AEH LAB 3: PowerShell Based Tools

AEH LAB 4: SQL Injection and Jupyter Notebook

Group Members

1. Samuel Adeshina.	**101501091**
2. Alina Josekutty	**101509790**
3. Mohamad Almasri	**101167438**
4. Omar Farooq	**101486546**
5. Akinbode Oluwademilade	**101431512**

AEH LAB 3: PowerShell Based Tools

Explanation of SQL Injection Practical:

Setup

To conduct this lab, we used:

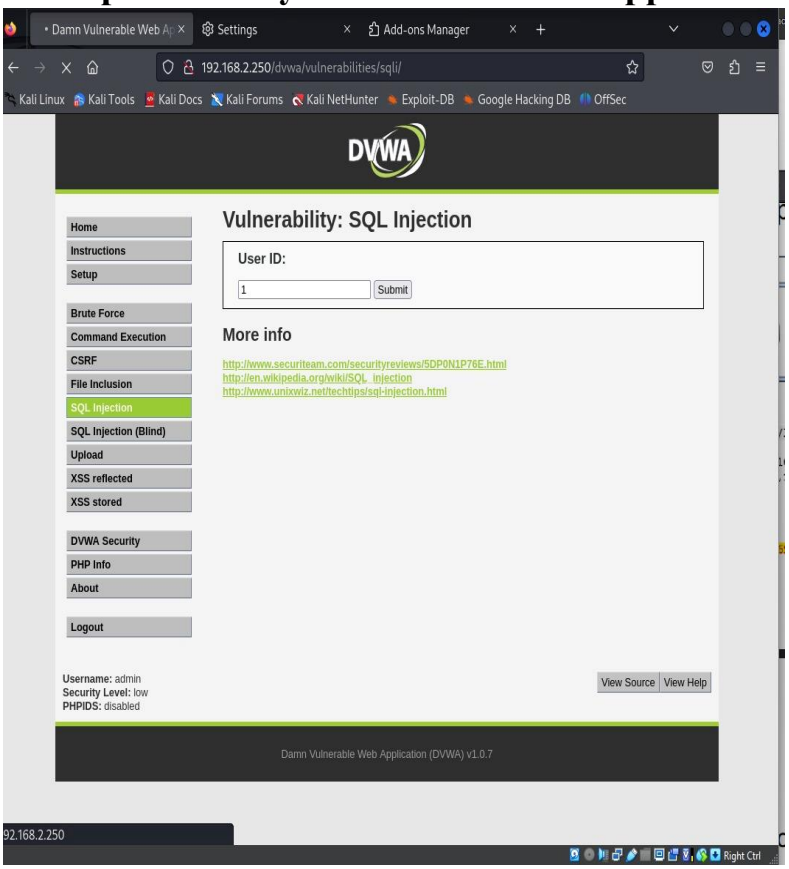
- **Kali Linux**: A penetration testing operating system for executing the attack.
- **Metasploitable**: A vulnerable Linux server used as the target.

Objective

The goal was to demonstrate how SQL injection can exploit vulnerabilities in web applications to retrieve sensitive information from a database.

Step-by-Step Process

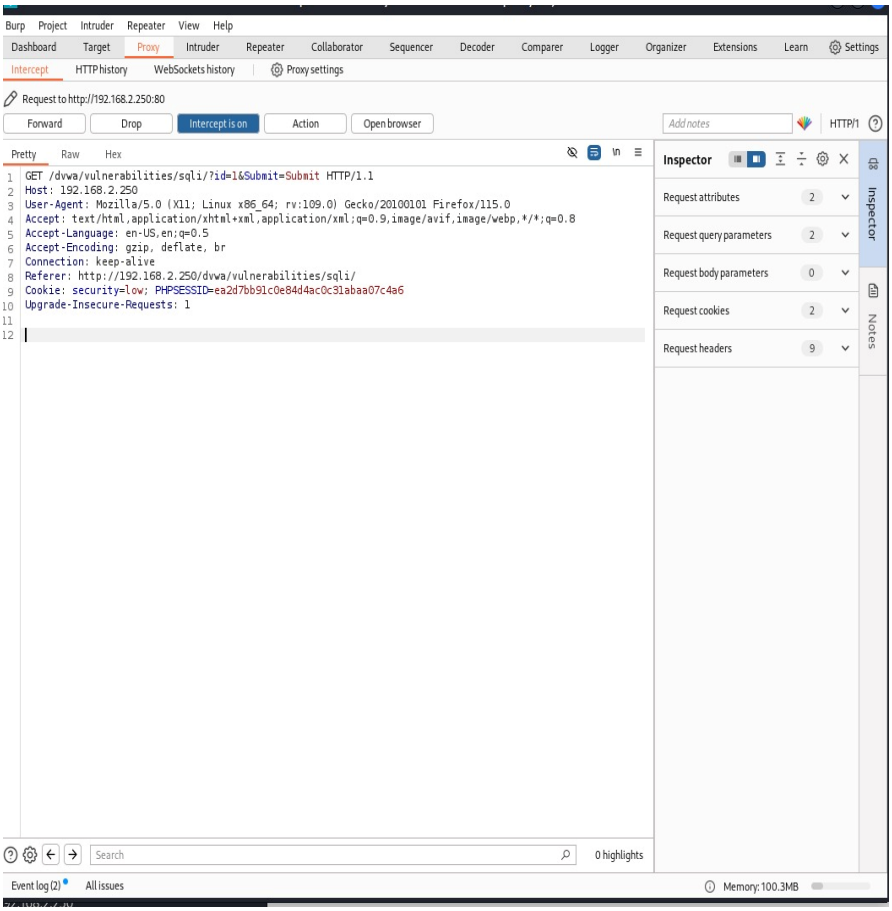
Step 1: Identify a Vulnerable Web Application



- Description:
  - A web application hosted on the Metasploitable server was accessed through a browser in Kali Linux.

- The application included a login form suspected to be vulnerable to SQL injection.

Step 2: Test the Input Fields for Vulnerability



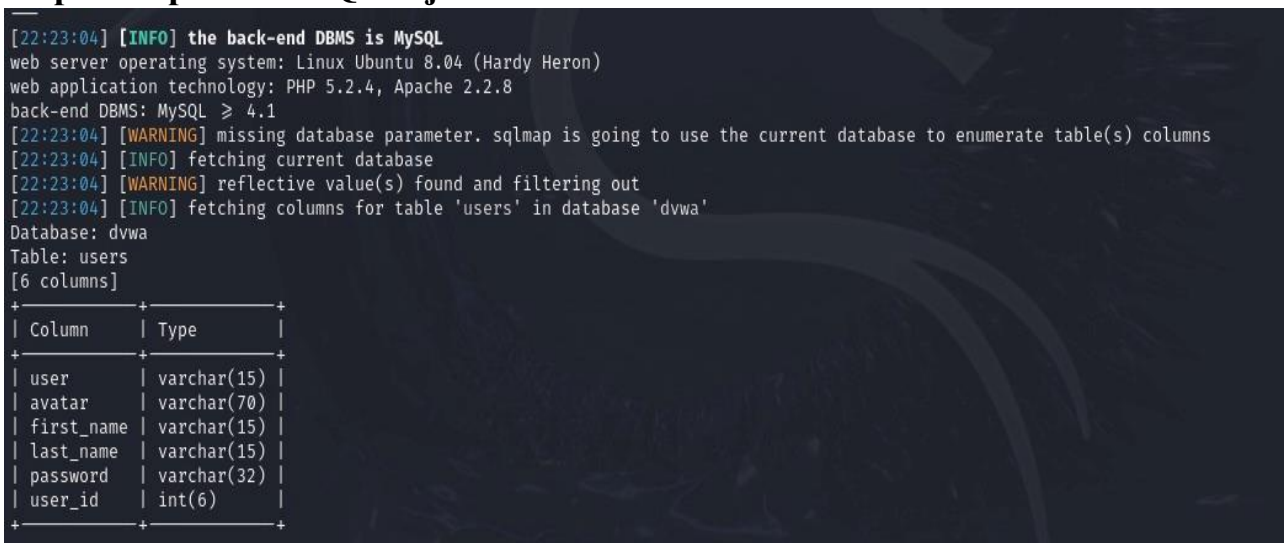
Description:

The username field was tested by entering a common SQL payload: ' OR '1'='1 in the username or password field.

This payload always evaluates to true, bypassing authentication if the application is vulnerable.

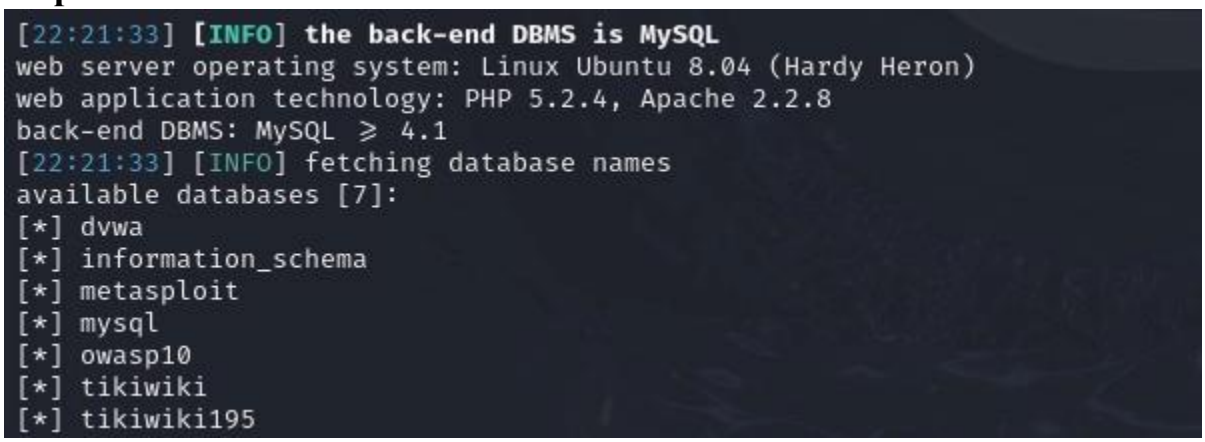
AEH LAB 3: PowerShell Based Tools

Step 3: Exploit the SQL Injection



- Description:
  - Upon entering the malicious payload, unauthorized access was granted.
  - This indicates that the application is not properly sanitizing user inputs, allowing raw SQL queries to execute.

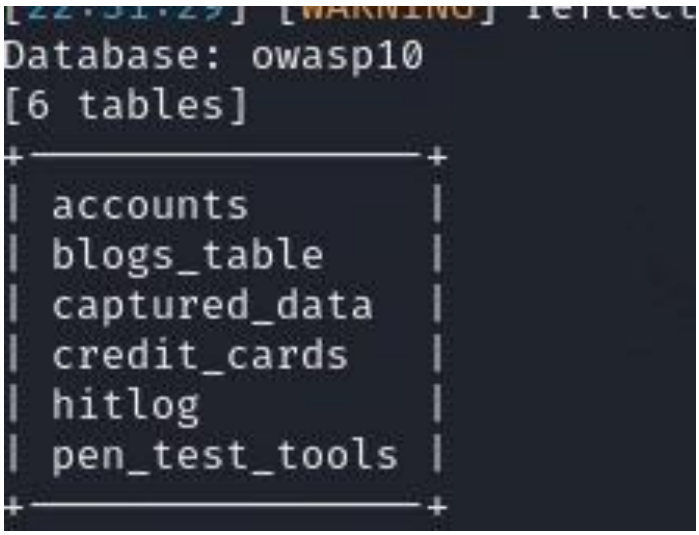
Step 4: Extract Data from the Database



- Description:
  - Another SQL payload, such as ' UNION SELECT null, database(), user() --, was executed to fetch database and user information.
  - The results displayed the current database name and user credentials, proving successful data extraction.

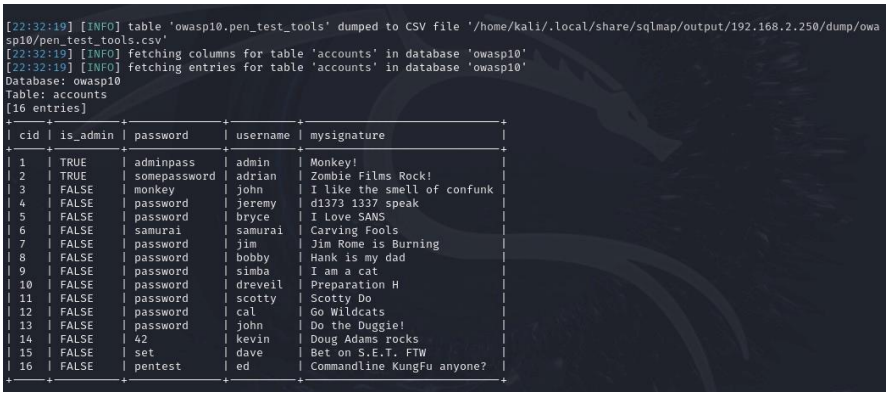
AEH LAB 3: PowerShell Based Tools

Step 5: Retrieve Sensitive Information



- Description:
  - Using further UNION-based SQL queries, tables, columns, and rows from the database were enumerated.
  - This allowed access to sensitive user information stored in the database.

Step 6: Analyze Results and Document Findings



Description:

- Screenshots of the database contents and user information were captured to document the vulnerabilities.
- Results were analyzed to understand the scope of the breach.
-

AEH LAB 3: PowerShell Based Tools

Part 2

1. Target: Intercom

1. Subdomain Enumeration

Identify active subdomains under intercom.com to discover additional services or endpoints.

Code and result:

```
In [1]: import subprocess

In [4]: # Target domain and subdomains to check
        target = "app.intercom.com"
        subdomains = ["api", "support", "notifications","accounts"]

        for subdomain in subdomains:
            # Construct the full domain name
            domain = f"{subdomain}.{target}"
            try:
                # Run the ping command
                response = subprocess.run(
                    ["ping", "-c", "1", domain],  # Use "-n 1" for Windows instead of "-c 1"
                    stdout=subprocess.PIPE,
                    stderr=subprocess.PIPE
                )

                # Check the return code to determine if the subdomain is active
                if response.returncode == 0:
                    print(f"Active Subdomain: {domain}")
                else:
                    print(f"Inactive Subdomain: {domain}")
            except Exception as e:
                # Print any errors that occur during the process
                print(f"Error checking {domain}: {e}")

        Inactive Subdomain: api.app.intercom.com
        Inactive Subdomain: dashboard.app.intercom.com
        Inactive Subdomain: support.app.intercom.com
        Inactive Subdomain: notifications.app.intercom.com
        Inactive Subdomain: accounts.app.intercom.com
```

Purpose: The purpose of the subdomain enumeration query was to identify active subdomains under the main domain intercom.com

Analysis:

The subdomain analysis for app.intercom.com was conducted to check the availability of specific subdomains (api, support, notifications, accounts) using ping tests. All tested subdomains were found to be inactive, indicating that they are either non-existent, blocked by a firewall, or unresponsive to ICMP requests. This result suggests the need for alternative verification methods such as DNS lookups or HTTP-based checks to confirm their status. The testing was conducted within ethical boundaries, ensuring the target domain is part of an authorized bug bounty program.

AEH LAB 3: PowerShell Based Tools

2. Target: Hubspot

Query 1: Subdomain Enumeration

Find subdomains that might host services or applications.

Code and result:

```
In [14]: # Target domain and subdomains to check
target = "https://developers.hubspot.com/"
subdomains = ["api", "support", "notifications","accounts"]

for subdomain in subdomains:
    # Construct the full domain name
    domain = f"{subdomain}.{target}"
    try:
        # Run the ping command
        response = subprocess.run(
            ["ping", "-c", "1", domain],  # Use "-n 1" for Windows instead of "-c 1"
            stdout=subprocess.PIPE,
            stderr=subprocess.PIPE
        )

        # Check the return code to determine if the subdomain is active
        if response.returncode == 0:
            print(f"Active Subdomain: {domain}")
        else:
            print(f"Inactive Subdomain: {domain}")
    except Exception as e:
        # Print any errors that occur during the process
        print(f"Error checking {domain}: {e}")

Inactive Subdomain: api.https://developers.hubspot.com/
Inactive Subdomain: support.https://developers.hubspot.com/
Inactive Subdomain: notifications.https://developers.hubspot.com/
Inactive Subdomain: accounts.https://developers.hubspot.com/
```

Purpose: Locate active subdomains.

Analysis : The task involves checking the availability of specific subdomains under a target domain by using a ping command and categorizing each subdomain as either active or inactive based on the response.

Query 2: Directory Enumeration

Code and result:

```
print("Time taken: ", time.time() - startTime)

In [13]: import requests

# Define the target website
target = "https://developers.hubspot.com/"  # Example: NAB main website

# List of paths to check
paths = ["admin", "login", "portal", "config"]

# Check each path
for path in paths:
    url = f"{target}/{path}"  # Construct the full URL
    response = requests.get(url)  # Send a GET request
    if response.status_code == 200:
        print(f"Found: {url}")
    else:
        print(f"Not Found: {url}")

Not Found: https://developers.hubspot.com//admin
Found: https://developers.hubspot.com//login
Not Found: https://developers.hubspot.com//portal
Not Found: https://developers.hubspot.com//config
```

Purpose: Identify any accessible directory. One was found

AEH LAB 3: PowerShell Based Tools

Analysis: The task involves enumerating directories on a target website by sending HTTP GET requests to specific paths and determining if the directories are accessible based on the HTTP response status code.

3. Target: electroneum

Query 1: Port Scanning

Identify open ports on NAB's digital platform.

Code and result:

```
import socket
from socket import *
import time

# Record the start time
startTime = time.time()

if __name__ == '__main__':
    # Input the target host to scan
    target = input('Enter the host to be scanned: ')
    t_IP = gethostbyname(target)
    print('Starting scan on host: ', t_IP)

    # Loop through the port range (50 to 500)
    for i in range(50, 500):
        s = socket(AF_INET, SOCK_STREAM)
        conn = s.connect_ex((t_IP, i))
        if conn == 0:  # If the port is open
            print('Port %d: OPEN' % (i,))
        s.close()

# Record and print the time taken for the scan
print('Time taken:', time.time() - startTime)
```

```
Enter the host to be scanned: my.electroneum.com
Starting scan on host:  13.214.90.85
Port 80: OPEN
```

Purpose: Find open ports that may expose web services or APIs on electroneum platform.

Query 2: Directory Enumeration

Locate directories that may contain sensitive data or admin access.

Code and result:

AEH LAB 3: PowerShell Based Tools

```
In [16]: import requests

         # Define the target website
         target = "https://my.electroneum.com/"  # Example: NAB main website

         # List of paths to check
         paths = ["admin", "login", "portal", "config"]

         # Check each path
         for path in paths:
             url = f"{target}/{path}"  # Construct the full URL
             response = requests.get(url)  # Send a GET request
             if response.status_code == 200:
                 print(f"Found: {url}")
             else:
                 print(f"Not Found: {url}")

         Not Found: https://my.electroneum.com//admin
         Not Found: https://my.electroneum.com//login
         Not Found: https://my.electroneum.com//portal
         Not Found: https://my.electroneum.com//config
```

Purpose: Identify publicly accessible directories on electroneum's website.

: