

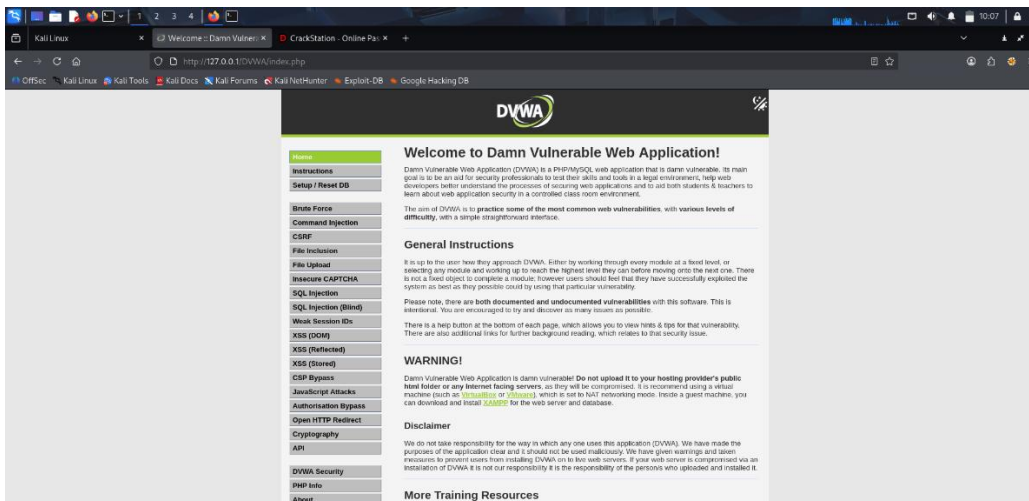
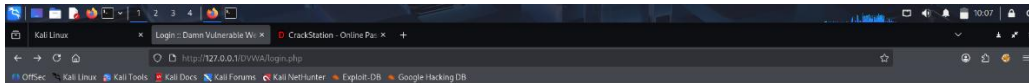
# Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

## Objective:-

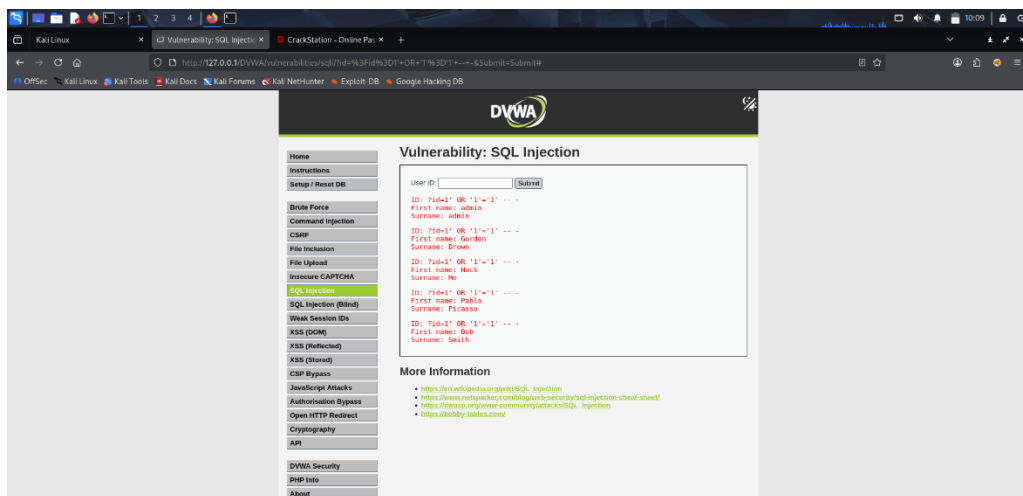
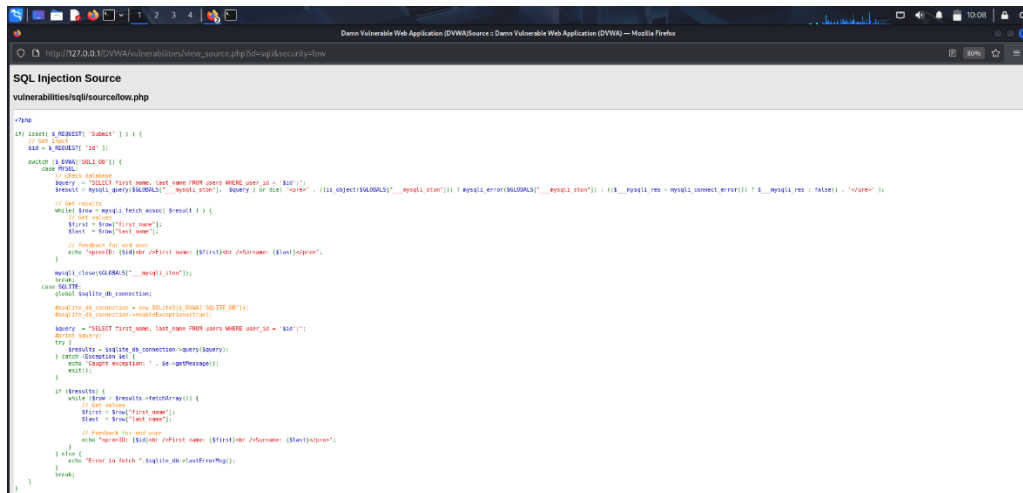
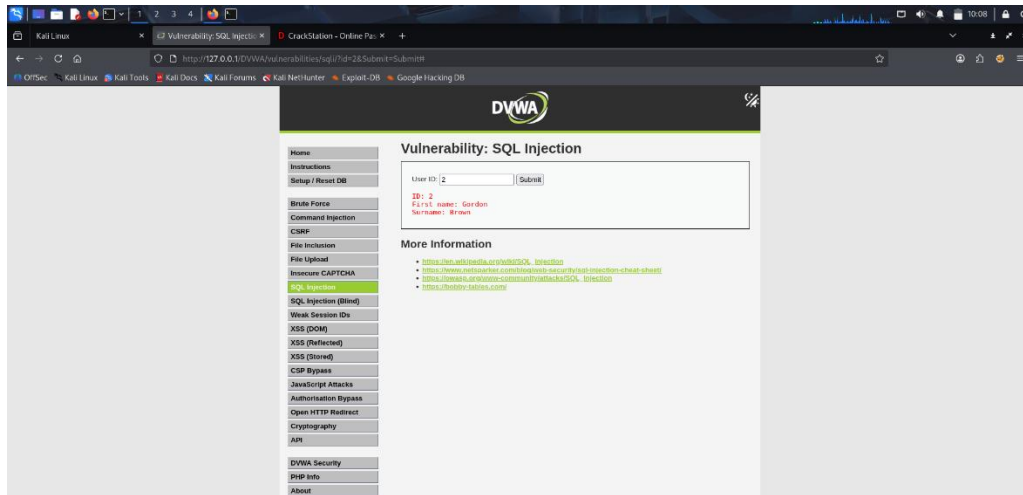
In a controlled lab environment identify, validate and safely exploit OWASP Top 10 vulnerability to assess application security, all testing will be authorized, non-destructive, and aimed at improving security posture through documented findings and remediation recommendations.

## 1. SQL Injection:-

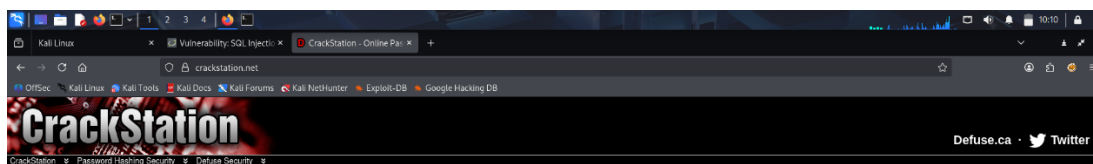
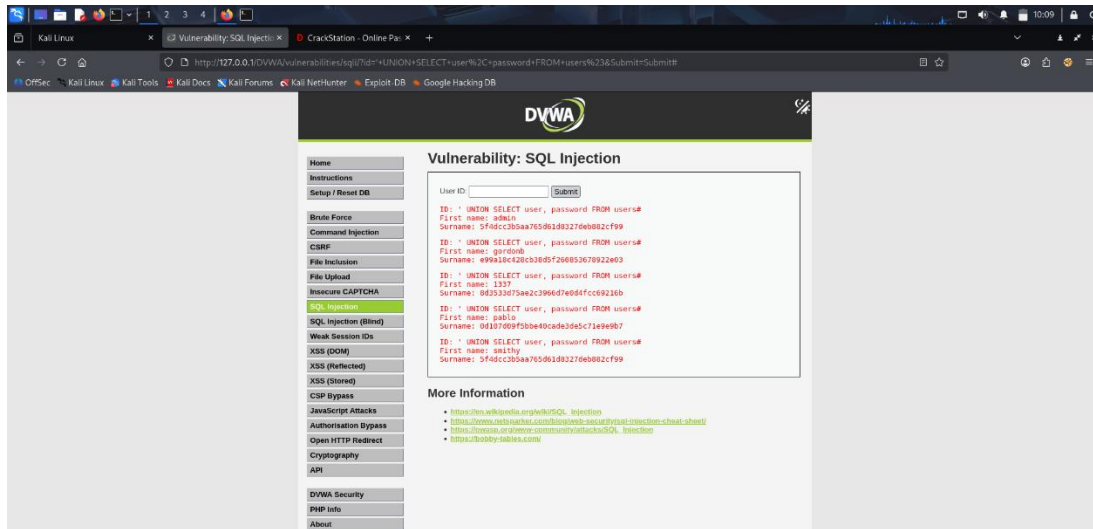


# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025



Intern Name: **Nitesh Sharma**    Date: **05/10/2025**



## Free Password Hash Cracker

Enter up to 20 non-salted hashes, one per line:

5f4ecc3b5aa765d61d8327deb882cf99



Crack Hashes

**Supports:** LM, NTLM, md2, md4, md5, md5(md5\_hex), md5-half, sha1, sha224, sha256, sha384, sha512, rfcMD160, whirlpool, MySQL 4.1+ (sha1 sha1\_bin), QubesV3.1BackupDefault

Hash	Type	Result
5f4dcc3b5aa765d618327deb882c199	md5	password

Color Codes: Green Exact match, Yellow Partial match, Red Not found

[Download CrackStation's Wordlist](#)

## How CrackStation Works

CrackStation uses massive pre-computed lookup tables to crack password hashes. These tables store a mapping between the hash of a password, and the correct password for that hash. The hash values are indexed so that it is possible to quickly search the database for a given hash. If the hash is present in the database, the password can be recovered in a fraction of a second. This only works for "unsalted" hashes. For information on password hashing systems that are not vulnerable to pre-computed lookup tables, see our [hashing security page](#).

Crackstation's lookup tables were created by extracting every word from the Wikipedia databases and adding with every password list we could find. We also applied intelligent word mangling (brute force hybrid) to our wordlists to make them much more effective. For MD5 and SHA1 hashes, we have a 190GB, 15-billion-entry lookup table, and for other hashes, we have a 19GB 1.5-billion-entry lookup table.

## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

### (Demonstrate prevention using Prepared Statement):-

```
<?php

if( isset( $_REQUEST['Submit'] ) ){
    // Get input
    $id = $_REQUEST['id'];

    switch ( $_DVWA['SQLI_DB'] ){
        case MYSQL:
            // Check database connection
            $query = "SELECT first_name, last_name FROM users WHERE user_id = ?";

            // Prepare statement
            if ( $stmt = mysqli_prepare($GLOBALS['__mysqli_ston'], $query) ) {
                // Bind the input parameter
                mysqli_stmt_bind_param($stmt, "i", $id); // "i" means integer

                // Execute the statement
                mysqli_stmt_execute($stmt);

                // Get the result
                $result = mysqli_stmt_get_result($stmt);

                // Fetch results
                while ( $row = mysqli_fetch_assoc($result) ) {
                    $first = $row["first_name"];
                    $last = $row["last_name"];
                    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
                }

                // Close statement
                mysqli_stmt_close($stmt);
            } else {
                echo "Failed to prepare the query.";
            }

            // Close connection
            mysqli_close($GLOBALS['__mysqli_ston']);
            break;

        case SQLITE:
            global $sqlite_db_connection;

            $query = "SELECT first_name, last_name FROM users WHERE user_id = :id";

            try {
                // Prepare statement
                $stmt = $sqlite_db_connection->prepare($query);
```

## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

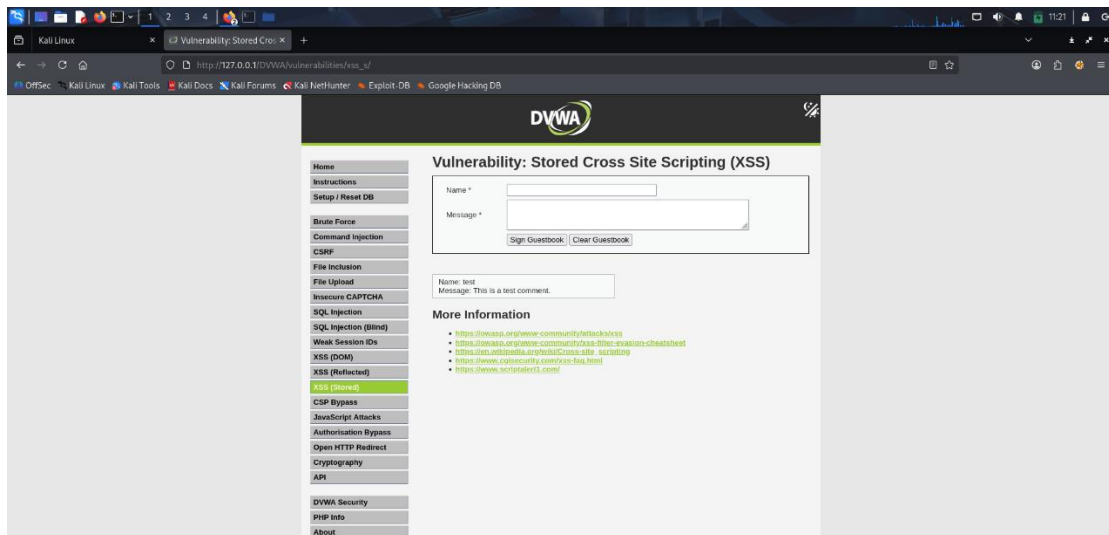
```
// Bind parameters
$stmt->bindValue(':id', $id, SQLITE3_INTEGER);

// Execute the query
$results = $stmt->execute();

// Fetch results
while ($row = $results->fetchArray()) {
    $first = $row["first_name"];
    $last = $row["last_name"];
    echo "<pre>ID: {$id}<br />First name: {$first}<br />Surname: {$last}</pre>";
}

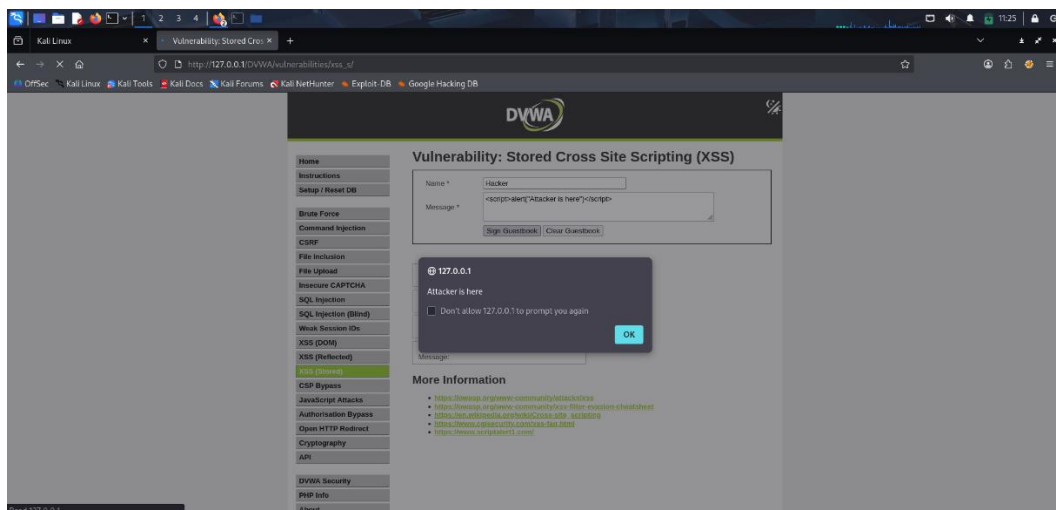
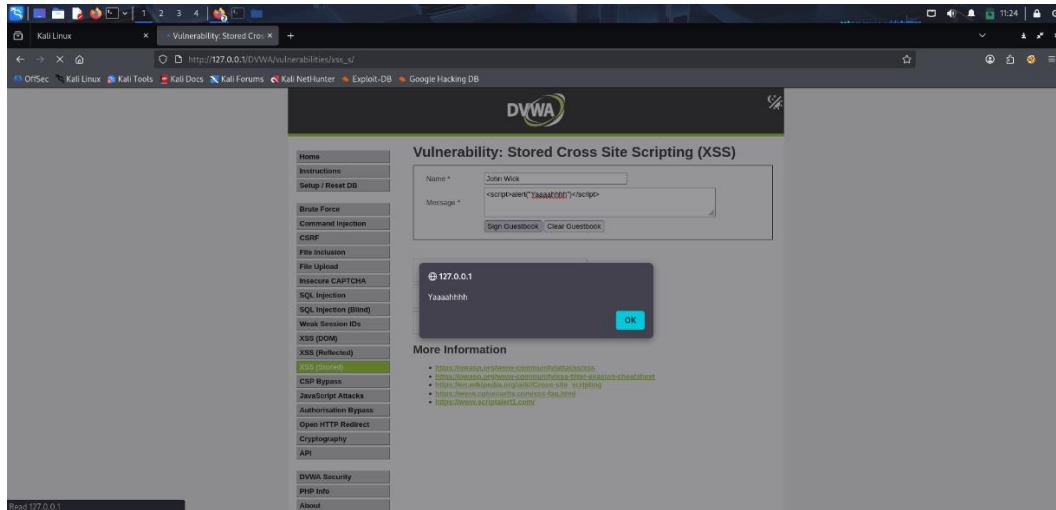
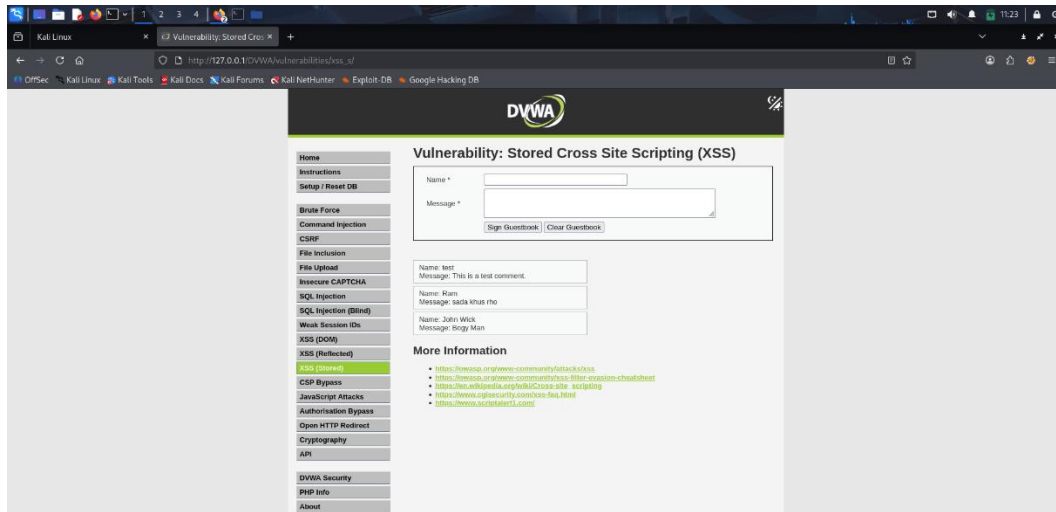
} catch (Exception $e) {
    echo 'Caught exception: ' . $e->getMessage();
}
break;
}
}
?>
```

## 2. Cross-Site Scripting (XSS):-



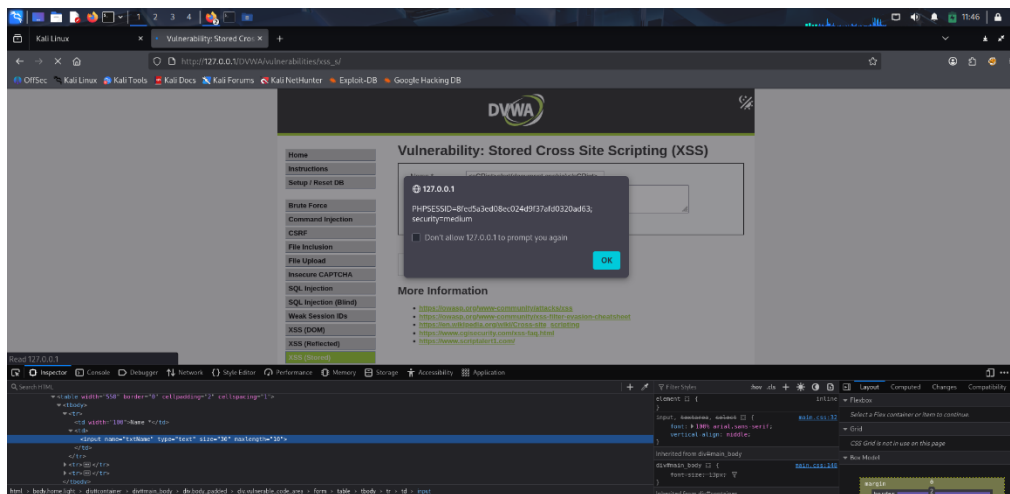
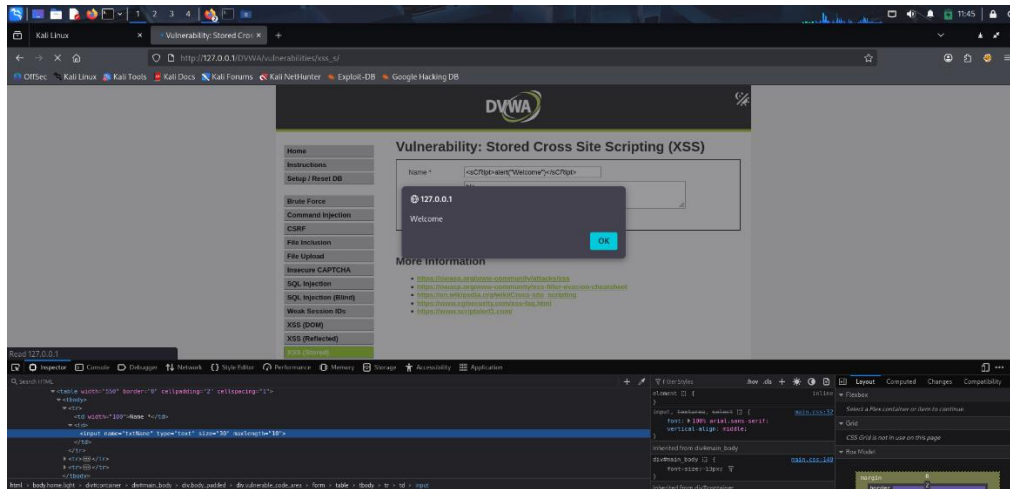
## Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025

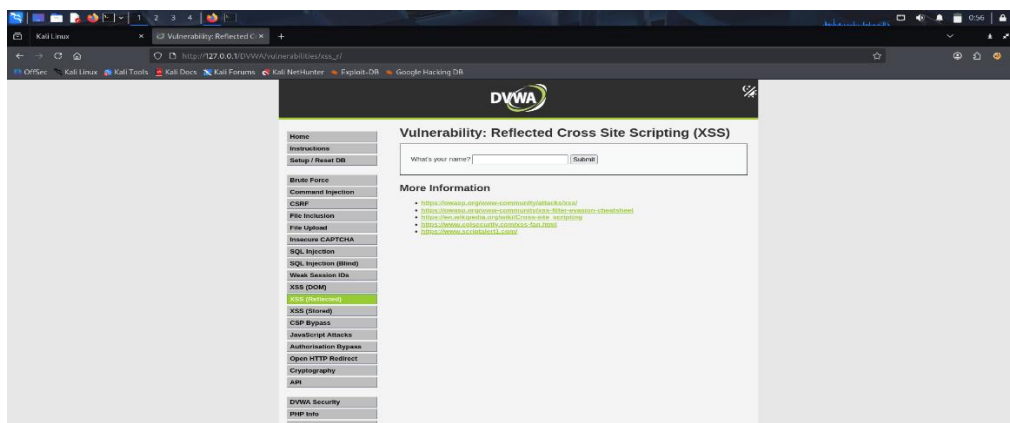


## Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025

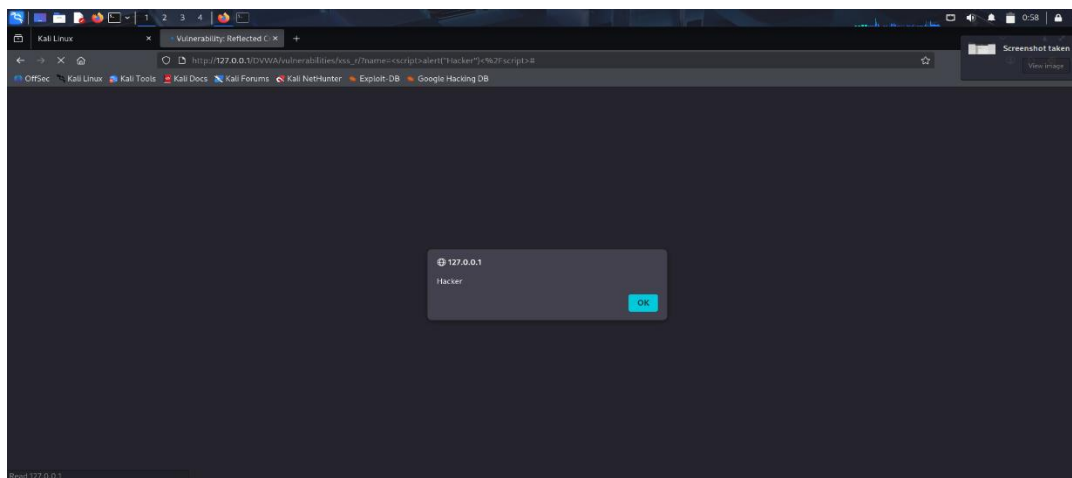
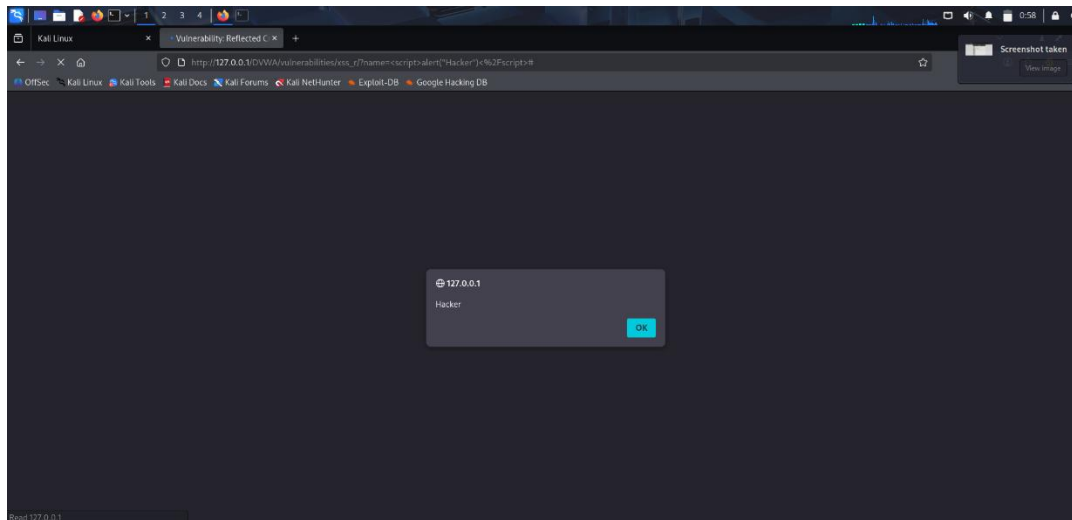
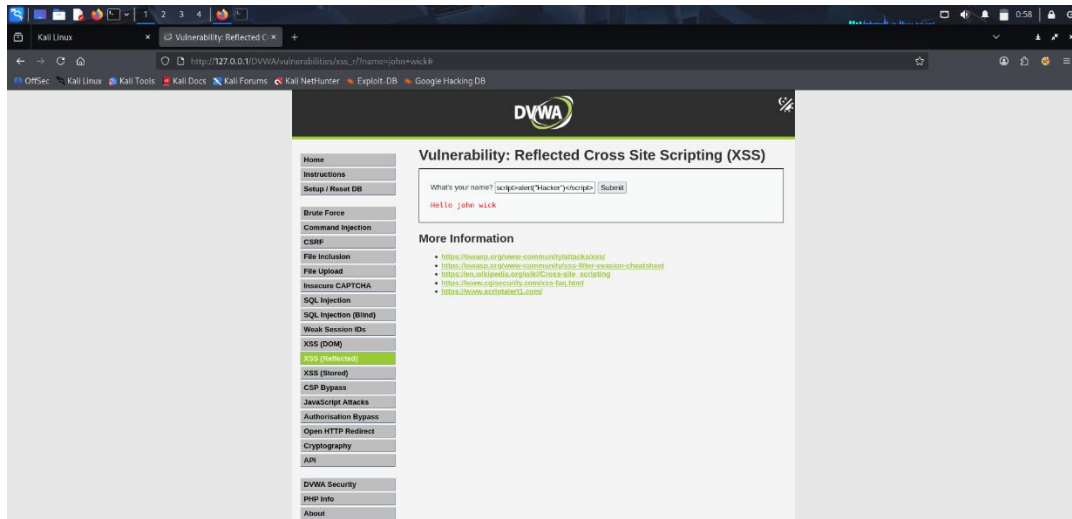


### Reflected XSS using query parameters:-



## Task 3 — Web Application Security

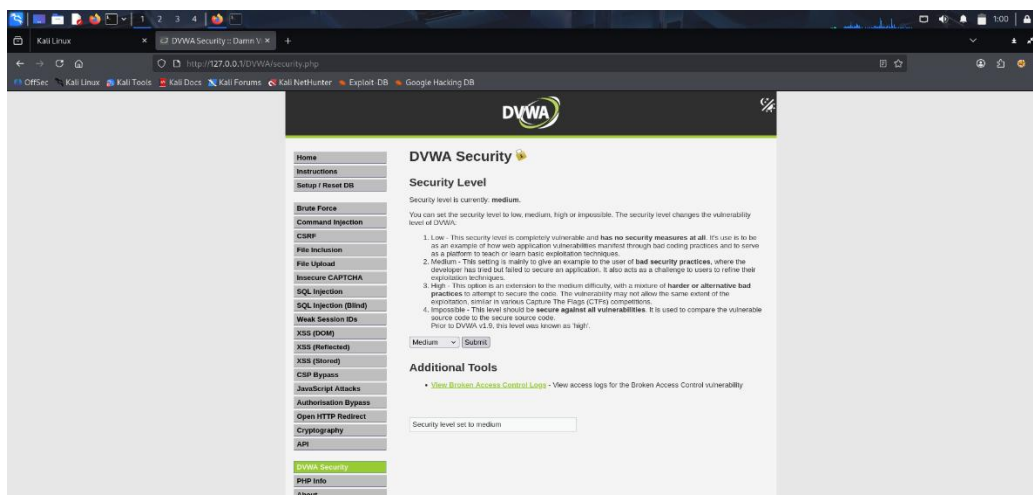
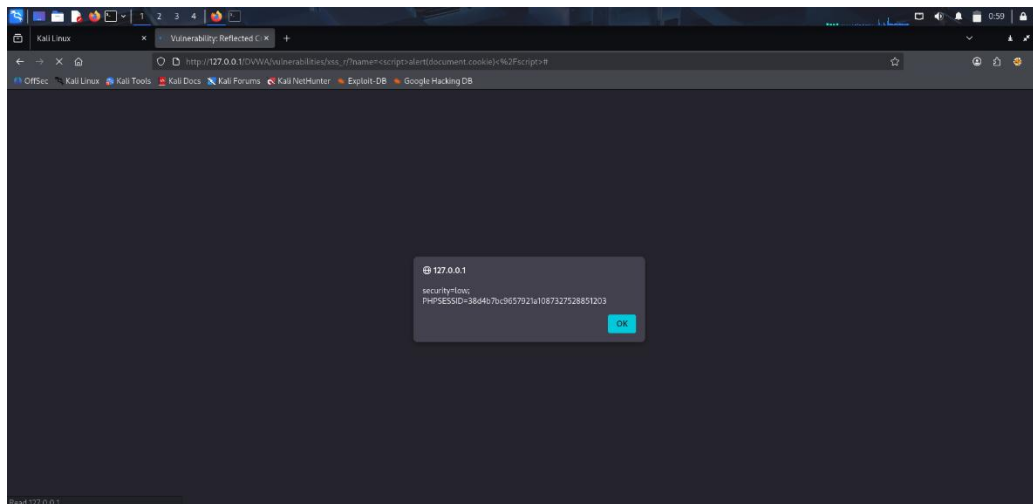
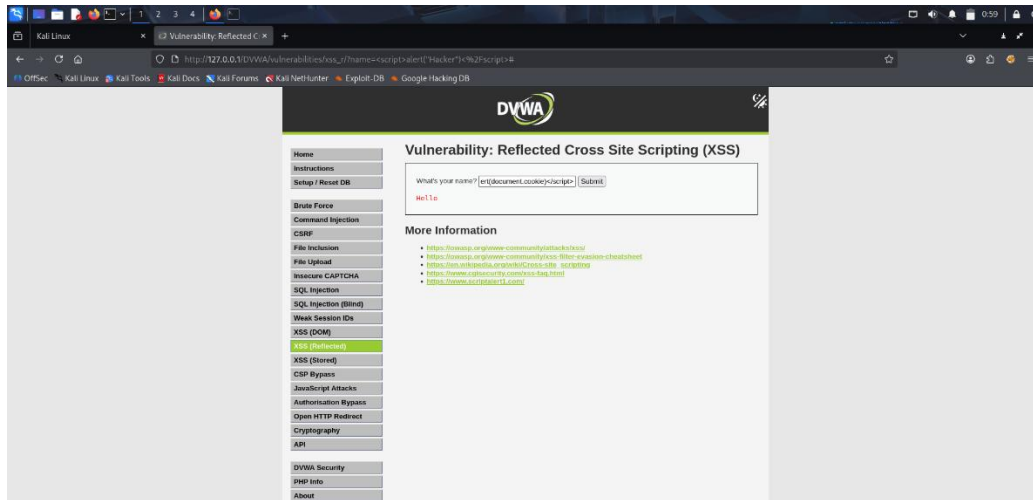
Intern Name: **Nitesh Sharma** Date: **05/10/2025**





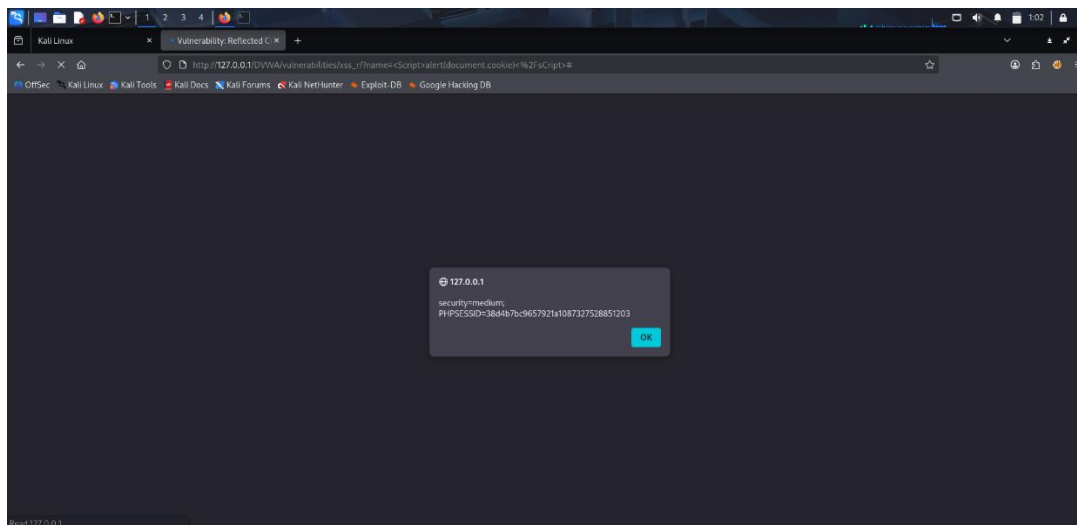
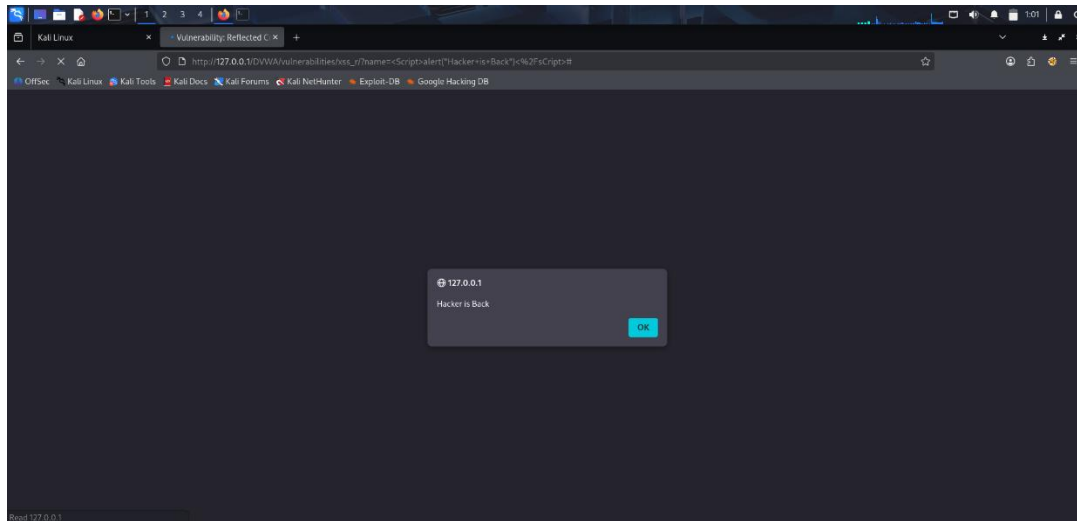
# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025



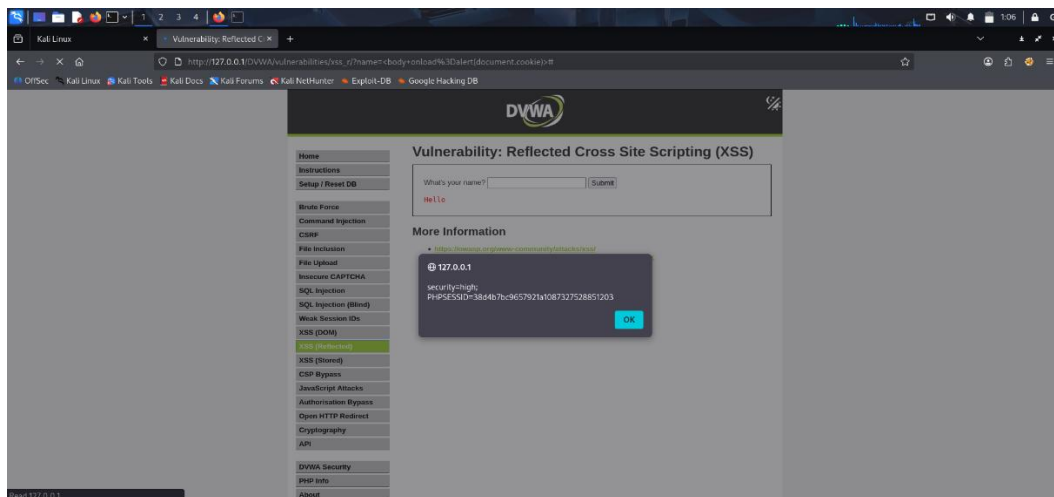
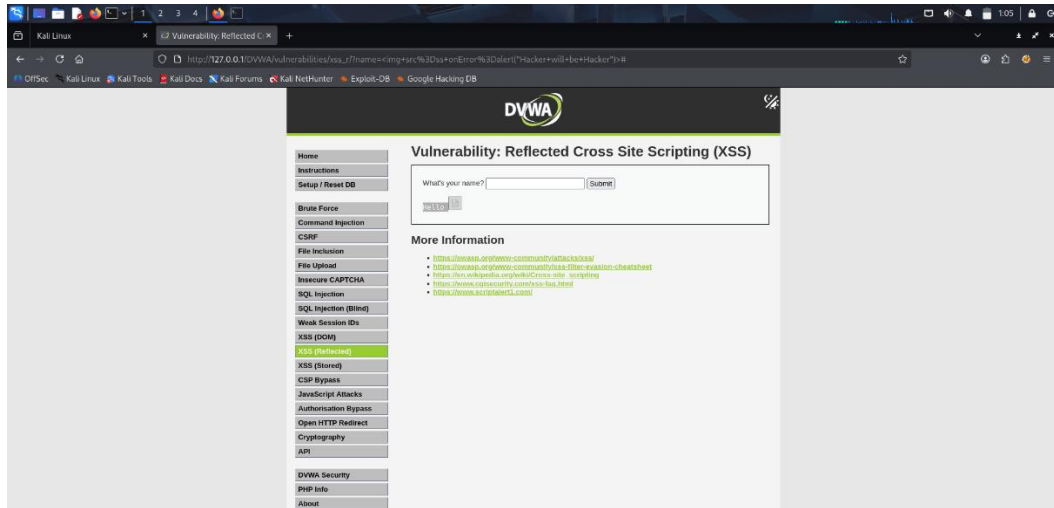
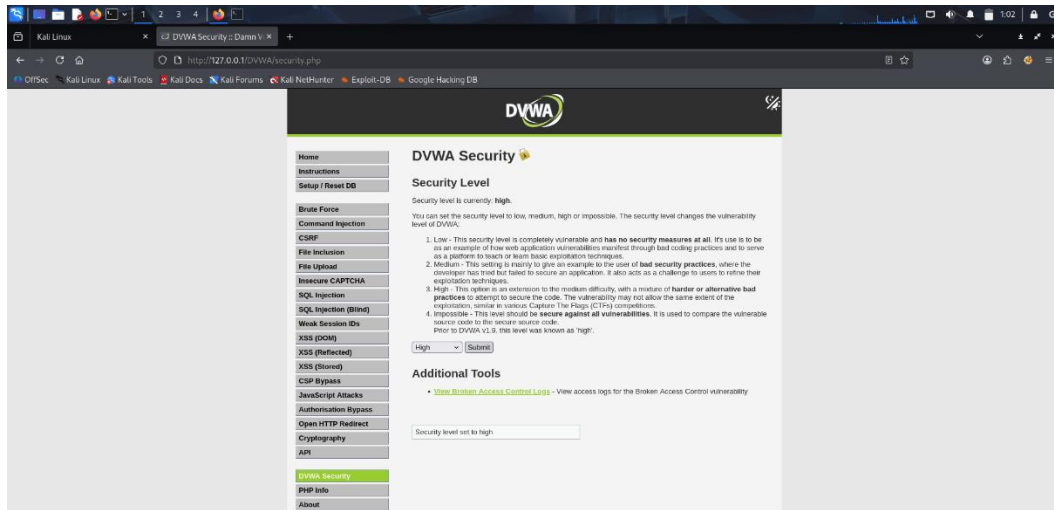
## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**



# Task 3 — Web Application Security

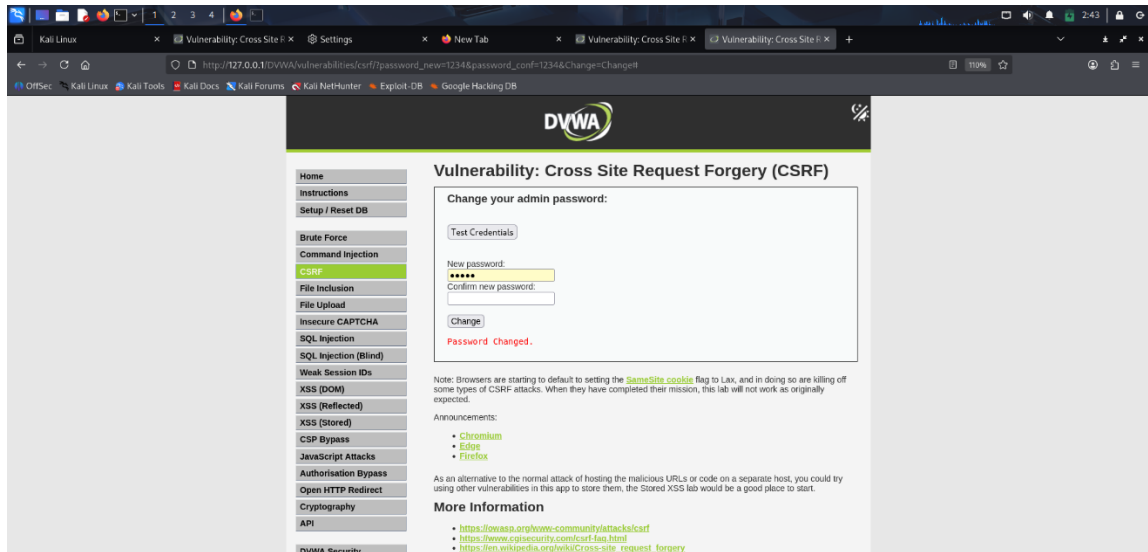
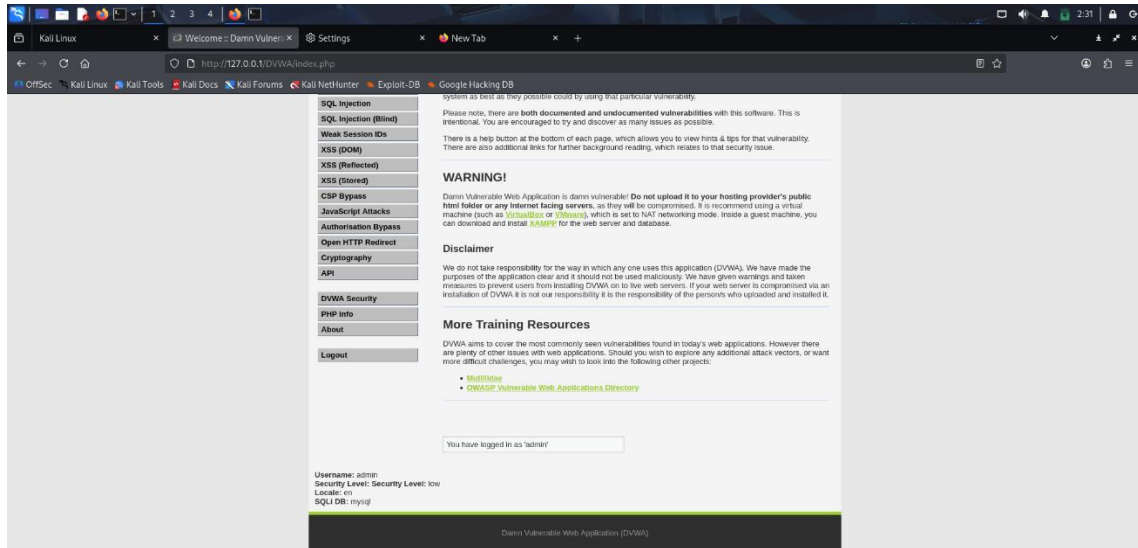
Intern Name: Nitesh Sharma Date: 05/10/2025



## Task 3 — Web Application Security

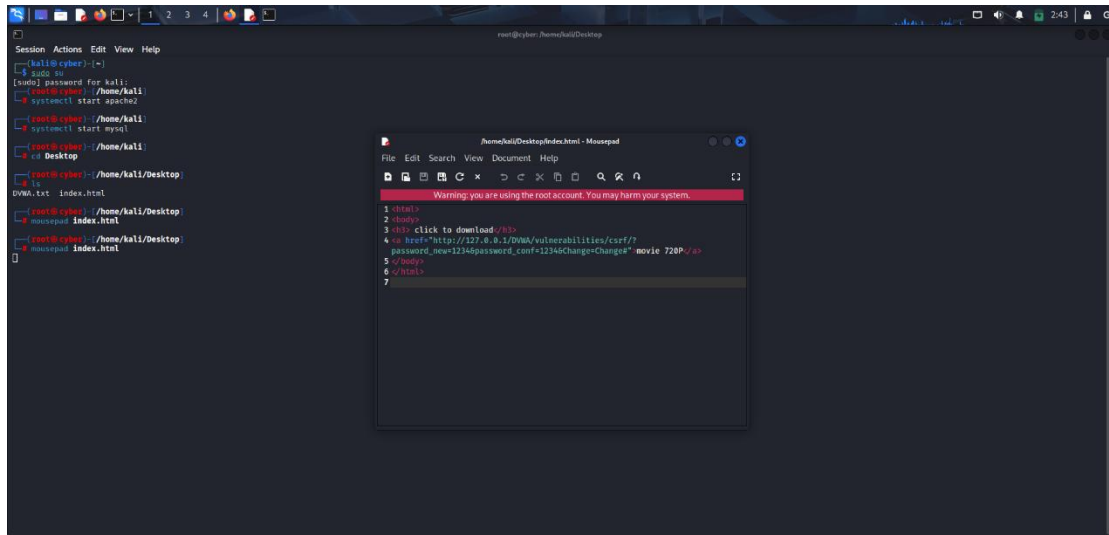
Intern Name: Nitesh Sharma Date: 05/10/2025

### 3. Cross-Site Request Forgery (CSRF):-



## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

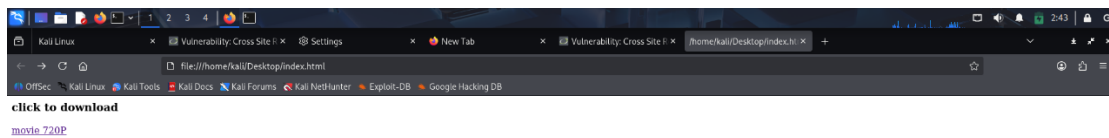


The screenshot shows a Kali Linux desktop environment. In the background, a terminal window displays the following commands and output:

```
root@kali:~/Desktop
kali@kali:~$ sudo su
[sudo] password for kali:
root@kali:~# systemctl start apache2
root@kali:~# systemctl start mysql
root@kali:~# cd Desktop
root@kali:~/Desktop# mousepad index.html
root@kali:~/Desktop# mousepad index.html
```

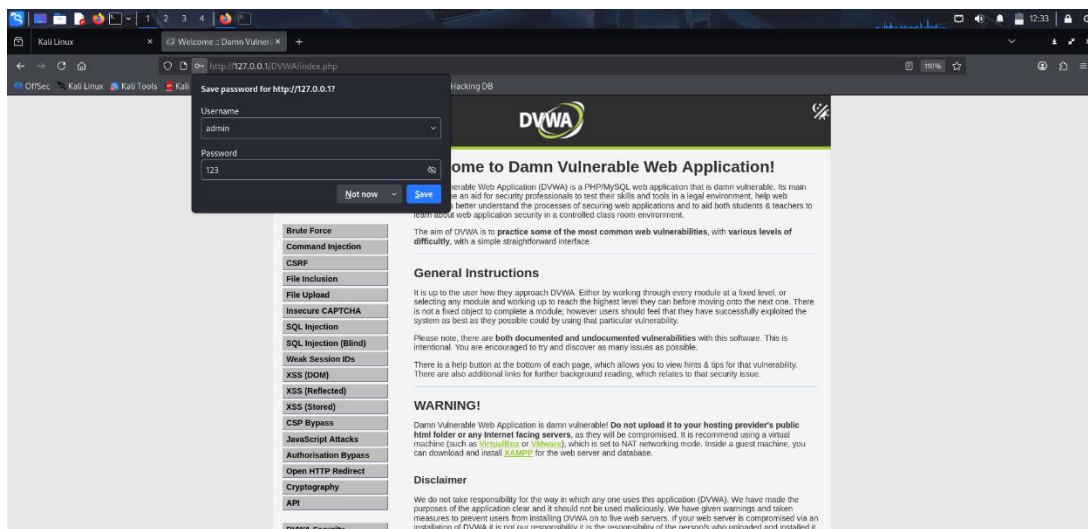
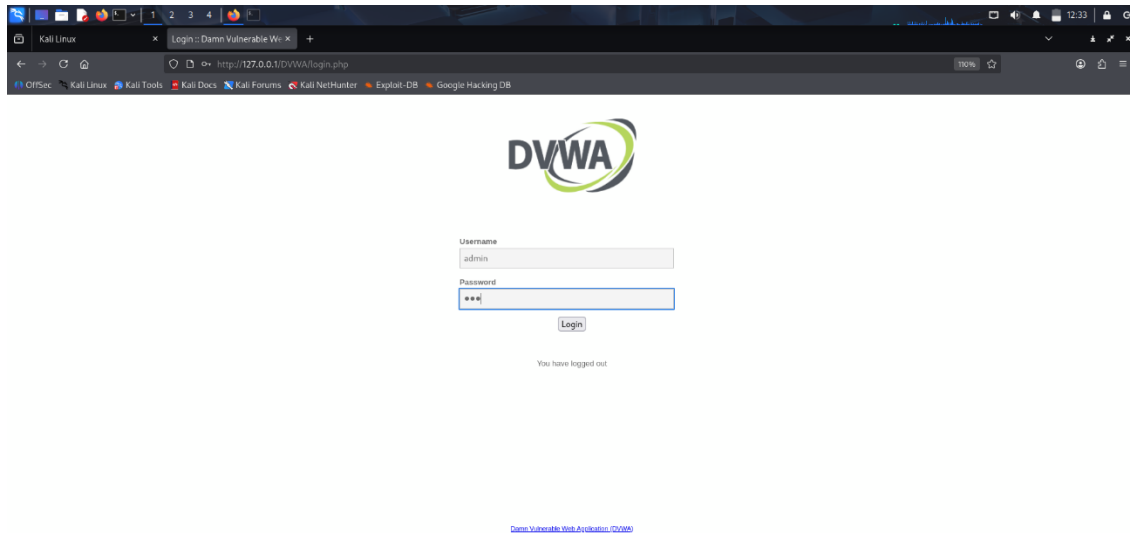
In the foreground, a mousepad editor window titled "index.html - Mousepad" is open, showing a CSRF payload:

```
1 <html>
2 <body>
3 <div> click to download </div>
4 <a href="http://127.0.0.1/DVWA/vulnerabilities/csrf/?
  password_new=123456password_conf=123456Change=Change"> movie 720P </a>
5 </body>
6 </html>
7
```

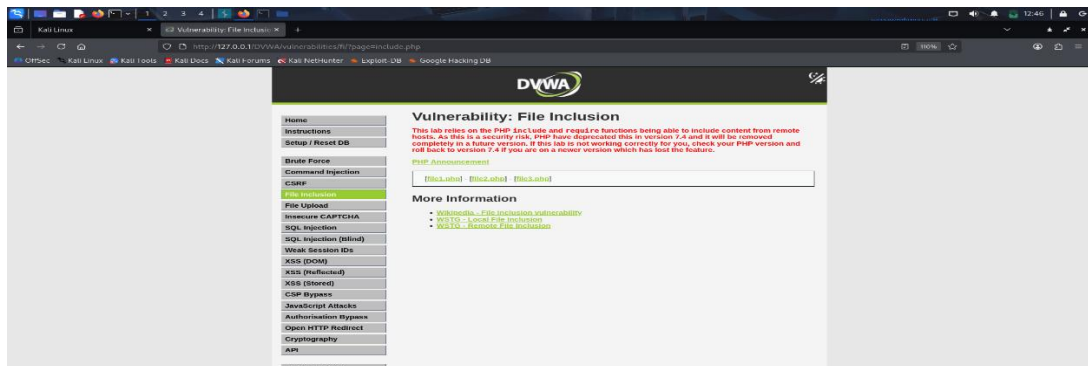


## Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025

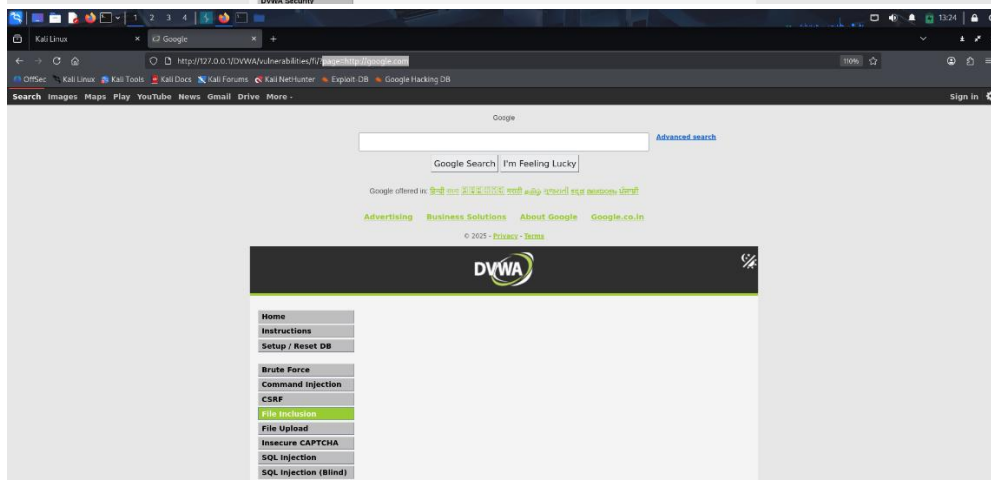
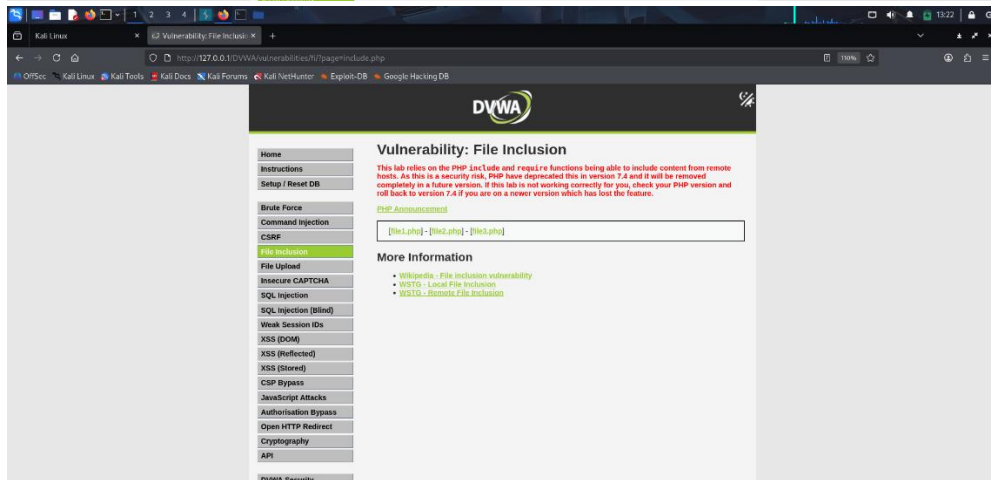
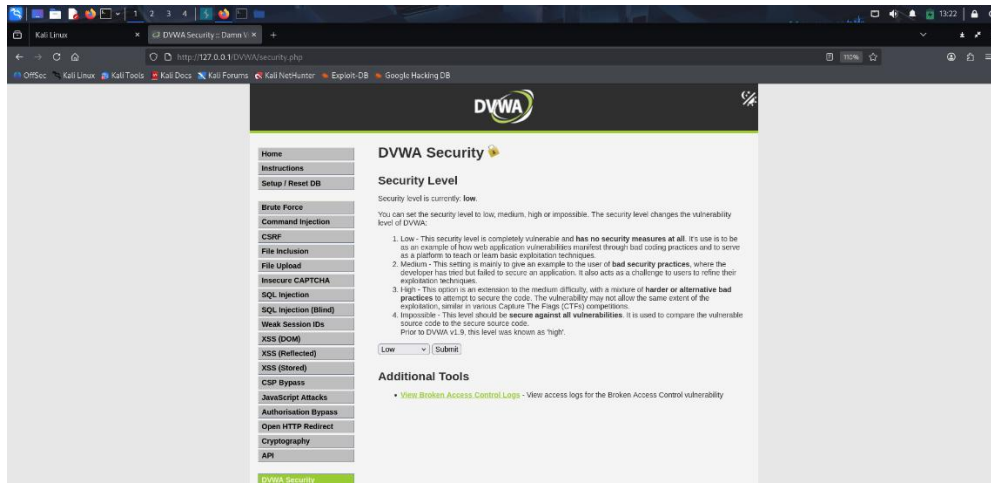


### 4. File inclusion attack:-



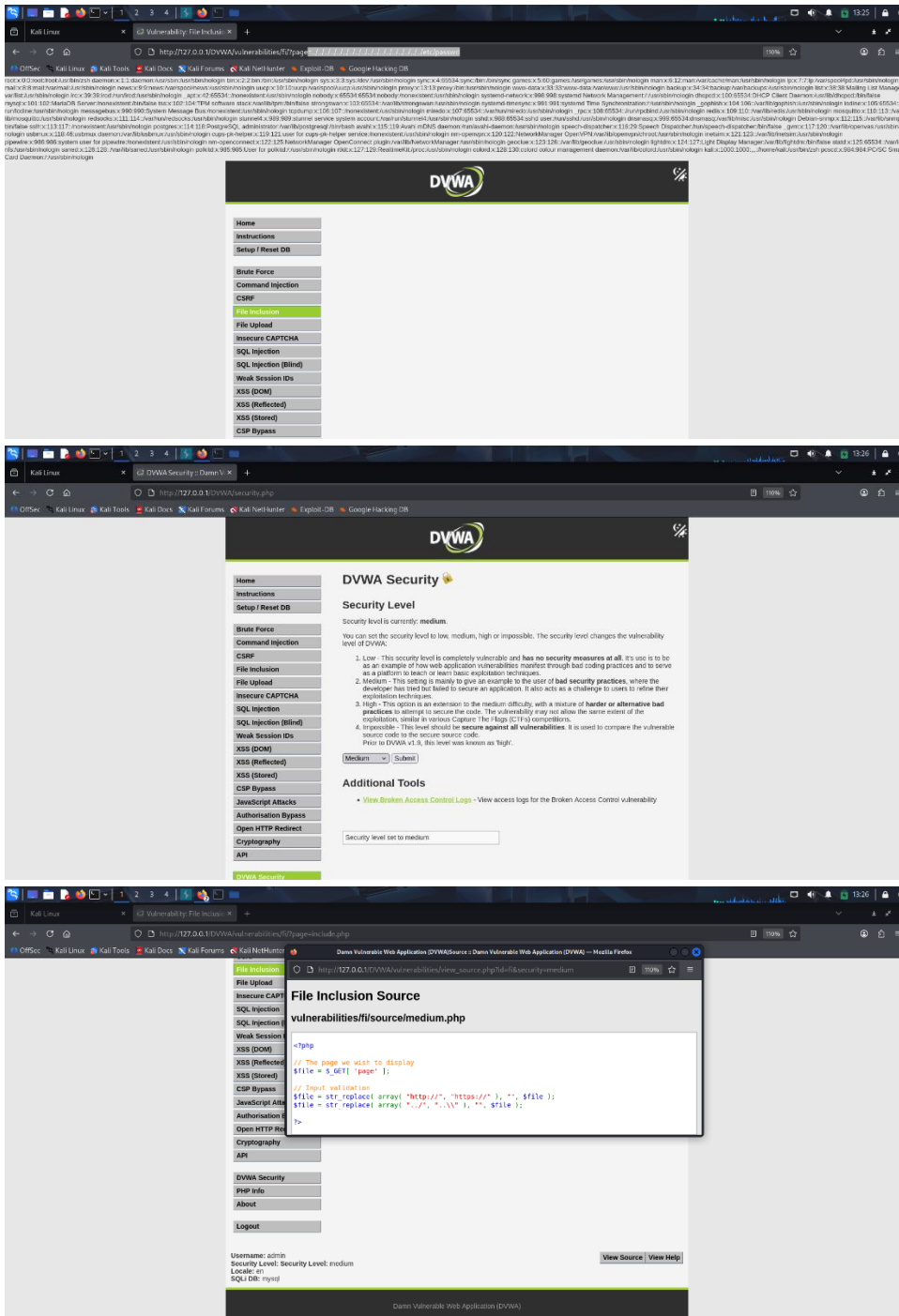
# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025



# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025



The screenshot shows a Kali Linux terminal window with a web browser displaying the DVWA (Damn Vulnerable Web Application) homepage. The browser's address bar shows the URL `http://172.0.0.10/VWw/vulnerabilities/fi/?page=...`. The DVWA homepage features a sidebar menu with various attack categories like Brute Force, Command Injection, CSRF, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), and CSP Bypass. The main content area displays the 'DVWA Security' section, indicating the current security level is 'medium'. Below this, there are instructions on how to set the security level and a list of additional tools. A 'File Inclusion Source' window is open, showing the source code of the 'vulnerabilities/fi/source/medium.php' file. The code includes a header, a form, and a footer. The form has a 'Submit' button. The footer contains the text 'Damn Vulnerable Web Application (DVWA)'.

```
<?php
// The page we wish to display
$file = $_GET['page'];

// Input validation
$file = str_replace(array("http://", "https://", ""), "", $file);
$file = str_replace(array("../", "..\\", ""), "", $file);

?>
```

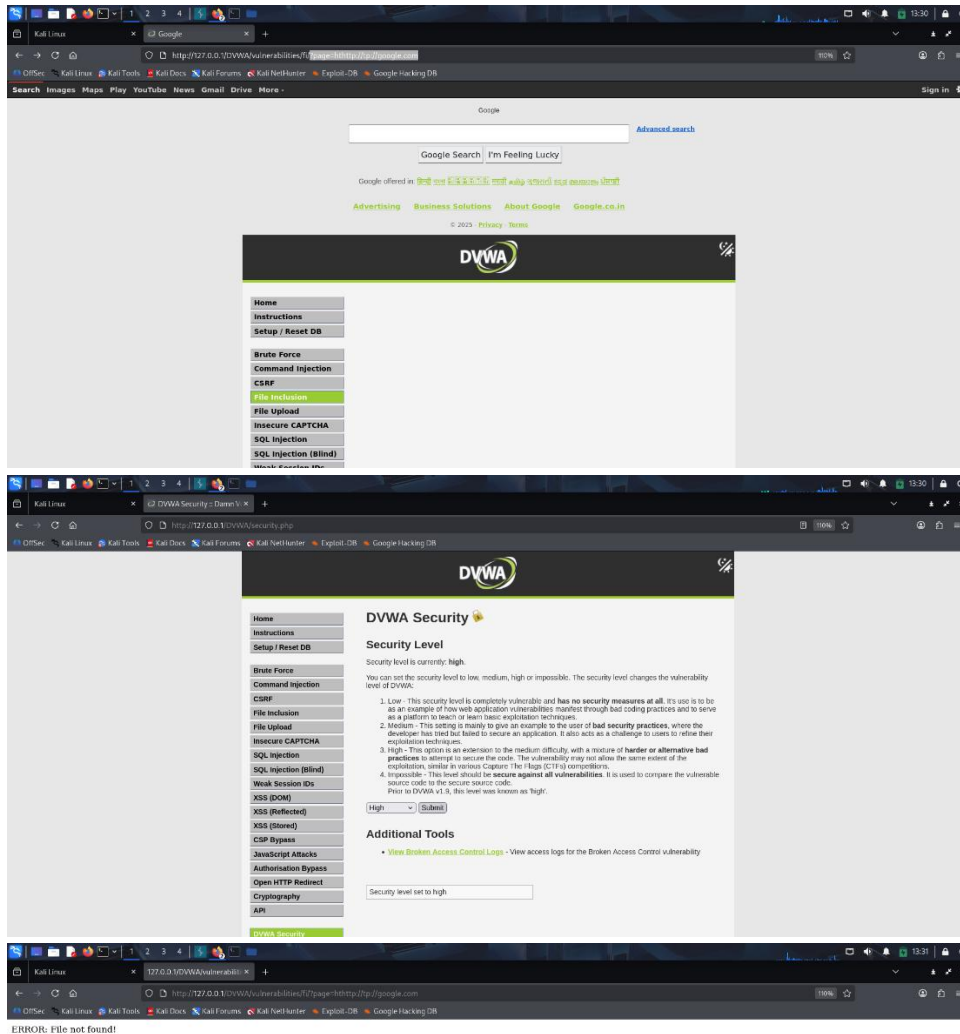
Username: admin  
Security Level: Security Level: medium  
Location: /  
SQL DB: mysql

Damn Vulnerable Web Application (DVWA)



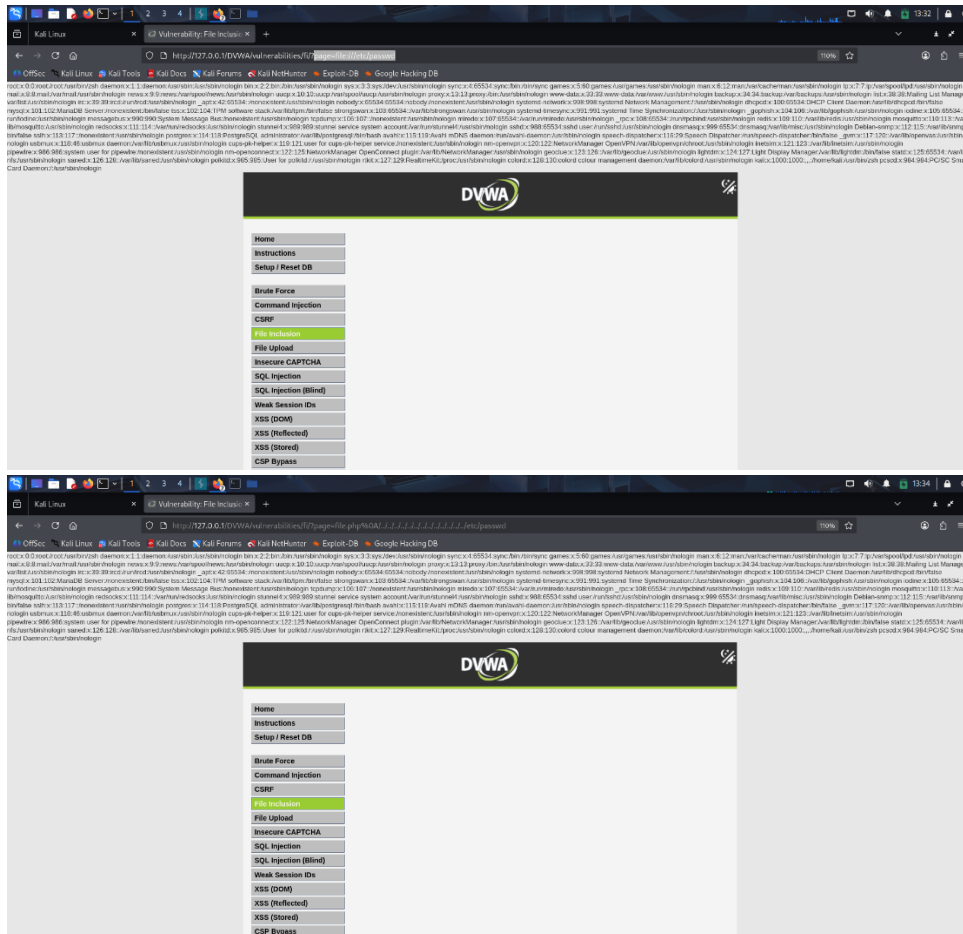
## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

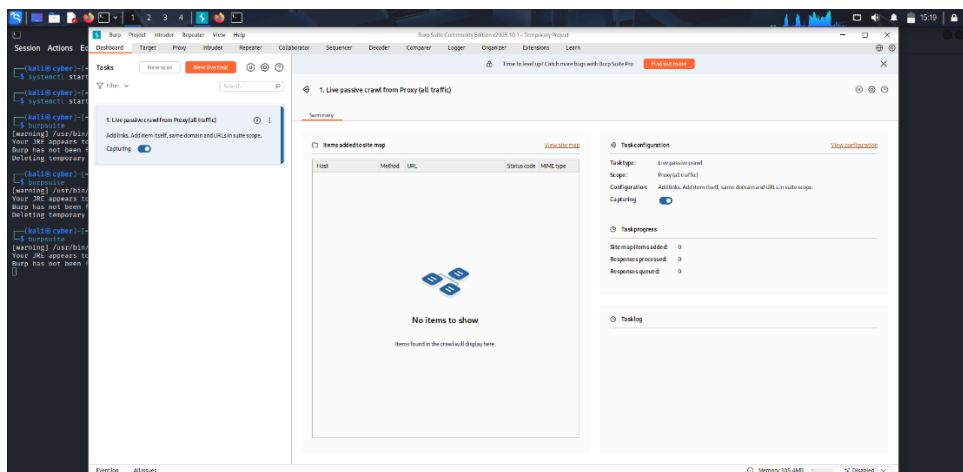


# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025



## 5. Burp-Suite Advanced:-



## Task 3 — Web Application Security

Intern Name: **Nitesh Sharma** Date: **05/10/2025**

The top screenshot shows a web browser window displaying the DVWA (Damn Vulnerable Web Application) login page. The page has a logo at the top, followed by input fields for 'Username' (containing 'admin') and 'Password' (containing '---'). A 'Login' button is below the fields. A message 'CSRF token is incorrect' is displayed below the button. A link 'Damn Vulnerable Web Application (DVWA)' is at the bottom.

The bottom screenshot shows the Burp Suite interface. The 'HTTP History' tab is active, showing a list of requests. The first request is a POST to 'http://127.0.0.1/DVWA/login.php' at 15:20:20.5. The 'Request' tab is selected, showing the raw HTTP request details. The 'Inspector' tab is also visible on the right, showing the request attributes and headers.

**Request Details:**

- 1 POST /DVWA/login.php HTTP/1.1
- 2 Host: 127.0.0.1
- 3 Content-Length: 83
- 4 Cache-Control: max-age=0
- 5 sec-ch-ua: "Chrome";v="141", "Not7A\_Brand";v="8"
- 6 sec-ch-ua-mobile: ?0
- 7 sec-ch-ua-platform: "Linux"
- 8 Accept-Language: en-GB,en;q=0.9
- 9 Origin: http://127.0.0.1
- 10 Content-Type: application/x-www-form-urlencoded
- 11 Upgrade-Insecure-Requests: 1
- 12 User-Agent: Mozilla/5.0 (X11; Linux x86\_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
- 13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,\*/\*;q=0.8,application/signed-exchange;v=b3;q=0.7
- 14 Sec-Fetch-Site: same-origin
- 15 Sec-Fetch-Mode: navigate
- 16 Sec-Fetch-User: ?1
- 17 Sec-Fetch-Dest: document
- 18 Referer: http://127.0.0.1/DVWA/login.php
- 19 Accept-Encoding: gzip, deflate, br

# Task 3 — Web Application Security

Intern Name: Nitesh Sharma Date: 05/10/2025

The image displays two screenshots from a web application security exercise. The top screenshot shows the Burp Suite interface with a captured HTTP request to `http://127.0.0.1/DVWA/login.php`. The request is a POST method with a body containing session cookies and a token. The bottom screenshot shows a web browser displaying the 'Damn Vulnerable Web Application' (DVWA) homepage. The page includes a navigation menu on the left with links like Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript Attacks, Authorisation Bypass, Open HTTP Redirect, Cryptography, API, DVWA Security, PHP Info, and About. The main content area welcomes the user and provides general instructions and a warning about the application's purpose as a security training tool.

**Burp Suite Request Details:**

Time	Type	Direction	Method	URL	Status code	Length
15:20:20.5	New	Request	Post	http://127.0.0.1/DVWA/login.php		

**Request Body (Pretty):**

```
5 sec-ch-ua: "Chromium"v="141", "Not?A_Brand"v="0"
6 sec-ch-ua-mobile: ?0
7 sec-ch-ua-platform: "Linux"
8 Accept-Language: en-gb,en;q=0.9
9 Origin: http://127.0.0.1
10 Content-Type: application/x-www-form-urlencoded
11 Upgrade-Insecure-Requests: 1
12 User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/141.0.0.0 Safari/537.36
13 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
14 Sec-Fetch-Site: same-origin
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Dest: document
17 Referer: http://127.0.0.1/DVWA/login.php
18 Accept-Encoding: gzip, deflate, br
19 Cookie: security=impossible; PHPSESSID=f260623455500846979f6d5b870b44
20 Connection: keep-alive
21
22 username=admin&password=admin&login_token=2bce6df991d969388b0f02e77778815
23
```

**Inspector Decoded Content:**

```
username=admin&password=admin&login_token=2bce6df991d969388b0f02e77778815
```

**Web Browser Content:**

## Welcome to Damn Vulnerable Web Application!

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

### General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level you can before moving onto the next one. There is not a fixed object to complete a module, however users should feel that they have successfully exploited the system as best as they possibly could by using that particular vulnerability.

Please note, there are both documented and undocumented vulnerabilities with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

### WARNING!

Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommended using a virtual machine (such as [VirtualBox](#) or [Vagrant](#)), which is set to NAT networking mode. Inside a guest machine, you can download and install [XAMPP](#) for the web server and database.

### Disclaimer

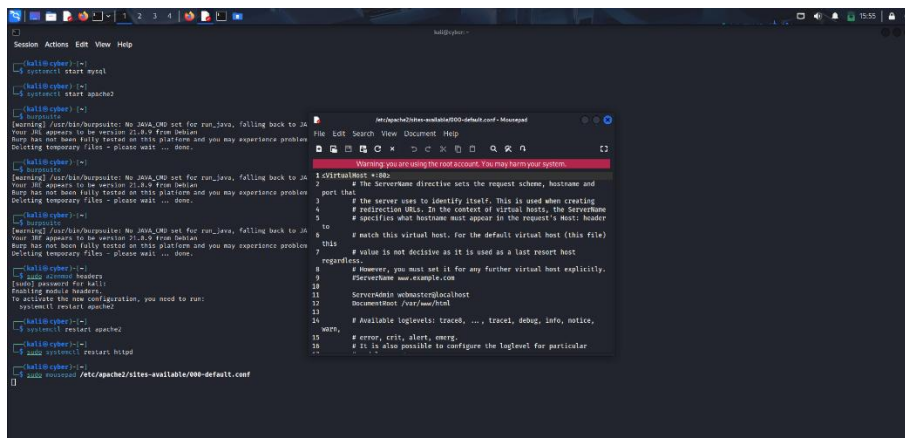
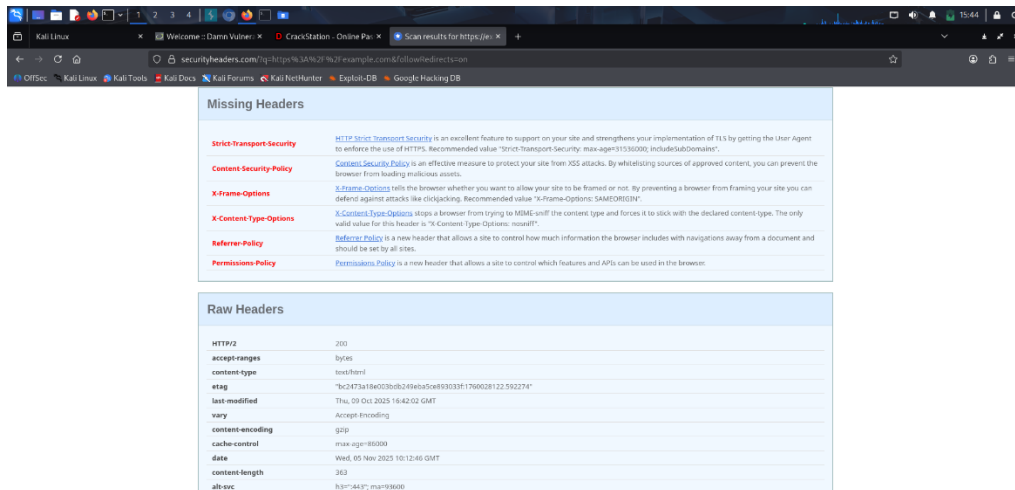
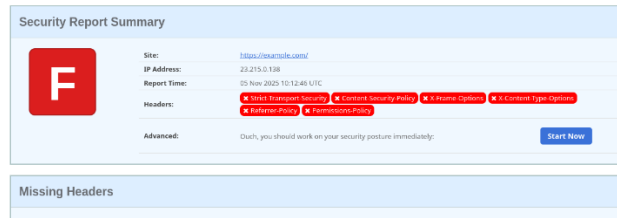
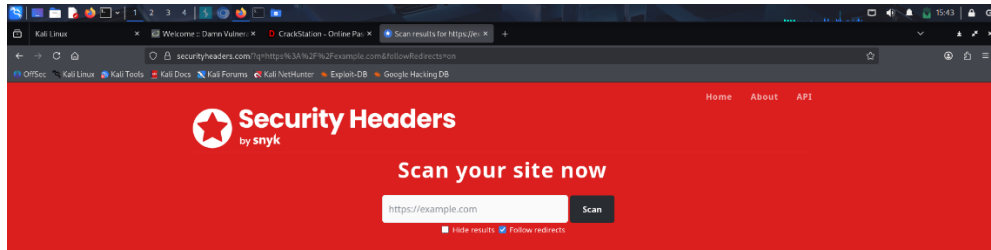
We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person who uploaded and installed it.

### More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want

Intern Name: **Nitesh Sharma**    Date: **05/10/2025**

## 6. Web-Security Headers:-



### **Task 3 — Web Application Security**

Intern Name: **Nitesh Sharma** Date: **05/10/2025**