**WAPH-Web Application Programming and Hacking**

**Instructor: Dr. Phu Phung**

**Student**

**Name**: Sai Keerthi Vadnala

**Email**: vadnalsi@ucmail.uc.edu

**Short bio**: Sai Keerthi Vadnala has great interest in learning web development.

and wants to explore more about it by doing hands-on projects.

Figure 1: Sai Keerthi vadnala headshot

**Repository Information**

**Repository's URL:** https://github.com/Saikeerthi72/waph-vadnalsi.git

This is a private repository for Sai Keerthi Vadnala to store all code from the course.

**Hackathon 2 Overview**

• Hackathon 1 focuses on SQL Injection Attacks

• This hackathon includes contains level: 0,1,2,

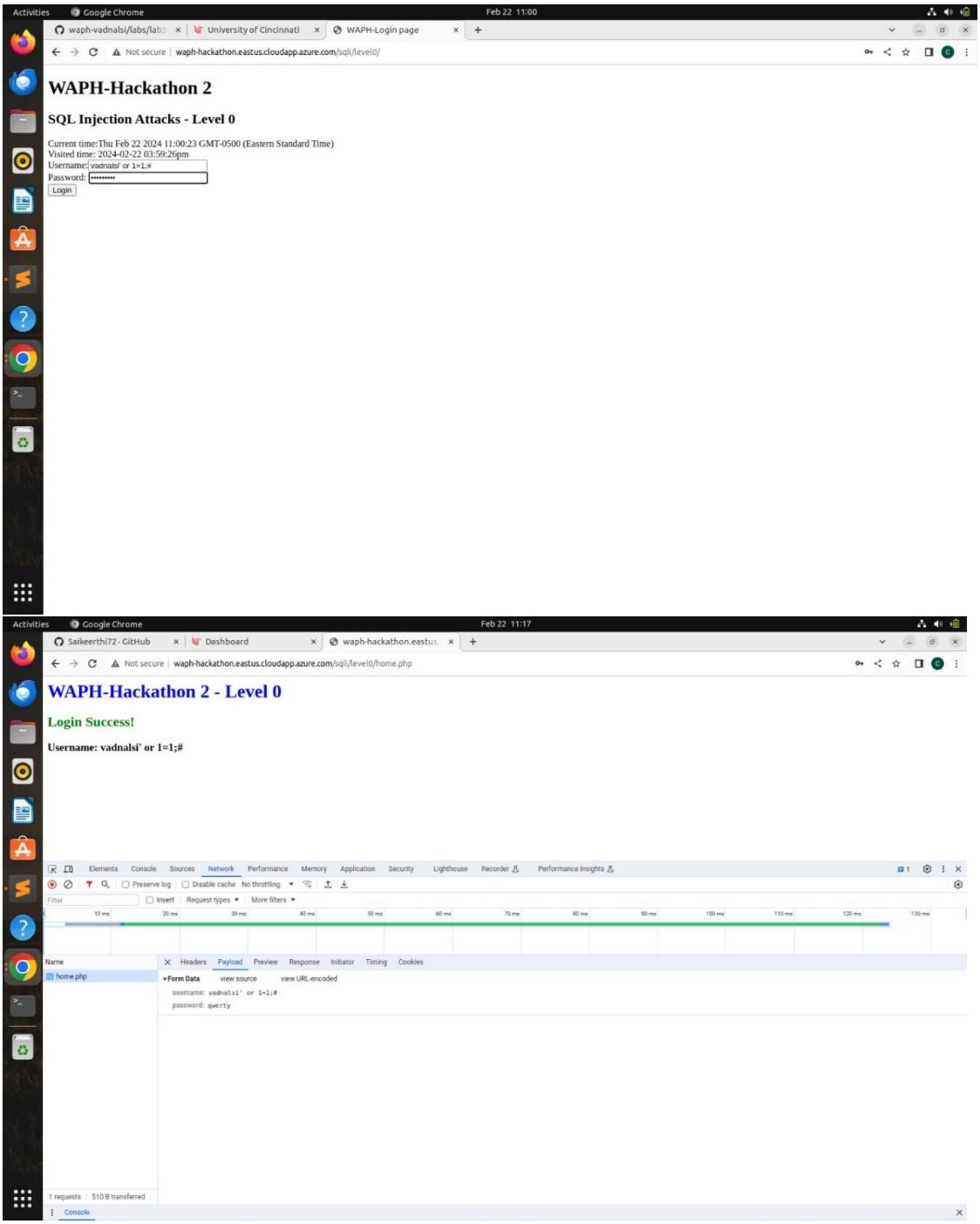
• In level 0, A task of injecting SQL code using UC username is given to bypass the login, which aims at familiarizing with SQL injection vulnerabilities.

• In level 1, it is about guessing the SQL string used in the back end and injecting SQL code to bypass the login check. This level deepens the understanding of SQL injection.

• In level 2, Identification of SQLi vulnerabilities across the application, using injections to understand the database structures, and extract data, including usernames and passwords. Here we demonstrate hackers demonstrating their ability to bypass security measures, access unauthorized data, and gain system access with stolen credentials. This level not only tests technical skills but also emphasizes the importance of understanding and mitigating SQLi vulnerabilities to enhance application security.

**Level 0:**

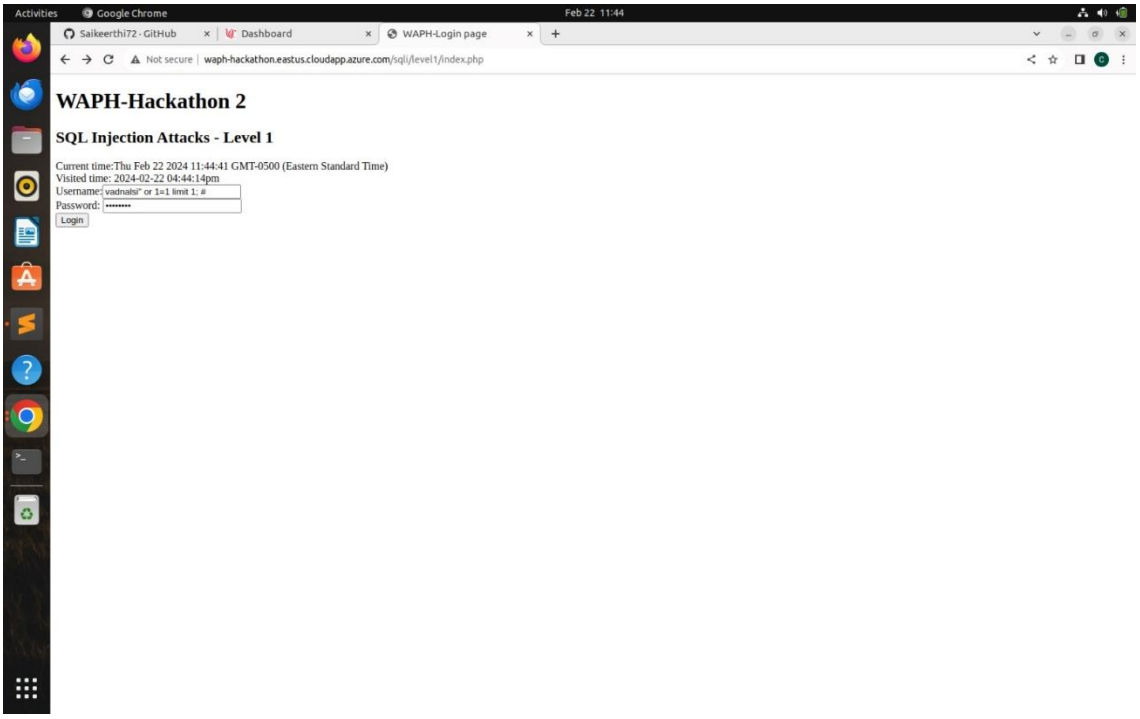
- Level 0 focuses on utilizing SQL injection by entering a UC username to bypass the login process and access the system.
- In this level, the username is the “username or the universally true condition '1 = 1' “ is given as input.
- The system verifies either the provided username or the always-true condition. And I have ended the injection with '#' which acts as a comment out for the password field.
- By submitting the login with 'username' or 1=1# along with any arbitrary password, the system grants access successfully.
- Below is the output for this level:

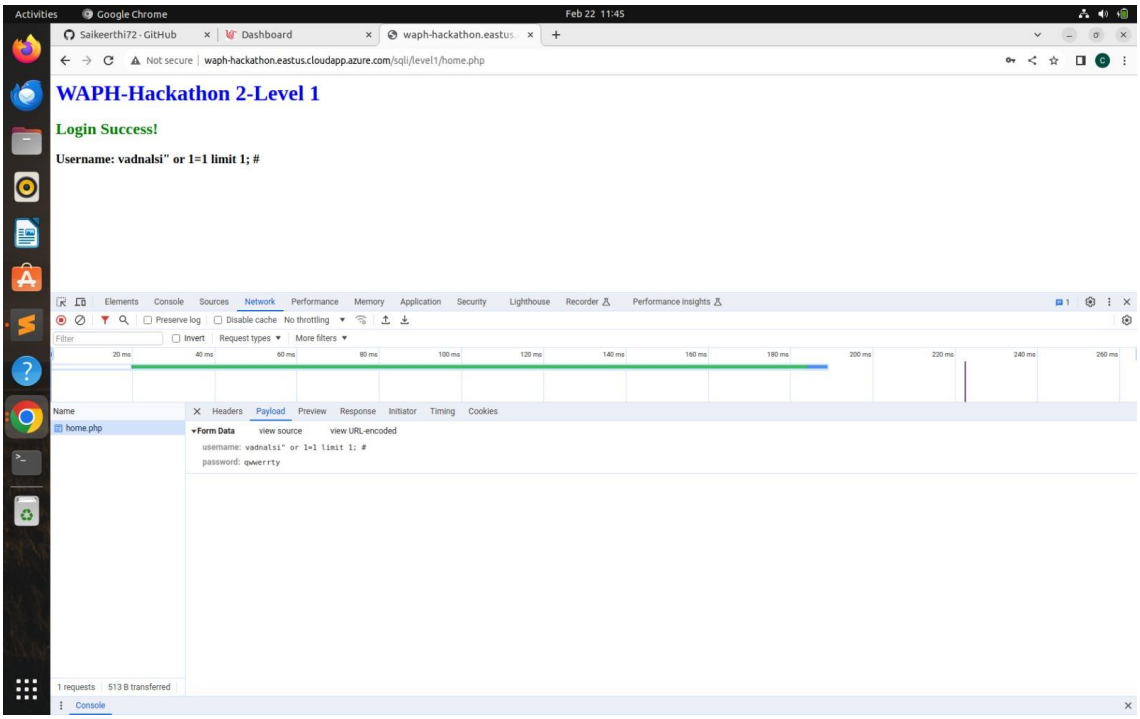



**Level 1:**

- Level 1 introduces a higher level of complexity compared to Level 0.
- When the database contains only a single record, the condition '1 = 1' is invariably true. However, in Level 1, there are multiple records in the database which necessitates a method to retrieve just one specific entry.

- The SQL 'LIMIT' function is used to restrict the query to returning only a single row. So I have appended a 'LIMIT' clause which will enable successful system login.
- The backend SQL query is: `select * from users where username = 'Uc_username" AND password = md5('any')`.
- By modifying the query to `select * from users where username = 'vadnalsi" or 1 = 1 limit 1; # AND password = md5('any') `, this manipulates the backend command and gives system access successfully.
- Below is the output for this level:


Activities
Google Chrome
Feb 22 11:44
Saikeerthi72 - GitHub
Dashboard
WAPH-Login page
Not secure | waph-hackathon.eastus.cloudapp.azure.com/sqli/level1/index.php
WAPH-Hackathon 2
SQL Injection Attacks - Level 1
Current time:Thu Feb 22 2024 11:44:41 GMT-0500 (Eastern Standard Time)
Visited time: 2024-02-22 04:44:14pm
Username: vadnalsi" or 1=1 limit 1; #
Password: ••••••••
Login

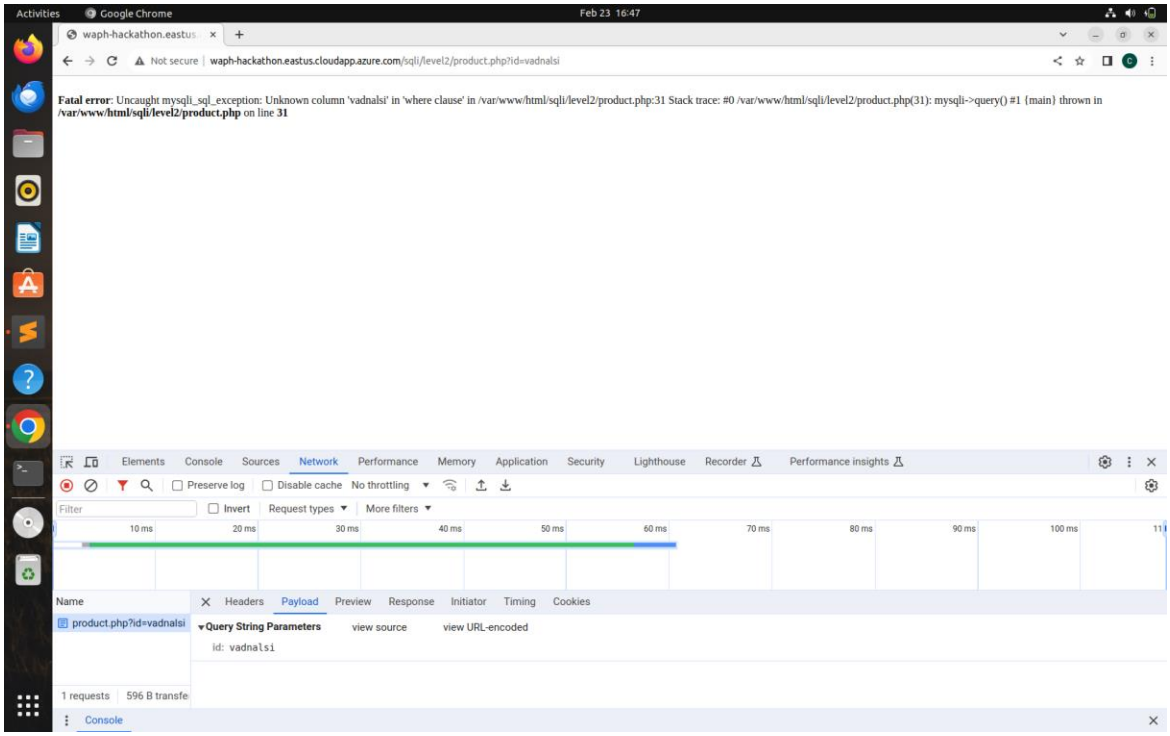

Activities
Google Chrome
Feb 22 11:45
Saikeerthi72 - GitHub
Dashboard
waph-hackathon.eastus...
Not secure | waph-hackathon.eastus.cloudapp.azure.com/sqli/level1/home.php
WAPH-Hackathon 2-Level 1
Login Success!
Username: vadnalsi" or 1=1 limit 1; #
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Preserve log
Disable cache
No throttling
Filter
Invert
Request types
More filters
Name
home.php
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
Form Data
view source
view URL-encoded
username: vadnalsi" or 1=1 limit 1; #
password: qwwerrty
1 requests
513 B transferred
Console

**Level 2:**

- This level involves working on advanced SQL injection (SQLi) techniques to discover vulnerabilities in a secure website.
- This task involves identifying specific queries within the system that are susceptible to SQLi, paving the way for unauthorized access.

**a. Regarding SQLi vulnerabilities:**

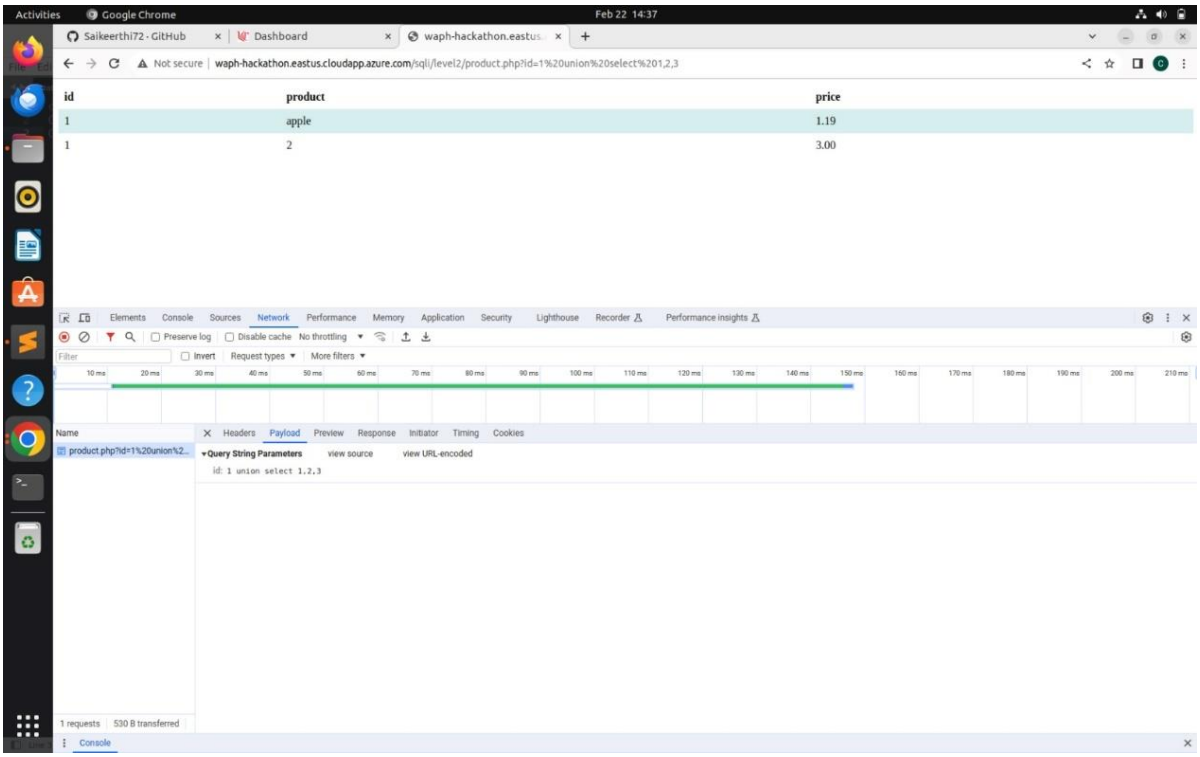
- After a thorough examination understood that attempts to inject SQL in the login form are like the techniques used in Levels 0 and 1 were unsuccessful, indicating backend protection against such attacks.
- Further investigation into the product categories revealed a potential vulnerability. By altering the ID parameter in the URL from 1 to 2, and then experimenting with different ID values and the UNION SQL command, it was possible to manipulate the system's output. This indicated a vulnerability that could be exploited through SQL injection by adjusting ID parameters and using UNION commands to retrieve data, showcasing a significant security vulnerability within the application.



## b. Exploiting SQLi to Access Data

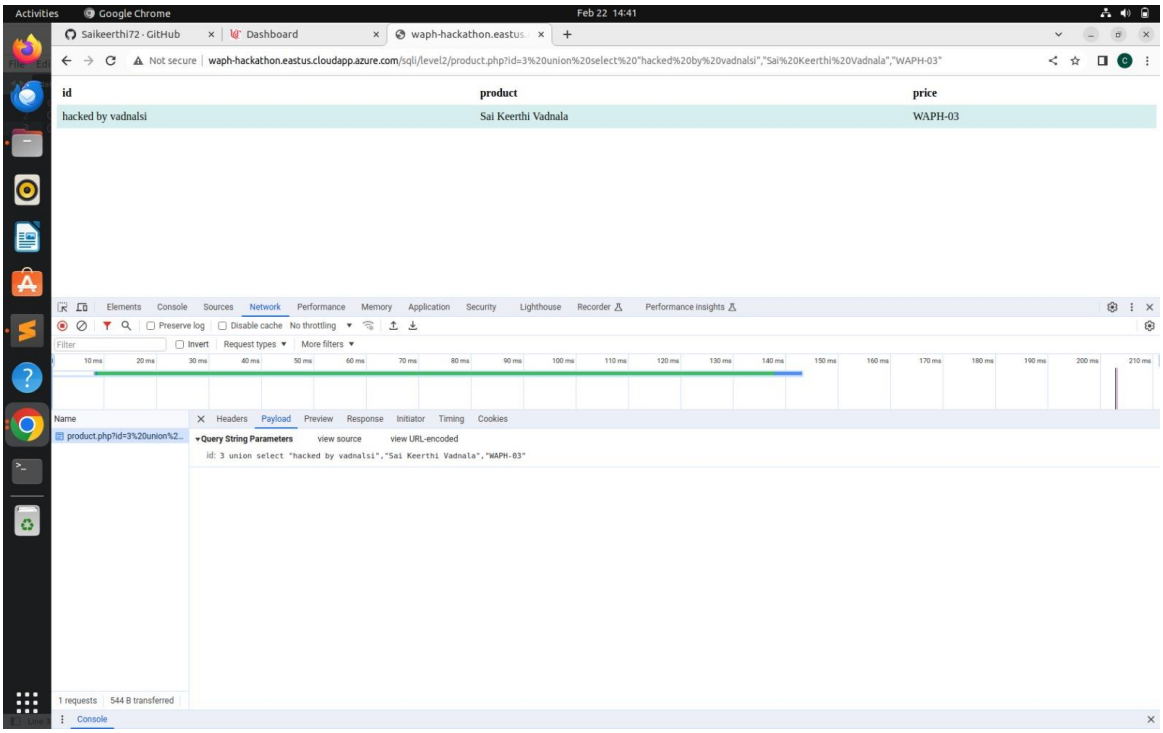
### i. Identify the Number of Columns

- I experimented with SELECT statements to determine the number of columns in the database.
- When I used "UNION SELECT 1,2,3", I understood that there are three columns in the database.



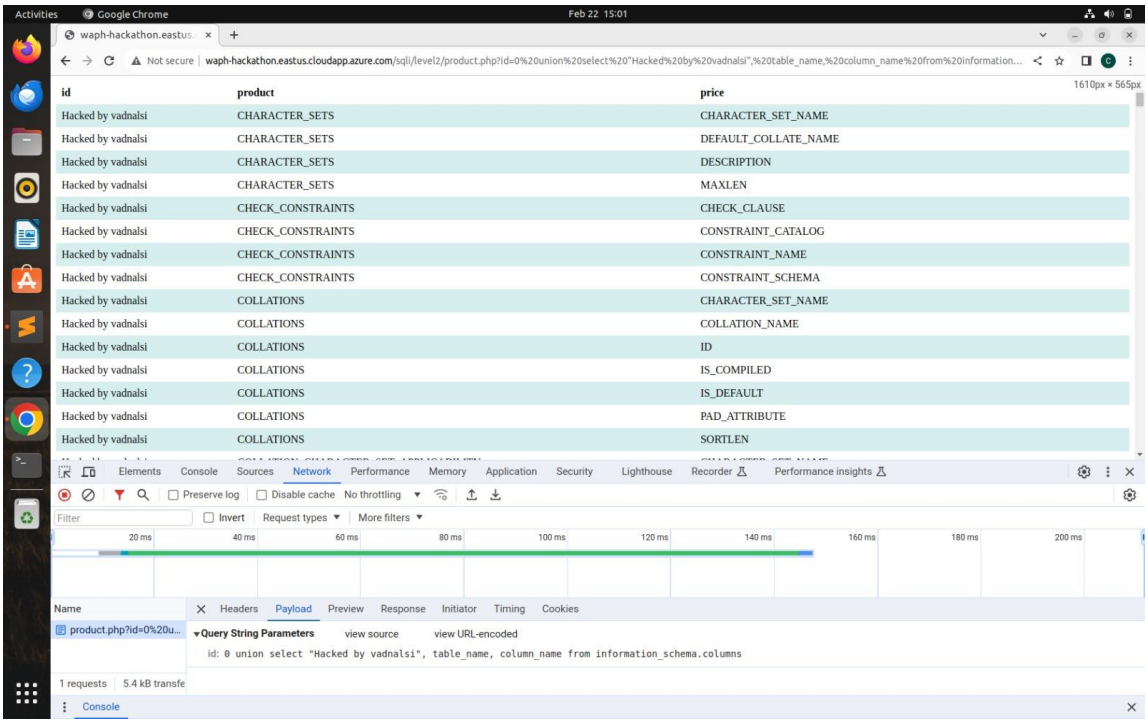
### ii. Display Your Information

- I retrieved username, name, and section by using a SELECT statement, by inserting each information into three separate strings.



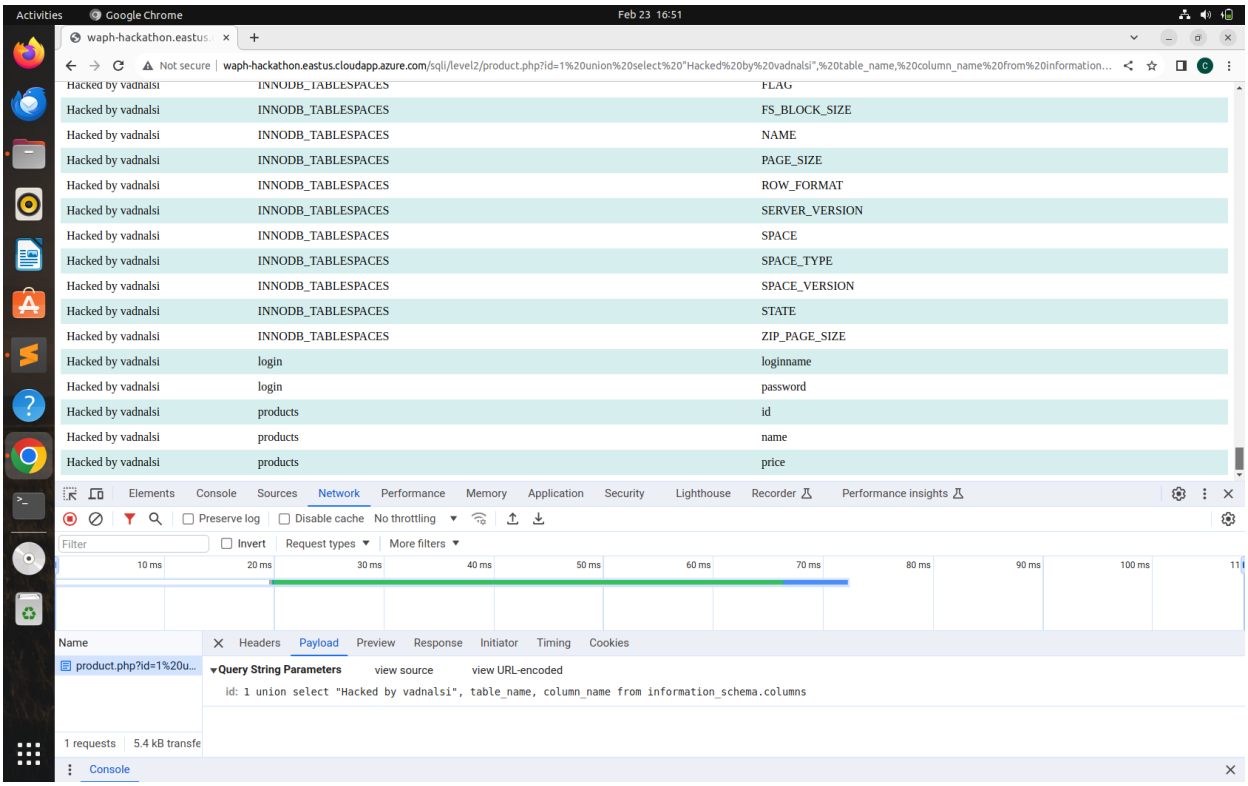
### iii. Display the Database Schema

- I executed a SQL query using UNION, followed by SELECT "Hacked by vadnalsi", table_name, column_name FROM information_schema.columns to point the tables and their respective columns.

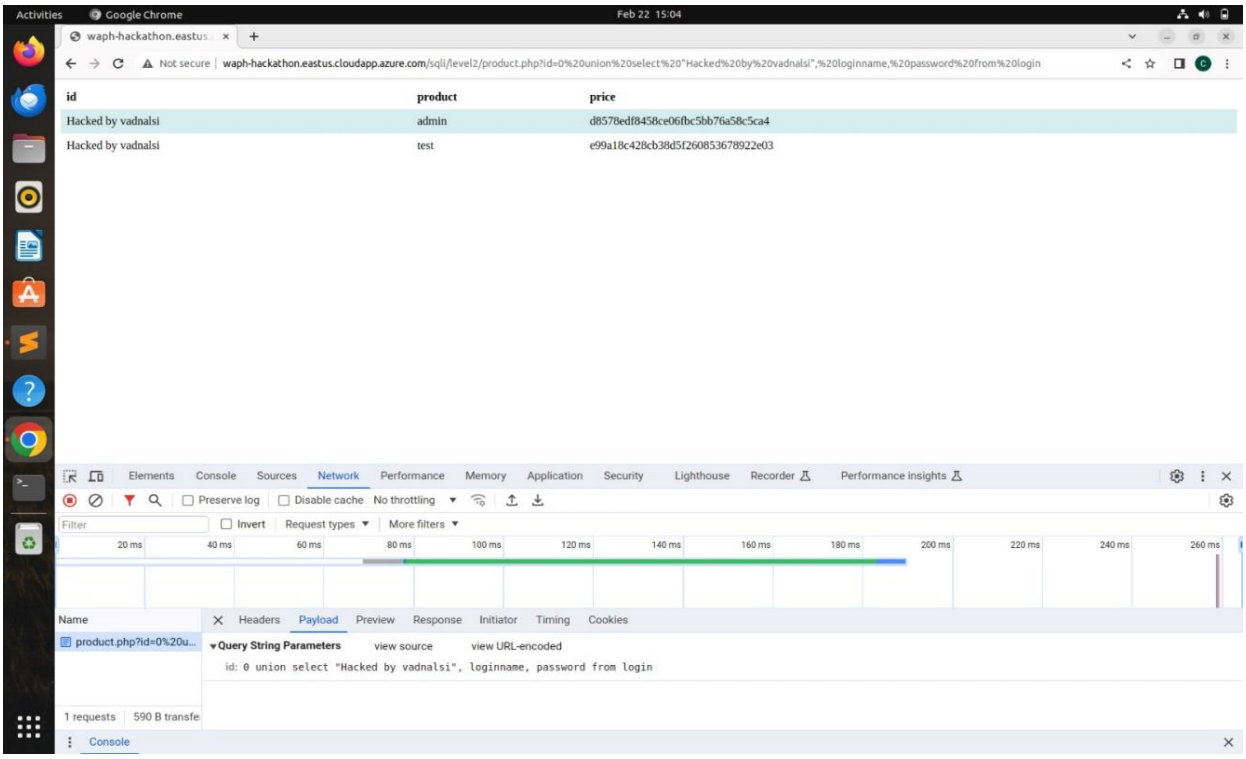



### iv. Display Login Credentials

- By leveraging the SQL query mentioned earlier and navigating through the results, I was able to identify the specific table and columns where the usernames and passwords are stored.

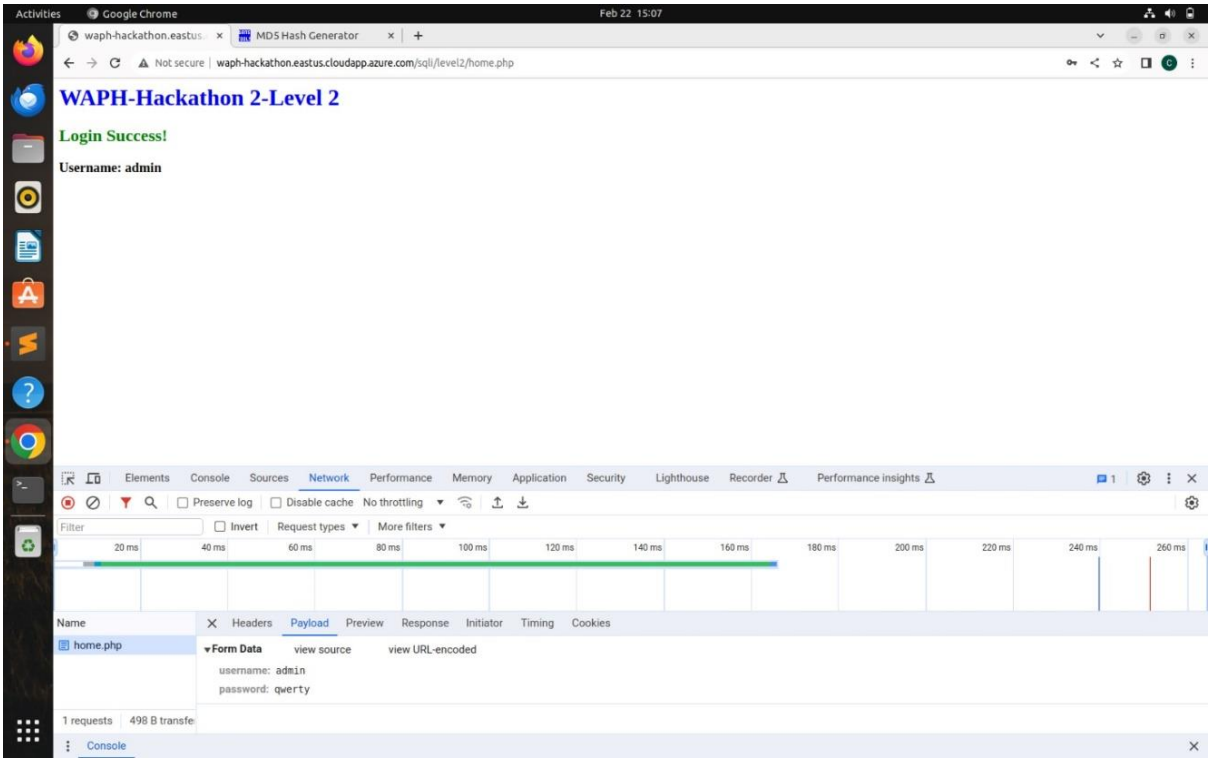


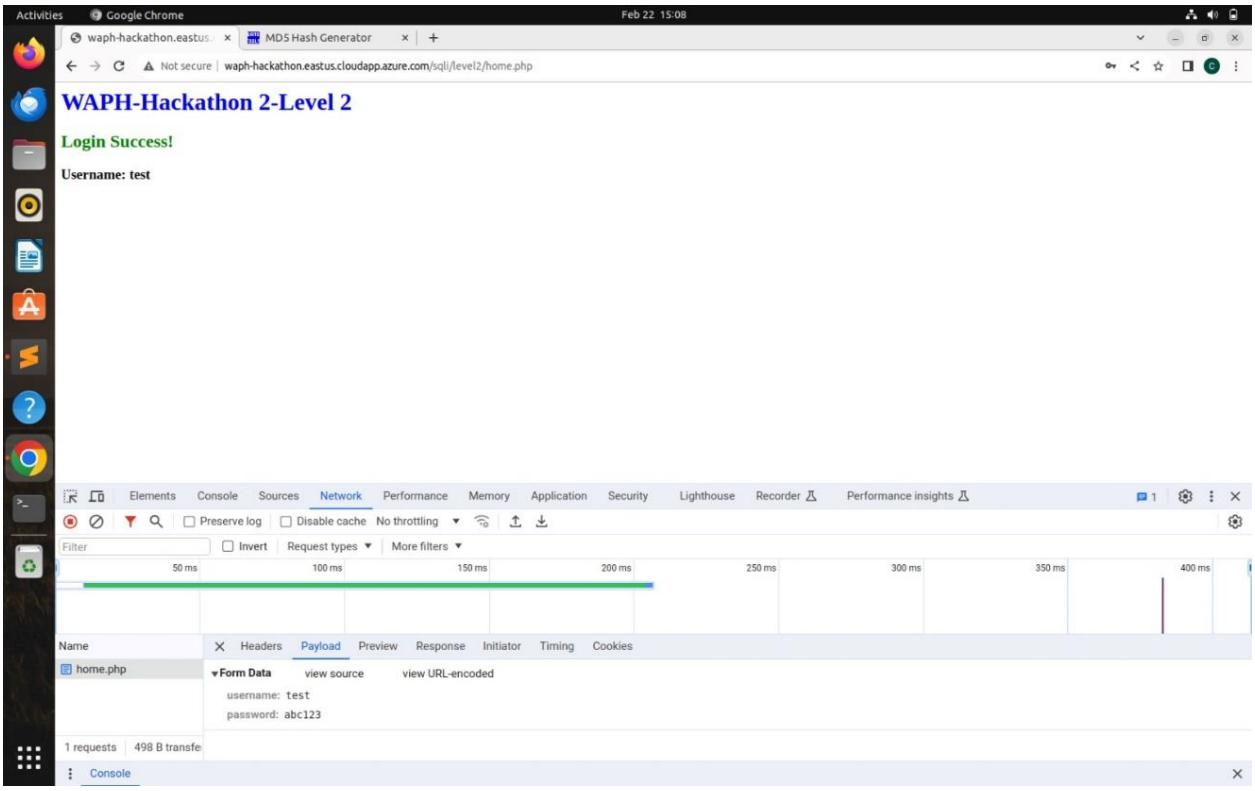

- The query I used was "SELECT ‘Hacked by vadnalsi’, loginname, password FROM login", which resulted in displaying all the usernames and passwords.



### c. Login with Stolen Credentials

- Using the identified usernames and passwords, I attempted to log into the system, and the login was successful.




Activities
Google Chrome
Feb 22 15:08
waph-hackathon.eastus
MD5 Hash Generator
Not secure | waph-hackathon.eastus.cloudapp.azure.com/sqli/level2/home.php
WAPH-Hackathon 2-Level 2
Login Success!
Username: test
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Performance insights
Preserve log
Disable cache
No throttling
Filter
Invert
Request types
More filters
50 ms
100 ms
150 ms
200 ms
250 ms
300 ms
350 ms
400 ms
Name
home.php
Headers
Payload
Preview
Response
Initiator
Timing
Cookies
Form Data
view source
view URL-encoded
username: test
password: abc123
1 requests
498 B transfe
Console