

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.

```
(cyberpen㉿Cyberpen)-[~]
$ sudo systemctl status apache2
● apache2.service - The Apache HTTP Server
  Loaded: loaded (/usr/lib/systemd/system/apache2.service; enabled; preset: disabled)
  Active: active (running) since Mon 2025-06-16 12:21:57 WAT; 1h 31min ago
  Invocation: 3188ecbf6b2b4fdda210447d522375a0
    Docs: https://httpd.apache.org/docs/2.4/
   Process: 1187 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Process: 11914 ExecReload=/usr/sbin/apachectl graceful (code=exited, status=0/SUCCESS)
 Main PID: 1269 (apache2)
    Tasks: 6 (limit: 2093)
   Memory: 23.5M (peak: 34.1M)
      CPU: 2.243s
     CGroup: /system.slice/apache2.service
             └─1269 /usr/sbin/apache2 -k start
                  ├─11921 /usr/sbin/apache2 -k start
                  ├─11923 /usr/sbin/apache2 -k start
                  ├─11924 /usr/sbin/apache2 -k start
                  ├─11925 /usr/sbin/apache2 -k start
                  └─11926 /usr/sbin/apache2 -k start

Jun 16 12:21:55 Cyberpen systemd[1]: Starting apache2.service - The Apache HTTP Server ...
Jun 16 12:21:57 Cyberpen systemd[1]: Started apache2.service - The Apache HTTP Server.
Jun 16 12:42:26 Cyberpen systemd[1]: Reloading apache2.service - The Apache HTTP Server ...
Jun 16 12:42:26 Cyberpen systemd[1]: Reloaded apache2.service - The Apache HTTP Server.

(cyberpen㉿Cyberpen)-[~]
```

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

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

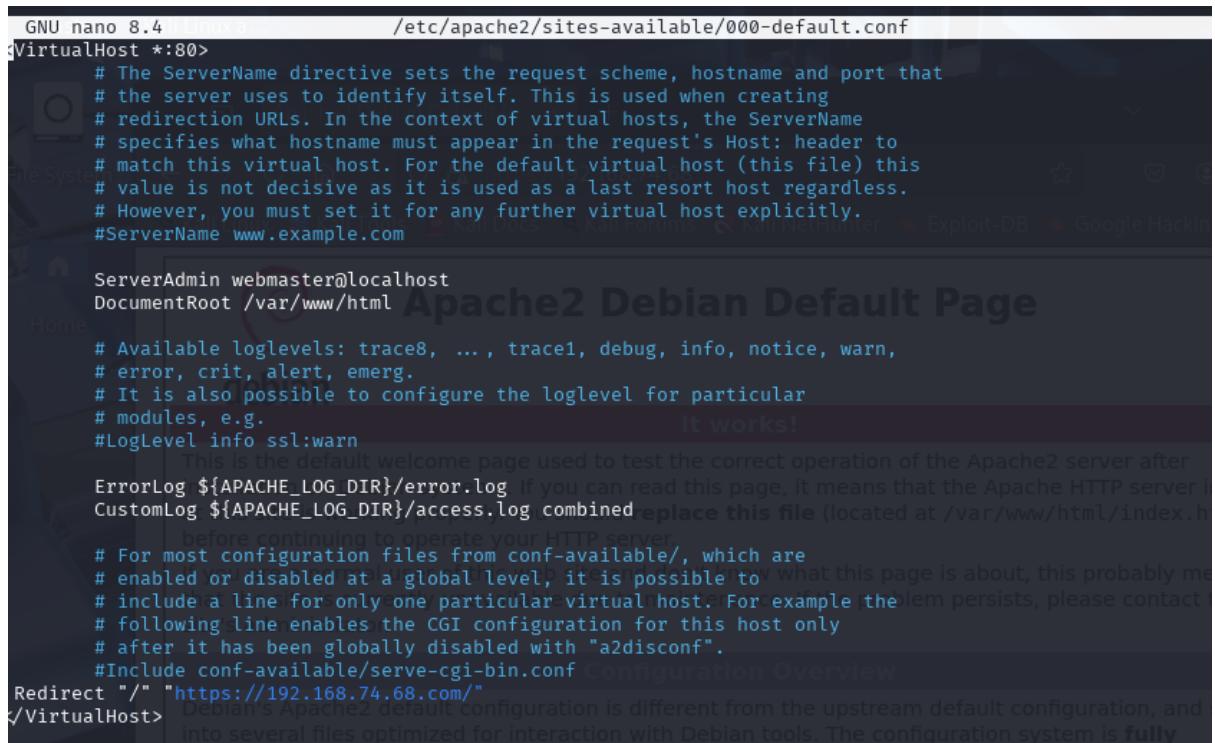
```
(cyberpen㉿Cyberpen)=[~] files optimized for inter-  
$ sudo a2enmod ssl  
Considering dependency mime for ssl:  
Module mime already enabled  
Module socache_shmcb for ssl:  
Module socache_shmcb already enabled  
Module ssl already enabled  
/etc/apache2/  
(cyberpen㉿Cyberpen)=[~] e2.conf
```

```
File Actions Edit View Help
GNU nano 8.4          /etc/apache2/sites-available/www.10.0.2.15.com-ss
<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.

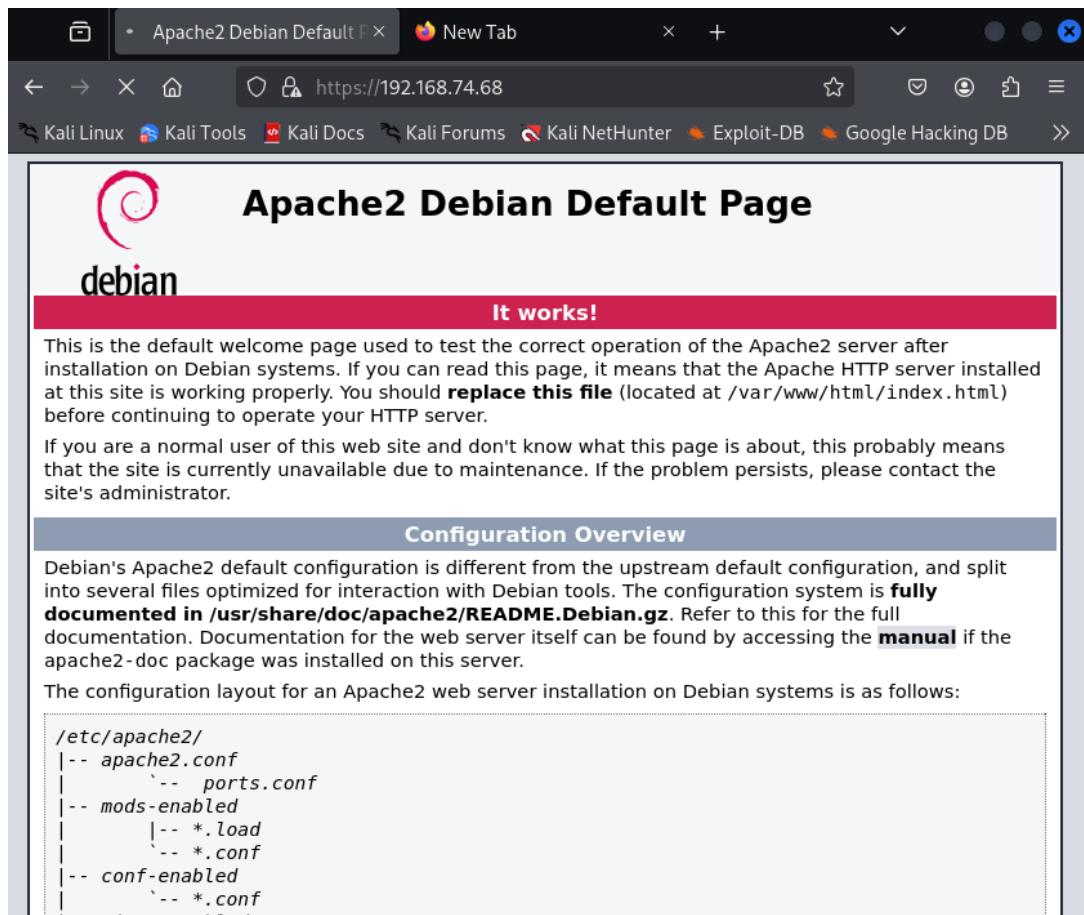


```
GNU nano 8.4  [DEB] /etc/apache2/sites-available/000-default.conf
VirtualHost *:80>
    # The ServerName directive sets the request scheme, hostname and port that
    # the server uses to identify itself. This is used when creating
    # redirection URLs. In the context of virtual hosts, the ServerName
    # 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
    Home

    # 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
        This is the default welcome page used to test the correct operation of the Apache2 server after
        ErrorLog ${APACHE_LOG_DIR}/error.log If you can read this page, it means that the Apache HTTP server i
        CustomLog ${APACHE_LOG_DIR}/access.log combined replace this file (located at /var/www/html/index.h
        before continuing to operate your HTTP server.
        # For most configuration files from conf-available/, which are
        # enabled or disabled at a global level, it is possible to what this page is about, this probably me
        # include a line for only one particular virtual host. For example the problem persists, please contact
        # 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 Configuration Overview
    Redirect "/" "https://192.168.74.68.com/"
/VirtualHost> Debian's Apache2 default configuration is different from the upstream default configuration, and
           into several files optimized for interaction with Debian tools. The configuration system is fully
```

- 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:

```
└─(cyberpen㉿Cyberpen)─[~]
$ ipsec pki --gen --outform pem > ~/pki/private/ca-key.pem
ipsec pki --self --ca --lifetime 3650 --in ~/pki/private/ca-key.pem --type
rsa --dn "CN=VPN-CA" --outform pem > ~/pki/cacerts/ca-cert.pem
TPM 2.0 - could not load "libtss2-tcti-tabrmd.so.0"
plugin 'tpm': failed to load - tpm_plugin_create returned NULL
/usr/bin/pki: unrecognized option '--gen--outform'
Error: invalid operation
strongSwan 6.0.1 PKI tool
loaded plugins: test-vectors pkcs11 aesni aes rc2 sha2 sha1 md5 mgf1 rdrand random x509 revocation constrain
ts pubkey pkcs1 pkcs7 pkcs12 dnskey sshkey pem openssl gcrypt pkcs8 af-alg gmp curve25519 hmac kdf drbg curl
usage:
  pki --acert  (-z)  issue an attribute certificate
  pki --dn     (-d)  extract the subject DN of an X.509 certificate
  pki --est    (-E)  Enroll an X.509 certificate with an EST server
  pki --estca  (-e)  get CA certificate[s] from an EST server
```

- Generate server certificate

```
└─(cyberpen㉿Cyberpen)─[~]
$ ipsec pki --gen --outform pem > ~/pki/private/server-key.pem

ipsec pki --pub --in ~/pki/private/server-key.pem --type rsa | \
ipsec pki --issue --lifetime 1825 \
--cacert ~/pki/cacerts/ca-cert.pem \
--cakey ~/pki/private/ca-key.pem \
--dn "CN=vpnserver" \
--san "vpnserver" \
--flag serverAuth \
--outform pem > ~/pki/certs/server-cert.pem

TPM 2.0 - could not load "libtss2-tcti-tabrmd.so.0"
plugin 'tpm': failed to load - tpm_plugin_create returned NULL
TPM 2.0 - could not load "libtss2-tcti-tabrmd.so.0"
plugin 'tpm': failed to load - tpm_plugin_create returned NULL
TPM 2.0 - could not load "libtss2-tcti-tabrmd.so.0"
plugin 'tpm': failed to load - tpm_plugin_create returned NULL
```

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

Edit /etc/ipsec.conf

```
GNU nano 8.4                               /etc/ipsec.conf
# ipsec.conf - strongSwan IPsec configuration file

# basic configuration

config setup

charondebug="ike 2, knl 2, cfg 2, net 2, esp 2, dmn 2, mgr 2"
conn vpn
keyexchange=ikev2
auto=add
left=%any
leftcert=server-cert.pem
leftsubnet=0.0.0.0/0
right=%any
rightauth=eap-mschapv2
rightdns=8.8.8.8
rightsourseip=10.10.10.0/24
eap_identity=%identity

# strictcrlpolicy=yes
# uniqueids = no

# Add connections here.
```

- Ensure secure communication between clients and the server:

```
GNU nano 8.4                               /etc/ipsec.secrets
: RSA server-key.pem
testuser : EAP "testpassword"># This file holds shared secrets or RSA private keys for authentication.

# RSA private key for this host, authenticating it to any other host
# which knows the public part.
```

- 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.

```
(cyberppen@CyberPen)@[~]
$ sudo systemctl status ssh

● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: ▾)
  Active: active (running) since Sun 2025-06-22 03:39:29 CDT; 34s ago
    Invocation: 038138ad5a354ae296160f83b94b99f2
      Docs: man:sshd(8)
             man:sshd_config(5)
   Process: 45934 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUC>
 Main PID: 45936 (sshd)
    Tasks: 1 (limit: 4547)
   Memory: 1.6M (peak: 2.3M)
```

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

```
# OpenSSH is to specify options with their default value where
# possible, but leave them commented. Uncommented options
# have their default value.

Include /etc/ssh/sshd_config.d/*.conf
Port 2222

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::
```

- Disable root login.

```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes
```

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

```
(cyberpen@CyberPen)@[~]
$ ssh -p 2222 cyberpen@localhost location /etc/ssh/sshd_config

Enter passphrase for key '/home/cyberpen/.ssh/id_rsa':
Linux CyberPen 6.12.25-1kali1 #1 SMP PREEMPT_DYNAMIC Kali 6.12.25-1kali1 (2025-04-30) x86_64
      ↗ PAM authentication, then enable this or set PasswordAuthentication
The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/

(Run: "touch ~/.hushlogin" to hide this message)
(cyberpen@CyberPen)@[~]
```

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

```
(cyberpen@CyberPen)@[~]
$ sudo ufw allow 2222/tcp
sudo ufw enable
sudo ufw status

Rules updated
Rules updated (v6)
Firewall is active and enabled on system startup
Status: active

To                         Action      From
--                         ALLOW       Anywhere
2222/tcp                   ALLOW       Anywhere
2222/tcp (v6)               ALLOW       Anywhere (v6)

(cyberpen@CyberPen)@[~]
$ █
```

5. Using 802.1X for Network Access Control

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

```
(cyberrpen㉿CyberPen)-[~]
$ sudo apt update
sudo apt install freeradius freeradius-utils

[sudo] password for cyberrpen:
Sorry, try again.
[sudo] password for cyberrpen:
Hit:1 http://http.kali.org/kali kali-rolling InRelease
884 packages can be upgraded. Run 'apt list --upgradable' to see them.
The following packages were automatically installed and are no longer require
d:
base58
cython3
debtags
debugedit
dnsmap
-----
```

- Configure a RADIUS server to handle authentication requests.

```
(cyberrpen㉿CyberPen)-[~]
$ radtest bob password123 127.0.0.1 0 testing123
8-> sudo nano /etc/freeradius/clients.conf
Sent Access-Request Id 176 from 0.0.0.0:51069 to 127.0.0.1:1812 length 73
client lUser-Name = "bob"
    ipadUser-Password = "password123"
    secrNAS-IP-Address = 127.0.1.1
    shorNAS-Port = 0
)
    Message-Authenticator = 0x00
    Cleartext-Password = "password123"
Received Access-Accept Id 176 from 127.0.0.1:1812 to 127.0.0.1:51069 length 3
8-> sudo nano /etc/freeradius/3.0/mods-config/files/authorize
    Message-Authenticator = 0xd357fb2236e3a55c60c4293a4df461fa
```

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

```

}
}-(cyberpen@CyberPen)-[~]
listen { 8.8.8.8 -c 4
    type = "auth"
PING 8.8.ipaddr=127.0.0.1(84) bytes of data.
64 bytes port =18120: icmp_seq=1 ttl=128 time=34.6 ms
}, bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=37.4 ms
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, 529/22.051 ms
Listening on proxy address :: port 49750
Ready to process requests[~]
█ $ █

```

```

    type = "acct"
└─(cyberpen@CyberPen)-[~]
$ sudo cat /etc/freeradius/clients.conf
limit {
client localhost { options = 16
    ipaddr = 127.0.0.1
    secret = testing12330
    shortname = localhost
}
listen {
    (cyberpen@CyberPen)-[~]
$ sudo nano /etc/freeradius/3.0/mods-config/files/authorize
        port = 18120
}
└─(cyberpen@CyberPen)-[~] port 1812 bound to server default
$ ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data. to server default

```

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

```
(cyberrpen® CyberPen)-[~]
$ ip addr show eth0:25 : Warning: Shared secret for client localhost_ipv6 is short, and likely can be broken by an attacker.
2: eth0:2 <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 00:0c:29:21:b9:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.119.134/24 brd 192.168.119.255 scope global dynamic noprefixroute
        valid_lft 1502sec preferred_lft 1502sec
    inet 192.168.119.135/24 brd 192.168.119.255 scope global secondary dynamic
        linklayer
        valid_lft 1308sec preferred_lft 1308sec
    inet6 fe80::20c:29ff:fe21:b90f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
Sun Jun 22 11:06:19 2025 : Info: Loaded virtual server inner-tunnel
```

- 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.

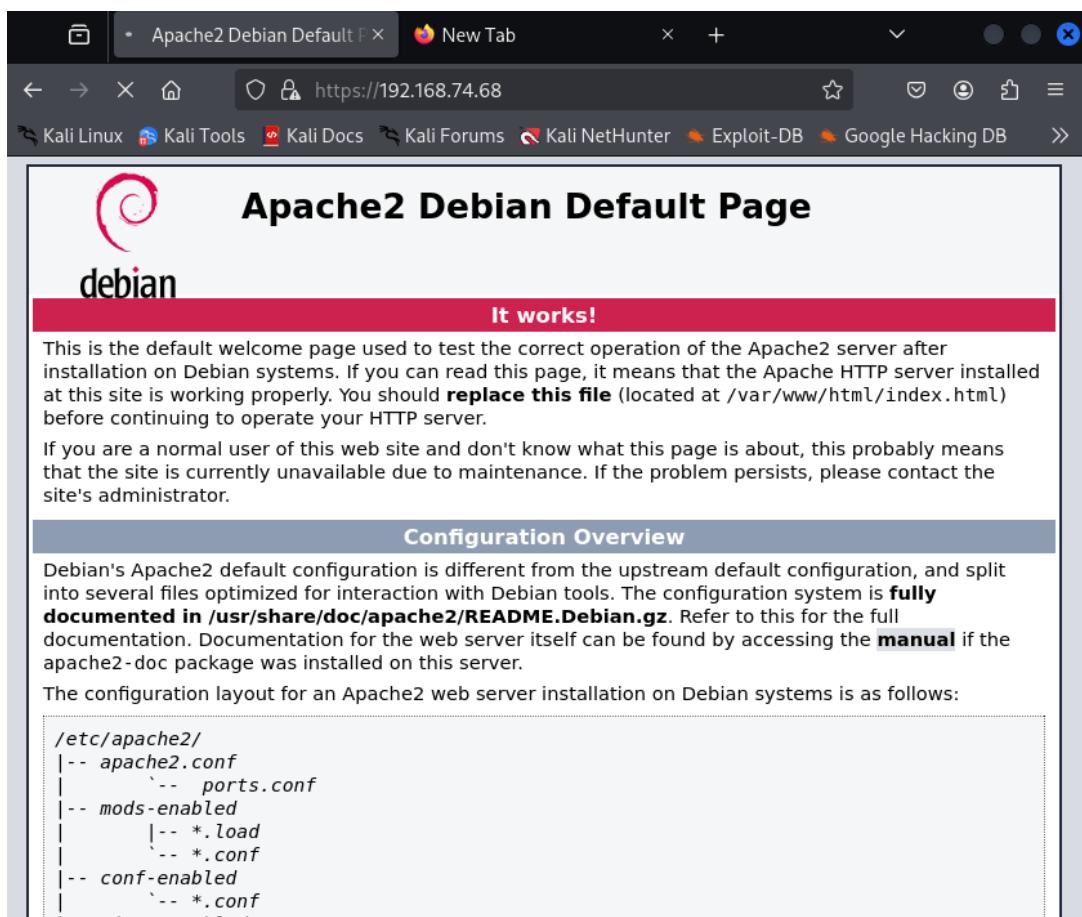
Summary Table

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.

```
(cyberpen@CyberPen)-[~]
$ sudo systemctl status ssh

● ssh.service - OpenBSD Secure Shell server
  Loaded: loaded (/usr/lib/systemd/system/ssh.service; disabled; preset: >
    Active: active (running) since Sun 2025-06-22 03:39:29 CDT; 34s ago
  Invocation: 038138ad5a354ae296160f83b94b99f2
    Docs: man:sshd(8)
          man:sshd_config(5)
  Process: 45934 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUC>
  Main PID: 45936 (sshd)
    Tasks: 1 (limit: 4547)
   Memory: 1.6M (peak: 2.3M)
```

- Test network access with and without 802.1X authentication

```
(cyberpen@CyberPen)-[~]
$ radtest bob password123 127.0.0.1 0 testing123
$ sudo cat /etc/freeradius/clients.conf
Sent Access-Request Id 176 from 0.0.0.0:51069 to 127.0.0.1:1812 length 73
client !User-Name = "bob"
  ipaddrUser-Password1= "password123"
  sec NAS-IP-Address3= 127.0.1.1
  shortNAS-Port = 0localhost
}
  Message-Authenticator = 0x00
  Cleartext-Password = "password123"
Received Access-Accept Id 176 from 127.0.0.1:1812 to 127.0.0.1:51069 length 3
8 $ sudo nano /etc/freeradius/3.0/mods-config/files/authorize
  Message-Authenticator = 0xd357fb2236e3a55c60c4293a4df461fa
```

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.

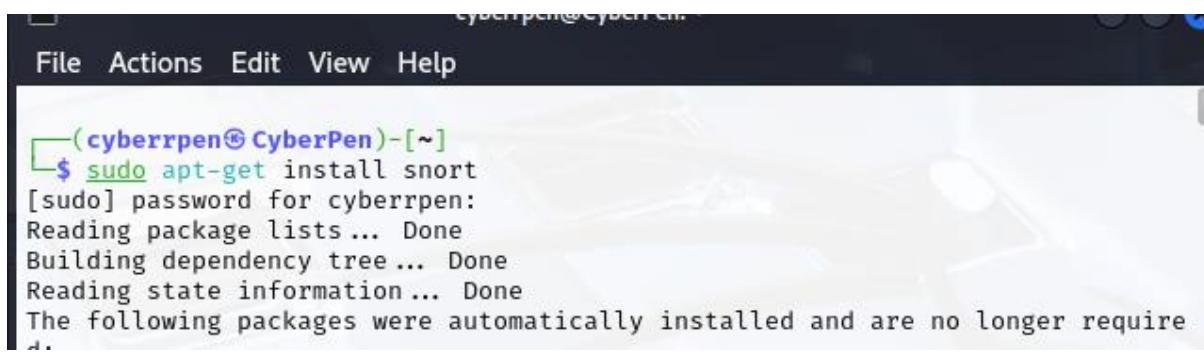
Materials Required:

- 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).



The screenshot shows a terminal window with a dark theme. The title bar says "CyberPen® CyberPen". The menu bar includes "File", "Actions", "Edit", "View", and "Help". The terminal prompt is "(cyberrpen® CyberPen)-[~]". The user has run the command "\$ sudo apt-get install snort". The terminal displays the output of the apt-get command, which includes prompts for a password, package lists, dependency trees, state information, and a message about automatically installed packages. The text is in a monospaced font.

```
(cyberrpen® CyberPen)-[~]
$ sudo apt-get install snort
[sudo] password for cyberrpen:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer require
d.
```

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

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=1 Ack=1 Win=64240
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=64240
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=64240
8	0.105235594	35.190.72.216	192.168.119.134	TCP	1514	443 → 45188 [ACK] Seq=1401 Ack=518 Win=64240
9	0.105304299	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=518 Ack=2861 Win=64240
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=64240
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=64240
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=64240
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.

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 TStamp=2830444443 TSectr=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 TStamp=3563363377 TSectr=0 WS=128
55	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 TStamp=1441493299 TSectr=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 TStamp=836091739 TSectr=0 WS=128
85	7.40938818	192.168.119.134	52.24.74.7	TCP	74	57574 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=4141945776 TSectr=0 WS=128
88	7.549355939	192.168.119.134	35.190.72.216	TCP	74	48756 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=2830444443 TSectr=0 WS=128
129	9.126697058	192.168.119.134	2.18.199.288	TCP	74	47048 → 89 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=1133204968 TSectr=0 WS=128
142	8.491965751	192.168.119.134	2.18.199.288	TCP	74	39162 → 89 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=2234913877 TSectr=0 WS=128
165	8.839454409	192.168.119.134	34.149.190.209	TCP	74	35048 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=2506019384 TSectr=0 WS=128
166	8.922665746	192.168.119.134	34.107.243.93	TCP	74	39296 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=3464180652 TSectr=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 TStamp=2448436892 TSectr=0 WS=128
233	9.389426448	192.168.119.134	34.120.158.37	TCP	74	56028 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=2948437273 TSectr=0 WS=128
244	9.433157374	192.168.119.134	34.167.243.93	TCP	74	39312 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=3464181163 TSectr=0 WS=128
341	10.645983836	192.168.119.134	34.169.144.191	TCP	74	35568 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=3522617962 TSectr=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 TStamp=2948438180 TSectr=0 WS=128
456	10.956496784	192.168.119.134	34.120.158.37	TCP	74	56052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TStamp=2948438920 TSectr=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.

```
(cyberppen㉿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.

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=64240
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.mozillabot.com)
5	0.031039960	35.190.72.216	192.168.119.134	TCP	60	443 → 45188 [ACK] Seq=1 Ack=518 Win=64240
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=64240
8	0.105235594	35.190.72.216	192.168.119.134	TCP	1514	443 → 45188 [ACK] Seq=1401 Ack=518 Win=64240
9	0.105304299	192.168.119.134	35.190.72.216	TCP	54	45188 → 443 [ACK] Seq=518 Ack=2861 Win=64240
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=64240
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=64240
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=64240
16	0.190408187	192.168.119.134	35.190.72.216	TLSv1.3	224	Application Data

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.