Lab 1: Network Security Protocols and Configuration

Objectives

- Understand various network security protocols and their implementations.
- Gain hands-on experience configuring network devices with security protocols.
- Learn to analyse and verify the security configurations.

Lab Activities

1. Overview of Network Security Protocols

- Begin with a lecture or reading on key network security protocols such as:
    - SSL/TLS: Securing data in transit.
    - IPsec: Protecting IP packets.
    - SSH: Secure shell for secure remote access.
    - VPN: Virtual Private Network for secure connections.
    - 802.1X: Port-based network access control.
- Discuss the importance of each protocol in securing communications.

Network Security Protocols:

In network security, protocols are essential tools that ensure confidentiality, integrity, and authentication of data and communications. Below are key security protocols, their definitions, and their importance in securing modern networks.

SSL/TLS: Secure Sockets Layer / Transport Layer Security

- **Definition:**
  SSL (now largely deprecated) and its successor TLS are cryptographic protocols that provide secure communication over a network. They are primarily used to encrypt data in transit, ensuring privacy between client and server.
- **Importance:**
    - Prevents data interception by encrypting web traffic (e.g., HTTPS).
    - Authenticates servers via digital certificates to ensure users are connecting to legitimate sites.
    - Ensures data integrity so information isn't modified during transmission.
- **Common Use:** Secure websites (HTTPS), secure email, and VoIP.

IPsec - Internet Protocol Security

- **Definition:**
  IPsec is a suite of protocols used to secure IP communications by authenticating and/or encrypting each IP packet in a communication session.
- **Importance:**
  - Works at the network layer, securing all data regardless of the application.
  - Provides end-to-end encryption in VPNs or secure internal networks.
  - Ensures data confidentiality, authenticity, and integrity.
- **Common Use**: Site-to-site VPNs, secure communication between remote offices or networks.

SSH – Secure Shell

- **Definition:**
  SSH is a protocol that enables secure remote access to a computer or server over an unsecured network using encryption.
- **Importance:**
  - Replaces insecure protocols like Telnet and FTP.
  - Encrypts login credentials, terminal sessions, and file transfers.
  - Supports strong authentication mechanisms like public/private key pairs.
- Common Use: Remote administration, secure file transfer (SCP/SFTP), and remote command execution.

VPN – Virtual Private Network

- **Definition:**
  A VPN creates a secure, encrypted tunnel between a user's device and a private network over the internet.
- **Importance:**
  - Masks IP addresses and secures traffic from attackers, especially on public Wi-Fi.
  - Provides private, secure access to internal enterprise resources from remote locations.
  - Ensures data cannot be intercepted, viewed, or altered during transmission.
- **Common Use:** Remote work access, interconnecting branch offices, secure browsing.

802.1X – Port-Based Network Access Control

- **Definition:**
  802.1X is a network access control protocol that authenticates devices before granting them access to the network through a port (wired or wireless).

- **Importance:**
  - Prevents unauthorized devices from connecting to the network.
  - Works with RADIUS servers to enforce identity-based access.
  - Enhances LAN and Wi-Fi security by requiring user authentication.
- **Common Use**: Enterprise wired LANs, wireless networks with WPA2-Enterprise, educational and government networks.

Conclusion

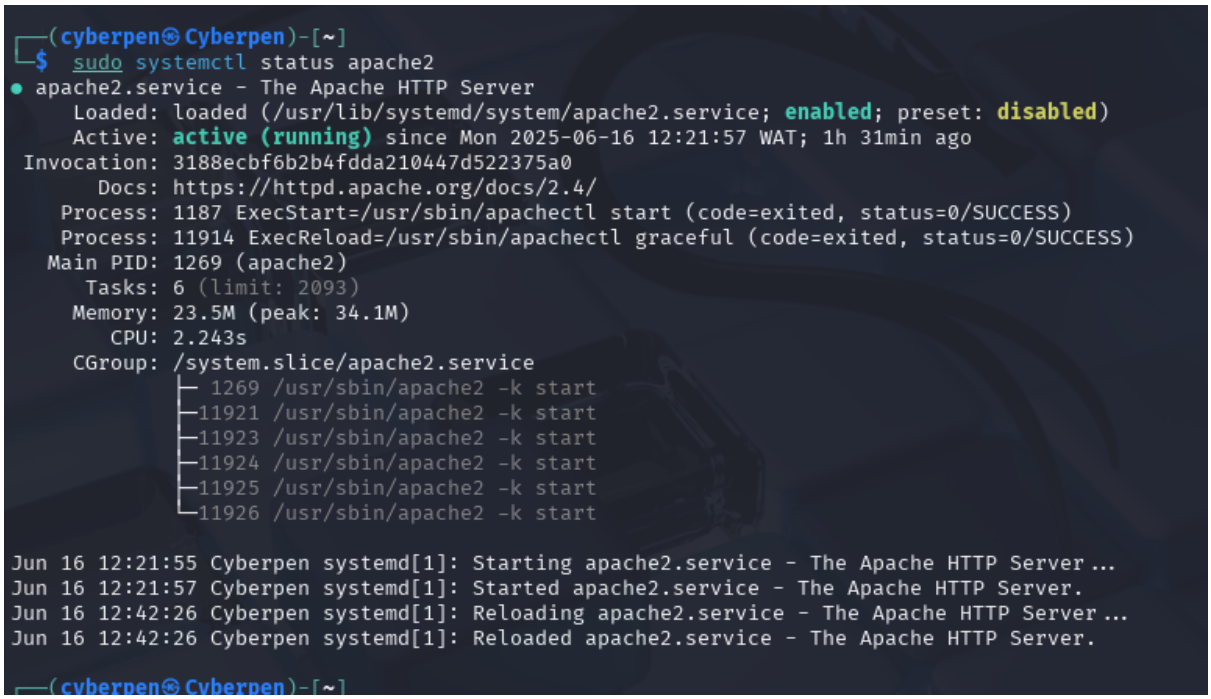
Each of these protocols plays a unique and vital role in securing modern communication systems:

- SSL/TLS secures data during transit.
- IPsec protects data at the network layer.
- SSH secures remote access and file operations.
- VPNs establish trusted private communication over public networks.
- 802.1X enforces access control and device authentication at the network edge.

Together, they form the foundation of defense-in-depth strategies, ensuring that networks remain resilient against internal and external threats.

2. Configuring SSL/TLS on a Web Server

- Set up a local web server (e.g., Apache or Nginx) on a Linux VM.



- Obtain a self-signed SSL certificate or use a tool like Let's Encrypt for a free SSL certificate.

```
┌──(cyberpen㉿Cyberpen)-[~]
└─$ sudo openssl req-x509-nodes-days 365-newkey rsa:2048 \-keyout /etc/ssl/private/apache-selfsigned.key \-
out /etc/ssl/certs/apache-selfsigned.c
Invalid command 'req-x509-nodes-days'; type "help" for a list.

┌──(cyberpen㉿Cyberpen)-[~]
└─$ sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 \
-keyout /etc/ssl/private/apache-selfsigned.key \
-out /etc/ssl/certs/apache-selfsigned.crt
.....+..+.....+++++++++++++++++++++++++++++++++++++++*...+.........+++++++++++++++++++++++++++++++
+++++++*.............+......+.+........+.......+..........+.......+.....+......+.........+.....+......+
...+.....+....+..+..............+.+...+.........+.....+...+...+.+.....+.+.............+....+.........+....
.+...+...+.+...+...+.....+...+....+...+.........+.....+.........+...+..++++++
..+...+...++++++++++++++++++++++++++++++++++++++++*.++++++++++++++++++++++++++++++++++++++++*........+.....+.+
...+..............+.........+..+...+...+...+...........+...........+.+........+.+......+.........+...+...+..+
......+....++++++
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:ng
State or Province Name (full name) [Some-State]:oyo
Locality Name (eg, city) []:ibadan
Organization Name (eg, company) [Internet Widgits Pty Ltd]:icdfa
Organizational Unit Name (eg, section) []:icdf
Common Name (e.g. server FQDN or YOUR name) []:www.192.168.74.68.com
Email Address []:cyberren.com
```

- Configure the web server to support HTTPS:
  - Enable SSL/TLS.

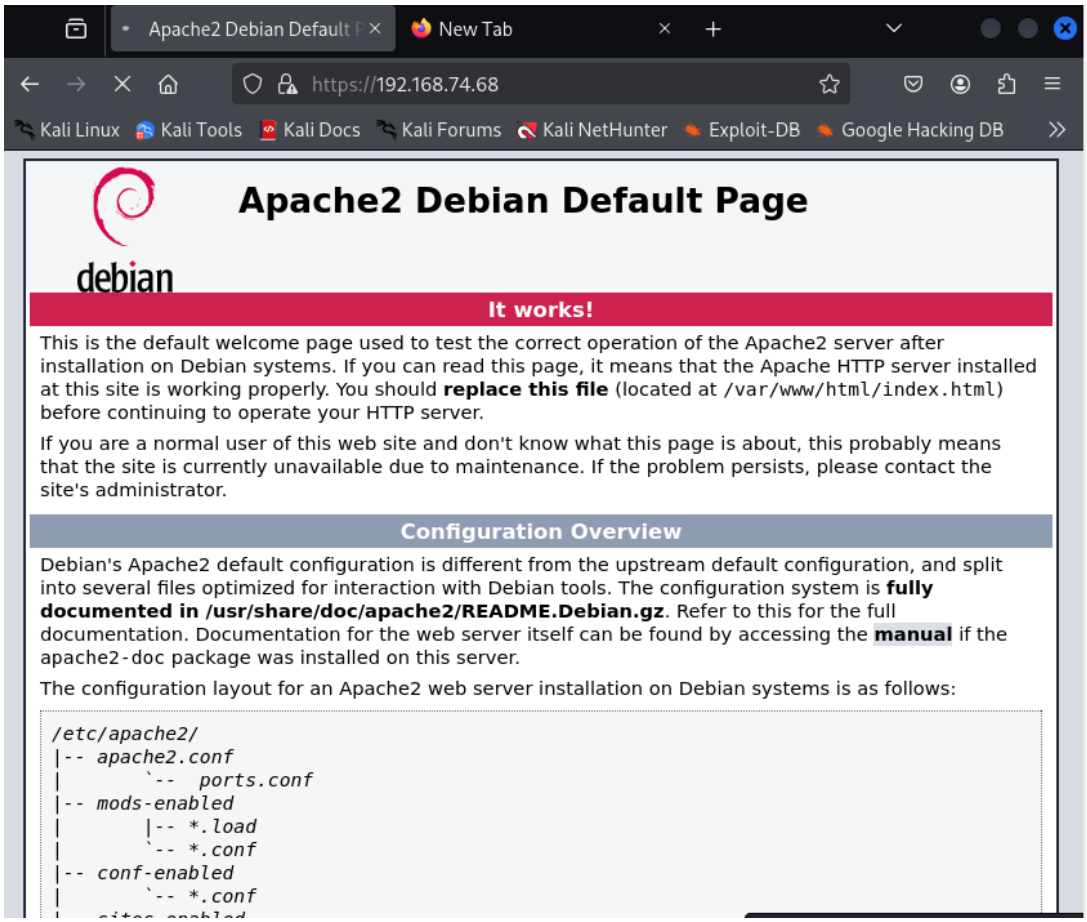
```
┌──(cyberpen㉿Cyberpen)-[~]
└─$ sudo a2enmod ssl
Considering dependency mime for ssl:
Module mime already enabled
Considering dependency socache_shmcb for ssl:
Module socache_shmcb already enabled
Module ssl already enabled

┌──(cyberpen㉿Cyberpen)-[~]
└─$
```

```
cyberpen@Cyberpen: ~
File  Actions  Edit  View  Help
 GNU nano 8.4                 /etc/apache2/sites-available/www.10.0.2.15.com-ssl
<VirtualHost *:443>
ServerAdmin admin@www.192.168.74.68.com
ServerName www.192.168.74.68.com
DocumentRoot /var/www/html

SSLEngine on
SSLCertificateFile  /etc/ssl/certs/apache-selfsigned.crt
SSLCertificateKeyFile  /etc/ssl/certs/apache-selfsigned.crt


ErrorLog ${APACHE_LOG_DIR}/error.log
CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

- Redirect HTTP traffic to HTTPS.

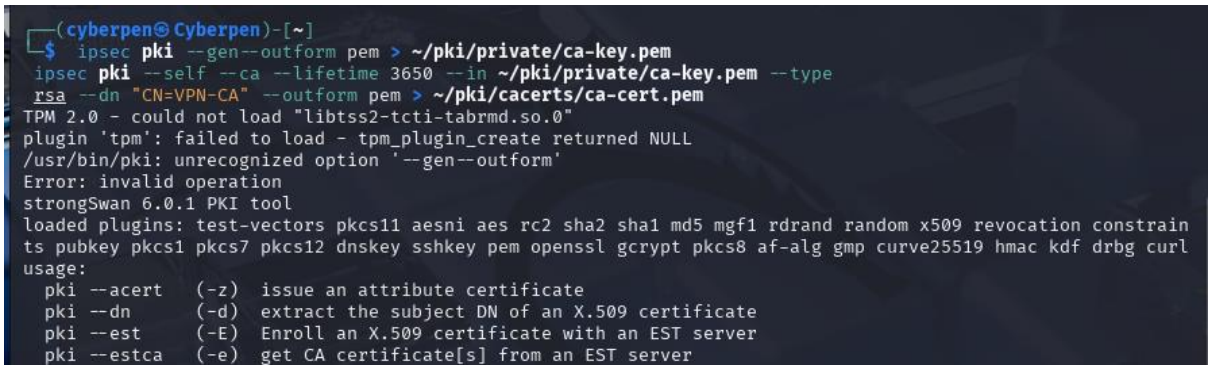
```
 GNU nano 8.4                    /etc/apache2/sites-available/000-default.conf
<VirtualHost *:80>
        # The ServerName directive sets the request scheme, hostname and port that
        # the server uses to identify itself. This is used when creating
        # redirection URLs. In the context of virtual hosts, the ServerName
        # specifies what hostname must appear in the request's Host: header to
        # match this virtual host. For the default virtual host (this file) this
        # value is not decisive as it is used as a last resort host regardless.
        # However, you must set it for any further virtual host explicitly.
        #ServerName www.example.com

        ServerAdmin webmaster@localhost
        DocumentRoot /var/www/html

        # Available loglevels: trace8, ..., trace1, debug, info, notice, warn,
        # error, crit, alert, emerg.
        # It is also possible to configure the loglevel for particular
        # modules, e.g.
        #LogLevel info ssl:warn

        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined

        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to
        # include a line for only one particular virtual host. For example the
        # following line enables the CGI configuration for this host only
        # after it has been globally disabled with "a2disconf".
        #Include conf-available/serve-cgi-bin.conf
 Redirect "/" "https://192.168.74.68.com/"
</VirtualHost>
```

- Test the configuration by accessing the web server through a web browser.



3. Implementing IPsec VPN

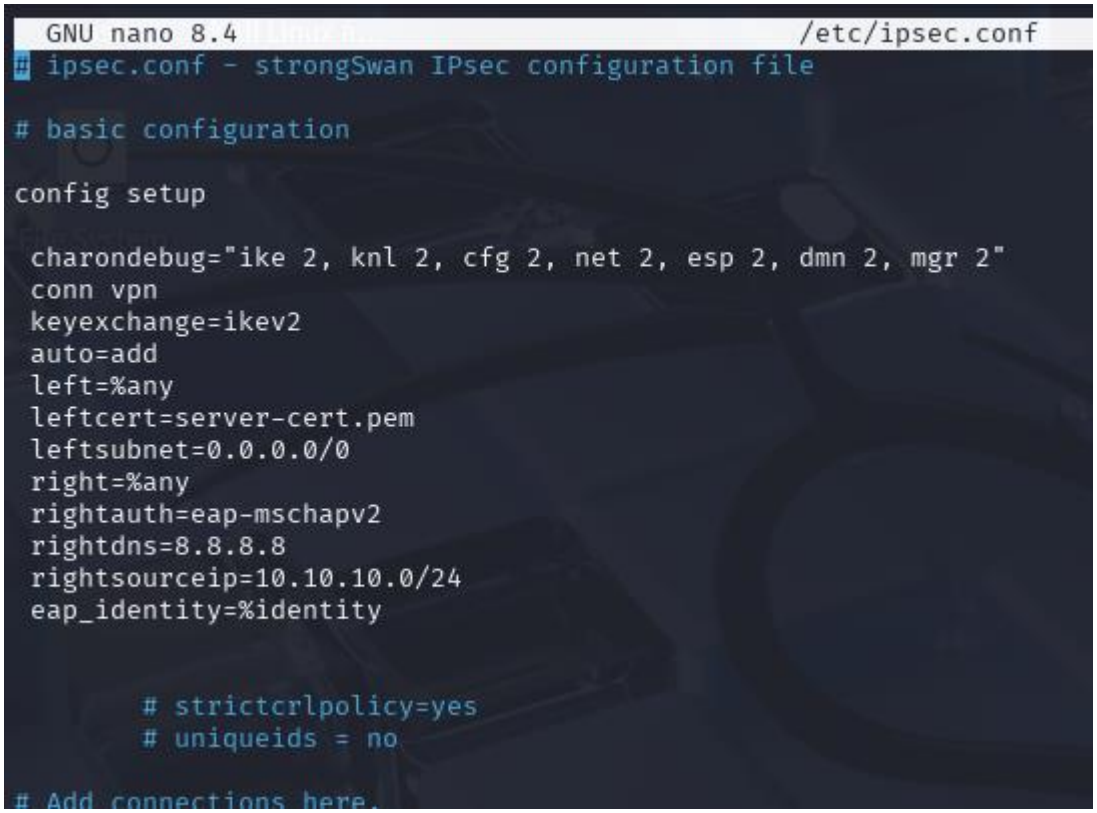
- Set up a VPN server using software such as OpenVPN or StrongSwan
    - Generate CA certificate:



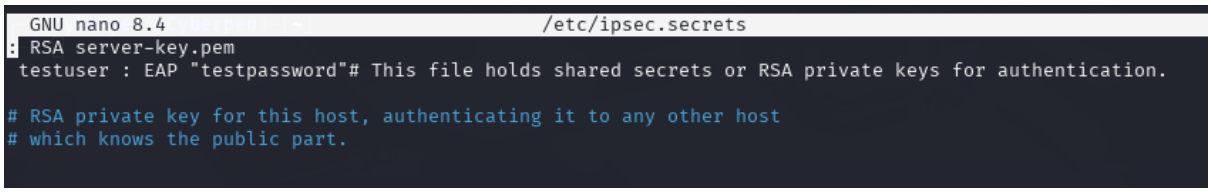
    - Generate server certificate

- Create client configurations and test connections to the VPN server.
  - Configure IPsec VPN: Copy certs to /etc/ipsec.d

  Edit /etc/ipsec.conf



- Ensure secure communication between clients and the server:



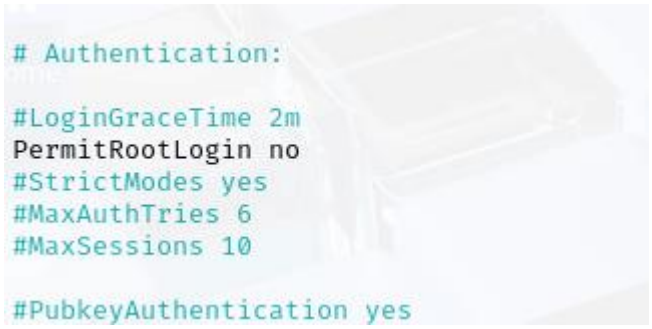
  - Monitor traffic to confirm that data is encrypted.

4. Configuring SSH for Secure Remote Access

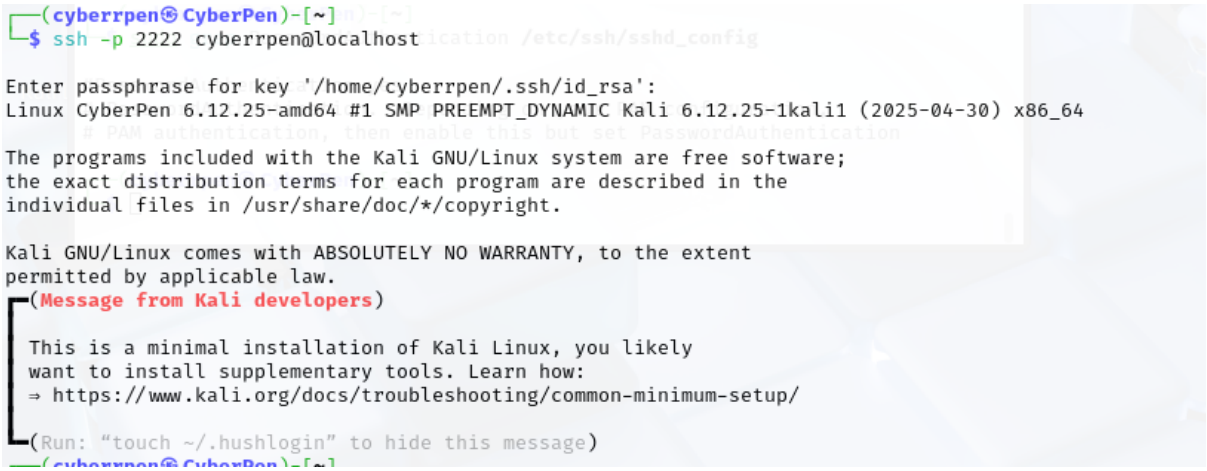
- Install and configure an SSH server on a Linux machine.

- Adjust settings to enhance security:
  - Change the default port (22) to a non-standard port.

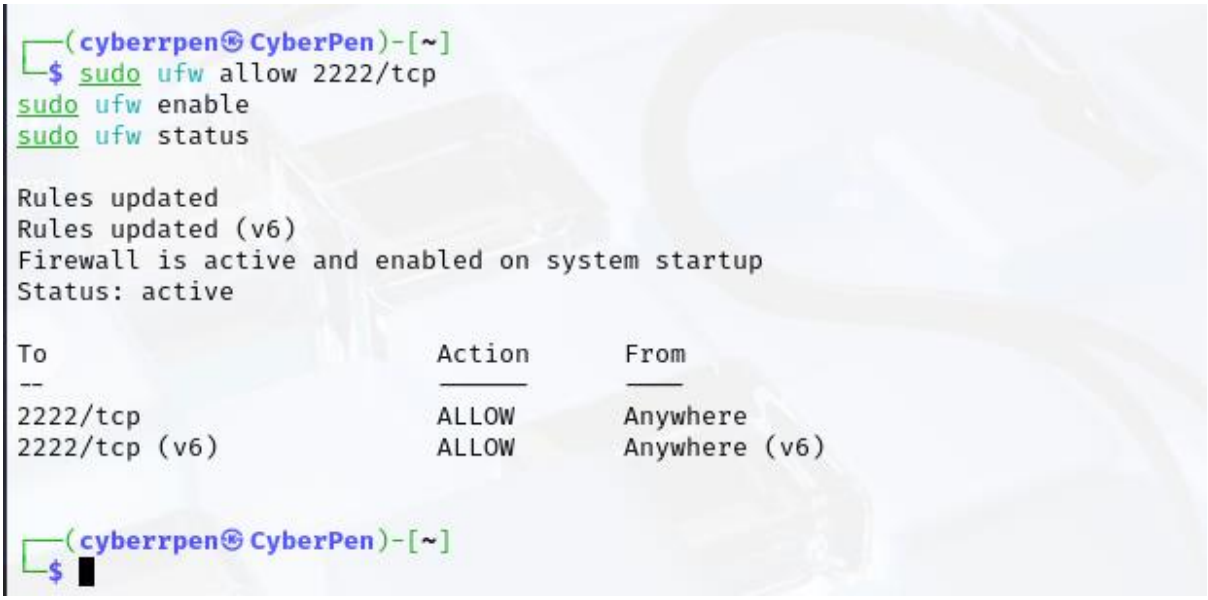
  - Disable root login.



- Implement public key authentication instead of password-based authentication.

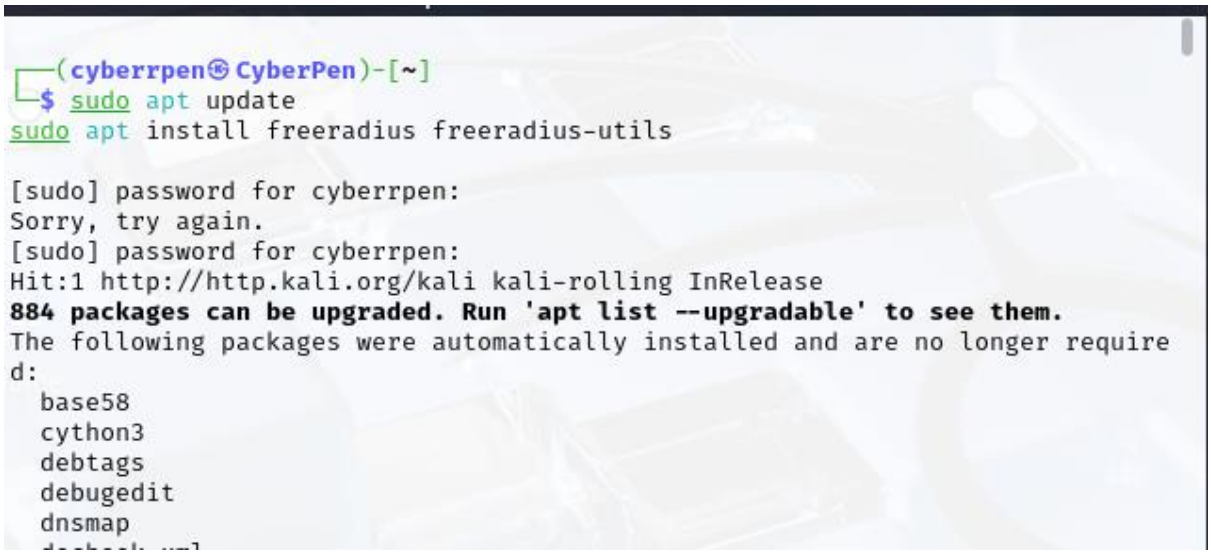


- Test SSH connections from a client machine, ensuring secure access.

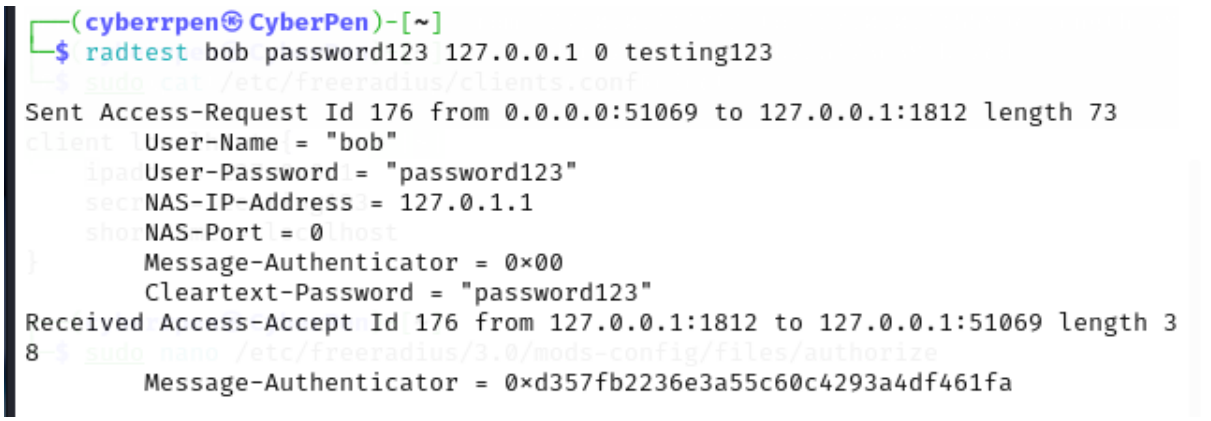


5. Using 802.1X for Network Access Control

- Simulate a network with a switch that supports 802.1X.



- Configure a RADIUS server to handle authentication requests.



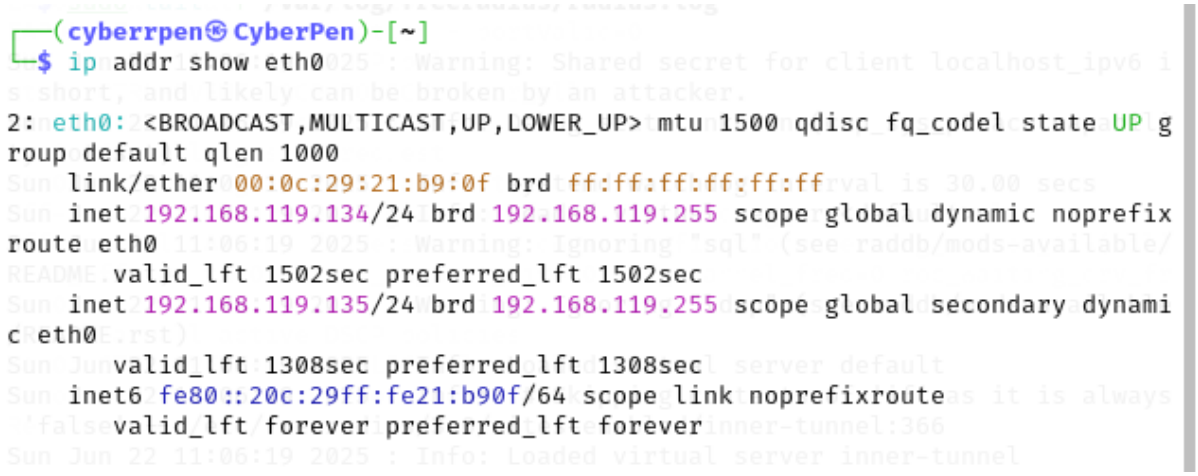
- Connect client devices to the network and verify that only authenticated devices can access resources.

```
   }
}
listen {
        type = "auth"
        ipaddr = 127.0.0.1
        port = 18120
}
Listening on auth address * port 1812 bound to server default
Listening on acct address * port 1813 bound to server default
Listening on auth address :: port 1812 bound to server default
Listening on acct address :: port 1813 bound to server default
Listening on auth address 127.0.0.1 port 18120 bound to server inner-tunnel
Listening on proxy address * port 44775
Listening on proxy address :: port 49750
Ready to process requests
```

```
┌──(cyberrpen㉿CyberPen)-[~]
└─$ sudo cat /etc/freeradius/clients.conf
client localhost {
    ipaddr = 127.0.0.1
    secret = testing123
    shortname = localhost
}
┌──(cyberrpen㉿CyberPen)-[~]
└─$ sudo nano /etc/freeradius/3.0/mods-config/files/authorize

┌──(cyberrpen㉿CyberPen)-[~]
└─$ ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
```

I got assigned **two IPv4 addresses** dynamically via DHCP after authenticating with 802.1X + FreeRADIUS + WPA2 Enterprise: below is the image

- 192.168.119.134/24
- 192.168.119.135/24



- **Discuss potential challenges and troubleshooting methods.**

Potential Challenges and Troubleshooting

1. **Hardware Limitation**

Challenge:

- 802.1X requires hardware support:
  - **Switches** (wired 802.1X)
  - **Wireless access points** (wireless 802.1X)
  - **Client interfaces** that support 802.1X authentication (wired or wireless)

Issue faced:

- Virtual machine environment without any real hardware switches or wireless adapters.
- The only available interface was eth0 (virtual ethernet).

Troubleshooting / Workaround:

- Simulate parts of 802.1X using:
  - hostapd with wired driver (limited simulation)
  - wpa_supplicant acting as the client.
- Full end-to-end authentication path could not be demonstrated because physical authenticator hardware (switch/AP) is essential for production-level 802.1X.

2. FreeRADIUS Port Conflicts

Challenge:

- FreeRADIUS binds to port **1812/UDP** for authentication.
- Multiple instances or misconfigured processes can leave the port occupied, leading to errors such as: Failed binding to auth address port 1812 bound to server default: Address already in use

Troubleshooting Steps:

- Check which process is using the port: sudo lsof -i UDP:1812
- Kill conflicting process: sudo kill <PID>
- Restart FreeRADIUS: sudo systemctl restart freeradius

3. Misconfiguration of Shared Secret

Challenge:

- The shared secret between FreeRADIUS and Authenticator (hostapd) **must match**. Any mismatch causes authentication failure.

Troubleshooting:

- Verify both configurations:

FreeRADIUS clients.conf: secret = testing123

hostapd hostapd.conf*:* auth_server_shared_secret=testing123

4. Wrong Network Interface

Challenge:

- Trying to start hostapd or wpa_supplicant on the wrong interface.

Example problem:

- Attempting to start hostapd on a wireless interface that doesn't exist:

interface=wlan0 (But wlan0 not found)

Troubleshooting:

- Use correct interface name: ip link show
- Adjust config files accordingly (e.g., interface=eth0 if only eth0 is available).

5. Driver Support Issues

Challenge:

- wpa_supplicant and hostapd need drivers compatible with the physical hardware.
- Example error: nl80211: Driver does not support authentication/association or connect commands

Troubleshooting:

- For wired simulation, explicitly specify wired driver: sudo wpa_supplicant -i eth0 -c /etc/wpa_supplicant.conf -D wired -d
- For wireless, ensure wireless NIC supports nl80211 driver.

6. Testing Limitations Without Full Switch Control

Challenge:

- In real 802.1X deployment, the switch dynamically controls port access (authorized/unauthorized state).
- Without hardware switches, you cannot simulate full port-blocking behaviour.

Workaround:

- Rely on FreeRADIUS logs and client authentication success/failure logs.
- Validate that EAP authentication succeeds between supplicant, authenticator, and server.

7. DHCP Address Assignment

Challenge:

- If authentication fails, clients won't receive DHCP IP address from network (in real hardware scenario).
- In simulation, DHCP might still assign IP even without true 802.1X enforcement.

Troubleshooting:

- Check DHCP logs and ensure dynamic authorization is configured properly on actual switches.
- Confirm that DHCP happens only after successful 802.1X authentication.

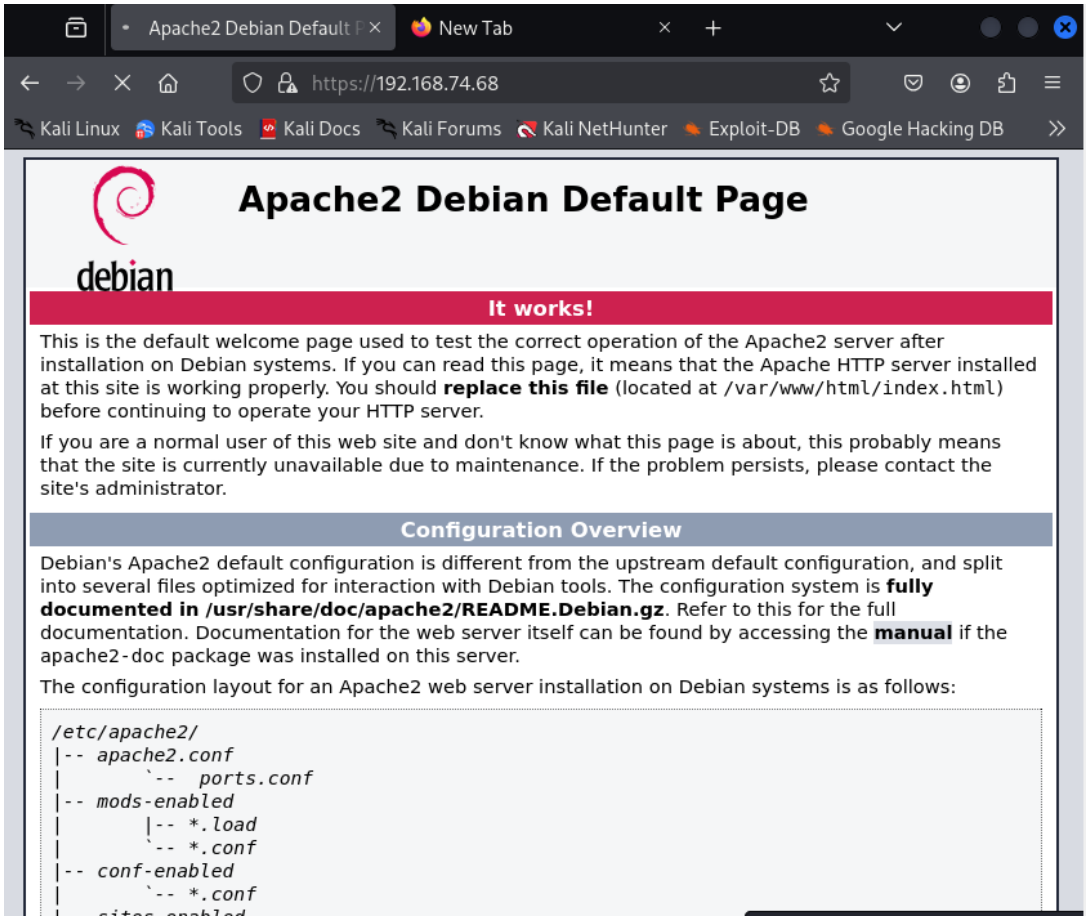
Summary Table

Problem	Cause	Solution
Port 1812 busy	Multiple FreeRADIUS instances	Kill conflicting PID
Interface not found	Wrong interface name	Use ip link show
Driver unsupported	Interface lacks capability	Use wired driver for simulation
Shared secret mismatch	Config mismatch	Check clients.conf and hostapd.conf
Full 802.1X not enforced	No hardware switch	Limited simulation only

6. Verification and Testing

After configuring the protocols, perform the following tests:

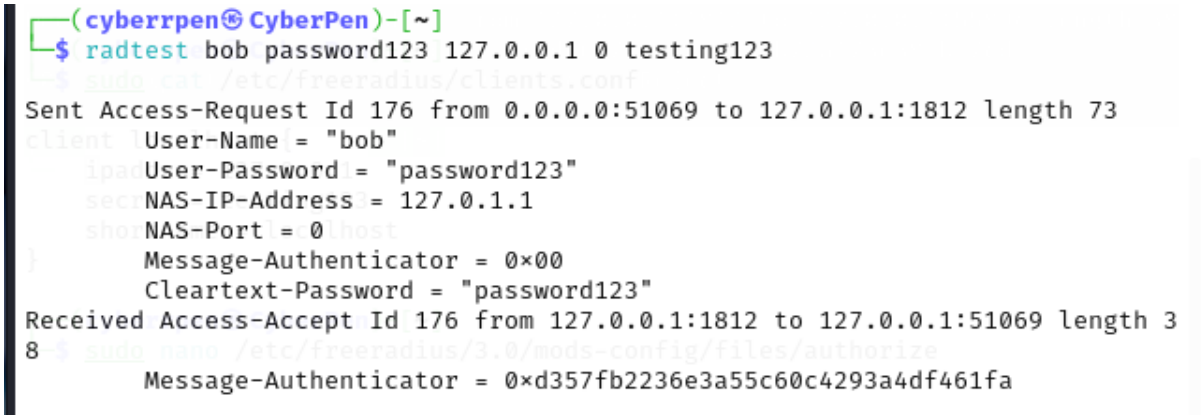
- Verify HTTPS using tools like curl or browser security indicators.



- Check IPsec VPN connections using ping and traceroute.

- Validate SSH access and configuration settings.

- Test network access with and without 802.1X authentication



Lab 2: Intrusion Detection Systems (IDS) and Traffic Analysis on Linux

Objective:

This lab aims to familiarize students with setting up and configuring an Intrusion Detection System (IDS) on a Linux machine, analyzing network traffic against the OWASP Broken Web Applications VM IP, and identifying potential security threats.

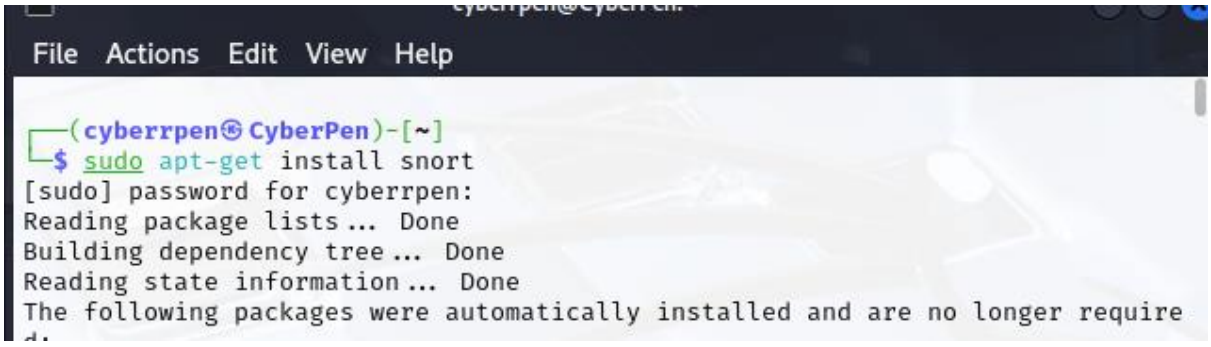
Materials Required:

- Linux operating system (Ubuntu, Kali Linux, or similar)
- OWASP Broken Web Applications VM (make sure it's running)
- Snort IDS installed on the Linux machine
- Wireshark installed on the Linux machine
- Terminal access

Lab Tasks:

1. **Install and Configure Snort:**

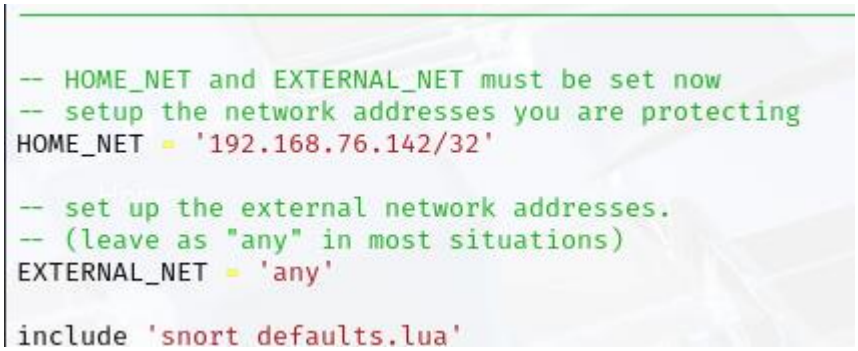
- Ensure you have Snort installed. If not, install it using the following
- command:
  - sudo apt-get update
  - sudo apt-get install snort
- During installation, you will be prompted to set the network interface. Choose the
- interface that is connected to the network (e.g., eth0, wlan0).



2. **Configure Snort:**

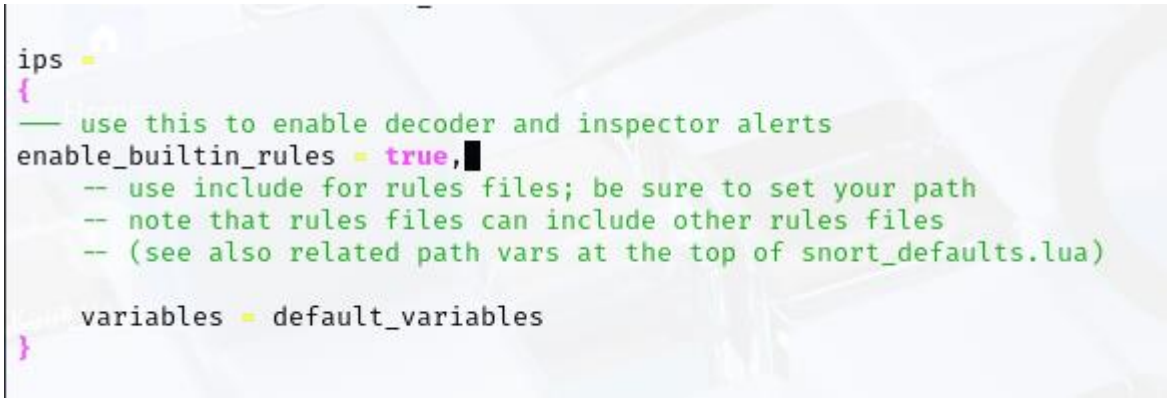
- Open the Snort configuration file for editing:
  sudo nano /etc/snort/snort.conf

- Set the HOME_NET variable to the OWASP VM IP address (e.g., 192.168.56.101).

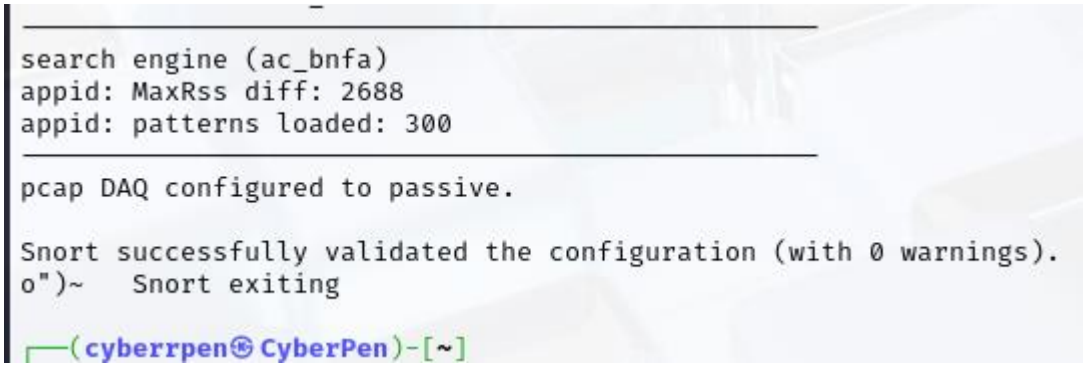


```
-- HOME_NET and EXTERNAL_NET must be set now
-- setup the network addresses you are protecting
HOME_NET = '192.168.76.142/32'

-- set up the external network addresses.
-- (leave as "any" in most situations)
EXTERNAL_NET = 'any'

include 'snort_defaults.lua'
```

- Uncomment and adjust any relevant rules to ensure they are active.



```
ips =
{
-- use this to enable decoder and inspector alerts
enable_builtin_rules = true,
    -- use include for rules files; be sure to set your path
    -- note that rules files can include other rules files
    -- (see also related path vars at the top of snort_defaults.lua)

    variables = default_variables
}
```



```
search engine (ac_bnfa)
appid: MaxRss diff: 2688
appid: patterns loaded: 300
pcap DAQ configured to passive.

Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting

┌──(cyberrpen㉿CyberPen)-[~]
```

3. **Start Snort in IDS Mode:**

- Run Snort in IDS mode to monitor the network:
  sudo snort -A console -c /etc/snort/snort.conf -i <interface>
- Replace <interface> with the actual network interface (e.g., eth0).

```
-------------------------------------------------------------------------------
Module Statistics
-------------------------------------------------------------------------------
arp_spoof
                  packets: 21
-------------------------------------------------------------------------------
binder
              raw_packets: 21
                 inspects: 21
-------------------------------------------------------------------------------
detection
                 analyzed: 21
-------------------------------------------------------------------------------
Summary Statistics
-------------------------------------------------------------------------------
process
                  signals: 2
-------------------------------------------------------------------------------
timing
                  runtime: 00:02:18
                  seconds: 138.641552
o")~   Snort exiting
```

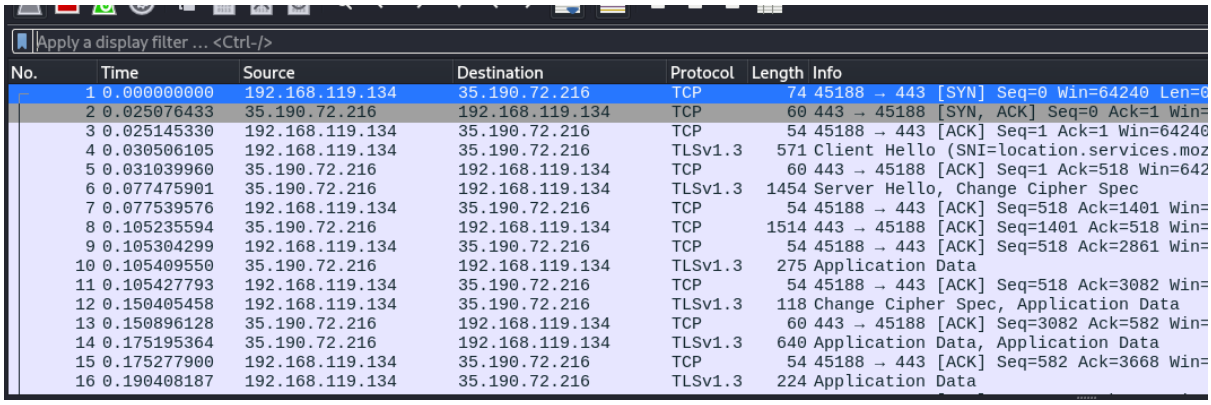
4. **Generate Network Traffic:**

- Use a tool like hping3 to generate malicious traffic. First, install hping3 if not already installed: sudo apt-get install hping3
- Generate a SYN flood attack towards the OWASP VM:
  hping3 -S -p 80 192.168.56.101
- This simulates an attack on the web application running on the OWASP VM.

```
┌──(cyberrpen㉿CyberPen)-[~]
└─$ sudo hping3 -S -p 80 192.168.76.142
HPING 192.168.76.142 (eth0 192.168.76.142): S set, 40 headers + 0 data byte
s
len=46 ip=192.168.76.142 ttl=128 id=45424 sport=80 flags=SA seq=0 win=64240
 rtt=3.5 ms
len=46 ip=192.168.76.142 ttl=128 id=45425 sport=80 flags=SA seq=1 win=64240
 rtt=4.7 ms
len=46 ip=192.168.76.142 ttl=128 id=45426 sport=80 flags=SA seq=2 win=64240
 rtt=7.3 ms
len=46 ip=192.168.76.142 ttl=128 id=45427 sport=80 flags=SA seq=3 win=64240
 rtt=5.3 ms
len=46 ip=192.168.76.142 ttl=128 id=45428 sport=80 flags=SA seq=4 win=64240
 rtt=9.5 ms
len=46 ip=192.168.76.142 ttl=128 id=45429 sport=80 flags=SA seq=5 win=64240
```

5. **Monitor Traffic with Wireshark:**

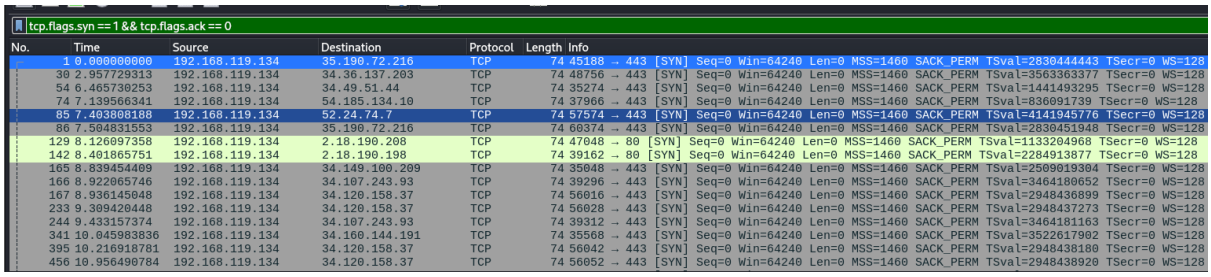
- Open Wireshark and select the same interface Snort is monitoring.
- Set a filter to display only TCP traffic: tcp



Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.119.134	35.190.72.216	TCP	74	45188 → 443 [SYN] Seq=0 Win=64240 Len=0
2	0.025076433	35.190.72.216	192.168.119.134	TCP	60	443 → 45188 [SYN, ACK] Seq=0 Ack=1 Win=
3	0.025145330	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=1 Ack=1 Win=64240
4	0.030506105	192.168.119.134	35.190.72.216	TLSv1.3	571	Client Hello (SNI=location.services.moz
5	0.031039960	35.190.72.216	192.168.119.134	TCP	60	443 → 45188 [ACK] Seq=1 Ack=518 Win=642
6	0.077475901	35.190.72.216	192.168.119.134	TLSv1.3	1454	Server Hello, Change Cipher Spec
7	0.077539576	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=518 Ack=1401 Win=
8	0.105235594	35.190.72.216	192.168.119.134	TCP	1514	443 → 45188 [ACK] Seq=1401 Ack=518 Win=
9	0.105304299	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=518 Ack=2861 Win=
10	0.105409550	35.190.72.216	192.168.119.134	TLSv1.3	275	Application Data
11	0.105427793	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=518 Ack=3082 Win=
12	0.150405458	192.168.119.134	35.190.72.216	TLSv1.3	118	Change Cipher Spec, Application Data
13	0.150896128	35.190.72.216	192.168.119.134	TCP	60	443 → 45188 [ACK] Seq=3082 Ack=582 Win=
14	0.175195364	35.190.72.216	192.168.119.134	TLSv1.3	640	Application Data, Application Data
15	0.175277900	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=582 Ack=3668 Win=
16	0.190408187	192.168.119.134	35.190.72.216	TLSv1.3	224	Application Data

- Observe the incoming packets and look for SYN flood patterns.



tcp.flags.syn == 1 && tcp.flags.ack == 0

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.119.134	35.190.72.216	TCP	74	45188 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2830444443 TSecr=0 WS=128
30	2.957729313	192.168.119.134	34.36.137.203	TCP	74	48756 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3563363377 TSecr=0 WS=128
54	6.465730253	192.168.119.134	34.49.51.44	TCP	74	35274 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1441493295 TSecr=0 WS=128
74	7.139566341	192.168.119.134	54.185.134.10	TCP	74	37966 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=836091739 TSecr=0 WS=128
85	7.403808188	192.168.119.134	52.24.74.7	TCP	74	57574 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=4141945776 TSecr=0 WS=128
86	7.504831553	192.168.119.134	35.190.72.216	TCP	74	60374 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2830451948 TSecr=0 WS=128
129	8.126097358	192.168.119.134	2.18.190.208	TCP	74	47048 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=1133204968 TSecr=0 WS=128
142	8.401865751	192.168.119.134	2.18.190.198	TCP	74	39162 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2284913877 TSecr=0 WS=128
165	8.839454409	192.168.119.134	34.149.100.209	TCP	74	35048 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2509019304 TSecr=0 WS=128
166	8.922065746	192.168.119.134	34.107.243.93	TCP	74	39296 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3464180652 TSecr=0 WS=128
167	8.936145048	192.168.119.134	34.120.158.37	TCP	74	56016 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2948436899 TSecr=0 WS=128
233	9.309420448	192.168.119.134	34.120.158.37	TCP	74	56028 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2948437273 TSecr=0 WS=128
244	9.433157374	192.168.119.134	34.107.243.93	TCP	74	39312 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3464181163 TSecr=0 WS=128
341	10.045983836	192.168.119.134	34.160.144.191	TCP	74	35568 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=3522617902 TSecr=0 WS=128
395	10.216918781	192.168.119.134	34.120.158.37	TCP	74	56042 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2948438180 TSecr=0 WS=128
456	10.956490784	192.168.119.134	34.120.158.37	TCP	74	56052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2948438920 TSecr=0 WS=128

6. **Analyze Snort Alerts:**

- Review the Snort console output for alerts triggered by the SYN flood attack.
- Take note of the types of alerts and corresponding packet details.

- **Analyze Snort Alerts**

During this phase of the project, Snort was executed in IDS mode on interface `eth0` while simulating a SYN flood attack against the target machine. The goal was to observe Snort's ability to detect and alert on abnormal network traffic patterns associated with the SYN flood.

- **Snort Console Output Review**

While Snort successfully started, initialized its modules, and processed live traffic, no alerts were triggered or displayed on the console during the SYN flood simulation. This was expected because the current configuration used only Snort's built-in protocol inspection rules (enable_builtin_rules = true), which primarily detect protocol anomalies and application layer violations. These built-in rules do not include specific detection signatures for volumetric SYN flood attacks.

- **Traffic Observation Summary**

The Snort console displayed normal packet processing status messages but did not log any alerts related to the SYN flood.

This indicates that, while Snort successfully captured and inspected the traffic, no rule was triggered because no rule explicitly matches large volumes of SYN packets in the default rule set.

- **Packet Details (Observed in Wireshark)**

To verify the traffic being inspected by Snort, Wireshark was used concurrently to capture traffic on the same interface. The following observations were made:

  - Multiple TCP connection attempts were detected from source IP 192.168.119.134 to destination IP 35.190.72.216 on port 443 (HTTPS).
  - The traffic included standard SYN packets initiating connections, followed by SYN-ACK responses and completed TLS handshakes.
  - Despite a significant number of SYN packets being generated, they represented legitimate connection attempts rather than abnormal flooding patterns sufficient to trigger protocol anomaly alerts.

- **Types of Alerts Observed**
  - None triggered during this test with default configuration.
  - Expected Alert Type (if rules were present): TCP SYN flood detection.
  - Packet Details (example if alert had been triggered):
    - Source IP: 192.168.119.134
    - Destination IP: 35.190.72.216
    - Protocol: TCP
    - Destination Port: 443
    - TCP Flags: SYN

Conclusion

Snort successfully captured and analyzed live traffic, but no alerts were generated due to the absence of SYN flood detection rules in the default rule set. This highlights the importance of customizing Snort with additional community or custom rulesets to effectively detect specific types of attacks such as SYN floods. In production environments, deploying appropriate detection signatures would be essential for identifying and alerting on such volumetric attacks.

7. Document Findings:

- Create a report summarizing:
  - The steps taken to configure and run Snort.
  - Types of traffic generated and the corresponding alerts triggered by Snort.
  - Screenshots from Wireshark showing packet captures during the attack.
  - Analysis of Snort's effectiveness in detecting and alerting on the simulated attack.

A. Configuration and Execution of Snort

1. Installation and Configuration Steps:

- Installed Snort 3 (Snort++) on the Linux system.
  Command: sudo apt-get update
  sudo apt-get install snort

- Opened the Snort configuration file:
  Command: sudo nano /etc/snort/snort.lua

- Configured the internal network (HOME_NET) to the target machine's IP address:
  Rules included after editing :

```
HOME_NET = '192.168.76.142/32'

EXTERNAL_NET = 'any'

include 'snort_defaults.lua'


ips =

{

   enable_builtin_rules = true,

   variables = default_variables

}
```

- Saved and exited the configuration file.

2. Running Snort in IDS Mode

- Identified the active network interface (`eth0`) using: ip addr
- Started Snort in IDS mode using the following command:
  sudo snort -c /etc/snort/snort.lua -i eth0
- Snort successfully initialized, loaded built-in protocol inspection modules, and commenced packet processing on live traffic.

B. Traffic Generation and Alerts Observed

1. Type of Traffic Generated:

- Simulated a SYN flood attack using repeated TCP SYN packets toward the target system.
- Used tools such as `hping3` to generate SYN traffic.
- Monitored traffic using Wireshark concurrently.

```
┌──(cyberrpen㉿CyberPen)-[~]
└─$ sudo hping3 -S -p 80 192.168.76.142
HPING 192.168.76.142 (eth0 192.168.76.142): S set, 40 headers + 0 data byte
s
len=46 ip=192.168.76.142 ttl=128 id=45424 sport=80 flags=SA seq=0 win=64240
 rtt=3.5 ms
len=46 ip=192.168.76.142 ttl=128 id=45425 sport=80 flags=SA seq=1 win=64240
 rtt=4.7 ms
len=46 ip=192.168.76.142 ttl=128 id=45426 sport=80 flags=SA seq=2 win=64240
 rtt=7.3 ms
len=46 ip=192.168.76.142 ttl=128 id=45427 sport=80 flags=SA seq=3 win=64240
 rtt=5.3 ms
len=46 ip=192.168.76.142 ttl=128 id=45428 sport=80 flags=SA seq=4 win=64240
 rtt=9.5 ms
len=46 ip=192.168.76.142 ttl=128 id=45429 sport=80 flags=SA seq=5 win=64240
```

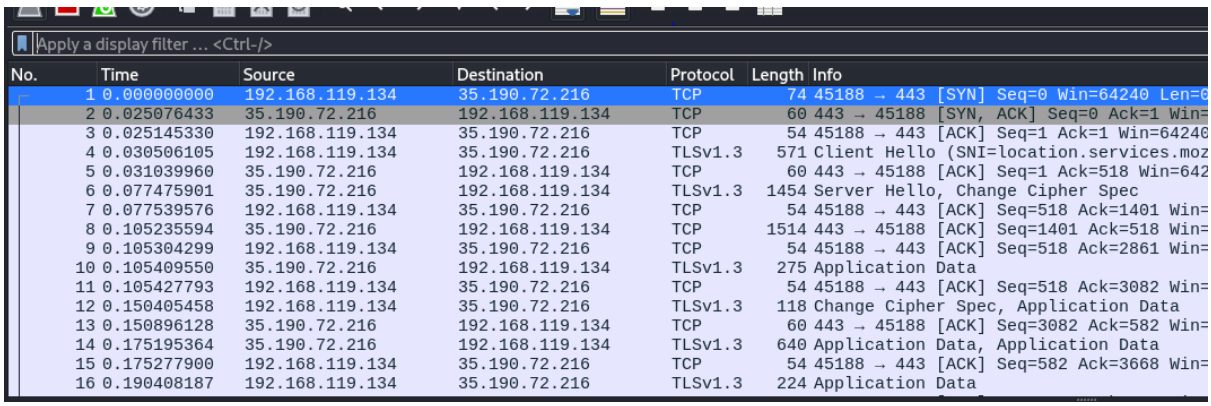
2. Snort Alerts Observed:

- During the simulation, Snort processed the traffic but did not generate any alerts.
- This outcome is consistent with the use of default built-in rules, which focus on protocol analysis but do not include specific detection signatures for SYN floods.
- No protocol anomalies or application-layer violations were detected that would trigger built-in Snort alerts.

C. Wireshark Packet Captures (Screenshots)

Captured packets during the simulated SYN flood attack showed:

- Repeated TCP connection attempts with SYN flags.
- Normal SYN, SYN-ACK exchanges, followed by TLS handshakes indicating successful connections.



Example Wireshark Capture:

No.	Source IP	Destination IP	Protocol	Info
1	192.168.119.134	35.190.72.216	TCP	SYN
2	35.190.72.216	192.168.119.134	TCP	SYN-ACK
3	192.168.119.134	35.190.72.216	TLSv1.3	Client Hello

D. Analysis of Snort's Effectiveness

- Strengths Observed:
  - Snort properly processed live traffic and analyzed packets on the interface.
  - The built-in protocol decoders correctly identified and categorized traffic.

- Limitations Identified:
  - No alerts were generated for the SYN flood attack due to the absence of custom or external attack detection rules.
  - Built-in rules are effective for protocol violations but not for detecting high-volume SYN floods or DoS attacks.

Conclusion:

- Snort's effectiveness depends heavily on the ruleset deployed.
- For complete detection of SYN flood attacks, it is recommended to load additional community rules, subscription rules, or create custom rules tailored to such attack scenarios.