Лабораторна робота №18(6.4.7)

Атаки впровадження (Injection Attacks)

Хід роботи:

Частина 1: Експлуатація вразливості SQL Injection на DVWA

Крок 1: Підготовка DVWA для експлойту SQL injection

Завдання: Налаштуйте DVWA на низький рівень безпеки

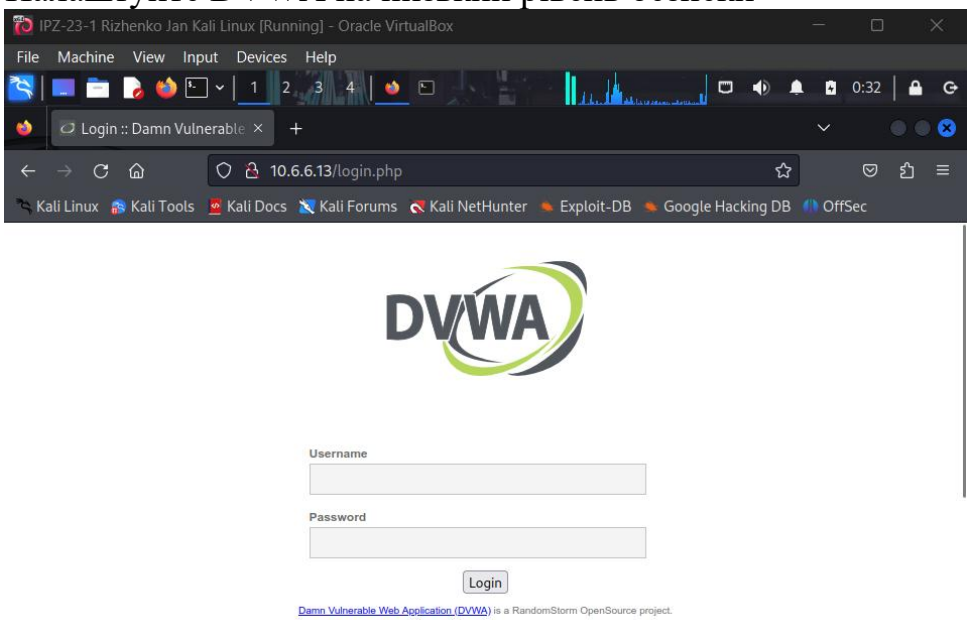

Рис. 1. Сторінка входу DVWA з полями для автентифікації

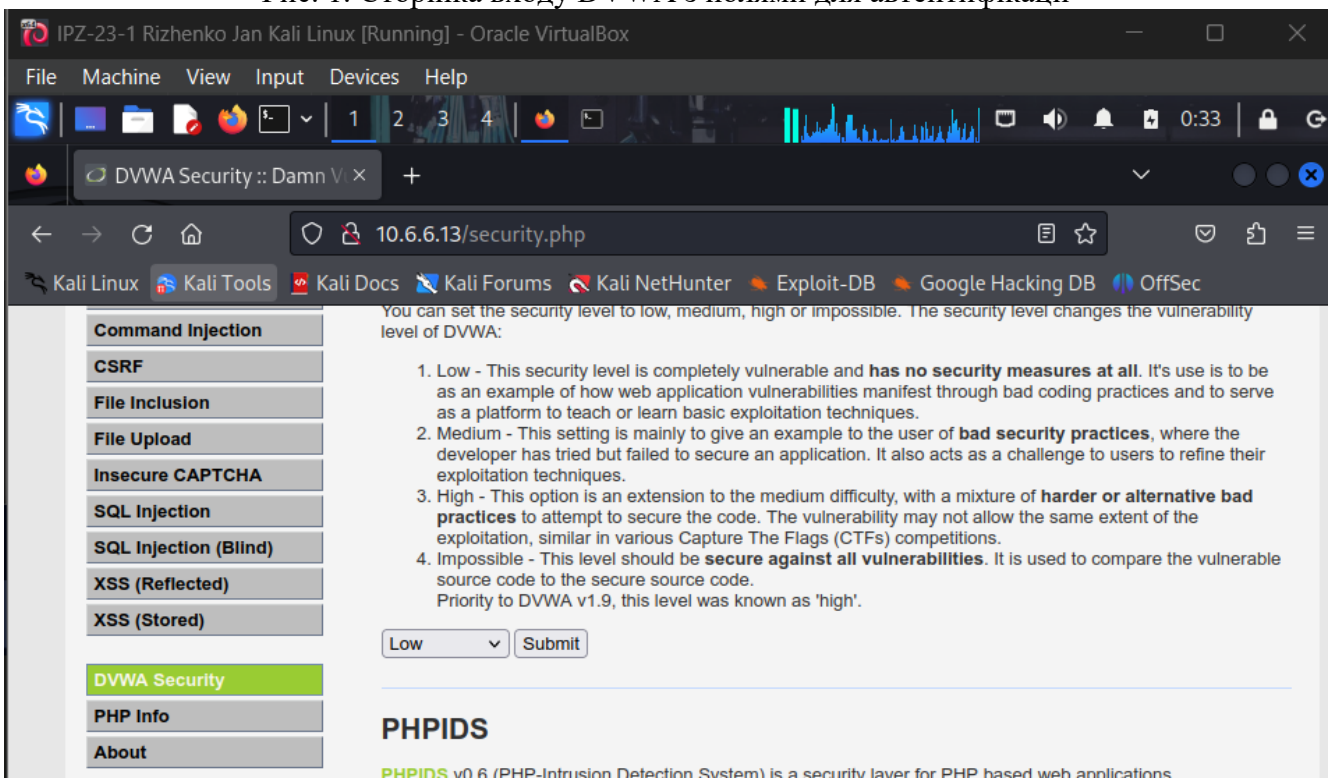

Рис. 2. Налаштування рівня безпеки DVWA на Low для тестування вразливостей

Крок 2: Перевірка наявності вразливості SQL injection
Завдання: Перевірте, чи дозволяє DVWA виконання SQL команд

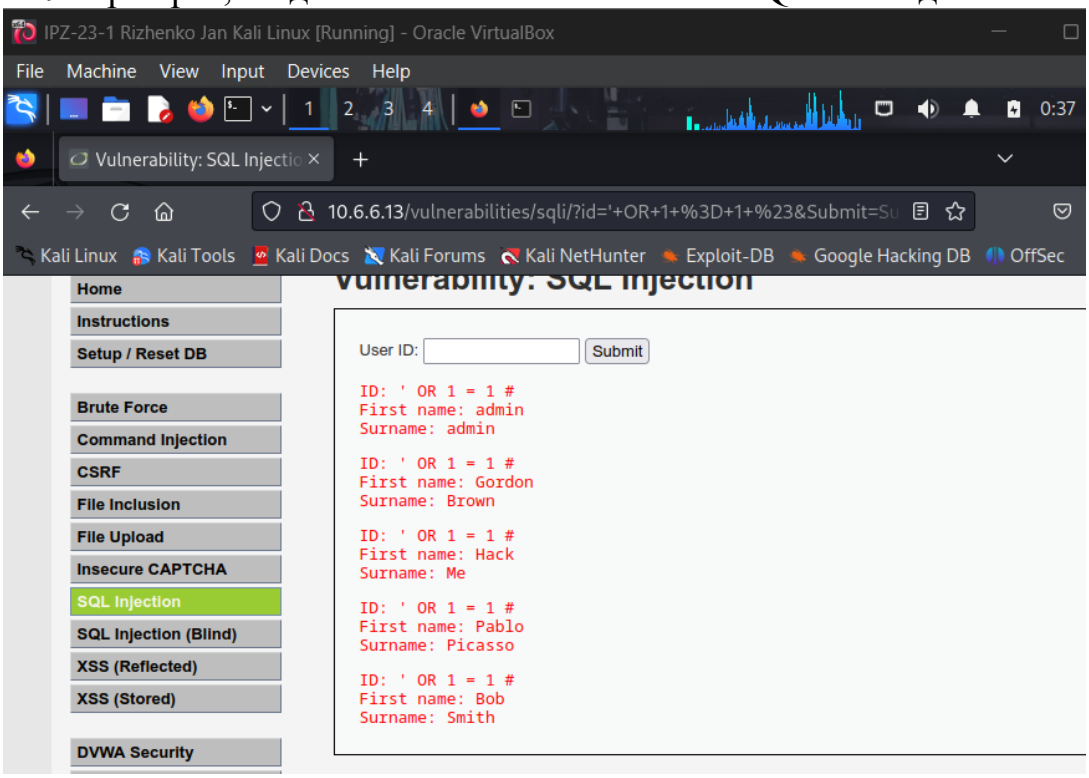

Рис. 3. Успішна SQL injection атака з виведенням всіх записів користувачів

Крок 3: Визначення кількості полів у запиті
Завдання: Дізнайтесь структуру SQL запиту через ORDER BY

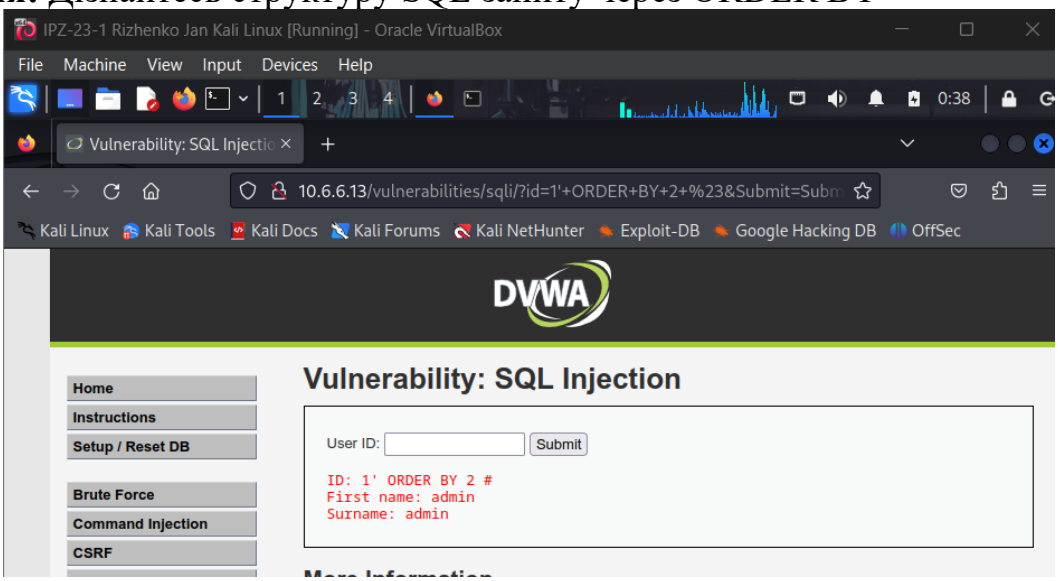

Рис. 4. Тестування структури запиту через ORDER BY для визначення кількості полів

ORDER BY сортує результати за вказаним номером колонки. Якщо номер більший за кількість полів у запиті, виникає помилка.

Крок 4: Визначення версії DBMS
Завдання: Отримайте інформацію про версію системи управління базами даних

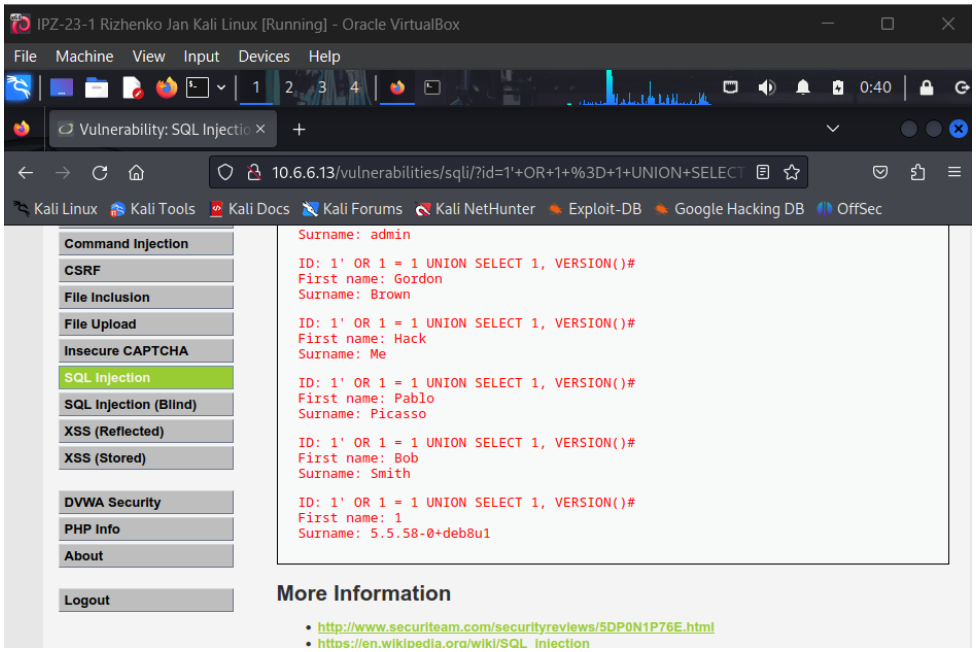

Рис. 5. Визначення версії MySQL через UNION SELECT та функцію VERSION()

UNION об'єднує результати двох SELECT запитів. Кількість та тип полів має співпадати, тому використовуємо SELECT 1, VERSION() (2 поля).

Крок 5: Визначення імені бази даних

Завдання: Дізнайтесь назву поточної бази даних

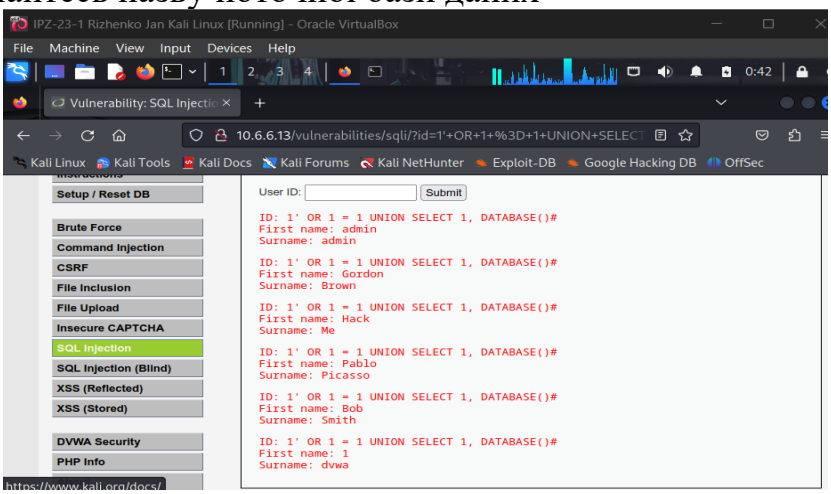

Рис. 6. Визначення імені бази даних через функцію DATABASE()

Крок 6: Отримання імен таблиць з бази dvwa

Завдання: Переліччіть всі таблиці в базі даних dvwa

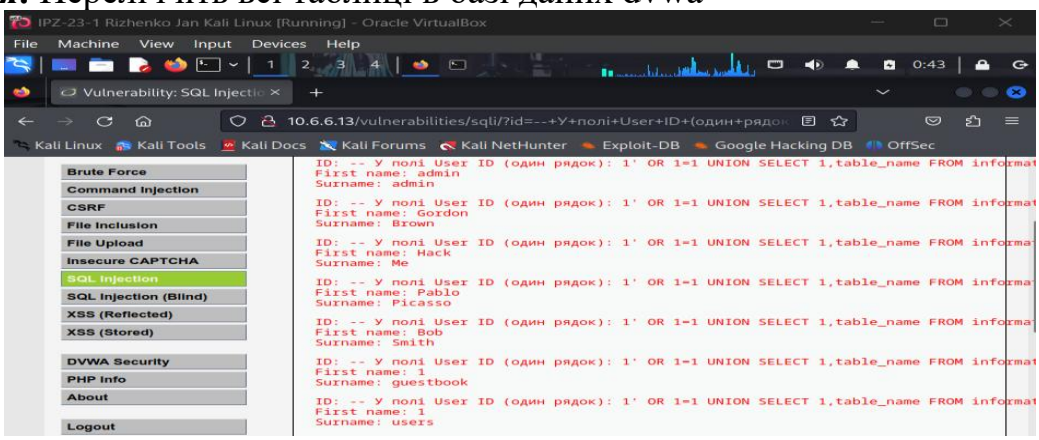

Рис. 7. Перерахування таблиць бази даних через information_schema

Питання: Які дві таблиці були знайдені?
Відповідь: У базі даних dvwa знайдено дві таблиці: таблиця guestbook (для зберігання записів гостьової книги з коментарями користувачів) та таблиця users (для зберігання облікових записів користувачів з автентифікаційними даними, іменами та іншою особистою інформацією).
Питання: Яка таблиця, на вашу думку, найцікавіша для тесту на проникнення?
Відповідь: Таблиця users є найцікавішою для тесту на проникнення, оскільки вона містить критичну інформацію для безпеки: імена користувачів (usernames), паролі (зазвичай у вигляді хешів), email адреси, рівні привілеїв та іншу автентифікаційну інформацію. Компрометація цієї таблиці дозволяє атакуючому отримати облікові дані для несанкціонованого доступу до системи, виконати lateral movement, ескалювати привілеї або використовувати здобуті паролі для атак на інші системи через password reuse.
Крок 7: Отримання імен колонок з таблиці users
Завдання: Дізнайтесь структуру таблиці users

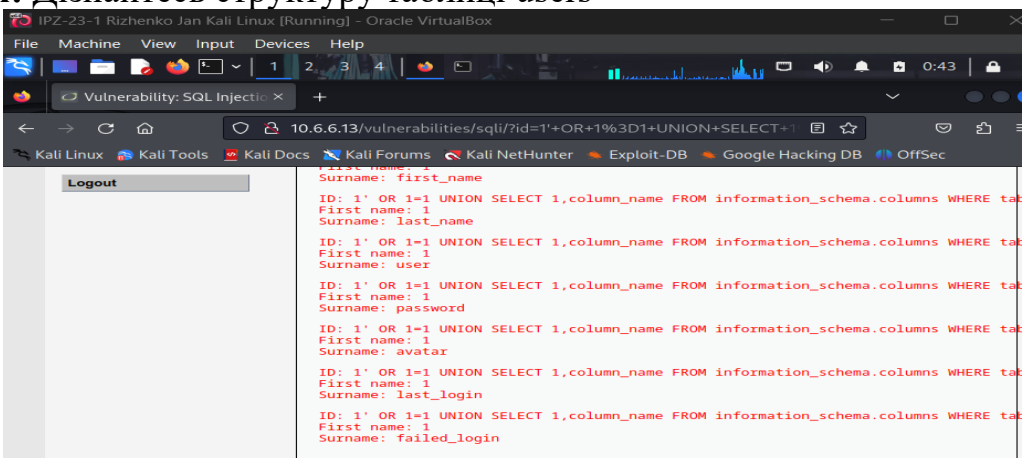

Рис. 8. Перерахування колонок таблиці users через information_schema.columns

Питання: Інформація з яких двох колонок представляє інтерес для тесту на проникнення? Поясніть.
Відповідь: Найбільший інтерес представляють колонки user та password. Колонка user містить імена користувачів (логіни) для автентифікації в системі, тоді як колонка password містить паролі користувачів, зазвичай у вигляді криптографічних хешів (MD5, SHA-1, bcrypt тощо). Комбінація цих двох колонок надає повні облікові дані, які атакуючий може використати для: безпосереднього входу в систему (якщо паролі слабко захищені), offline cracking хешів паролів через John the Ripper або Hashcat, credential stuffing атак на інші системи, де користувачі могли використати ті самі паролі, та ескалації привілеїв через компрометацію адміністративних облікових записів.
Крок 8: Отримання облікових даних користувачів
Завдання: Витягніть usernames та password хеші з таблиці users

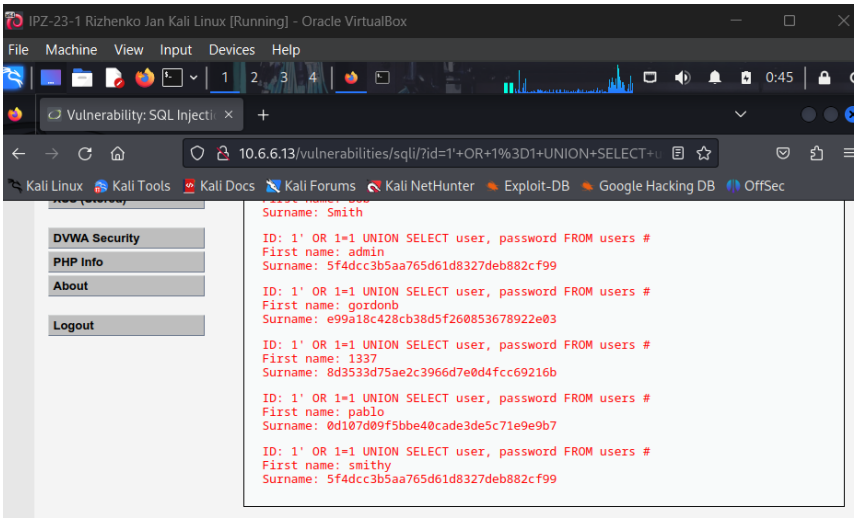

Рис. 9. Витягування облікових даних користувачів з паролями у вигляді хешів

Питання: Який обліковий запис може бути найціннішим у нашому інтересі? Поясніть.

Відповідь: Обліковий запис admin є найціннішим для тесту на проникнення, оскільки він зазвичай має найвищі привілеї в системі. Доступ до облікового запису адміністратора дозволяє: повний контроль над DVWA додатком та його конфігурацією, можливість створювати, модифікувати або видаляти інших користувачів, доступ до всіх функцій та даних системи без обмежень, потенційну можливість виконання команд на сервері через адміністративні інтерфейси, використання легітимних адміністративних функцій для подальшої експлуатації або lateral movement в мережі. Крім того, облікові дані адміністратора часто використовуються повторно на інших системах організації, що розширює можливості атаки.

Питання: Яка різниця між полями user_id та user?

Відповідь: Поле user_id - це унікальний числовий ідентифікатор (primary key) у базі даних, який автоматично генерується системою для внутрішньої ідентифікації записів користувачів (наприклад, 1, 2, 3, 4, 5). Це поле використовується для зв'язків між таблицями та забезпечення унікальності записів. Поле user - це текстове поле, що містить фактичне ім'я користувача (username/login), яке використовується для автентифікації та відображення в інтерфейсі (наприклад, admin, gordonb, 1337, pablo, smithy). User_id є технічним внутрішнім ідентифікатором, тоді як user є зрозумілим для людини логіном, який користувач вводить при вході в систему.

Крок 9: Злам хешів паролів

Завдання: Використайте онлайн сервіс для злому хешів паролів

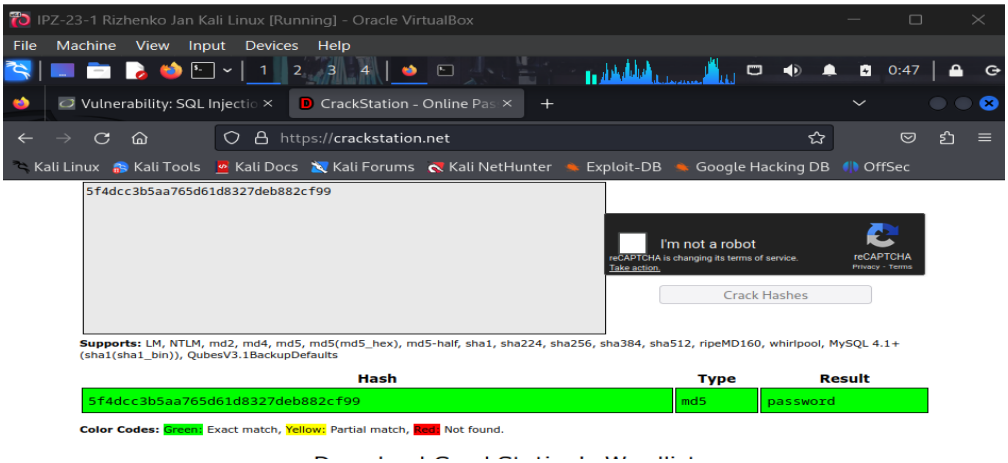

Рис. 10. Використання CrackStation для злому MD5 хешів паролів

Питання: Який пароль облікового запису admin?

Відповідь: Пароль облікового запису admin: password. Це надзвичайно слабкий пароль, який є одним з найпоширеніших паролів у світі та присутній у всіх password dictionaries. Використання такого простого пароля для адміністративного облікового запису є критичною вразливістю безпеки, яка робить систему вразливою навіть до найпростіших brute-force або dictionary атак.

Питання: Який пароль користувача pablo?

Відповідь: Пароль користувача pablo: letmein. Це також дуже слабкий та поширений пароль, який легко піддається злому через словникові атаки. Фраза "letmein" (пусти мене) є класичним прикладом небезпечного пароля, який базується на звичайних словах англійської мови та не має достатньої складності (немає цифр, спеціальних символів або змішаного регістру).

Частина 2: Дослідження методів захисту від SQL Injection

Крок 1: Проведення онлайн дослідження захисту від SQL injection

Завдання: Вивчіть методи запобігання та пом'якшення SQL injection атак

Я ознайомився з документацією OWASP та рекомендаціями SANS Instutute.

Рефлексивне питання

Питання: Які три методи пом'якшення для запобігання експлойтам SQL injection?

Відповідь:

1. Параметризовані запити (Prepared Statements/Parameterized Queries):

Це найефективніший метод захисту від SQL injection. Параметризовані запити відокремлюють SQL код від даних користувача, використовуючи placeholders для параметрів. База даних розглядає вхідні дані користувача виключно як дані, а не як виконуваний код. Приклад у PHP з MySQLi:

```
$stmt = $mysqli->prepare("SELECT * FROM users WHERE user = ? AND password = ?");
$stmt->bind_param("ss", $username, $password);
$stmt->execute();
```

При такому підході навіть якщо користувач введе ' OR 1=1 #, це буде оброблено як літеральний рядок, а не SQL команда.

2. Валідація та санітизація вхідних даних (Input Validation & Sanitization):

Впровадження суворої валідації всіх даних користувача перед використанням у SQL запитах. Це включає: whitelist валідацію (дозволяти лише передбачувані символи/формати), відхилення спеціальних SQL символів (', ", ;, --, #, /*, */), використання регулярних виразів для перевірки формату (email, номери, дати), обмеження довжини введення, type checking (гарантування, що числові поля містять лише числа).
Важливо: санітизація є другорядним захистом і не повинна бути єдиним методом, оскільки складні атаки можуть обійти фільтри.

3. Принцип найменших привілеїв для database accounts:

Налаштування облікових записів бази даних з мінімальними необхідними привілеями для функціонування додатку. Веб-додаток не повинен використовувати облікові записи з правами DBA або root. Конкретно: використовувати окремі облікові записи для read-only операцій, обмежити доступ до системних таблиць (information_schema), заборонити виконання небезпечних команд (DROP, ALTER, GRANT), обмежити доступ лише до необхідних таблиць і баз даних, заборонити file operations (LOAD_FILE, INTO OUTFILE). Це зменшує шкоду від успішної SQL injection - навіть якщо атака пройде, зловмисник не зможе отримати адміністративний контроль над СУБД.

Додаткові методи захисту:

- Використання ORM (Object-Relational Mapping) frameworks, які автоматично параметризують запити
- Web Application Firewalls (WAF) для виявлення та блокування SQL injection патернів
- Регулярний code review та security testing (SAST/DAST)
- Error handling - не показувати детальні SQL помилки користувачам
- Database encryption для захисту даних у разі успішної експлуатації
- Security headers та Content Security Policy

Висновок: У процесі виконання лабораторної роботи було детально вивчено механізм атак SQL injection на веб-додатки з backend базами даних через експлуатацію DVWA. Послідовна серія SQL injection атак продемонструвала повний цикл компрометації: від початкового виявлення вразливості через тестовий payload ' OR 1=1 #, визначення структури запиту методом ORDER BY, отримання метаданих про версію MySQL та назву бази даних, перерахування таблиць через information_schema, витягування структури таблиці users до фінального отримання облікових даних всіх користувачів з хешами паролів. Успішний злам MD5 хешів через CrackStation виявив критично слабкі паролі admin:password та pablo:letmein, що підкреслює важливість політик складних паролів. Дослідження методів захисту показало, що параметризовані запити є найефективнішим технічним рішенням, доповнені валідацією вхідних даних та принципом найменших привілеїв для database accounts. Лабораторна робота продемонструвала критичну небезпеку SQL injection як однієї з найпоширеніших та найнебезпечніших вразливостей веб-додатків згідно OWASP Top 10, підкресливши необхідність комплексного підходу до безпеки на всіх рівнях розробки додатків.