються в базах даних. Таким чином, компанії, які працюють в ЄС або з клієнтами з ЄС, повинні відповідати високим стандартам захисту персональних даних, а також повідомляти про можливі порушення безпеки.

У США, HIPAA вимагає від організацій охорони здоров'я захищати електронні медичні записи і персональні дані пацієнтів. Це також включає захист від несанкціонованого доступу та перешкоджання втраті даних.

Україна має закон "Про захист персональних даних", який встановлює правила збору, зберігання та обробки персональних даних в Україні. Закон також вимагає від підприємств захищати персональні дані своїх клієнтів.

Отже, законодавче та нормативне регулювання може значно впливати на рівень захисту веб–застосунків. Компанії повинні дотримуватися цих норм, щоб забезпечити високий рівень захисту персональних даних своїх клієнтів та уникнути можливих порушень безпеки.

РОЗДІЛ 3. ОЦІНКА ЕФЕКТИВНОСТІ СУЧАСНИХ МЕТОДІВ ЗАХИСТУ ВЕБ–ЗАСТОСУНКІВ

Оцінка ефективності сучасних методів захисту веб–застосунків залежить від конкретного методу та використаної конфігурації. Однак, деякі загальні методи захисту можуть бути дуже ефективними при запобіганні багатьом загрозам.

Наприклад, використання коректних методів аутентифікації та авторизації, таких як міцна хешування паролів, двофакторна аутентифікація та використання різних рівнів доступу для різних користувачів, може значно зменшити ризик вразливостей типу Broken authentication and session management.

Щодо захисту від ін'єкцій, то ефективним методом є використання параметризованих запитів у базу даних, що дозволяє уникнути виконання зловмисних запитів через введення користувачами шкідливого коду.

Щодо захисту від атак типу DDoS, то використання спеціального програмного забезпечення, такого як firewalls, load balancers та веб–проксі може допомогти у запобіганні чи пом'якшенні наслідків таких атак.

Нарешті, щодо захисту від соціального інженерінгу та фішингу, важливо проводити навчання та тренінги з працівниками, щоб вони могли розрізнити підозрілі запити та повідомлення та уникнути небезпеки. Також можуть використовуватися різні технології захисту, такі як антивірусні програми та системи виявлення вторгнень.

У кожного веб–застосунку свої унікальні вимоги щодо безпеки та можливих загроз, тому важливо проводити аналіз ризиків та підходити до питання захисту індивідуально.

В загальному сучасні методи захисту веб–додатків може виконуватись на різних рівнях. Це залежить від вимог до додатку, та проблеми яку він вирішує.

Рівні веб–додатку:

1. **Рівень мережі**: на цьому рівні встановлюються засоби захисту на рівні мережі, такі як мережеві екрані, виявлення вторгнень, системи запобігання DDoS–атак тощо;
2. **Рівень сервера**: на цьому рівні встановлюються засоби захисту на рівні сервера, такі як захист від SQL–ін'єкцій, захист від кросс–сайт скриптів, захист від фішингу, контроль доступу тощо;
3. **Рівень додатка**: на цьому рівні встановлюються засоби захисту на рівні додатка, такі як перевірка введених даних, обмеження доступу, виявлення та блокування зловживань, моніторинг активності користувачів тощо;
4. **Рівень бази даних**: на цьому рівні встановлюються засоби захисту на рівні бази даних, такі як шифрування даних, обмеження доступу, аудит доступу тощо.

3.1 Захист веб–застосунку на рівні мережі

Захист веб–сайтів на рівні мережі може бути здійснений різними методами. Основною метою захисту на цьому рівні є забезпечення надійної та безпечної передачі даних між клієнтами та серверами.

Один з методів захисту на рівні мережі – це встановлення фаєрволу. Фаєрвол (firewall) є одним з основних інструментів для захисту веб–додатків на рівні мережі. Його основна функція полягає у фільтрації мережевого трафіку та блокуванні небезпечних з'єднань.

Фаєрвол здатний блокувати певні типи трафіку, такі як SQL–ін'єкції, крос–сайт скриптінг (XSS), крос–сайт запити підрядка (CSRF), DDoS–атаки та інші, що зменшує ризики появи таких загроз.

Окрім того, фаєрвол може контролювати трафік до і з веб–додатків, а також моніторити вхідні та вихідні запити. Він також здатний захистити веб–

додатки від атак на рівні мережі, таких як IP–флуди, ARP–атаки, ICMP–атаки та інші.

Фаєрвол може бути реалізований як програмна апаратна засоби, або комбінацією обох. В будь–якому випадку, він є дуже важливим елементом захисту веб–додатків на рівні мережі та допомагає запобігати багатьом загрозам, зокрема, захист від перехоплення даних, ін'єкцій, вірусів та інших атак. Однак, слід зауважити, що фаєрвол не є універсальним рішенням для всіх видів загроз, тому потрібно поєднувати його з іншими методами захисту на рівні мережі та додатку (рис. 3.1)

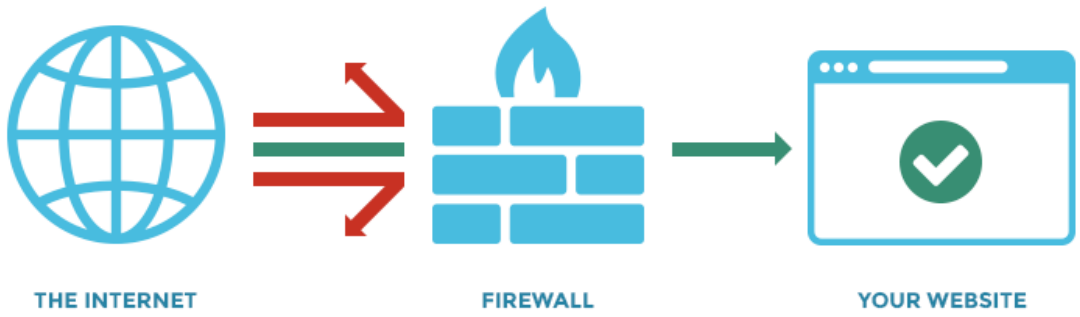


Рисунок 3.1 – Інтеграція фаєрволу для захисту веб–додатку

Інший метод захисту на рівні мережі – це використання віртуальних приватних мереж (VPN). VPN дозволяє створити захищену мережеву інфраструктуру, де всі дані передаються через шифроване з'єднання [27]. Це забезпечує конфіденційність даних та захищає їх від злочинних дій.

VPN (Virtual Private Network) – це технологія, яка забезпечує безпечний канал зв'язку між двома або більше точками через інтернет. Використання VPN може мати позитивний вплив на захист веб додатку та запобігання загрозам.

Основні переваги використання VPN для захисту веб додатків:

1. Захист від перехоплення даних – VPN шифрує весь трафік, що пересилається між веб–сайтом та користувачем, що знижує ризик перехоплення даних з мережі;

2. Забезпечення анонімності – VPN дозволяє користувачеві приховати свою IP–адресу та інші персональні дані, що допомагає запобігти відстеженню та збору даних;

3. Захист від DDoS–атак – VPN може допомогти у захисті веб–сайту від DDoS–атак, зменшивши кількість запитів, що досягають сервера;

4. Захист від шкідливих програм – VPN може допомогти у захисті веб–додатків від шкідливих програм, вірусів та інших загроз;

5. Захист від фішингу – VPN може допомогти у захисті від фішингу, відображаючи віртуальну IP–адресу користувача та забезпечуючи безпечний канал зв'язку з веб–сайтом.

Утім, варто зазначити, що VPN не є універсальним методом захисту веб додатків та має свої обмеження. Наприклад, VPN не може захистити веб–сайт від SQL–ін'єкцій, Cross–Site Scripting (XSS) або інших вразливостей додатків. Тому важливо використовувати комплексний підхід до захисту веб додатків, включаючи використання фаєрволу, мережевих систем інтрузії, антивірусного програмного забезпечення.

Отже, VPN може бути ефективним засобом захисту веб–додатків від деяких загроз, але він не може забезпечити повний захист усіх типів атак.

SSL. SSL (Secure Sockets Layer) або **TLS (Transport Layer Security)** – це криптографічний протокол, який забезпечує захищене з'єднання між клієнтом і сервером в Інтернеті [28]. Це один з найбільш поширених протоколів безпеки, що використовується веб–сайтами для захисту від різних

загроз (рис.2.2).

Рисунок 3.2 – Порівняння схеми з використанням SSL та без нього

Використання SSL у захисті веб–застосунків має наступні переваги:

1. **Конфіденційність**: SSL дозволяє захистити дані, що передаються між користувачем та веб–сайтом від перехоплення іншими особами. Для цього використовується криптографічний протокол, що шифрує дані перед їх відправкою та розшифровує їх на стороні отримувача.

2. **Аутентифікація**: SSL також дозволяє перевірити, що веб–сайт, з якого отримуються дані, є дійсним і відповідає вимогам безпеки. Для цього використовуються цифрові сертифікати, видані надійними організаціями.

3. **Інтегрітет даних**: SSL дозволяє перевірити, що дані, що передаються між користувачем та веб–сайтом, не були змінені під час їх передачі. Для цього використовуються механізми контролю цілісності, такі як код хешування.

4. **Захист від атаки "людина посередині" (Man–in–the–Middle)**: SSL дозволяє запобігти атакам типу "людина посередині", коли зловмисник може перехоплювати і змінювати передачу даних між користувачем та веб–сайтом.

Принцип дії SSL полягає в тому, що веб–сервер встановлює захищене з'єднання з веб–браузером. Це з'єднання використовує криптографічні протоколи для шифрування даних, що передаються між ними. SSL забезпечує конфіденційність, цілісність та автентичність даних, що передаються.

Крім шифрування даних, SSL також забезпечує аутентифікацію веб–сервера та веб–браузера. Коли веб–браузер підключається до веб–сервера, сервер надсилає свій сертифікат SSL, який містить публічний ключ сервера та інформацію про організацію, яка контролює сервер. Веб–браузер перевіряє цей сертифікат, щоб переконатися, що він дійсний та виданий відповідною довіреною організацією. Якщо сертифікат перевірено та є дійсним, то веб–браузер підтверджує автентичність веб–сервера.

Коли дані передаються від сервера до клієнта, вони спочатку шифруються за допомогою SSL на сервері. Потім вони пересилаються через Інтернет у зашифрованому вигляді і дешифруються на стороні клієнта за допомогою ключа шифрування SSL.

Клієнт і сервер обмінюються цифровими сертифікатами, щоб підтвердити свою ідентичність. Це забезпечується за допомогою цифрових підписів, які встановлюються у вигляді сертифікатів на стороні сервера і клієнта.

Коли сертифікати перевірені і ідентичність сторін підтверджена, SSL встановлює безпечне з'єднання між сервером і клієнтом, що забезпечує конфіденційність, цілісність і автентичність передачі даних.

Отже, коли дані потрапляють до користувача, вони є зашифрованими і можуть бути розшифровані лише з допомогою ключа шифрування SSL на стороні клієнта, що забезпечує безпеку передачі даних.

HTTP та HTTPS протокол. HTTPS – це протокол забезпечення безпеки під час передачі даних між веб–сервером та браузером. Він забезпечує захист конфіденційної інформації, такої як логіни, паролі, банківські дані тощо [29].

Принцип дії HTTPS полягає в застосуванні SSL/TLS протоколів для шифрування даних перед їх відправкою. Під час першого підключення до веб–сайту, браузер встановлює захищене з'єднання із сервером, використовуючи SSL/TLS протокол. Цей протокол забезпечує шифрування передаваних даних та перевірку автентичності веб–сайту.

SSL/TLS використовують симетричне та асиметричне шифрування для забезпечення безпеки передачі даних. Ключі шифрування передаються між веб–сервером і браузером, тому злочинці не можуть перехопити ці ключі та декодувати дані.

Після успішної автентифікації сервера, веб–браузер отримує публічний ключ для розшифрування даних, який використовується в подальшому

обміні даними між веб–сервером та браузером. Цей публічний ключ захищений від несанкціонованого доступу та забезпечує конфіденційність даних, переданих через захищене з'єднання HTTPS.

Отже, принцип дії HTTPS полягає в захищенні передачі даних між веб–сервером та браузером за допомогою SSL/TLS протоколів, що забезпечують конфіденційність та цілісність передаваних даних.

Ефективність засобів захисту веб–додатків на рівні мережі залежить від різноманітних факторів, таких як типи атак, характеристики мережі та веб–додатку, якість і налаштування захисних засобів тощо.

Щодо типів атак, то засоби захисту на рівні мережі дозволяють ефективно боротися з багатьма з них, такими як DDoS, перехоплення пакетів, мережеві вразливості тощо. Проте, інші типи атак, наприклад, SQL Injection та Cross–Site Scripting, не можуть бути повністю запобіженні за допомогою захисту на рівні мережі і потребують захисту на більш вищих рівнях.

Щодо характеристик мережі та веб–додатку, то ефективність захисту на рівні мережі залежить від швидкості та якості мережевого з'єднання, пропускної здатності мережі, наявності і правильності налаштування брандмауера та інших засобів захисту.

На якість захисту на рівні мережі впливає також якість і правильність налаштування захисних засобів, таких як брандмауер, вторинний брандмауер, інтрузійна детекція та інші засоби. Відповідно, якщо налаштування не вірне або не відповідає характеристикам мережі та веб–додатку, ефективність захисту може значно зменшитися.

Отже, для досягнення максимальної ефективності захисту веб–додатків на рівні мережі, необхідно враховувати всі ці фактори та налаштовувати захисні засоби відповідно до конкретних характеристик мережі та веб–додатку.

3.2 Захист веб–застосунку на рівні серверу

Захист веб–додатку на рівні серверу включає в себе ряд заходів, які допоможуть запобігти можливим атакам та зберегти цілісність, конфіденційність та наявність даних.

Основні методи захисту веб–додатку на рівні серверу:

1. **Використання більш безпечних протоколів**. Якщо ваш веб–додаток використовує HTTP, рекомендується перехід на більш безпечний протокол HTTPS, який шифрує передачу даних між клієнтом та сервером.

2. **Налаштування файрволу на рівні серверу**. Файрвол дозволяє блокувати небезпечний трафік, фільтрувати певні типи запитів та обмежувати доступ до деяких ресурсів.

3. **Використання сучасних алгоритмів шифрування**. Важливо використовувати сучасні алгоритми шифрування, які не мають відомих вразливостей та запобігають підміні даних.

4. **Захист від DDoS атак**. Налаштування спеціальних засобів захисту від DDoS атак, які дозволяють розпізнавати та блокувати трафік від зловмисників, що намагаються перевантажити сервер.

5. **Налаштування прав доступу**. Налаштування прав доступу до файлів та каталогів сервера, щоб забезпечити обмежений доступ до конфіденційних даних та зберігати цілісність додатку.

6. **Використання віртуальних приватних серверів (VPS)**. Використання VPS може допомогти зменшити ризики зв'язані зі спільним використанням сервера та зберегти веб–додаток від можливих атак на інші сайти, розташовані на цьому сервері.

DDoS–атака є однією з найбільш розповсюджених атак на веб–сайти, і її наслідки можуть бути дуже серйозними [30]. Зазвичай DDoS–атаки здійснюються шляхом перевантаження веб–сервера запитами, що призводить до його падіння.

Основними методами захисту від DDoS–атак на рівні сервера є:

1. Оптимізація веб–сервера: підвищення продуктивності сервера, розширення каналу зв'язку, налаштування оптимальних параметрів сервера для роботи з великою кількістю запитів.

2. Використання спеціальних програмних і апаратних засобів захисту, які можуть відбирати запити, що надходять до сервера, та визначати, які з них є легітимними, а які є атакою.

3. Використання CDN: збільшення мережевого простору засобами CDN може допомогти в розподілі навантаження між серверами і зменшенні ймовірності перевантаження одного конкретного сервера.

4. Налаштування фаєрволу: фаєрвол може відфільтрувати вхідні запити та заборонити підозрілі запити, що можуть бути частиною DDoS–атаки.

5. Налаштування IPS (Intrusion Prevention System): IPS може відфільтрувати вхідні запити та заборонити підозрілі запити, що можуть бути частиною DDoS–атаки. Іншими словами, IPS може виявити та зупинити DDoS–атаку, ще до того, як вона завдасть шкоди серверу.

Ефективність засобів захисту веб–додатків на рівні серверу залежить від різноманітних факторів, включаючи тип застосунку, особливості мережевого середовища, архітектуру системи та інші технічні та організаційні аспекти.

Деякі методи захисту на рівні серверу, такі як використання фаєрволу, IPS та WAF, можуть бути досить ефективними в запобіганні різних типів атак, включаючи DDoS, SQL Injection та Cross–Site Scripting. Однак, ці засоби можуть бути вразливі до певних типів атак, і їхню ефективність може залежати від правильної конфігурації та регулярного оновлення.

Також важливо застосовувати додаткові заходи безпеки, такі як валідації даних на стороні сервера, розсіювання навантаження, моніторинг системи та аналіз журналів. Крім того, ефективність заходів безпеки може

бути покращена за допомогою регулярних аудитів безпеки та пенетраційних тестів для виявлення потенційних вразливостей та забезпечення належної реагування на нові загрози.

Засоби захисту веб–додатків на рівні серверу можуть бути дуже ефективними в захисті від багатьох типів атак, таких як SQL Injection, XSS атаки, DDoS атаки, і т.д. Однак, їх ефективність залежить від правильної налаштування та використання.

Наприклад, для захисту від SQL Injection можна використовувати методи підготовки запитів та параметризованих запитів, а також використовувати бібліотеки для обробки та валідації вхідних даних.

Для захисту від XSS атак можна використовувати методи екранування та фільтрації вхідних даних, а також використовувати Content Security Policy.

Для захисту від DDoS атак можна використовувати різні методи, такі як захист від SYN–флуду, захист від HTTP флуду, захист від DNS атак, захист від атак на рівні мережі, та інші.

Однак, важливо пам'ятати, що найбільш ефективним заходом є комплексний підхід до захисту веб–додатку на різних рівнях, від мережевого до додаткового рівня захисту.

3.3 Захист веб–застосунків на рівні додатку

Захист веб–застосунку на рівні додатку є одним з ключових аспектів забезпечення безпеки. На цьому рівні застосовуються різноманітні методи та підходи для запобігання атакам та захисту від вразливостей, серед найпоширеніших можна виділити наступні:

Валідація вхідних даних – перевірка коректності та допустимості введених користувачем даних, таких як параметри запиту, форми, файлові завантаження та інші. Валідація дозволяє уникнути SQL ін'єкцій, XSS, LDAP ін'єкцій та інших вразливостей, пов'язаних з введенням некоректних даних.

Санітізація HTTP запитів – це процес очищення та перевірки даних, що передаються у HTTP запитах, з метою запобігання вразливостей безпеки, таких як SQL Injection, Cross–site scripting (XSS), Command Injection та інших [31].

Під час валідації, введені дані перевіряються на наявність недопустимих символів, форматування, тип даних, обсяг тощо. Якщо виявляється, що дані не відповідають вимогам, то користувач повідомляється про помилку і вимагається введення коректних даних.

Основні методи валідації даних на рівні додатку включають:

1. Перевірку типу даних;
2. Перевірку наявності недопустимих символів;
3. Перевірку форматування даних;
4. Перевірку обсягу даних;
5. Перевірку на наявність SQL ін'єкцій;
6. Перевірку на наявність XSS ін'єкцій.

Валідація даних є важливим методом захисту веб–додатків від атак з використанням вразливостей на рівні додатку. Правильно налаштована валідація даних може значно зменшити ризик вразливостей та забезпечити безпеку веб–додатків.

Алгоритм виявлення SQL ін'єкцій може бути реалізований за допомогою наступних кроків (рис. 3.3)

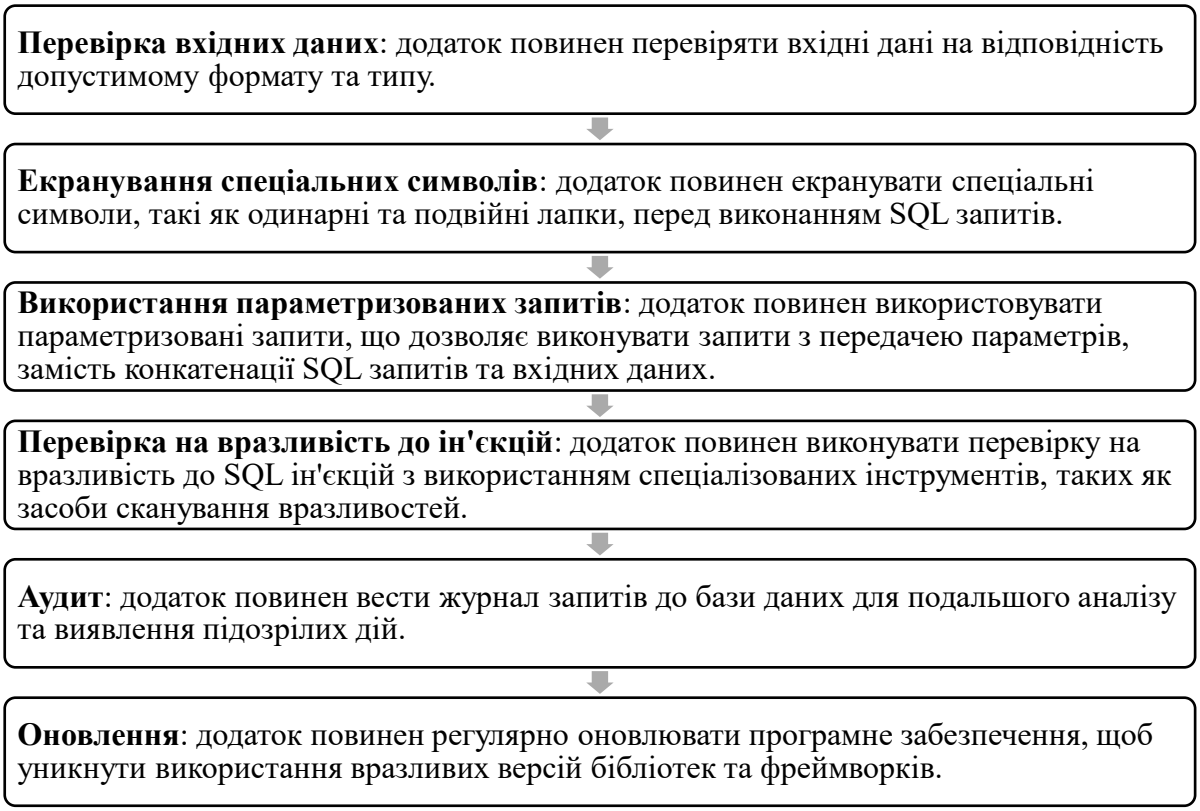


Рисунок 3.3 – Алгоритм виявлення SQL ін'єкцій

Ці кроки допомагають уникнути SQL ін'єкцій та забезпечити безпеку додатку. Однак, важливо розуміти, що захист від SQL ін'єкцій – це постійний процес, оскільки нові вразливості можуть виникати з часом. Тому, потрібно регулярно перевіряти та оновлювати захист додатку.

З точки зору розробки (написання коду) веб–застосунків виявлення та запобігання SQL ін'єкцій можна здійснити за допомогою певних методів та підходів. Ось кілька прикладів:

1. Використання параметризованих запитів: замість використання рядкових конкатенацій для створення SQL запиту, краще використовувати параметризовані запити. Це дозволяє відокремити дані від запиту і уникнути можливості SQL ін'єкції.

2. Використання фільтрації введених даних: перевіряйте введені користувачем дані на наявність недозволених символів, таких як одинарні лапки, двійкові числа та інші.

3. Використання ORM: ORM (Object–Relational Mapping) – це технологія, що дозволяє взаємодіяти з базою даних за допомогою об'єктів, а не напряму через SQL запити. ORM може автоматично генерувати безпечні SQL запити та запобігати SQL ін'єкціям [32].

4. Використання спеціальних інструментів для виявлення SQL ін'єкцій: існують спеціальні інструменти, які допомагають виявляти SQL ін'єкції в коді, наприклад, SQLMap, SonarQube, FindBugs та інші.

Алгоритм перевірки на наявність **XSS** ін'єкцій може бути наступним (рис. 3.4).

Ці методи можуть бути використані як окремо, так і в комбінації для ефективного захисту від XSS–ін'єкцій.

Автентифікація та авторизація. Визначення прав користувачів та їхнього доступу до різних функцій та ресурсів веб–застосунку. Для цього використовуються різноманітні методи, такі як парольна автентифікація, двофакторна аутентифікація, OAuth та інші.

Автентифікація та авторизація є ключовими поняттями у веб–додатках для забезпечення безпеки та контролю доступу до різних ресурсів.

Рисунок 3.4 – **Алгоритм** перевірки на наявність **XSS** ін'єкцій

Автентифікація визначає, чи є користувач валідним і дозволяє йому ввійти до системи. Зазвичай, це включає в себе введення ідентифікатора користувача та пароля. Іноді можуть бути використані інші форми автентифікації, такі як використання біометричних даних або сертифікатів.

Авторизація визначає, які дії дозволені для користувача після входу до системи. Наприклад, які сторінки він може переглядати, які дії він може виконувати на цих сторінках, які дані він може змінювати і т. д. Авторизація зазвичай контролюється різними ролями користувачів, що надають різні рівні доступу до ресурсів.

У веб–додатках зазвичай використовується комбінація автентифікації та авторизації для забезпечення безпеки та контролю доступу до ресурсів.

OAuth2 – це протокол авторизації, що дозволяє користувачам надавати доступ до своїх ресурсів стороннім додаткам, не передаючи при цьому свій логін та пароль (рис. 3.5) [33].

Основними етапами протоколу є:

1. Реєстрація додатку – створення облікового запису додатку та отримання унікального ідентифікатора (client ID) та секретного ключа (client secret).

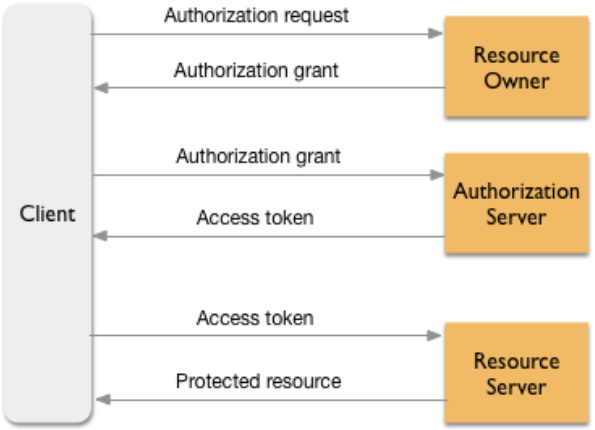
2. Авторизація користувача – перенаправлення користувача на сторінку авторизації, де йому запропонують надати доступ до своїх ресурсів сторонньому додатку. Після погодження, користувач перенаправляється на сторінку додатку з унікальним кодом авторизації.

3. Отримання токену – за допомогою унікального коду авторизації, додаток звертається до авторизаційного сервера та отримує токен доступу.

4. Використання токену – додаток використовує токен доступу для отримання доступу до ресурсів користувача, запитуючи його з авторизаційним сервером при необхідності отримання нового токену.

Рисунок 3.5 – Схема протоколу OAuth2

Алгоритм OAuth2 дозволяє забезпечити безпеку авторизації



користувачів та контролювати доступ до ресурсів веб–застосунку з боку сторонніх додатків.

Захист від атак на сесії. Захист від атак, пов'язаних з підробкою сесій, включаючи Cross–Site Request Forgery (CSRF) та Session Fixation.

Захист від атак на сесії включає в себе декілька методів:

1. **Встановлення короткого таймауту сесії**: встановлення таймауту на сесію може зменшити ризик крадіжки сесії. Якщо користувач не використовує свою сесію протягом певного періоду часу, сесія автоматично закінчується.

2. **Використання HTTPS**: HTTPS забезпечує безпеку передачі даних між веб–сервером і браузером. Це унеможливлює зловмисникам перехоплювати трафік і зчитувати дані, включаючи інформацію про сесії.

3. **Використання cookie HttpOnly**: HttpOnly cookie дозволяє уникнути атак, які спрямовані на отримання доступу до даних з cookies від некоректного використання JavaScript.

4. **Використання двофакторної автентифікації**: Для підвищення рівня безпеки можна використовувати двофакторну автентифікацію, що вимагає від користувача додаткового підтвердження (наприклад, коду з SMS або додаткового пароля).

5. **Захист від CSRF (Cross–Site Request Forgery)** атак полягає в тому, щоб унеможливити зловмиснику виконати певні дії в ім'я авторизованого користувача, який відвідує зловмисний сайт.

Розглянемо детальніше деякі з вище перерахованих методів.

Двофакторна аутентифікація – це процес перевірки ідентифікації користувача, який включає в себе використання двох факторів: щось, що ви знаєте (наприклад, пароль) та щось, що ви маєте (наприклад, фізичний пристрій або програмне забезпечення для генерації одноразових паролів) (рис.3.6) [34].

Рисунок 3.6 – Принцип дії двофакторної аутентифікації

Для прикладу, при вході в онлайн–банк користувач буде запрошений ввести свій логін та пароль (щось, що він знає), а потім він отримає одноразовий код, що генерується мобільним додатком (щось, що він має) із можливістю введення його в систему для підтвердження своєї ідентифікації.

Така система зменшує ризик несанкціонованого доступу до важливих акаунтів, таких як банківські або поштові.

Захист від CSRF (Cross–Site Request Forgery) – це процес захисту веб–застосунку від атак, коли зловмисник намагається використати авторизовану сесію користувача, щоб виконати певну дію на його ім'я без його згоди [35].

Основним методом захисту від CSRF є використання токенів. Токени генеруються сервером під час запиту користувача на сторінку або форму, і повинні бути включені в кожен наступний запит, що здійснює користувач. Кожен токен повинен бути унікальним для кожного запиту і повинен мати обмежений термін дії.

Алгоритм захисту від CSRF (Cross–Site Request Forgery) можна розбити на кілька етапів:

1. **Перевірка HTTP запиту**: сервер повинен перевіряти, що HTTP запит містить токен CSRF.

2. **Генерація та використання токену CSRF**: при відображенні форми аутентифікації користувача сервер повинен згенерувати унікальний токен CSRF, який буде доданий до форми. Кожен раз, коли користувач відправляє дані через форму, сервер повинен перевіряти, чи збігається токен CSRF з тим, що зберігся на сервері.

3. **Використання SameSite cookies**: SameSite cookies є механізмом захисту від CSRF, який дозволяє веб–сайту визначати, чи можуть бути cookie відправлені на інші домени. Використання SameSite cookies може значно зменшити ризик CSRF.

4. **Використання Captcha(опціональний)**: Captcha є методом захисту від CSRF, який вимагає від користувача введення додаткової інформації для підтвердження того, що він людина, а не бот. Captcha може бути використана для запобігання автоматизованому введенню даних.

5. **Використання HTTP заголовків**: сервер може вимагати від браузера включення додаткових HTTP заголовків для підтвердження того, що запит прийшов з відповідного веб–сайту.

6. **Перевірка HTTP відповіді**: сервер може перевіряти HTTP відповідь на наявність небезпечних кодів, що можуть виконатися в браузері користувача (рис.3.7)

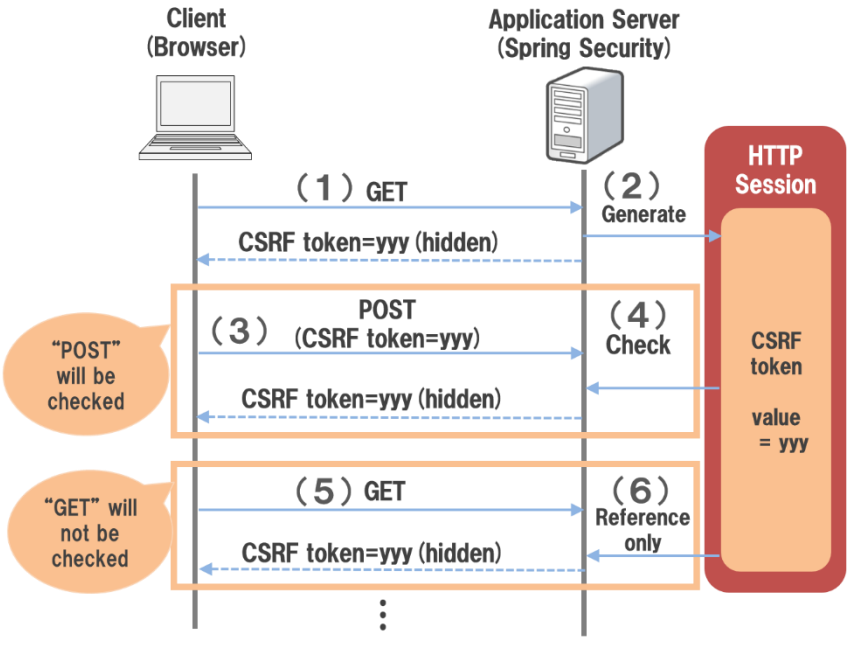


Рисунок 3.7 – Схема роботи HTTP запитів з валідацією CSRF

Ці етапи можуть бути поєднані разом, щоб забезпечити найвищий рівень захисту від CSRF.

Використання бібліотек та фреймворків. Такий підхіж може значно полегшити процес захисту веб–застосунків. Нижче наведені приклади деяких бібліотек та фреймворків, які можуть допомогти в забезпеченні безпеки веб–застосунків:

1. **OWASP ESAPI** – це бібліотека, яка надає різноманітні інструменти для захисту веб–застосунків від різноманітних загроз безпеки, таких як SQL Injection, Cross–Site Scripting (XSS), аутентифікації та авторизації, і т.д. [36].
2. **Spring Security** – це фреймворк для захисту веб–застосунків, який надає інструменти для автентифікації, авторизації та керування сесіями користувачів [37].
3. **Apache Shiro** – це фреймворк для захисту веб–застосунків, який надає інструменти для автентифікації, авторизації, керування сесіями користувачів, захисту від CSRF–атак та інших загроз безпеки [38].
4. **Microsoft ASP.NET Identity** – це фреймворк для захисту веб–застосунків на основі платформи .NET, який надає інструменти для автентифікації, авторизації та керування ролями користувачів [39].
5. **Ruby on Rails Security** – це набір інструментів та практик для захисту веб–застосунків на базі фреймворку Ruby on Rails від різних загроз безпеки [40].
6. **Django Security Middleware** – це бібліотека, яка надає різноманітні інструменти для захисту веб–застосунків на базі фреймворку Django від різних загроз безпеки [41].

Ці інструменти та фреймворки допоможуть розробникам забезпечити безпеку веб–застосунків, але важливо також пам'ятати про необхідність проведення регулярних аудитів безпеки.

Ефективність захисту веб–додатків на рівні додатку полягає в тому, що це дозволяє захистити додаток від вразливостей, що можуть бути використані для атак з боку зловмисників. Захист на рівні додатку дозволяє забезпечити безпеку шляхом валідації та санітізації вхідних даних, автентифікації та авторизації користувачів, використанням механізмів шифрування та підписування даних, захисту від атак на сесії та інших типів атак.

Основні методи захисту на рівні додатку включають в себе належну валідацію та санітизацію даних, використання механізмів автентифікації та авторизації, використання двофакторної аутентифікації, захист від атак на сесії та захист від Cross–Site Scripting (XSS) та Cross–Site Request Forgery (CSRF) атак.

Використання бібліотек та фреймворків також може покращити ефективність захисту веб–додатків на рівні додатку, оскільки ці інструменти зазвичай містять вбудовані механізми захисту від вразливостей, які дозволяють забезпечити безпеку додатка без необхідності написання власного коду захисту. Однак, важливо пам'ятати, що використання бібліотек та фреймворків не є панацеєю і потребує правильної настройки та налагодження для забезпечення максимальної ефективності захисту.

3.4 Захист веб–застосунків на рівні бази даних

Захист веб–додатків на рівні бази даних полягає в захисті самої бази даних та даних, які зберігаються в ній, від різних видів атак, таких як SQL Injection, NoSQL Injection, XSS–атак і т.д. Основні методи захисту на рівні бази даних включають:

1. **Встановлення обмежень доступу до бази даних** – встановлення прав доступу на рівні бази даних для забезпечення обмеженого доступу до даних.

2. **Використання шифрування** – шифрування даних в базі даних та на зв'язку між додатком та базою даних.

3. **Використання моніторингу та журналювання** – встановлення механізмів моніторингу та журналювання на рівні бази даних для виявлення та аналізування вразливостей.

4. **Використання різних інструментів захисту бази даних** – таких як фаєрволи, інструменти виявлення вразливостей та антивірусні програми.

Отже, ефективний захист веб–додатків на рівні бази даних передбачає комплексний підхід та використання різних методів захисту.

Встановлення обмежень доступу до бази даних є важливою складовою захисту веб–додатків. Це означає, що додаток має дозволи лише на ті дії з базою даних, які є необхідними для його коректної роботи, і не має доступу до непотрібних даних.

Один зі способів встановлення обмежень доступу до бази даних – це використання механізму контролю доступу до бази даних. Наприклад, в MySQL це можна зробити, використовуючи спеціальні користувачів з обмеженими правами, які можуть виконувати тільки обмежений набір запитів.

Також можна використовувати технології аудиту доступу до бази даних, щоб відстежувати дії користувачів з базою даних і виявляти можливі порушення безпеки.

Нарешті, для забезпечення безпеки даних в базі даних необхідно правильно налаштувати і захистити саму базу даних від зовнішніх атак. Для цього можна використовувати різні механізми, такі як шифрування даних, автоматичне резервне копіювання, перевірку цілісності даних і так далі.

Використання шифрування на рівні бази даних є ефективним методом захисту веб–додатків від несанкціонованого доступу до даних. Шифрування може бути застосовано на різних рівнях: на рівні поля, таблиці або бази даних в цілому.

Рівні шифрування даних:

1. **Шифрування на рівні поля** полягає в тому, що дані, що зберігаються в конкретному полі таблиці, шифруються за допомогою певного алгоритму, що робить їх незрозумілими для тих, хто намагається отримати до них доступ без належних прав.

2. **Шифрування на рівні таблиці** полягає в тому, що всі дані, що зберігаються в таблиці, шифруються за допомогою певного алгоритму. Цей метод може бути ефективним для захисту від несанкціонованого доступу до всіх даних в таблиці.

3. **Шифрування на рівні бази даних** полягає в тому, що всі дані, що зберігаються в базі даних, шифруються за допомогою певного алгоритму. Цей метод може бути ефективним для захисту від несанкціонованого доступу до всіх даних, що зберігаються в базі даних.

Використання шифрування на рівні бази даних може бути ефективним методом захисту від різних типів атак, включаючи SQL ін'єкції та крадіжку даних. Однак, важливо знати, що шифрування може знизити продуктивність бази даних, тому його необхідно використовувати з обережністю і правильно налаштовувати для максимальної ефективності.

Моніторинг та журналювання бази даних є важливими практиками для забезпечення безпеки веб–додатків. Основні переваги використання цих практик полягають в тому, що вони дозволяють виявляти можливі вразливості та випадки несанкціонованого доступу до даних.

Моніторинг бази даних зазвичай забезпечується за допомогою спеціальних програмних засобів, які відстежують активність користувачів та виконання запитів до бази даних. Це дозволяє вчасно виявляти небезпечні дії та реагувати на них.

Журналювання бази даних полягає в записуванні дій користувачів та виконаних запитів у спеціальний журнал. Цей журнал може бути

використаний для аналізу активності користувачів, виявлення небезпечних дій та відновлення даних у разі їх втрати або пошкодження.

Підсумовуючи, моніторинг та журналювання бази даних є важливими практиками для забезпечення безпеки веб–додатків, оскільки дозволяють виявляти можливі вразливості та небезпечні дії користувачів та реагувати на них вчасно.

Ефективність захисту баз даних визначається комплексом заходів, що забезпечують захист від потенційних загроз. До таких загроз можна віднести SQL–ін’єкції, витоки даних, вразливості автентифікації, а також ризики викрадення даних з мережі.

Для забезпечення ефективного захисту баз даних можна використовувати такі методи:

2. Використання параметризованих запитів, що унеможливлює SQL–ін’єкції.

3. Встановлення обмежень доступу до бази даних для зменшення ризиків витоку даних.

4. Використання шифрування на рівні бази даних для забезпечення конфіденційності даних.

5. Моніторинг та журналювання бази даних для відстеження можливих загроз та виявлення несправностей.

6. Регулярне оновлення баз даних та патчів, що дозволяє запобігти використання вразливостей, які можуть бути виявлені в майбутньому.

Застосування цих методів може допомогти підвищити ефективність захисту баз даних від потенційних загроз та забезпечити високий рівень безпеки веб–додатків.

Отже, з метою максимальної ефективності захисту веб–додатків, потрібно розглядати заходи безпеки на всіх рівнях, а саме на рівні мережі, серверу, додатку та бази даних. Кожен з цих рівнів має власні загрози та захистові методи, і оптимальна стратегія захисту повинна включати заходи

на кожному з них. Для максимальної ефективності, заходи на рівнях повинні доповнювати один одного, і підходити комплексно до захисту веб–додатків.

Оцінка ефективності методів захисту веб–додатків залежить від багатьох факторів, таких як тип додатку, рівень захисту, типи загроз і т.д. Проте деякі загальні висновки можна зробити:

1. **Захист веб–додатків на рівні мережі** є важливим і повинен включати фільтрацію та моніторинг трафіку, виявлення і блокування DDoS атак, використання VPN та фаєрволів.

2. **Захист веб–додатків на рівні серверу** може включати захист від SQL–ін'єкцій, використання обмежень доступу до баз даних та параметризованих запитів, встановлення фаєрволів та захист від Cross–Site Scripting (XSS) атак.

3. **Захист веб–додатків на рівні додатку** повинен включати валідацію вхідних даних, автентифікацію та авторизацію користувачів, захист від Cross–Site Request Forgery (CSRF) атак, використання SSL / TLS та двофакторної аутентифікації.

4. **Захист веб–додатків на рівні бази** даних повинен включати використання шифрування, обмежень доступу до баз даних та параметризованих запитів, а також моніторинг та журналювання баз даних.

Загалом, найбільш ефективний захист веб–додатків досягається за допомогою комплексного підходу, включаючи захист на різних рівнях і використання різноманітних методів. Також важливо регулярно оновлювати захист і вдосконалювати його, оскільки загрози постійно змінюються та еволюціонують.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ МЕТОДІВ ЗАХИСТУ ВЕБ–ЗАСТОСУНКІВ

Реалізація методів захисту веб–застосунків зазвичай відбувається шляхом поєднання різноманітних підходів та технік на різних рівнях захисту (мережевому, серверному, додатковому та баз даних) з використанням відповідних інструментів, бібліотек та фреймворків.

Крім того, важливо забезпечити правильну конфігурацію та налаштування окремих компонентів веб–застосунку, таких як веб–сервери, бази даних, проксі–сервери та інші, а також використовувати найбільш сучасні технології захисту та регулярно проводити аудит безпеки для виявлення потенційних вразливостей та їх виправлення. Комплексний підхід до захисту веб–застосунків дозволяє забезпечити максимальний рівень безпеки та уникнути можливих атак.

4.1 Постановка задачі та вибір технологій веб–застосунку

У Розділі 1 досліджено перелік можливих загроз та вразливостей веб–застосунків. З огляду на загрози, вони ефективні по відношенню до одних типів додатків, та можуть бути не ефективними до інших.

Для даної роботи обрано напрямок, який найкраще може продемонструвати практичне застосування методів захисту веб–додатків – це **додаток для керування власними фінансами**.

Додатки такого типу можуть бути підвищеної уразливості до різноманітних атак з приводу обробки великих обсягів користувацької інформації та її передачі. Деякі загрози, які можуть стати причиною атак на додатки керування власними фінансами, включають:

1. Фішинг;

2. XSS–атаки;
3. Атаки на перехоплення сесій;
4. Атаки на перехоплення даних;
5. Атаки на переповнення буфера;
6. Атаки на мережу.

Оскільки такі додатки містять велику кількість чутливої інформації, такої як паролі, особисті дані та фінансові дані, захист від цих загроз має бути на високому рівні.

Компоненти на, які обрано на глобальному рівні для реалізації захисту веб–додатку:

1. Хмарна платформа Microsoft Azure;
2. Atlas MongoDB cluster
3. .NET/C# (.NET6)
4. .NET Azure SDK
5. Blazor WASM
6. MongoDB Driver

4.2 Архітектура веб–застосунку

Застосування методів захисту веб–додатків повинно бути вбудовано в їх архітектуру, щоб забезпечити найвищий рівень безпеки. Основні принципи архітектури безпеки веб–додатків включають:

1. Розмежування повноважень (privilege separation): цей принцип полягає в тому, що кожен елемент системи повинен мати лише ті права, які необхідні для його функціонування [42]. Наприклад, база даних повинна мати лише ті права, які необхідні для збереження та витягування даних, а не повні права адміністратора.

2. Мінімізація атакованої поверхні (attack surface minimization): цей принцип полягає в тому, щоб зменшити кількість вразливостей, доступних

для атакування [43]. Це можна зробити шляхом відключення не використовуваних функцій, видалення непотрібного коду, застосування найновіших версій програмного забезпечення та усунення потенційних вразливостей.

3. Захист даних (data protection): цей принцип полягає в захисті конфіденційної та особистої інформації, що зберігається на веб–серверах та базах даних [44]. Це може бути зроблено шляхом шифрування даних, застосування механізмів аутентифікації та авторизації, та забезпечення захищеного каналу передачі даних.

4. Захист від вразливостей в коді (code vulnerability protection): цей принцип полягає в тому, щоб перевіряти веб–додатки на вразливості в коді, такі як SQL–ін'єкції та XSS–атаки, та вживати заходів для їх запобігання [45].

При розробці соціальної мережі з точки зору захисту даних, слід притримуватись комплексного підходу до захисту на різних рівнях (мережевому, серверному, додатковому та бази даних).

Крім того, необхідно враховувати наступні рекомендації з точки зору архітектури:

1. Використовувати мікросервісну архітектуру зі зменшенням зв'язків між компонентами системи та забезпеченням гнучкості управління.

2. Забезпечити доступ до даних через відкриті API з контролем доступу і використанням токенів авторизації.

3. Захистити конфіденційні дані, використовуючи шифрування з точністю до поля або рядка.

4. Застосовувати перевірку введення даних на вході та відповідну обробку помилкових запитів з метою запобігання атакам на основі введення даних.

5. Застосовувати перевірку та валідацію введених даних перед збереженням в базу даних.

6. Застосовувати механізми захисту від атак типу DDoS.

7. Застосовувати двофакторну аутентифікацію та відповідні методи захисту від атак на сесії.

8. Проводити регулярні оновлення та патчі для компонентів системи з метою запобігання використання вразливостей системою атакувальниками.

4.3 Опис компонентів веб–застосунку та технологій

Під час реалізації веб застосунку було використано наступні компоненти, які мають безпосередній вплив на захист веб–застосунку(табл. 4.1) (візуалізація у Додатку А):

Таблиця 4.1 – Основні компоненти веб–застосунку та їхні функції захисту

Назва	**Функції захисту**	**Рівень захисту веб–додатку**
Azure Active Directory B2C	– Автентифікація і авторизація – Захист персональних даних – Моніторинг та журналювання – Багаторівнева автентифікація – Захист від DDoS атак – Моніторинг	Рівень **додатку**
Azure Static WebSite	– Протокол HTTPS – Використання CDN – Інтегрований WAF – Захист від DDoS – Моніторинг	Рівень **серверу** та рівень **мережі**
Azure Function	– Підтримка SSL – Автентифікація та авторизація – Обмеження доступу – Моніторинг – Захист від DDoS атак – Захист від SQL ін'єкцій – Захист від Cross–Site Scripting (XSS) – Захист від Cross–Site Request Forgery (CSRF)	Рівень **додатку** та рівень **мережі**
Azure Blob Storage	– Аутентифікація та авторизація – Шифрування даних – Перевірка цілісності даних – Відновлення даних – Захист від DDoS–атак	Рівень **даних**

Назва	**Функції захисту**	**Рівень захисту веб–додатку**
Azure Key Vault	– Зберігання та управління ключами шифрування – Зберігання та управління сертифікатами SSL/TLS – Зберігання та управління секретами – Підтримка доступу до збережених ключів – Надання контролю доступу до ключів	Рівень **даних та додатку**
Azure App Configuration	– Керування доступом – Шифрування – Моніторинг – Аудит – Резервне копіювання та відновлення	
Atlas MongoDB	– Автентифікація – Шифрування даних – Захист від атак – Резервне копіювання – Моніторинг та аналітика	Рівень **даних**

Azure Active Directory B2C (Business to Customer) – це ідентифікаційний сервіс Azure, який дозволяє забезпечувати захист веб–застосунку і даних від несанкціонованого доступу [46]. Деякі функції захисту, які надає Azure Active Directory B2C, включають:

1. **Автентифікація і авторизація**: Azure Active Directory B2C надає можливість налаштування рівня автентифікації та авторизації для ваших додатків. Це дозволяє вам забезпечити доступ до ресурсів тільки після виконання валідації ідентифікатора користувача та його прав доступу.

2. **Захист персональних даних**: Azure Active Directory B2C забезпечує захист персональних даних, включаючи конфіденційну інформацію, таку як імена користувачів та паролі. Ви можете налаштувати рівні захисту та шифрування для захисту даних від несанкціонованого доступу.

3. **Моніторинг та журналювання**: Azure Active Directory B2C надає інструменти моніторингу та журналювання, які дозволяють вам відстежувати активність користувачів і перевіряти наявність можливих загроз безпеці.

4. **Багаторівнева автентифікація**: Azure Active Directory B2C надає можливість використовувати багаторівневу автентифікацію, що дозволяє забезпечувати додатковий рівень захисту для ваших додатків та даних.

5. **Захист від DDoS атак**: Azure Active Directory B2C надає захист від DDoS атак, який дозволяє забезпечити безпеку ваших додатків від перевантаження і недоступності.

Завдяки цьому компоненту вирішується принцип **Azure Static WebSite** надає різноманітні функції захисту, серед яких можна виділити наступні:

1. **HTTPS** – підтримка протоколу HTTPS, який шифрує дані між клієнтом та сервером і забезпечує їх конфіденційність та цілісність.

2. **Access Control** – можливість обмежити доступ до статичного веб–сайту шляхом конфігурування доступу до ресурсів за допомогою Azure Active Directory або за допомогою ключа доступу до SAS (Shared Access Signature).

3. **IP Restrictions** – можливість обмежити доступ до статичного веб–сайту за IP–адресами або діапазонами IP–адрес.

4. **CDN** – можливість використання Content Delivery Network (CDN) для швидкого та надійного доставлення вмісту статичного веб–сайту користувачам з різних частин світу.

5. **WAF** – можливість використання Azure Web Application Firewall (WAF), який забезпечує захист статичного веб–сайту від атак, таких як SQL Injection, Cross–Site Scripting (XSS) і т.д.

6. **DDoS Protection** – можливість використання Azure DDOS Protection, який забезпечує захист статичного веб–сайту від DDOS атак.

7. **Monitoring** – можливість використання Azure Monitor для відстеження та моніторингу статичного веб–сайту.

Azure Function надає ряд функцій захисту, зокрема:

1. **Підтримка SSL**: Azure Function підтримує SSL–шифрування, яке дозволяє захищати передачу даних між веб–додатком та Azure Function.

2. **Автентифікація та авторизація**: Azure Function може інтегруватись з Azure Active Directory для забезпечення механізмів автентифікації та авторизації.

3. **Обмеження доступу**: Azure Function може бути налаштований для обмеження доступу до функцій, щоб запобігти несанкціонованому використанню.

4. **Моніторинг**: Azure Function надає можливість моніторити виконання функцій та виявляти можливі атаки або проблеми з безпекою.

5. **Захист від DDoS атак**: Azure Function може бути захищений від DDoS атак, завдяки інтеграції з Azure DDoS Protection.

6. **Захист від SQL ін'єкцій**: Azure Function може бути налаштований для використання параметризованих запитів до бази даних, що унеможливлює SQL ін'єкції.

7. **Захист від Cross–Site Scripting (XSS)**: Azure Function може бути налаштований для перевірки на наявність XSS ін'єкцій та очищення введених даних перед обробкою.

8. **Захист від Cross–Site Request Forgery (CSRF)**: Azure Function може бути налаштований для захисту від CSRF, зокрема за допомогою використання токенів аутентифікації.

Azure Blob Storage – це хмарне сховище, яке надає можливість зберігати великі об'єми даних у вигляді об'єктів (блоків). Для захисту даних в Blob Storage використовуються наступні функції:

1. **Аутентифікація та авторизація**: Blob Storage підтримує доступ до даних за допомогою токенів SAS (Shared Access Signature), що дозволяє обмежувати доступ до об'єктів за потребою.

2. **Шифрування даних**: Blob Storage підтримує шифрування даних у спокійному стані та в русі. Для захисту даних в русі використовуються протоколи HTTPS або HTTP.

3. **Перевірка цілісності даних**: Blob Storage може перевіряти цілісність даних, що зберігаються, за допомогою контрольної суми.

4. **Відновлення даних**: Blob Storage може зберігати копії даних для забезпечення їх відновлення в разі втрати.

5. **Захист від DDoS–атак**: Blob Storage підтримує захист від DDoS–атак, що забезпечує безпеку даних і зменшує ризик втрати даних.

Atlas MongoDB надає декілька функцій захисту для даних, що зберігаються в базі даних:

1. **Автентифікація**: Atlas MongoDB дозволяє налаштувати автентифікацію користувачів та керувати доступом до бази даних на рівні користувача.

2. **Шифрування даних**: Atlas MongoDB надає можливість застосовувати шифрування на рівні бази даних, що забезпечує конфіденційність даних в разі їх втрати або викрадення.

3. **Захист від атак**: Atlas MongoDB має вбудований захист від SQL–ін'єкцій та інших відомих атак, що можуть бути використані для отримання несанкціонованого доступу до бази даних.

4. **Резервне копіювання**: Atlas MongoDB надає можливість регулярного резервного копіювання бази даних, що забезпечує захист даних в разі їх втрати або пошкодження.

5. **Моніторинг та аналітика**: Atlas MongoDB надає можливість моніторити та аналізувати діяльність бази даних, що дозволяє вчасно виявляти та усувати потенційні загрози безпеці.

Застосування вище перерахованих компонентів має безпосередній захист функцій веб–застосунку.

4.4 Проведення тестування додатку на вразливості

Тестуванню буде підлягати сайт: https://happy–mud–04f7d1203.1.azurestaticapps.net

Для проведення тестів обрано https://pentest–tools.com/ – популярний та відомий сервіс для тестування вразливостей веб–застосунків та мереж. На ньому можна знайти широкий спектр інструментів для виявлення різних видів вразливостей, таких як SQL ін'єкції, XSS атаки, переповнення буфера та інші.

4.4.1 Перевірка вразливостей SSL/TLS.

Після виконання тесту на вказаному вище сайті для пенетраційного тестування отримано наступний результати (рис 4.1):

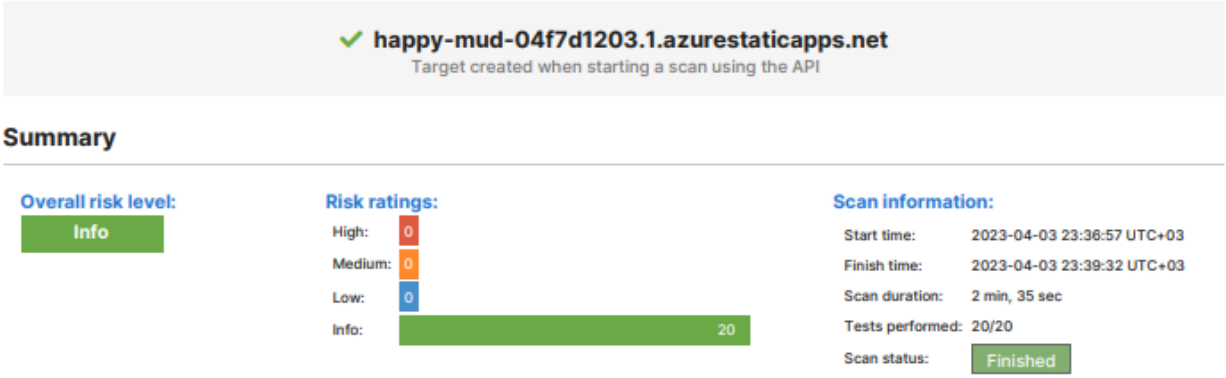


Рисунок 4.1 – Результат перевірки вразливостей SSL/TLS

1. Кількість критичних помилок та вразливостей – 0;
2. Кількість помилок середнього рівня – 0;
3. Кількість помилок низького рівня – 0

4. Повідомлень – 20.

Виконано перевірку наступних пунктів:

1. Довіреність сертифікатів SSL/TLS – всі сертифікати довірені та валідні та не вразливі;
2. Порт публічного доступу до сайту – 443;
3. Жодних вразливостей протоколу.

4.4.2 Перевірка вразливостей мережі.

Після виконання тесту на вказаному вище сайті для пенетраційного тестування отримано наступний результати (рис 4.2):

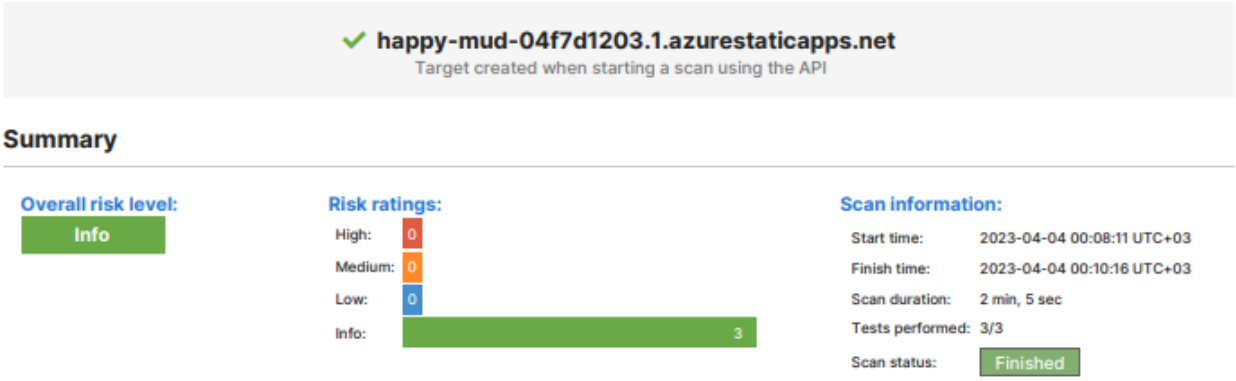


Рисунок 4.2 – Результат тестування вразливостей мережі

1. Кількість критичних помилок та вразливостей – 0;
2. Кількість помилок середнього рівня – 0;
3. Кількість помилок низького рівня – 0
4. Повідомлень – 3.

Таким чином, на рис.4.2, застосунок для пенетраційного тестування не виявив вразливостей мережі.

4.4.3 Перевірка вразливостей веб–додатку.

Після виконання тесту на вказаному вище сайті для пенетраційного тестування отримано наступний результати (рис 4.3):

1. Кількість критичних помилок та вразливостей – 0;
2. Кількість помилок середнього рівня – 0;
3. Кількість помилок низького рівня – 3;
4. Повідомлень – 16.

1. **Відсутність X–Frame–Options заголовку. Опис ризику**: Оскільки заголовок `X–Frame–Options` не надсилається сервером, зловмисник може вставити цей веб–сайт у фрейм iframe стороннього веб–сайту. Маніпулюючи атрибутами зображення iframe, зловмисник міг обманом змусити користувача виконувати клацання мишею в програмі, таким чином виконуючи дії без згоди користувача (наприклад, видалення користувача, підписка на інформаційний бюлетень тощо). Це називається атакою Clickjacking і детально описано тут: https://owasp.org/www–community/attacks/Clickjacking.

Рекомендації по усуненню: *додати HTTP–заголовок `X–Frame–Options` зі значеннями `DENY` або `SAMEORIGIN` до кожної сторінки, для захисту від атак* ***Clickjacking***.

2. **Відсутність Content–Security–Policy. Опис ризику:** Заголовок Content–Security–Policy (CSP) активує механізм захисту, реалізований у веб–браузерах, який запобігає використанню вразливості міжсайтового сценарію (XSS). Якщо цільова програма вразлива до XSS, відсутність цього заголовка робить її легкою для використання зловмисниками.

Рекомендації по усуненню: *налаштувати заголовок* ***Content–Security–Header****, який надсилатиметься з кожною відповіддю HTTP, щоб застосувати певні політики, необхідні програмі*.

3. **Виявлено технології на яких розроблений сайт. Опис ризику**: Зловмисник може використати цю інформацію для здійснення конкретних атак на ідентифікований тип і версію програмного забезпечення.

Рекомендації по усуненню: видалити інформацію, яка дозволяє ідентифікувати програмну платформу, технологію, сервер і операційну систему: заголовки HTTP–сервера, метаінформацію HTML тощо.

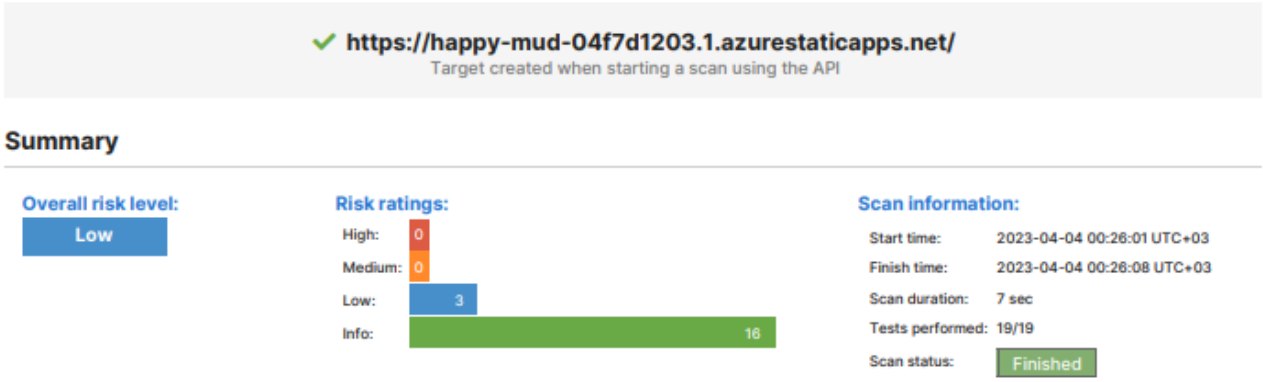


Рисунок 4.3 – Результати тестування вразливостей веб–застосунку

4.4.4 Перевірка вразливостей на XXS атаки

Рисунок 4.4 – Результати тестування вразливостей на XXS атаки

Після виконання тесту на вказаному вище сайті для пенетраційного тестування отримано наступний результати (рис 4.4):

1. Кількість критичних помилок та вразливостей – 0;
2. Кількість помилок середнього рівня – 0;
3. Кількість помилок низького рівня – 0
4. Повідомлень – 3.

Застосунок для пенетраційного тестування не виявив вразливостей мережі.

Таким чином, при побудові веб–застосунку на основі Azure Cloud можна забезпечити високий рівень захисту веб–додатку, використовуючи різноманітні засоби захисту, такі як Azure Active Directory B2C, Azure Static WebSite, Azure Function, Load Balancer, WAF, Azure Key Vault та інші.

Azure надає гнучку архітектуру та розширюваність, що дозволяє забезпечувати ефективний захист веб–застосунку на різних рівнях, включаючи мережевий рівень, рівень сервера, рівень додатку та рівень бази даних.

Проте, слід пам'ятати, що використання Azure не є гарантією повного захисту веб–застосунку від усіх можливих загроз. При розробці веб–додатку необхідно ретельно проаналізувати потенційні загрози та виконати комплекс заходів захисту на всіх рівнях.

За результатами тестування додатку на вразливості виявлено, що додаток має не значні відхилення. Однак, в ході тестування виявлено, що в загальному додаток немає ані вразливостей критичного чи середнього рівня. Для досягнення максимально можливого рівня захисту від загроз необхідно виконати незначні зміни в налаштуваннях додатку чи обновити певні бібліотеки.

ВИСНОВКИ

В ході аналізу потенційних загроз та вразливостей веб–застосунків було виявлено, що безпека веб–застосунків є складною та незавершеною проблемою. Загрози можуть прийти з багатьох джерел, включаючи кібератаки, витоки даних та вразливості в самому застосунку. Для запобігання цим загрозам потрібно застосовувати комплексний підхід до захисту веб–застосунків, що включає в себе методи захисту на різних рівнях (мережі, сервера, додатку, бази даних), а також використання сучасних технологій та практик, таких як шифрування, автентифікація та авторизація, валідація вхідних даних, моніторинг та журналювання подій, захист від CSRF та SQL ін'єкцій. Крім того, важливо дотримуватися законодавчих та нормативних вимог щодо захисту персональних даних, таких як GDPR та HIPAA.

Захист веб–застосунків є надзвичайно важливим в сучасному цифровому світі, і він може бути забезпечений за допомогою різних методів. Оцінка ефективності сучасних методів захисту веб–застосунків показує, що вони є ефективними і можуть запобігати багатьом потенційним загрозам та вразливостям.

Наприклад, використання параметризованих запитів до бази даних унеможливлює SQL–ін'єкції, а використання моніторингу та журналювання баз даних дозволяє вчасно виявляти й усувати можливі атаки на дані.

Також використання комплексного підходу до захисту веб–застосунків, що включає в себе застосування різних методів на кожному рівні (мережі, сервера, додатку, бази даних) може забезпечити максимальний рівень захисту веб–застосунків.

Наприкінці, ефективний захист веб–застосунків потребує постійного вдосконалення та оновлення методів захисту, оскільки загрози та вразливості постійно змінюються і розвиваються.

На основі розробленого додатку на базі Azure та його компонентів можна зробити висновок, що безпека веб–додатку забезпечується за допомогою вбудованих засобів безпеки, що надаються Azure. Наприклад, Azure Active Directory B2C забезпечує безпеку аутентифікації та авторизації користувачів, Azure Static WebSite забезпечує безпеку веб–сайту, Azure Function забезпечує безпеку функцій, а Azure Key Vault забезпечує безпеку ключів та секретів. Застосування цих засобів безпеки дозволяє значно знизити ризики вразливостей та атак на веб–додаток. Однак, необхідно забезпечити правильну конфігурацію та налагодження цих компонентів для забезпечення максимальної безпеки веб–додатку.

Для забезпечення високого рівня безпеки веб–застосунку необхідно виконувати регулярні дії з його захисту, зокрема:

1. Оновлювати всі компоненти та бібліотеки додатку, включаючи оперативну систему, фреймворки, бібліотеки та інші залежності, щоб уникнути вразливостей, які можуть бути використані для атак.
2. Використовувати сучасні методи аутентифікації та авторизації, включаючи двофакторну аутентифікацію, OAuth2 та інші методи захисту.
3. Використовувати захист від XSS, CSRF та інших вразливостей, які можуть бути використані для атак.
4. Застосовувати захист на рівні бази даних, включаючи використання параметризованих запитів, обмеження доступу та шифрування даних.
5. Використовувати моніторинг та журналювання веб–додатку для виявлення потенційних загроз та вразливостей.
6. Регулярно проводити аудит безпеки веб–додатку та його компонентів для виявлення вразливостей та забезпечення високого рівня безпеки.

7. Регулярно навчати користувачів правилам безпеки та виконанню процедур безпеки, таким як складність паролів, двофакторна аутентифікація та інші методи захисту.

Виконання цих дій може допомогти забезпечити високий рівень безпеки веб–застосунку та захистити його від потенційних загроз.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кальченко В. Огляд методів проведення тестування на проникнення для оцінки захисту комп'ютерних систем. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2018. Т. 4. №. 50. С. 109–114.

2. Єфіменко А. О., Трифонова К. О., Зіновський В. Р. Аудит стеганографічної вразливості веб–додатку. З*бірник наукових праць Військового інституту Київського національного університету імені Тараса Шевченка*. 2015. №. 50. С. 187–197.

3. Кива В., Судніков Є., Войтко О. Методи розвідки кіберпростору. *Сучасні інформаційні технології у сфері безпеки та оборони*. 2018. Т. 33. №. 3. С. 45–52.

4. Киричок Р.В. та ін. Проблеми забезпечення контролю захищеності корпоративних мереж та шляхи їх вирішення. *Наукові записки Українського науково–дослідного інституту зв'язку*. 2016. №. 3. С. 48–61.

5. Прокопов В. та ін. Розробка системи виявлення кіберзагроз на основі аналізу даних з веб–ресурсів на мові програмування PYTHON. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2022. Т. 2. №. 68. С. 79–84.

6. Salem A. Intercepting filter approach to injection flaws. *Journal of Information Processing Systems*. 2010. Vol. 6. No 4. P. 563–574.

7. Clarke J. SQL injection attacks and defense. Elsevier, 2009. 761 p.

8. Степанов А., Микитишин А. Дослідження та удосконалення стандартних методів захисту web–додатків. *Матеріали IX науково–технічної конференції «Інформаційні моделі, системи та технології».* 2020. С. 75–76.

9. Su Zh., Wassermann G. The essence of command injection attacks in web applications. *Acm Sigplan Notices*. 2006. Vol. 41. No.1. P. 372–382.

10. Берлог Є., Роговенко А., Дивнич Г. Research of methods of automated search of" SQL injection" type vulnerabilities in web applications. *Технічні науки та технології.* 2022. № 4(30). С. 113–120.

11. Klein A. Blind XPath Injection. Whitepaper from Watchfire, 2005. 11 p.

12. Adhyaru R.P. Techniques for attacking web application security. *International Journal of Information.* 2016. Vol. 6. No.1/2. P. 45–52.

13. Endler D. The evolution of cross site scripting attacks. Technical report, iDEFENSE Labs, 2002. 26 p.

14. Hassan M.M. et al. Broken authentication and session management vulnerability: a case study of web application. *Int. J. Simul. Syst. Sci. Technol*. 2018. Vol. 19. No. 2. P. 1–11.

15. Янко А., Вишинський Р. Система захисту комп'ютерної мережі. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2022. Т. 2. № 68. С. 91–94.

16. Соляник Т.М. Розвиток фішинг–атак та методів боротьби з ними / Т.М. Соляник, Є.Р. Загорецька. *Застосування інформаційних технологій у діяльності правоохоронних органів : зб. матеріалів кругл. столу* (м. Харків, 9 груд. 2020 р.) / МВС України, Харків. нац. ун–т внутр. справ. Харків : ХНУВС, 2020. С. 120–122.

17. Соколов В.Ю., Курбанмурадов Д.М. Методика протидії соціальному інжинірингу на об'єктах інформаційної діяльності. *Науково–технічний журнал" Кібербезпека: освіта, наука, техніка"*. 2018. № 1. С. 6–16.

18. Zhao Y. et al. Design and Evaluation of a Policy–Based Security Routing and Switching System for Data Interception Attacks. *Big Data Computing and Communications: First International Conference, BigCom 2015*, Taiyuan, China, August 1–3, 2015, Proceedings 1. Springer International Publishing, 2015. P. 179–192.

19. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data and repealing Directive 95/46/EC (General Data Protection Regulation). URL: http://data.consilium.europa.eu/doc/document/ST–5419–2016–INIT/en/pdf.

20. Computer Fraud and Abuse Act, 18 U.S.C. § 1030 (2000). URL: https://www.justice.gov/jm/jm–9–48000–computer–fraud.

21. Huang Hsiu–Chuan et al. Web application security: threats, countermeasures, and pitfalls. *Computer*. 2017. Vol. 50. No. 6. P. 81–85.

22. Юзевич В.М., Юзевич Л.В., Крет І.З. Теоретичні та методичні засади кібернетичної безпеки підприємства. *Scientific notes of Lviv University of Business and Law*. 2018. Т. 20. С. 121–126.

23. Закон України. Про захист персональних даних: Закон України від 01.06.2010. Урядовий кур'єр. 2010. № 122.

24. Закон України. Про основні засади забезпечення кібербезпеки України. Відомості Верховної Ради (ВВР), 2017, 45: 2163–19.

25. Закон України. Про електронні довірчі послуги. Відомості Верховної Ради (ВВР), 2017, 45.

26. ACT, Accountability. Health insurance portability and accountability act of 1996. *Public law*. 1996. No. 104. P. 191.

27. Alshalan A., Pisharody S., Huang D. A survey of mobile VPN technologies. *IEEE Communications Surveys & Tutorials*. 2015. Vol. 18. No.2 P. 1177–1196.

28. Dastres R., Soori M. Secure socket layer (SSL) in the network and web security. *International Journal of Computer and Information Engineering*. 2020. Vol. 14. No.10. P. 330–333.

29. HTTPS–crippling attack threatens tens of thousands of Web and mail servers [Електронний ресурс]. URL: http://arstechnica.com/security/2015/05 /https–crippling–attack– threatens–tens–of–thousands–of–web–and–mail–servers/.

30. Баранов С.С. Уразливості комп’ютерних мереж. DDOS атака. Інформаційно–комунікаційні технології в освіті, 2014, 1. URL: https://e–journals.npu.edu.ua/index.php/ikt/article/view/10/pdf.

31. Díaz–Verdejo J.E., et al. A methodology for conducting efficient sanitization of http training datasets. *Future Generation Computer Systems.* 2020. No. 109. P. 67–82.

32. Attaullah Muhammad Usman, Muhammad F. Abrar, Najeeb Ullah, Ibrar A. Shah, Muhammad F. Nadeem. Systematic performance, and Security evaluation of .NET models for accessing database. *VFAST Transactions on Software Engineering*. Volume 9. Number 4. October–December 2021. P. 18–24.

33. The OAuth 2.0 Authorization Framework. URL: https://tools.ietf.org/html/rfc6749.

34. Онищенко, Ю.М., Петрова, К.К. Двофакторна автентифікація, як засіб захисту від несанкціонованого доступу. *Актуальні питання протидії кіберзлочинності та торгівлі людьми: зб. матеріалів Всеукр. наук.–практ. конф.*(м. Харків, 15 листоп. 2017 р.). Харків: ХНУВС, 2017. С. 146–148.

35. Jovanovic N., Kirda E., Kruegel Ch. Preventing cross site request forgery attacks. 2006 Securecomm and Workshops. IEEE, 2006. p. 1–10.

36. Hydara I. et al. Removing Cross–Site Scripting Vulnerabilities from Web Applications using the OWASP ESAPI Security Guidelines. *Indian Journal of Science and Technology*. 2015. Vol. 8. No.30. P. 1–5.

37. Nguyen Q., Baker O. F. Applying Spring Security Framework and OAuth2 To Protect Microservice Architecture API. *J. Softw.* 2019. Vol. 14. No.6. P. 257–264.

38. Javier Ochoa. Security for Java Web Applications Using Apache Shiro. Helsinki Metropolia University of Applied Sciences. November 2014. 74p.

39. Dorrans B. Beginning ASP. NET Security. John Wiley & Sons, 2010. 436 p.

40. Sonewar P.A., Mhetre N.A. A novel approach for detection of SQL Injection and cross site scripting attacks. *2015 International Conference on Pervasive Computing (ICPC). IEEE*. 2015. C. 1–4.

41. Shyam A., Mukesh N. A django based educational resource sharing website: shreic. *Journal of scientific research*. 2020. Vol. 64. No. 1. P. 138–152.

42. Felt A.P. et al. Diesel: Applying privilege separation to database access. *Proceedings of the 6th ACM symposium on information, computer and communications security*. 2011. P. 416–422.

43. Soule N. et al. Quantifying & minimizing attack surfaces containing moving target defenses. *2015 Resilience Week (RWS)*. IEEE, 2015. P. 1–6.

44. Liang G. et al. Distributed blockchain–based data protection framework for modern power systems against cyber-attacks. *IEEE Transactions on Smart Grid*. 2018. Vol. 10. No. 3. P. 3162–3173.

45. Kayaalp M. et al. Branch regulation: Low–overhead protection from code reuse attacks. *ACM SIGARCH Computer Architecture News*. 2012. Vol. 40. No. 3. P. 94–105.

46. Lad S. Identity and Access Management with Azure Active Directory. Azure Security For Critical Workloads: Implementing Modern Security Controls for Authentication, Authorization and Auditing. Berkeley, CA : Apress, 2022. P. 25–63.

47. OWASP. (2017). A1:2017–Injection. March 22, 2023. URL: https://owasp.org/www–project–top–ten/2017/A1_2017–Injection.

48. PortSwigger. (n.d.). OS command injection. March 22, 2023. URL: https://portswigger.net/web–security/os–command–injection.

49. OWASP. (n.d.). Command Injection. March 22, 2023. URL: https://owasp.org/www–community/attacks/Command_Injection.

50. Cloudflare. (n.d.). What is SSL/TLS? March 29, 2023. URL: https://www.cloudflare.com/learning/ssl/what–is–ssl/.

51. Moodle. (n.d.). OAuth 2 authentication. March 29, 2023. URL: https://docs.moodle.org/401/en/OAuth_2_authentication.

52. WebSan Solutions Inc. (n.d.). Microsoft Dynamics 365 Business Central OAuth2 Protocol. March 29, 2023. URL: https://www.websan.com/training/blog/business–central–oauth2–protocol.

53. Google Cloud. (n.d.). Introduction to OAuth. March 29, 2023. URL: https://cloud.google.com/apigee/docs/api–platform/security/oauth/oauth–introduction.

54. Imperva. (n.d.). Two–Factor Authentication (2FA). March 29, 2023. URL: https://www.imperva.com/learn/application–security/2fa–two–factor–authentication/.

55. Криворучко, Є. Огляд методів захисту web–застосунків від вразливостей типу CSRF (міжсайтова підробка запитів). *Системні технології*. 2017. № 3(106). URL: http://nbuv.gov.ua/UJRN/Syst_tech_2017_3_3.

56. Горбенко, С. В., & Ковальчук, І. В. (2019). Оцінювання технологій захисту веб–додатків від атак типу SQL Injection. *2019 IEEE 9th International Conference on Advanced Computer Information Technologies (ACIT)*. IEEE, 2019. С. 281–285.

57. Майстровський О.В., Шаповал Н.І. Методи та засоби виявлення та запобігання атакам на веб–додатки. *Системи обробки інформації*. 2020., №1 (161). С. 87–92.

58. Руденко О.О., Єрко О.В. Застосування машинного навчання для виявлення вразливостей у веб–додатках. *Проблеми телекомунікацій та інформаційної безпеки*. 2019. № 2 (37). С. 85–95.

59. Хорошевський В.В., Шаповал Н.І. Оцінка безпеки веб–додатків на основі тестування вразливостей. *Системи обробки інформації*. 2019. № 4(156). С. 137–141.

60. Шаповал Н.І., Хорошевський В.В. Методика тестування веб–додатків на вразливості за допомогою засобів автоматизації тестування. *Системи обробки інформації.* 2020. №2 (162). С. 147–152.

61. Razzaq A. et al. Semantic security against web application attacks. *Information Sciences.* 2014. No. 254. P. 19–38.

62. Kaur D., Kaur P. Empirical analysis of web attacks. *Procedia Computer Science*. 2016. No. 78. P. 298–306.

63. Ben–Asher N., Gonzalez C. Effects of cyber security knowledge on attack detection. *Computers in Human Behavior*. 2015. No. 48. P. 51–61.

64. Morgan D. Web application security–SQL injection attacks. *Network security*. 2006. No. 4. P. 4–5.

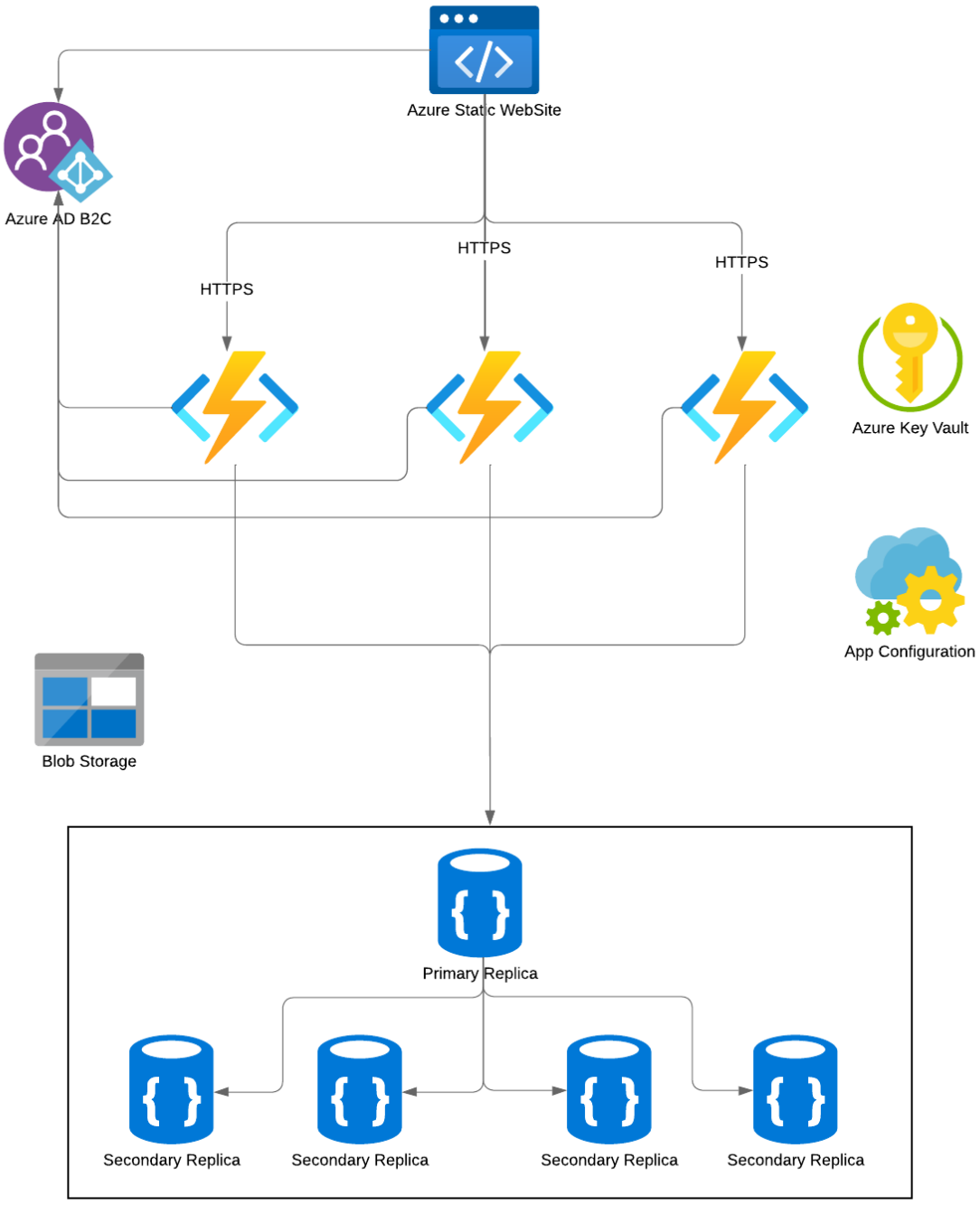
65. Яремчук К., Воскобойников Д., Мелкозьорова О. Сучасні загрози та способи забезпечення безпеки веб–застосунків. *Комп'ютерні науки та кібербезпека*. 2022. № 2. С. 28–34.

66. Слободянюк О., Хаханова А., Комолов Д. Безпека Інтернет ресурсів: аналіз розповсюдженості загроз та технології захисту. *Радиоэлектроника и информатика*. 2018. № 2. С. 30–34.

67. Жилін А., Дівіцький А., Козачок А. Проблематика захисту інформаційних ресурсів при використанні хмарних технологій. *Information Technology and Security: Ukrainian research papers collection*. 2019. Vol. 7. Iss. 2 (13). С. 171–180.

ДОДАТКИ

ДОДАТОК А. Архітектурна схема веб–застосунку



ДОДАТОК Б. Детальні результати тестування веб–застосунків на вразливості

Додаток Б.1. Результати тестування на вразливості веб–застосунку

Pentest Tools

Website Vulnerability Scanner Report (Light)

Unlock the full capabilities of this scanner

See what the FULL scanner can do

Perform in-depth website scanning and discover high risk vulnerabilities.

Testing areas	Light scan	Full scan
Website fingerprinting	✔	✔
Version-based vulnerability detection	✔	✔
Common configuration issues	✔	✔
SQL injection	–	✔
Cross-Site Scripting	–	✔
Local/Remote File Inclusion	–	✔
Remote command execution	–	✔
Discovery of sensitive files	–	✔

✔ **https://happy-mud-04f7d1203.1.azurestaticapps.net/**
Target created when starting a scan using the API

Summary

Overall risk level: Low

Risk ratings:
High: 0
Medium: 0
Low: 3
Info: 16

Scan information:
Start time: 2023-04-04 00:26:01 UTC+03
Finish time: 2023-04-04 00:26:08 UTC+03
Scan duration: 7 sec
Tests performed: 19/19
Scan status: Finished

Findings

Missing security header: X-Frame-Options

CONFIRMED

URL	Evidence
https://happy-mud-04f7d1203.1.azurestaticapps.net/	Response headers do not include the HTTP X-Frame-Options security header

Details

Risk description:
Because the `X-Frame-Options` header is not sent by the server, an attacker could embed this website into an iframe of a third party website. By manipulating the display attributes of the iframe, the attacker could trick the user into performing mouse clicks in the application, thus performing activities without user consent (ex: delete user, subscribe to newsletter, etc). This is called a Clickjacking attack and it is described in detail here:
https://owasp.org/www-community/attacks/Clickjacking

Recommendation:

We recommend you to add the `X-Frame-Options` HTTP header with the values `DENY` or `SAMEORIGIN` to every page that you want to be protected against Clickjacking attacks.

References:
https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Missing security header: Content-Security-Policy

CONFIRMED

URL	Evidence
https://happy-mud-04f7d1203.1.azurestaticapps.net/	Response headers do not include the HTTP Content-Security-Policy security header

Details

Risk description:
The Content-Security-Policy (CSP) header activates a protection mechanism implemented in web browsers which prevents exploitation of Cross-Site Scripting vulnerabilities (XSS). If the target application is vulnerable to XSS, lack of this header makes it easily exploitable by attackers.

Recommendation:
Configure the Content-Security-Header to be sent with each HTTP response in order to apply the specific policies needed by the application.

References:
https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy

Classification:
CWE : CWE-693
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Server software and technology found

UNCONFIRMED

Software / Version	Category
Microsoft ASP.NET	Web frameworks
Blazor	Web frameworks
Google Font API	Font scripts
HSTS	Security

Details

Risk description:
An attacker could use this information to mount specific attacks against the identified software type and version.

Recommendation:
We recommend you to eliminate the information which permits the identification of software platform, technology, server and operating system: HTTP server headers, HTML meta information, etc.

References:
https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/01-Information_Gathering/02-Fingerprint_Web_Server.html

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Security.txt file is missing

CONFIRMED

URL
Missing: https://happy-mud-04f7d1203.1.azurestaticapps.net/.well-known/security.txt

Details

Risk description:
We have detected that the server is missing the security.txt file. There is no particular risk in not creating a valid Security.txt file for your server. However, this file is important because it offers a designated channel for reporting vulnerabilities and security issues.

Recommendation:
We recommend you to implement the security.txt file according to the standard, in order to allow researchers or users report any security issues they find, improving the defensive mechanisms of your server.

References:
https://securitytxt.org/

Classification:
OWASP Top 10 - 2013 : A5 - Security Misconfiguration
OWASP Top 10 - 2017 : A6 - Security Misconfiguration

Website is accessible.

Nothing was found for vulnerabilities of server-side software.

Nothing was found for client access policies.

Nothing was found for robots.txt file.

Nothing was found for use of untrusted certificates.

Nothing was found for enabled HTTP debug methods.

Nothing was found for secure communication.

Nothing was found for directory listing.

Nothing was found for missing HTTP header - Strict-Transport-Security.

Nothing was found for missing HTTP header - X-XSS-Protection.

Nothing was found for missing HTTP header - X-Content-Type-Options.

Nothing was found for missing HTTP header - Referrer.

Nothing was found for domain too loose set for cookies.

Nothing was found for HttpOnly flag of cookie.

Nothing was found for Secure flag of cookie.

Scan coverage information

List of tests performed (19/19)

- Checking for website accessibility...
- Checking for missing HTTP header - X-Frame-Options...
- Checking for missing HTTP header - Content Security Policy...
- Checking for website technologies...
- Checking for vulnerabilities of server-side software...
- Checking for client access policies...
- Checking for robots.txt file...
- Checking for absence of the security.txt file...
- Checking for use of untrusted certificates...
- Checking for enabled HTTP debug methods...
- Checking for secure communication...
- Checking for directory listing...
- Checking for missing HTTP header - Strict-Transport-Security...
- Checking for missing HTTP header - X-XSS-Protection...
- Checking for missing HTTP header - X-Content-Type-Options...
- Checking for missing HTTP header - Referrer...
- Checking for domain too loose set for cookies...
- Checking for HttpOnly flag of cookie...
- Checking for Secure flag of cookie...

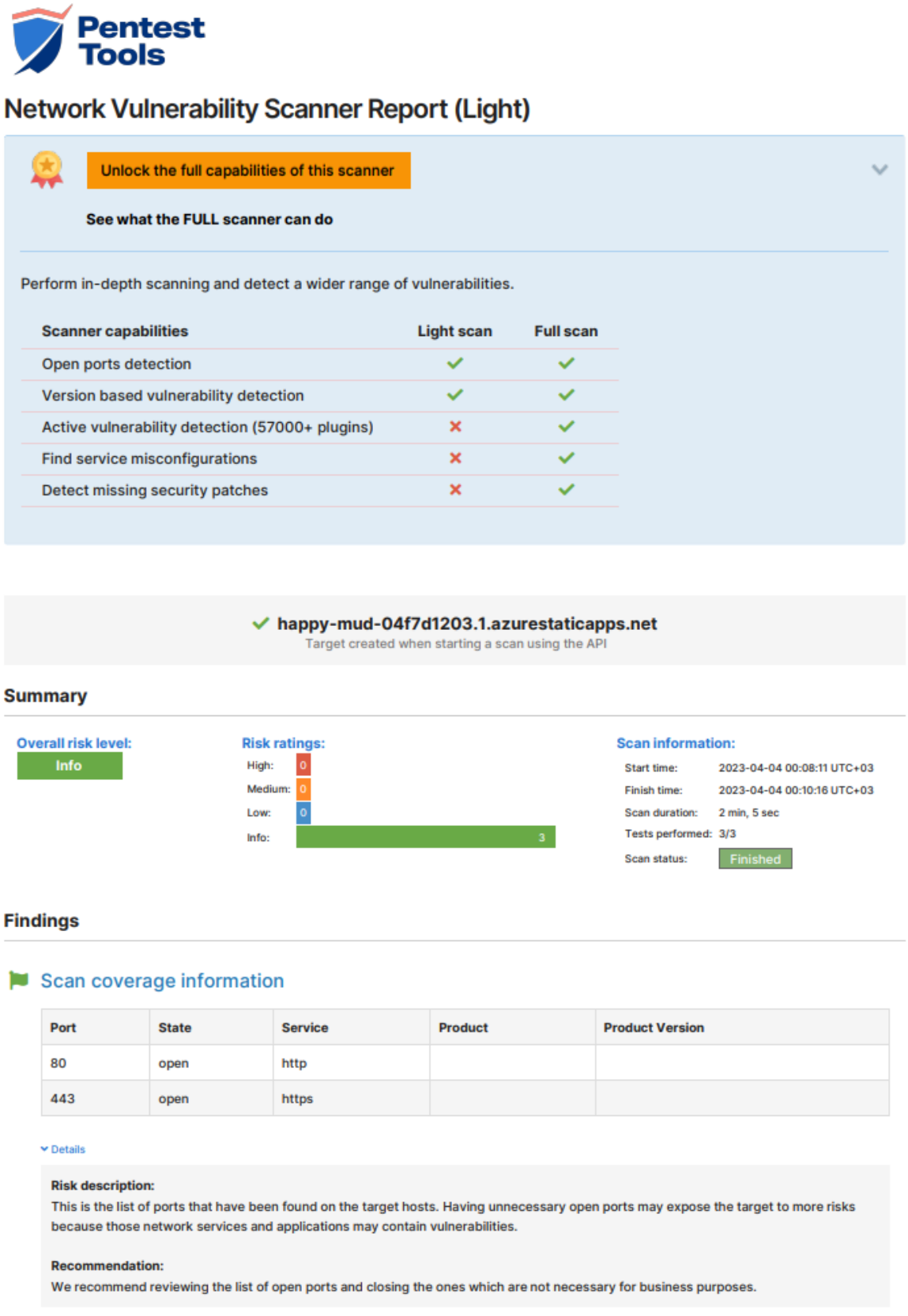
Scan parameters

Website URL:	https://happy-mud-04f7d1203.1.azurestaticapps.net/
Scan type:	Light
Authentication:	False

Scan stats

Unique Injection Points Detected:	1
URLs spidered:	1
Total number of HTTP requests:	9
Average time until a response was received:	52ms

Додаток Б.2. Результати тестування на вразливості мережі



Pentest Tools

Network Vulnerability Scanner Report (Light)

Unlock the full capabilities of this scanner

See what the FULL scanner can do

Perform in-depth scanning and detect a wider range of vulnerabilities.

Scanner capabilities	Light scan	Full scan
Open ports detection	✓	✓
Version based vulnerability detection	✓	✓
Active vulnerability detection (57000+ plugins)	✗	✓
Find service misconfigurations	✗	✓
Detect missing security patches	✗	✓

happy-mud-04f7d1203.1.azurestaticapps.net

Target created when starting a scan using the API

Summary

Overall risk level: Info

Risk ratings: High: 0, Medium: 0, Low: 0, Info: 3

Scan information:

Start time:	2023-04-04 00:08:11 UTC+03
Finish time:	2023-04-04 00:10:16 UTC+03
Scan duration:	2 min, 5 sec
Tests performed:	3/3
Scan status:	Finished

Findings

Scan coverage information

Port	State	Service	Product	Product Version
80	open	http		
443	open	https		

Details

Risk description:
This is the list of ports that have been found on the target hosts. Having unnecessary open ports may expose the target to more risks because those network services and applications may contain vulnerabilities.

Recommendation:
We recommend reviewing the list of open ports and closing the ones which are not necessary for business purposes.

No vulnerabilities found
port 80

No vulnerabilities found
port 443

Scan coverage information

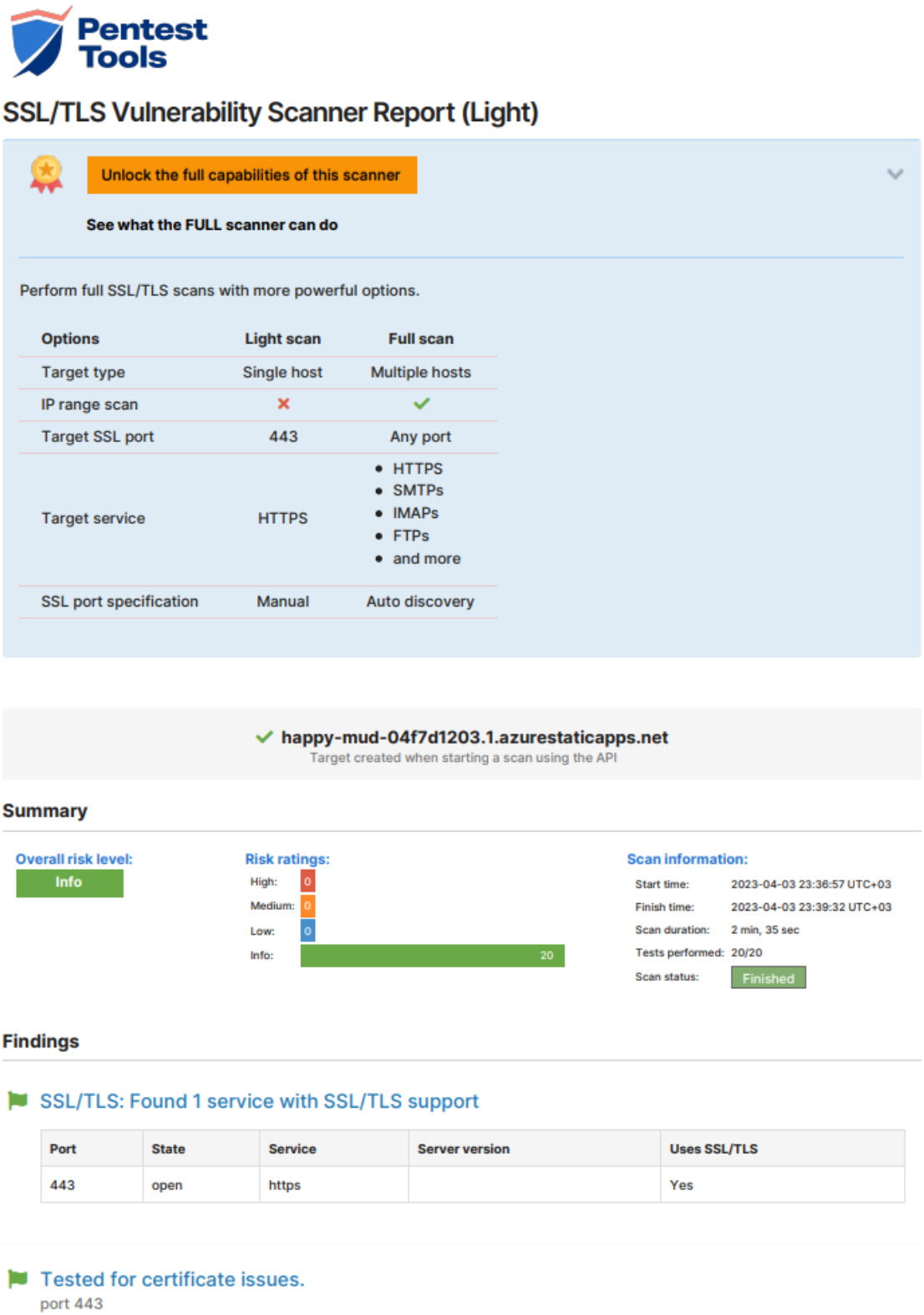
List of tests performed (3/3)

- Running port discovery phase...
- Scanning for vulnerabilities on port 80
- Scanning for vulnerabilities on port 443

Scan parameters

Target:	happy-mud-04f7d1203.1.azurestaticapps.net
Scan type:	Light
Check alive:	True
Extensive modules:	False
Protocol type:	Tcp
Ports to scan:	Top 100 ports

Додаток Б.3. Результати тестування на вразливості SSL/TLS



Pentest Tools

SSL/TLS Vulnerability Scanner Report (Light)

Unlock the full capabilities of this scanner

See what the FULL scanner can do

Perform full SSL/TLS scans with more powerful options.

Options	Light scan	Full scan
Target type	Single host	Multiple hosts
IP range scan	✗	✓
Target SSL port	443	Any port
Target service	HTTPS	• HTTPS • SMTPs • IMAPs • FTPs • and more
SSL port specification	Manual	Auto discovery

happy-mud-04f7d1203.1.azurestaticapps.net

Target created when starting a scan using the API

Summary

Overall risk level: Info

Risk ratings: High: 0, Medium: 0, Low: 0, Info: 20

Scan information:

Start time: 2023-04-03 23:36:57 UTC+03

Finish time: 2023-04-03 23:39:32 UTC+03

Scan duration: 2 min, 35 sec

Tests performed: 20/20

Scan status: Finished

Findings

SSL/TLS: Found 1 service with SSL/TLS support

Port	State	Service	Server version	Uses SSL/TLS
443	open	https		Yes

Tested for certificate issues.

port 443

Certificate number: #1
Issuer: Microsoft Azure TLS Issuing CA 02 (Microsoft Corporation from US)
Signature: SHA384 with RSA
Serial number: 33008A732935ECB01262EC27D40000008A7329

SSL/TLS: Certificate is trusted
port 443

The domain has been found among Subject Alternate Names (SAN) or is the Common Name (CN) itself.
Therefore, it is considered protected by the certificate.

The Server Name Indication (SNI) has also been found. SNI is an extension to the TLS protocol that allows a client or browser to indicate which hostname it is trying to connect to at the start of the TLS handshake.
This allows the server to present multiple certificates on the same IP address and port number.

SSL/TLS: Certificate Chain of Trust is valid
port 443

SSL/TLS: Certificate is Valid
port 443

SSL/TLS: Certificate Authority Issuer is valid
port 443

Tested for SSL/TLS vulnerabilities
port 443

SSL/TLS: Not vulnerable to Heartbleed
port 443

SSL/TLS: Not vulnerable to CCS Injection
port 443

SSL/TLS: Not vulnerable to Ticketbleed
port 443

SSL/TLS: Not vulnerable to ROBOT
port 443

SSL/TLS: Not vulnerable to Secure Renegotiation
port 443

SSL/TLS: Not vulnerable to CRIME
port 443

SSL/TLS: Not vulnerable to POODLE
port 443

SSL/TLS: Not vulnerable to SWEET32
port 443

SSL/TLS: Not vulnerable to FREAK
port 443

SSL/TLS: Not vulnerable to DROWN
port 443

SSL/TLS: Not vulnerable to LOGJAM
port 443

SSL/TLS: Not vulnerable to BEAST
port 443

SSL/TLS: Not vulnerable to RC4
port 443

Scan coverage information

List of tests performed (20/20)

- Checking if SSL/TLS is supported on port 443...
- Checking the certificate on port 443...
- Checking if the certificate is trusted...
- Checking for the certificate chain of trust...
- Checking if the certificate is expired...
- Checking for Certificate Authority Issuer...
- Checking for SSL/TLS vulnerabilities on port 443...
- Scanning for HEARTBLEED on port 443
- Scanning for CCS on port 443
- Scanning for TICKETBLEED on port 443
- Scanning for ROBOT on port 443
- Scanning for SECURE_RENEGO on port 443
- Scanning for CRIME_TLS on port 443
- Scanning for POODLE_SSL on port 443
- Scanning for SWEET32 on port 443
- Scanning for FREAK on port 443
- Scanning for DROWN on port 443
- Scanning for LOGJAM on port 443
- Scanning for BEAST on port 443
- Scanning for RC4 on port 443

Scan parameters

Target:	happy-mud-04f7d1203.1.azurestaticapps.net
Port:	443
Auto Detect SSL/TLS:	false

Додаток Б.4. Результати тестування на вразливості XSS атак

Pentest Tools

XSS Scanner Report (Light)

Unlock the full capabilities of this scanner

See what the FULL scanner can do

Perform a full XSS assessment of your website.

Scanner capabilities	Light scan	Full scan
Spider max URLs	20	500
Spider max duration	1 minute	15 minutes
Active scan max duration	2 minutes	30 minutes

✓ https://happy-mud-04f7d1203.1.azurestaticapps.net

Summary

Overall risk level:
Info

Risk ratings:
High: 0
Medium: 0
Low: 0
Info: 3

Scan information:
Start time: 2023-04-04 00:44:24 UTC+03
Finish time: 2023-04-04 00:44:57 UTC+03
Scan duration: 33 sec
Tests performed: 3/3
Scan status: Finished

Findings

Spider results

URL	Method	Parameters
https://happy-mud-04f7d1203.1.azurestaticapps.net	GET	Headers: User-Agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/108.0.0.0 Safari/537.36

Website is accessible.

Nothing was found for Cross-Site Scripting.

Scan coverage information

List of tests performed (3/3)

1 / 2

- ✔ Checking for website accessibility...
- ✔ Spidering target...
- ✔ Checking for Cross-Site Scripting...

Scan parameters

Website URL:	https://happy-mud-04f7d1203.1.azurestaticapps.net
Scan type:	Light
Authentication:	False

Scan stats

Unique Injection Points Detected:	1
URLs spidered:	1
Total number of HTTP requests:	29
Average time until a response was received:	581ms