

# CSCE 5585 – Part-B Secure Network Design and Implementation

**Project By:**

Group 9

Aparna Singh | Durga Shankar Dalayi | Nikhil Ethamukkala

## 1. INTRODUCTION

This project is about creating a secure virtual enterprise with multiple layers of security using GNS3, pfSense, VirtualBox, Suricata, and OpenVPN. The goal is to create a simulated enterprise network that will demonstrate important security principles including segmentation, perimeter security, intrusion detection, and secure remote connections.

The network has four segments: an internal LAN, a DMZ, a WAN connection, and a VPN segment. All these segments are connected together via a virtual switch, and multiple virtual machines (Ubuntu Server, Kali Linux, and Windows 10) are connected to this virtual switch and routed through a pfSense firewall. The pfSense firewall performs many functions and provides many security features, including routing, NAT, firewalling, and VPN services.

One of the most important aspects of this design is the DMZ segment, which includes an Ubuntu-based Apache web server. The web server can be accessed by the LAN workstations and by VPN users, however, it cannot communicate back into the LAN. This is typical of what you would find in an enterprise network. The LAN segment has two types of hosts on it, workstation hosts and an "attacker" host (the Kali Linux VM). These workstation hosts are used to simulate scanning, enumerating, and flooding the network with traffic. A Suricata IPS sensor is placed in inline IPS mode on both the LAN and DMZ segments to provide real-time intrusion detection and prevention.

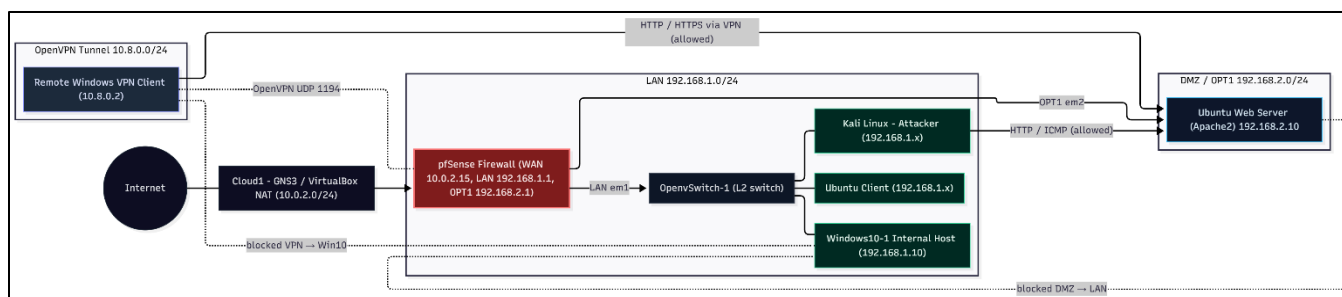
In general terms, the overall purpose of this project was to build and test a multi-layered security architecture, and validate the effectiveness of each layer through the use of a series of controlled attacks against the network. Through this process, all configuration steps, firewall rules, NAT configurations, VPN connections, and IPS alerts and responses have demonstrated how a secure network can be designed and protected.

## 2. NETWORK TOPOLOGY OVERVIEW

This network was created in GNS3 using pfSense for centralized firewall and routing functionality. It has been segmented into three distinct security zones:

1. WAN (external) representing the Internet;
2. LAN which contains internal hosts such as Windows, Ubuntu, and the Kali;
3. DMZ (OPT1) hosts the Apache web service.

Each of the LAN devices are connected via an Open vSwitch while the DMZ server connects directly to the OPT1 port on pfSense. VPN remote users are able to connect to this network through an OpenVPN tunnel providing secure connectivity, however this is not considered another network zone. In addition to segmentation, it allows the ability for LAN hosts to be able to communicate with DMZ services, DMZ is restricted from making connections to the LAN, and VPN remote users are allowed to communicate with the network, although with controlled access to certain areas.



This design replicates a typical enterprise defensive posture by including multiple layers of defense, Network Address Translation (NAT), Firewall filtering, and IPS inspection utilizing Suricata.

Segment / Interface	Device / Role	IP Address	Description
WAN (em0)	pfSense WAN Interface	10.0.3.16/24 (DHCP from Cloud1)	Connected to VirtualBox NAT (Internet edge)
LAN (em1) – 192.168.1.0/24	pfSense Gateway	192.168.1.1	Default gateway for internal hosts
	Windows10-1 Internal Host	192.168.1.10	LAN workstation used for testing
	Ubuntu LAN Client	192.168.1.x	Internal workstation
	Kali Linux (Attacker)	192.168.1.x	Used for scans, DoS, brute-force attacks
DMZ / OPT1 (em2) – 192.168.2.0/24	pfSense OPT1 Gateway	192.168.2.1	DMZ gateway
	Ubuntu Web Server (Apache2)	192.168.2.10	Public-facing DMZ web server
VPN Tunnel – 10.8.0.0/24	OpenVPN Windows Client	10.8.0.2	Assigned by pfSense OpenVPN server
	pfSense OpenVPN Server	(No static IP)	Terminates remote-access VPN tunnel

### 3. CONFIGURATION STEPS

This section summarizes the essential configuration elements required to build the project environment. These include GNS3 topology creation, pfsense interface assignment, firewall rule creation, and how NAT behaves. To provide a full visual representation of the steps described in this paper screenshots of each step are included in the appendix.

#### 3.1 GNS3 ENVIRONMENT SETUP

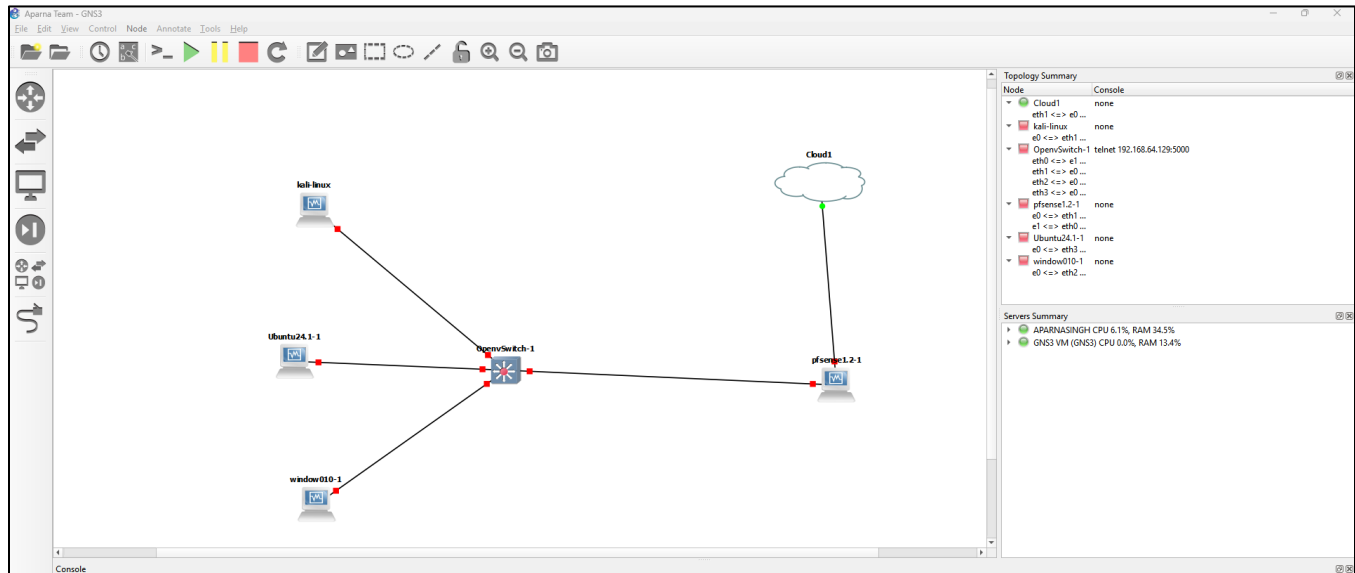
To create the GNS3 network, virtual machines running on VirtualBox were created using the Open vSwitch as the LAN switch. As the NAT network the "Cloud" node in the topology represents the VirtualBox NAT network, which provides the Internet connection to pfSense through its WAN interface. All of the internal hosts (Kali Linux, Windows 10, and Ubuntu client) were attached to the Open vSwitch to form the LAN segment (192.168.1.0/24). A dedicated network adapter was assigned to the

DMZ web server (ubuntu apache2), and it is directly connected to pfSense OPT1. In this way the layout models a segmented enterprise (WAN > Firewall > LAN/DMZ) and matches the topology shown.

### Virtual Machine Images Used:

- pfSense 2.8.1 – pfSense amd64 ISO - netgate-installer-v1.1-RELEASE-amd64.iso
- Ubuntu 24.04 LTS – ubuntu-24.04.3-desktop-amd64.iso
- Windows 10 (x64) – Microsoft Windows 10 ISO
- Kali Linux 2025.3 – kali-linux-2025.3-installer-amd64.iso

Below is the proof of Network topology implementation using GNS3:



### Key Steps:

1. Installed new devices in GNS3: pfSense, Kali Linux, Ubuntu, and a Windows10-1 device.
2. All LAN VMs was connected to an internal network through Open vSwitch with an IP address of 192.168.1.0/24.
3. pfSense LAN (em1) was connected to the switch; WAN (em0) was connected to the Cloud NAT and OPT1 (em2) was connected to the DMZ server.
4. The connection of all topology elements is verified by the GNS3 Topology Summary Panel.
5. pfSense was booted first to allow interface identification and IP assignment.

### Traffic Flow Summary:

- Internet <=> WAN:
  - Internet host > Cloud1 (VirtualBox NAT 10.0.2.0/24) > pfSense WAN (em0) > internal networks (only VPN/allowed traffic).
  - LAN host > pfSense > WAN (em0) > Cloud1 (NAT) > Internet.
- LAN > DMZ:
  - LAN host (192.168.1.0/24) > pfSense LAN (em1) > pfSense OPT1 (em2) > Apache web server 192.168.2.10
  - (HTTP/HTTPS/ICMP allowed).
- DMZ > LAN (blocked):
  - DMZ host 192.168.2.10 > pfSense OPT1 (em2) > blocked from reaching 192.168.1.0/24 LAN.
- VPN > DMZ / LAN:
  - VPN client 10.8.0.2 > pfSense OpenVPN interface > DMZ 192.168.2.10 allowed
  - VPN > LAN 192.168.1.0/24 blocked (by firewall/IPS policy).

### 3.2 PFSENSE INTERFACE CONFIGURATION

The pfSense firewall automatically detected three interfaces WAN (em0), LAN (em1), and OPT1 (em2) during boot. Interface assignments and IPs were configured as seen in the below pfSense console output:

```
.168.1.100 (Local Database)

FreeBSD/amd64 (pfSense.lab.local) (ttyv0)
VirtualBox Virtual Machine - Netgate Device ID: 03259eca7731c596fd09
*** Welcome to pfSense 2.8.1-RELEASE (amd64) on pfSense ***

WAN (wan)  -> em0 -> v4/DHCP4: 10.0.2.15/24
LAN (lan)   -> em1 -> v4: 192.168.1.1/24
OPT1 (opt1) -> em2 -> v4: 192.168.2.1/24

0) Logout / Disconnect SSH          9) pfTop
1) Assign Interfaces                 10) Filter Logs
2) Set interface(s) IP address       11) Restart GUI
3) Reset admin account and password  12) PHP shell + pfSense tools
4) Reset to factory defaults         13) Update from console
5) Reboot system                    14) Enable Secure Shell (sshd)
6) Halt system                      15) Restore recent configuration
7) Ping host                        16) Restart PHP-FPM
8) Shell

Enter an option: 
```

### 3.3 FIREWALL RULES

After verifying that the pfSense VM had initialized correctly and all interfaces were assigned, the pfSense web GUI was accessed from the Windows10-1 LAN machine using <https://192.168.1.100>. From the dashboards, we configured necessary firewall policies were created for the WAN, LAN, and OPT1 (DMZ) interfaces as shown in this chart. This chart outlines the overall rules that will be used to segment networks, provide VPN access to users on the DMZ network, and govern how traffic flows between the WAN, LAN, and DMZ networks.

Rule for	Rule	Traffic	Action	Purpose / What Happens
WAN	Block Private Networks	RFC1918 ranges > WAN	Block	Prevents spoofed private-IP traffic from appearing on the WAN interface; default pfSense protection.
	Block Bogon Networks	Unassigned/Reserved IP ranges > WAN	Block	Blocks traffic from IP ranges that should never originate on the public internet to reduce scanning/spoofing attempts.
	Allow OpenVPN (1194/UDP)	Any > WAN Address:1194	Allow	Allows legitimate VPN clients to reach the OpenVPN server on the firewall.
	OpenVPN Wizard Rule	Any > WAN Address:1194	Allow	Auto-generated rule required for the OpenVPN server configuration and tunnel establishment.
LAN	Anti-Lockout Rule	LAN > Firewall LAN IP (TCP/443)	Allow	Ensures continuous admin access to the pfSense web interface; auto-generated by pfSense.
	Default IPv4 LAN Rule	LAN subnets > Any (IPv4)	Allow	Allows unrestricted outbound IPv4 traffic from LAN hosts for initial routing/NAT and connectivity testing.
	Default IPv6 LAN Rule	LAN subnets > Any (IPv6)	Allow	Allows unrestricted outbound IPv6 traffic from LAN hosts for testing and basic functionality validation.
OPT1	Allow DNS to Firewall	OPT1 > Firewall (TCP/53)	Allow	We enabled DNS queries from DMZ hosts for name resolution.
	Allow ICMP to Firewall	OPT1 > Firewall (ICMP echo)	Allow	We allowed DMZ hosts to ping the gateway for connectivity diagnostics.
	Allow DMZ Outbound Internet	OPT1 > Any	Allow	We allowed DMZ outbound traffic to the internet as required for updates/testing.
	Block DMZ to LAN	OPT1 > LAN subnets	Block	We enforced strict isolation by blocking all DMZ-to-LAN access.

	Allow LAN to DMZ Web Server	LAN subnets > 192.168.2.10:80	Allow	We permitted controlled LAN access to the DMZ web server over HTTP.
--	-----------------------------	-------------------------------	-------	---

<div> <div>Floating</div> <div>WAN</div> <div>LAN</div> <div>OPT1</div> <div>OpenVPN</div> </div>											
Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	0/0 B	*	RFC 1918 networks	*	*	*	*	*		Block private networks	
<input checked="" type="checkbox"/>	0/0 B	*	Reserved Not assigned by IANA	*	*	*	*	*		Block bogon networks	
<input type="checkbox"/>	0/0 B	IPv4 UDP	*	*	WAN address	1194 (OpenVPN)	*	none		Allow OpenVPN	
<input type="checkbox"/>	0/0 B	IPv4 UDP	*	*	WAN address	1194 (OpenVPN)	*	none		OpenVPN OpenVPN-Server-Cert wizard	

<div> <div>Floating</div> <div>WAN</div> <div>LAN</div> <div>OPT1</div> <div>OpenVPN</div> </div>											
Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input checked="" type="checkbox"/>	3/1.30 MiB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
<input type="checkbox"/>	36/22.88 MiB	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule	
<input type="checkbox"/>	0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	

<div> <div>Floating</div> <div>WAN</div> <div>LAN</div> <div>OPT1</div> <div>OpenVPN</div> </div>											
Rules (Drag to Change Order)											
<input type="checkbox"/>	States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
<input type="checkbox"/>	0/47 KiB	IPv4 TCP	OPT1 subnets	*	This Firewall (self)	53 (DNS)	*	none		Allow DMZ DNS	
<input type="checkbox"/>	0/5 KiB	IPv4 ICMP	OPT1 subnets	*	This Firewall (self)	*	*	none		Allow DMZ ping to firewall	
<input type="checkbox"/>	0/1.70 MiB	IPv4 *	OPT1 subnets	*	*	*	*	none		Allow DMZ Outbound interent	
<input type="checkbox"/>	<input checked="" type="checkbox"/> 0/840 B	IPv4 *	OPT1 subnets	*	LAN subnets	*	*	none		block DMZ access to LAN	
<input type="checkbox"/>	0/0 B	IPv4 TCP	LAN subnets	*	192.168.2.10	80 (HTTP)	*	none		Allow LAN to DMZ Web Server HTTP	

### 3.3 NAT CONFUGIRATION

Hybrid Outbound NAT was enabled with a custom NAT rule for the DMZ to ensure secure outbound connectivity.

Component	Configuration / Traffic	Purpose / What We Did
NAT Mode	Hybrid Outbound NAT	We enabled Hybrid Mode to use both auto-generated NAT rules and custom mappings for specific subnets (e.g., DMZ 192.168.2.0/24).
Custom NAT Mapping	DMZ (192.168.2.0/24) > WAN Address	We created a manual rule, so all outbound DMZ traffic is translated to the WAN public IP, enabling DMZ systems to access the internet while staying behind NAT.
Automatic NAT Rules	LAN + VPN subnets > WAN Address	pfSense automatically created NAT rules for internal LAN and VPN networks to ensure proper outbound connectivity without manual configuration.
Security Effect	NAT boundary for LAN + DMZ	Ensure that internal addressing is hidden externally and unsolicited inbound traffic cannot reach DMZ/LAN hosts.

Firewall / NAT / Outbound

Port Forward
1:1
Outbound
NPT

### Outbound NAT Mode

Mode	Automatic outbound NAT rule generation. (IPsec passthrough included)	Hybrid Outbound NAT rule generation. (Automatic Outbound NAT + rules below)	Manual Outbound NAT rule generation. (AON - Advanced Outbound NAT)	Disable Outbound NAT rule generation. (No Outbound NAT rules)
------	--	---	--	---

Save

### Mappings

	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description	Actions
<input type="checkbox"/>	WAN	192.168.2.0/24	*	*	*	WAN address	*		DMZ outbound NAT	

Add
Add
Delete
Toggle
Save

### Automatic Rules

	Interface	Source	Source Port	Destination	Destination Port	NAT Address	NAT Port	Static Port	Description
<input checked="" type="checkbox"/>	WAN	127.0.0.0/8 :: 1/128 192.168.1.0/24 192.168.2.0/24 10.8.0.0/24	*	*	500	WAN address	*	<input checked="" type="checkbox"/>	Auto created rule for ISAKMP
<input checked="" type="checkbox"/>	WAN	127.0.0.0/8 :: 1/128 192.168.1.0/24 192.168.2.0/24 10.8.0.0/24	*	*	*	WAN address	*	<input checked="" type="checkbox"/>	Auto created rule

### 3.5 BEHAVIOR VALIDATION

We validated the expected LAN, DMZ, VPN, and WAN behavior using a series of connectivity tests including ping, curl, and Nmap-based checks, as demonstrated below:

- Verified Apache web server by checking localhost page and confirming port 80 is listening.
- Confirmed DMZ server can reach OPT1 gateway and the internet (ping 192.168.2.1, ping 8.8.8.8).
- Validated LAN access to DMZ web server using ping and curl from Kali and browser access from Windows.
- Confirmed DMZ isolation by blocking access to LAN host (failed ping and curl to 192.168.1.10).
- Verified pfSense redirect and web portal protection using curl -I to 192.168.2.1.

```

ubuntu@Aparna:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:4c:f8:84 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.10/24 brd 192.168.2.255 scope global noprefixroute enp0s8
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe4c:f884/64 scope link
        valid_lft forever preferred_lft forever
ubuntu@Aparna:~$ sudo ss -tulnp | grep ':80'
tcp        LISTEN    0      128          *:*          *:*    users:((("apache2",pid=1430,fd=4),("apache2",pid=1431,fd=5)))
ubuntu@Aparna:~$ curl http://localhost
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<!--
  Modified from the Debian original for Ubuntu
  Last updated: 2022-03-22
  See: https://launchpad.net/bugs/1966804
-->
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<title>Apache2 Ubuntu Default Page: It works</title>
<style type="text/css" media="screen">
* {
margin: 0px 0px 0px 0px;
padding: 0px 0px 0px 0px;
}

body, html {
padding: 3px 3px 3px 3px;
background-color: #D8DBE2;

font-family: Ubuntu, Verdana, sans-serif;
font-size: 11pt;
text-align: center;

```

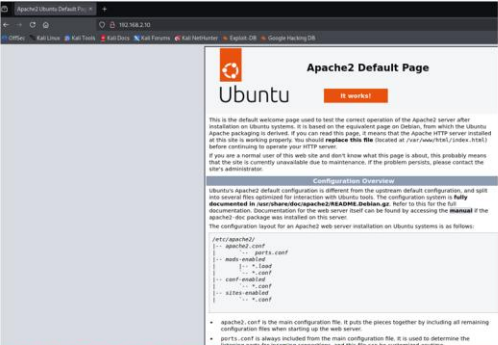
```

ubuntu@Aparna:~$ ping 192.168.2.1
PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=13.7 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=6.53 ms
64 bytes from 192.168.2.1: icmp_seq=3 ttl=64 time=2.74 ms
64 bytes from 192.168.2.1: icmp_seq=4 ttl=64 time=3.16 ms
64 bytes from 192.168.2.1: icmp_seq=5 ttl=64 time=6.94 ms
^C
--- 192.168.2.1 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4221ms
rtt min/avg/max/mdev = 2.742/6.617/13.723/3.939 ms
ubuntu@Aparna:~$ ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=109 time=23.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=109 time=36.7 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=109 time=25.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=109 time=25.4 ms
64 bytes from 8.8.8.8: icmp_seq=5 ttl=109 time=22.2 ms
^C
--- 8.8.8.8 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4011ms
rtt min/avg/max/mdev = 22.198/26.661/36.659/5.154 ms
ubuntu@Aparna:~$

```



```
kali@kali: ~  
Session Actions Edit View Help  
  
kali@kali:~$ ping 192.168.2.10  
PING 192.168.2.10 (192.168.2.10) 56(84) bytes of data:  
64 bytes from 192.168.2.10: icmp_seq=1 ttl=63 time=6.36 ms  
64 bytes from 192.168.2.10: icmp_seq=2 ttl=63 time=6.20 ms  
64 bytes from 192.168.2.10: icmp_seq=3 ttl=63 time=7.26 ms  
64 bytes from 192.168.2.10: icmp_seq=4 ttl=63 time=9.73 ms  
64 bytes from 192.168.2.10: icmp_seq=5 ttl=63 time=6.43 ms  
^C  
--- 192.168.2.10 ping statistics ---  
5 packets transmitted, 5 received, 0% packet loss, time 4010ms  
rtt min/avg/max/mdev = 6.204/7.196/9.731/1.319 ms  
  
kali@kali:~$ curl -v http://192.168.2.10  
  
* Trying 192.168.2.10:80...  
* Connected to 192.168.2.10 (192.168.2.10) port 80  
* using HTTP/1.x  
> GET / HTTP/1.1  
> Host: 192.168.2.10  
> User-Agent: curl/8.15.0  
> Accept: */*  
>  
* Request completely sent off  
< HTTP/1.1 200 OK  
< Date: Fri, 28 Nov 2025 08:13:38 GMT  
< Server: Apache/2.4.58 (Ubuntu)  
< Last-Modified: Thu, 27 Nov 2025 23:16:14 GMT  
< ETag: "29af-6449bb302d469"  
< Accept-Ranges: bytes  
< Content-Length: 10671  
< Vary: Accept-Encoding  
< Content-Type: text/html  
<  
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">  
<html xmlns="http://www.w3.org/1999/xhtml">  
  <!--  
    Modified from the Debian original for Ubuntu  
    Last updated: 2022-03-22  
    See: https://launchpad.net/bugs/1966004  
  -->  
  <head>  
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />  
    <title>Apache2 Ubuntu Default Page: It works</title>  
    <style type="text/css" media="screen">  
      * {  
        margin: 0px 0px 0px 0px;  
        padding: 0px 0px 0px 0px;  
      }  
  
      body, html {
```



```
ubuntu@Aparna:~$ curl -I http://192.168.2.1
HTTP/1.1 301 Moved Permanently
Server: nginx
Date: Fri, 28 Nov 2025 08:36:54 GMT
Content-Type: text/html
Content-Length: 162
Connection: keep-alive
Location: https://192.168.2.1/
X-Frame-Options: SAMEORIGIN

ubuntu@Aparna:~$
```

```
ubuntu@Aparna:~$ ping 192.168.1.10
PING 192.168.1.10 (192.168.1.10) 56(84) bytes of data.
^C
--- 192.168.1.10 ping statistics ---
9 packets transmitted, 0 received, 100% packet loss, time 8595ms

ubuntu@Aparna:~$ curl http://192.168.1.10
curl: (28) Failed to connect to 192.168.1.10 port 80 after 136549 ms: Couldn't connect to server
```

Commands used:

Command	Purpose
ip a	Check interface & IPs
sudo ss -tulnp   grep ':80'	Check Apache listening
curl http://localhost	Test Apache locally
ping 192.168.2.1	Ping pfSense DMZ gateway
ping 8.8.8.8	Test DMZ outbound internet
ping 192.168.2.10	Ping DMZ server from Kali
curl -v http://192.168.2.10	Test HTTP access from Kali

## 4. VPN CONFIGURATION

To validate secure remote access, we configured pfSense as an OpenVPN server (UDP/1194), exported a client profile, and connected from a Windows machine. Upon establishing a VPN tunnel, the client was assigned an IP from the 10.8.0.0/24 network and could then securely connect to the web server in the DMZ while at the same time being restricted from connecting to the LAN. This testing validated that all aspects of the OpenVPN service including certificates, routing and the firewalls segmenting policies functioned as intended.

### Key Validation Steps:

- Establishment of a working VPN connection (OpenVPN client on Windows, IP address assigned as 10.8.0.2).
- Web access via the DMZ Web Server (192.168.2.10) was possible via the VPN (HTTP web pages were accessed).
- The test pings/curls were completed confirming that VPN-to-DMZ is enabled while VPN-to-LAN is disabled.
- pfSense routing/firewall configuration is enforced by the OpenVPN rules page.
- LAN device separation from VPN clients was confirmed to be in place as required for security.

### Commands used:

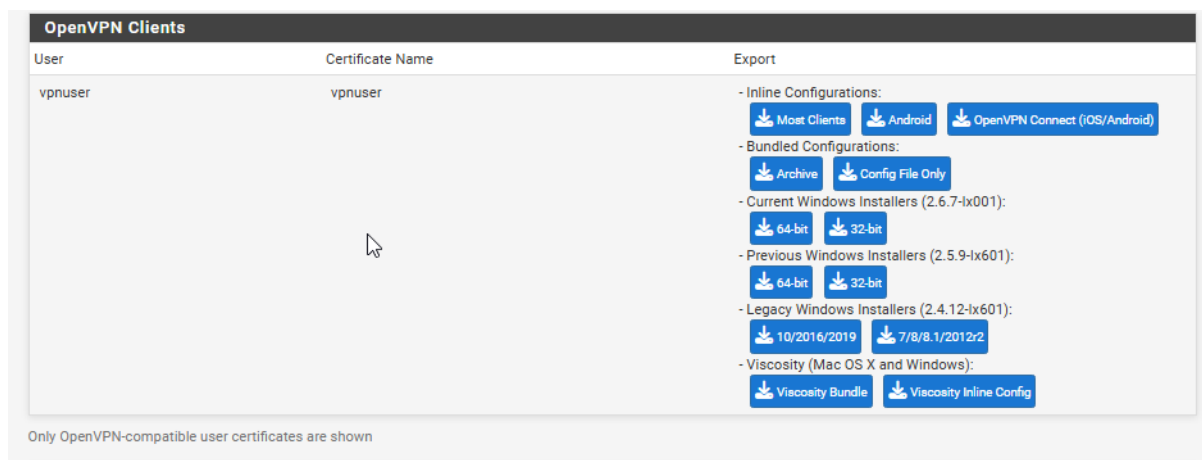
- ping 192.168.2.10 – succeeds - Traffic from the VPN tunnel can reach the DMZ network
- ping 192.168.1.10 – fails - VPN users cannot reach the internal LAN

The top section shows two screenshots. The left screenshot is a Windows command prompt window titled "OpenVPN Connection (pfSense-UDP4-1194-vpnuser-config)". It displays a log of the connection process, including the state "Connecting", the use of the "OpenVPN GUI" version 11.45.0.0/2.6.7, and the successful establishment of a tunnel on interface 3 with IP 10.8.0.2. The right screenshot is the "OpenVPN / Client Export Utility" web interface. It shows the "Client" tab selected, with the "Remote Access Server" set to "OpenVPN-Server-Cert UDP4:1194". The "Client Connection Behavior" section includes options for "Host Name Resolution" (Interface IP Address), "Verify Server CN" (Automatic), and "Block Outside DNS" (checked). The "Legacy Client" and "Silent Installer" options are also visible.

The bottom screenshot is a screenshot of the pfSense web interface, specifically the "VPN / OpenVPN / Servers" page. It shows a table of OpenVPN servers with the following columns: Interface, Protocol / Port, Tunnel Network, Mode / Crypto, Description, and Actions. The table contains one entry for the "WAN" interface, using "UDP4 / 1194" (TUN) protocol, with a "Tunnel Network" of "10.8.0.0/24". The "Mode / Crypto" section lists "Mode: Remote Access ( SSL/TLS + User Auth )", "Data Ciphers: AES-256-GCM, AES-128-GCM, CHACHA20-POLY1305, AES-256-CBC", "Digest: SHA256", and "D-H Params: 2048 bits". The "Description" is "OpenVPN-Server-Cert". The "Actions" column includes icons for edit, copy, and delete. A green "+ Add" button is located at the bottom right of the table.

Interface	Protocol / Port	Tunnel Network	Mode / Crypto	Description	Actions
WAN	UDP4 / 1194 (TUN)	10.8.0.0/24	Mode: Remote Access ( SSL/TLS + User Auth ) Data Ciphers: AES-256-GCM, AES-128-GCM, CHACHA20-POLY1305, AES-256-CBC Digest: SHA256 D-H Params: 2048 bits	OpenVPN-Server-Cert	





```

Aparna>ping 192.162.2.10
Pinging 192.162.2.10 with 32 bytes of data:
Reply from 192.162.2.10: bytes=32 time=467ms TTL=229
Reply from 192.162.2.10: bytes=32 time=500ms TTL=229
Reply from 192.162.2.10: bytes=32 time=331ms TTL=229
Reply from 192.162.2.10: bytes=32 time=289ms TTL=229

Ping statistics for 192.162.2.10:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 289ms, Maximum = 500ms, Average = 396ms

Aparna>
Aparna>ping 192.162.1.10
Pinging 192.162.1.10 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.

Ping statistics for 192.162.1.10:
    Packets: Sent = 4, Received = 0, Lost = 4 (100% loss),

Aparna>
Aparna>ipconfig

Windows IP Configuration

Unknown adapter OpenVPN Wintun:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :

Ethernet adapter Ethernet:

    Connection-specific DNS Suffix  . : lab.local
    Link-local IPv6 Address . . . . . : fe80::fd56:f43b:ec58:4df5%6
    IPv4 Address. . . . . : 192.168.1.100
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Unknown adapter OpenVPN TAP-Windows6:

    Connection-specific DNS Suffix  . :
    Link-local IPv6 Address . . . . . : fe80::23d2:64df:a96a:685c%3
    IPv4 Address. . . . . : 10.8.0.2
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . :

Unknown adapter OpenVPN Data Channel Offload:

    Media State . . . . . : Media disconnected
    Connection-specific DNS Suffix  . :
  
```

## 5. SURICATA SETUP

1. Enabled Suricata in inline IPS mode on LAN and OPT1 for real-time blocking.
2. Activated ET Open Ruleset, providing updated signatures for malware, scans, and anomalies.
3. Verified successful rules download and updates via the Suricata Updates tab.
4. Confirmed Suricata service was running and inspecting live traffic on both interfaces.
5. Ensured IPS blocked malicious activity during test scans and attack simulations.

Category	Technical Details
Mode	Inline IPS Mode (netmap)
Enabled Interfaces	LAN (em1), OPT1/DMZ (em2)
Rule Source	Emerging Threats Open (ET Open)
Ruleset Update	Updated successfully; MD5 verified
Blocking Mode	Enabled (drops malicious packets)
Engine Features	HTTP parser (libhttp), TCP stream reassembly, IP defrag, checksum validation
Pattern Matcher	Hyperscan

<b>Drop/Alert Events Observed</b>	SYN flood, ICMP flood, UDP flood, Nmap scan, DIRB brute-force, Slowloris, SQL injection attempts
<b>Common Alert Types</b>	TCP anomaly, ICMP anomaly, HTTP directory brute force, Slowloris detection, SQL injection signatures
<b>Log Output</b>	/var/log/suricata/eve.json
<b>Verification</b>	Rules loaded, Suricata running, attacks detected/blocked on LAN & DMZ

Services / Suricata

Interfaces

Global Settings

Updates

Alerts

Blocks

Files

Pass Lists

Suppress

Logs View

Logs Mgmt

SID Mgmt

Sync IP Lists

Interface Settings Overview

Interface	Suricata Status	Pattern Match	Blocking Mode	Description	Actions
<input checked="" type="checkbox"/> LAN (em1)		AUTO	INLINE IPS	LAN	
<input type="checkbox"/> OPT1 (em2)		AUTO	INLINE IPS	OPT1	

Services / Suricata / Global Settings

Interfaces

Global Settings

Updates

Alerts

Blocks

Files

Pass Lists

Suppress

Logs View

Logs Mgmt

SID Mgmt

Sync IP Lists

Please Choose The Type Of Rules You Wish To Download

Install ETOpen Emerging Threats rules

☒ ETOpen is a free open source set of Suricata rules whose coverage is more limited than ETPro.  
☐ Use a custom URL for ETOpen downloads  
 Enabling the custom URL option will force the use of a custom user-supplied URL when downloading ETOpen rules.

Install ETPro Emerging Threats rules

☐ ETPro for Suricata offers daily updates and extensive coverage of current malware threats.  
☐ Use a custom URL for ETPro rule downloads  
 The ETPro rules contain all of the ETOpen rules, so the ETOpen rules are not required and are disabled when the ETPro rules are selected. Sign Up for an ETPro Account. Enabling the custom URL option will force the use of a custom user-supplied URL when downloading ETPro rules.

Install Snort rules

☐ Snort free Registered User or paid Subscriber rules.  
☐ Use a custom URL for Snort rule downloads  
 Sign Up for a free Registered User Rules Account.  
 Sign Up for paid Snort Subscriber Rule Set (by Talos).  
 Enabling the custom URL option will force the use of a custom user-supplied URL when downloading Snort Subscriber rules.

Install Snort GPLv2 Community rules

☒ The Snort Community Ruleset is a GPLv2 Talos-certified ruleset that is distributed free of charge without any Snort Subscriber License restrictions.  
☐ Use a custom URL for Snort GPLv2 rule downloads  
 This ruleset is updated daily and is a subset of the subscriber ruleset. If you are a Snort Subscriber Rules customer (paid subscriber), the community ruleset is already built into your download of the Snort Subscriber rules, and there is no benefit in adding this rule set separately.

Install Feodo Tracker Botnet C2 IP rules

☐ The Feodo Botnet C2 IP Ruleset contains Dridex and Emotet/Feodo botnet command and control servers (C&Cs) tracked by Feodo Tracker.

Services / Suricata / Updates

Interfaces

Global Settings

Updates

Alerts

Blocks

Files

Pass Lists

Suppress

Logs View

Logs Mgmt

SID Mgmt

Sync IP Lists

INSTALLED RULE SET MD5 SIGNATURES

Rule Set Name/Publisher	MD5 Signature Hash	MD5 Signature Date
Emerging Threats Open Rules	7ca7ee0a2baca189d1c0b565231137	Tuesday, 18-Nov-25 16:47:16 CST
Snort Subscriber Rules	Not Enabled	Not Enabled
Snort GPLv2 Community Rules	336d858405e0daf071da4b66b6819b	Tuesday, 18-Nov-25 16:47:17 CST
Feodo Tracker Botnet C2 IP Rules	Not Enabled	Not Enabled
ABUSE.ch SSL Blacklist Rules	Not Enabled	Not Enabled

UPDATE YOUR RULE SET

Last Update: Nov-18 2025 16:48

Result: success

Update

Force

MANAGE RULE SET LOG

View

Clear

## 6. ATTACK SIMULATION & SECURITY CONTROLS VALIDATION

To validate the security controls, multiple attacks were launched from the Kali attacker system toward the DMZ web server (192.168.2.10) and the protected LAN host (192.168.1.10). Wireshark captures were taken on the target interfaces to observe real packet-level behavior, and Suricata alerts were reviewed to verify IPS action.

### Attacks Performed Summary:

- A reconnaissance NMAP scan was conducted on the DMZ web server (192.168.2.10); it is apparent that port 80 is active, and Apache was indicated via NMAP.
- Full NMAP Scan of the LAN host (192.168.1.10) - It appears the LAN host is unresponsive (PfSense+Suricata are rejecting all incoming packets).
- SYN Flood with Hping3 - Wireshark displays a never-ending stream of SYN packets directed at the DMZ Interface.
- UDP/ICMP floods - high-rate packets are visible in Wireshark showing that packets were generated at a high rate.
- Slowloris (http dos) sent to the DMZ webserver - Wireshark capture shows slow get requests (multiple http get requests made slowly over time).
- Directory brute-force (DIRB) against /index.html - http get requests are visible in Wireshark.

- SQL-injection attempts (http requests with malicious payload) are visible in Wireshark.

## What the Wireshark captures showed:

- Continuous ICMP, UDP, TCP, SYN packets hitting the DMZ interface (showing flood behavior).
- Bursts of http get/post requests from dirb, Slowloris and other enumeration tools.
- No response packets when attempting to communicate with the lan host that has been blocked by the firewall (proving the firewall is working as expected).
- Patterns of behavior seen in Wireshark that match Suricata's detections (e.g., malformed TCP, flood anomaly).

The image displays a Kali Linux terminal and Wireshark network traffic captures. The terminal shows the execution of Nmap and Hydra commands against the target IP 192.168.2.10. The Wireshark interface shows a flood of SYN, ICMP, and UDP packets, as well as HTTP requests from tools like Slowloris and Nmap. A blue circle with text 'SYN Flood', 'ICMP Flood', 'Slowloris', and 'Nmap Full Scan' is overlaid on the Wireshark interface.

The image shows a Wireshark packet capture of HTTP traffic. The packet list displays a series of HTTP requests, including GET and POST methods. The packet details pane shows the structure of an HTTP request, including the status bar, interface, Ethernet II, and Hypertext Transfer Protocol. The packet bytes pane shows the raw data of the HTTP request.



## Commands used for Attack:

Command	Attack Type / Purpose
<code>nmap -sS 192.168.2.10</code>	Reconnaissance (Port Scanning)
<code>nmap -A 192.168.2.10</code>	Reconnaissance (Deep Enumeration)
<code>nmap -A 192.168.1.10</code>	Reconnaissance (Service Enumeration)
<code>hydra -l admin -p /usr/share/wordlists/rockyou.txt 192.168.2.10 http-get /</code>	Brute Force Attack
<code>dirb http://192.168.2.10/</code>	Directory Enumeration
<code>sudo hping3 -S --flood -p 80 192.168.2.10</code>	SYN Flood (DoS Attack)
<code>sudo ping -f 192.168.2.10</code>	ICMP Flood (DoS Attack)
<code>sudo hping3 --flood -V 192.168.2.10</code>	ICMP Flood (DoS Attack)
<code>sudo hping3 -S --flood -p 22 192.168.1.104</code>	SYN Flood (DoS Attack)
<code>sudo hping3 --flood -V 192.168.1.104</code>	ICMP Flood (DoS Attack)
<code>slowhttptest -c 500 -H -g -o slowhttptest -u http://192.168.2.10/index.html</code>	Slowloris (Application Layer DoS)
<code>sudo hping3 --flood --udp -p 80 192.168.1.103</code>	UDP Flood Attack
<code>sudo ping -f 192.168.1.102</code>	ICMP Flood (Ping Flood)

```

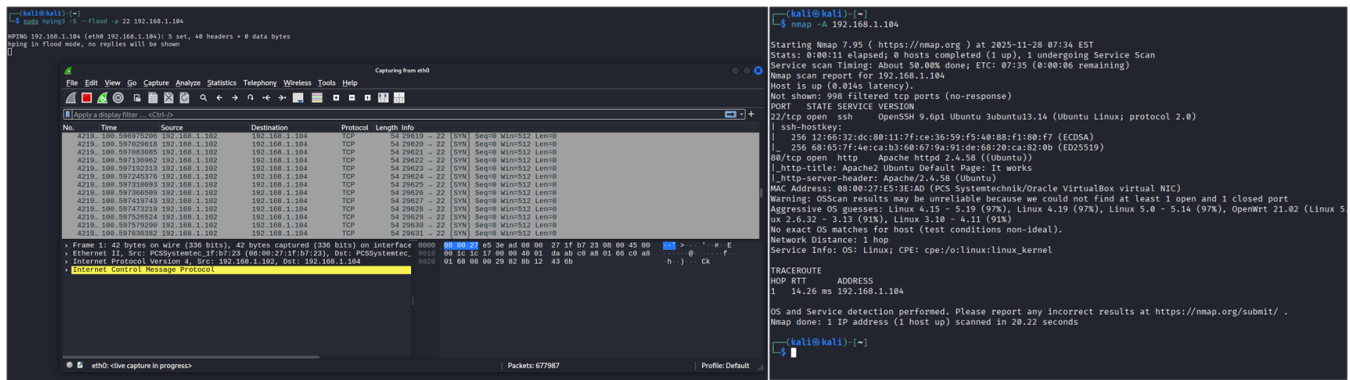
kali@kali:~$ nmap -sS 192.168.2.10
Starting Nmap 7.95 (https://nmap.org) at 2025-11-26 05:35 EST
Stats: 0/0/11 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 66.2% done; ETC: 03:35 (8:00:06 remaining)
Nmap scan report for 192.168.2.10
Host is up (0.11s latency).
Not shown: 996 filtered tcp ports (no-response)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http
443/tcp   open  https
8080/tcp  open  http
Service detected on 80: http
Service detected on 443: https
Service detected on 8080: http
Aggressive OS guesses: Linux 4.19 - 5.15 (97%), Linux 4.15 - 5.19 (91%), Linux 4.19 (91%), Linux 5.0 - 5.14 (91%), Open
6.4 (90%), Linux 5.4 (88%), Linux 6.0 (88%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 2 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
OS and Service detection performed. Please report any incorrect results at https://nmap.org/report/.
Nmap done: 1 IP address (1 host up) scanned in 33.69 seconds

```

```

kali@kali:~$ sudo hping3 --flood --udp -p 80 192.168.1.103
hping3 192.168.1.103 (eth0 192.168.1.103): udp mode set, 20 headers + 0 data bytes
hping3 in flood mode, no replies will be shown

```



## 7. SURICATA ALERTS

The Suricata intrusion detection system successfully detected and blocked traffic generated by simulated attacks using a number of different alert types; validating that Suricata can be used as an inline intrusion prevention system (IPS).

### Key Inline IPS Observations:

- Suricata was able to stop the TCP SYN flood traffic, the UDP burst traffic, and the ICMP anomaly traffic.
- Suricata reported on the following events:
  - Generic protocol commands in TCP
  - Flood/echo anomalies in ICMP
  - Anomalous HTTP activity
  - Possible patterns of a brute force attempt against directories
- The traffic being sent by the attacker was stopped by Suricata as the attacker attempted to send traffic to a LAN host, which allowed for firewalls to maintain segmentation of networks.
- Suricata also blocked several malicious packets that were produced during the high-rate flooding testing from reaching the DMZ server.

Alert Log View Filter										
Last 250 Alert Entries. (Most recent entries are listed first)										
Note: Alerts triggered by DROP rules that resulted in dropped (blocked) packets are shown with highlighted rows below.										
Date	Action	Pri	Proto	Class	Src	SPort	Dst	DPort	GID:SID	Description
11/28/2025 04:16:59		3	TCP	Generic Protocol Command Decode	192.168.2.10	80	192.168.1.101	34054	1:2260002	SURICATA Applayer Detect protocol only one direction
11/28/2025 04:16:57		3	ICMP	Generic Protocol Command Decode	192.168.2.10	0	192.168.1.101	9	1:2200025	SURICATA ICMPv4 unknown code
11/28/2025 04:16:57		3	ICMP	Generic Protocol Command Decode	192.168.1.101	8	192.168.2.10	9	1:2200025	SURICATA ICMPv4 unknown code
11/28/2025 04:16:55		3	ICMP	Generic Protocol Command Decode	192.168.2.10	0	192.168.1.101	9	1:2200025	SURICATA ICMPv4 unknown code
11/28/2025 04:16:55		3	ICMP	Generic Protocol Command Decode	192.168.1.101	8	192.168.2.10	9	1:2200025	SURICATA ICMPv4 unknown code
11/28/2025 04:10:44		3	TCP	Generic Protocol Command Decode	20.25.227.174	443	192.168.1.100	49843	1:2210038	SURICATA STREAM FIN out of window
11/28/2025 04:10:32		3	TCP	Generic Protocol Command Decode	20.25.227.174	443	192.168.1.100	49843	1:2210038	SURICATA STREAM FIN out of window

## 8. CONCLUSION

This project effectively demonstrated a well-structured, multi-zone network utilizing current industry best practice. pfSense proved to be a strong firewall enforcer; NAT (Network Address Translation) correctly routed and maintained privacy for the internal network; and the DMZ segmented the internal network from external networks thus preventing unauthorized access to internal resources. Only the appropriate networks had access to the Apache Server; therefore, it was secure. A VPN was optionally configured to provide additional secure remote access. An IDS/IPS system (Suricata) was also configured to monitor for malicious activity at all times.

The environment is a good representation of what can be found in a production environment that uses multiple defense-in-depth strategies. Therefore, it reinforces concepts such as segmentation, least privilege access, intrusion detection, and securely accessing a remote location. Implementing this project reinforced my knowledge of how I would perform many tasks in today's network security environment.