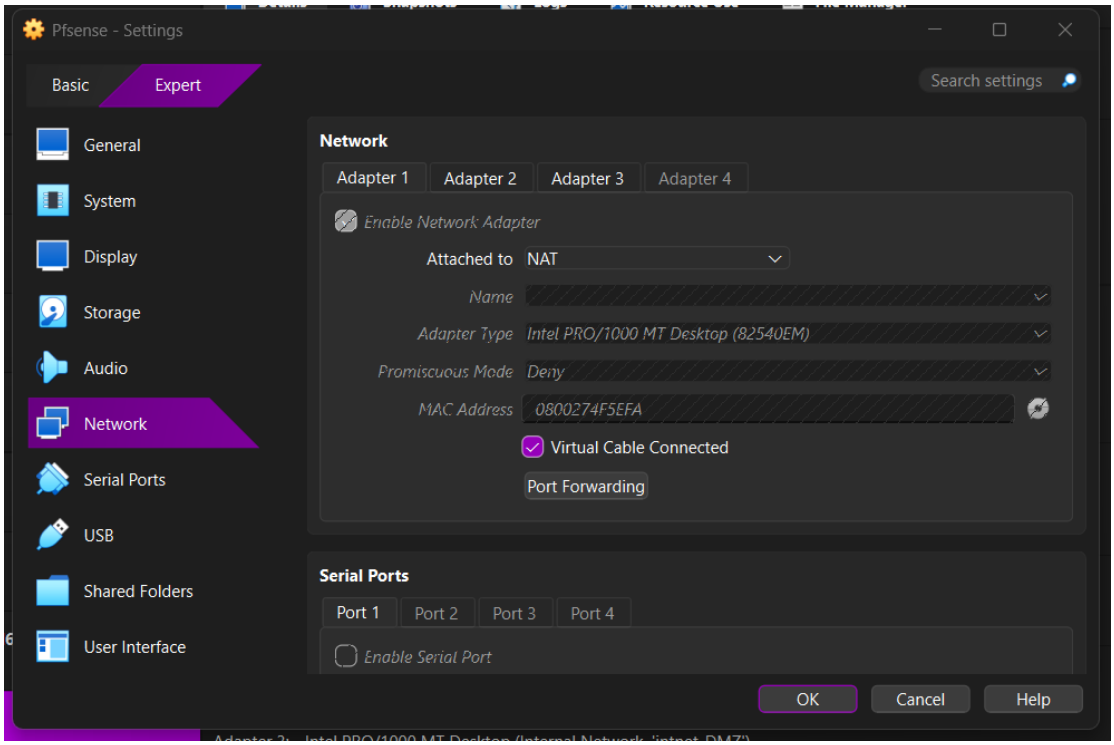
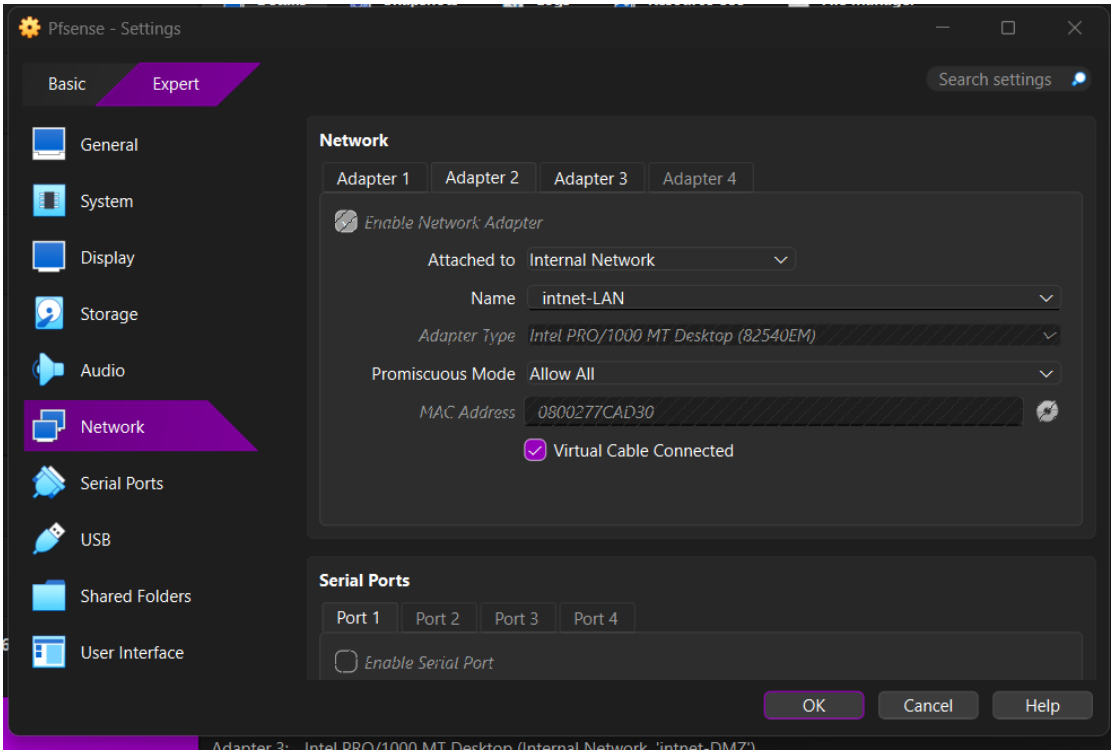
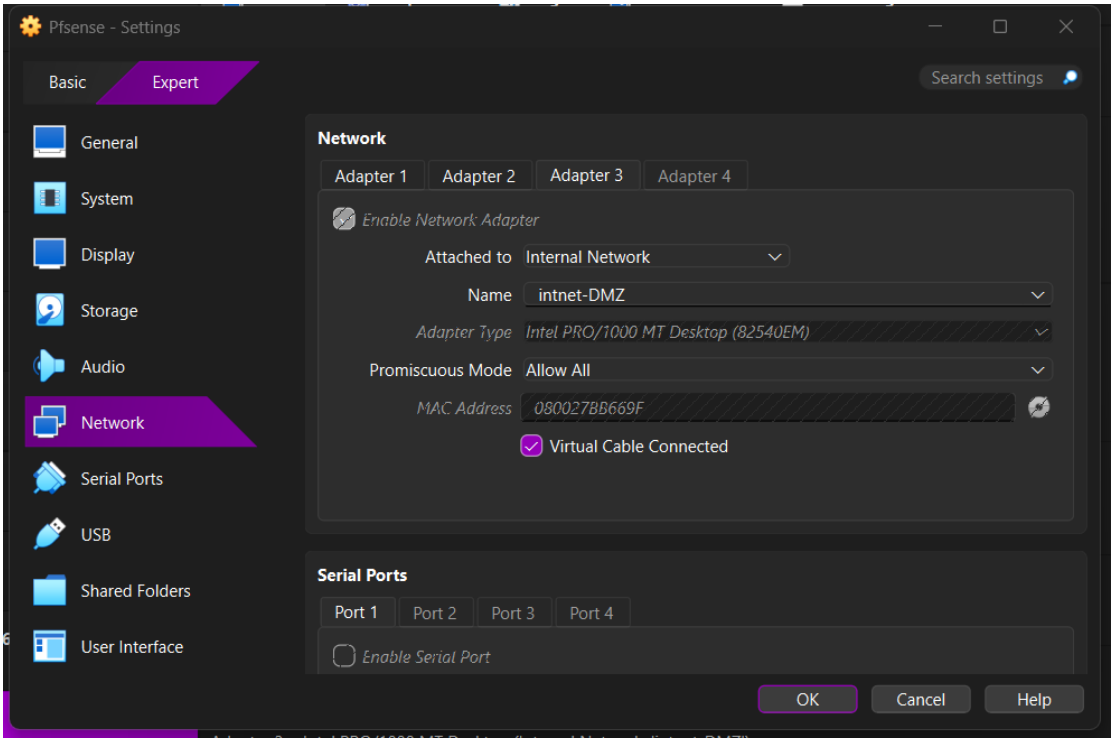
The project instructions suggested using GNS3, but due to connection issues, I couldn't finish the setup of the server between the GNS3 GUI and the local GNS3 VM.
After reinstalling the VM, changing network adapters, enabling the GNS3 VM in preferences, and testing loopback connectivity, the GNS3 client still showed a "Server is not connected" status. This made it impossible to create topologies or start appliances.
As the problem was with the environment and not the project, I decided to continue with VirtualBox that gave me a stable environment to configure pfSense, LAN, and DMZ networks as per the project requirements. With the VirtualBox setup, I was able to complete NAT, firewall rules, DMZ isolation, penetration testing, and monitoring functionalities in a reliable and reproducible manner.



The screenshot portrays the setup of the first adapter on the pfSense virtual machine that is connected to the NAT network. This interface is the WAN port of the firewall; thus it is the one that through VirtualBox's NAT engine provides Internet access to the internal LAN and DMZ networks. So, assigning the WAN interface to NAT, pfSense gets a private IP from VirtualBox but it is still capable of routing the outbound traffic to the Internet. Hence, this becomes the external facing "WAN side" of the topology which makes possible NAT outbound mappings, firewall rules, and the real-world simulation of external connections.

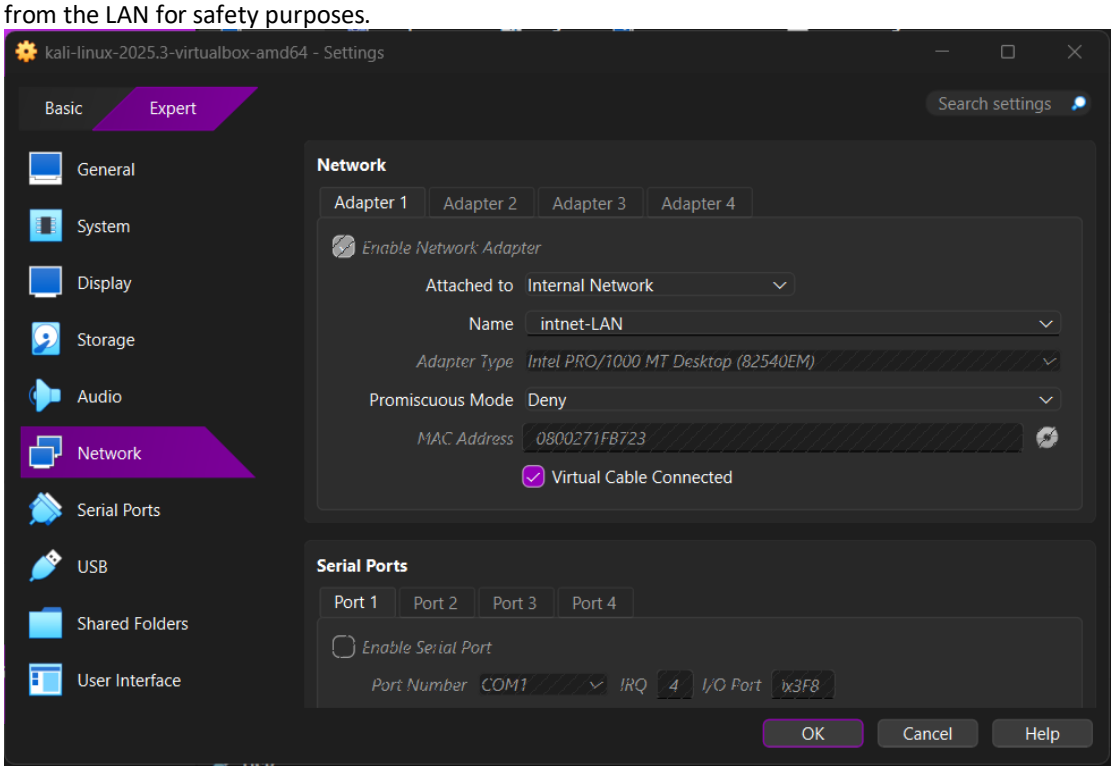


This screenshot depicts pfSense Adapter 2 that has been set up as an "Internal Network" with the label intnet-LAN. In this virtual topology, the interface is the LAN gateway. All LAN-side devices (Kali and Windows) are linked to this internal network, thus, they can get DHCP addresses from pfSense and communicate with each other via the LAN subnet (192.168.7.0/24). Promiscuous mode is configured as "Allow All", which guarantees that pfSense will be able to handle ARP, DHCP, and broadcast packets properly, these packets are coming from the LAN devices and are necessary for routing and firewall functions.

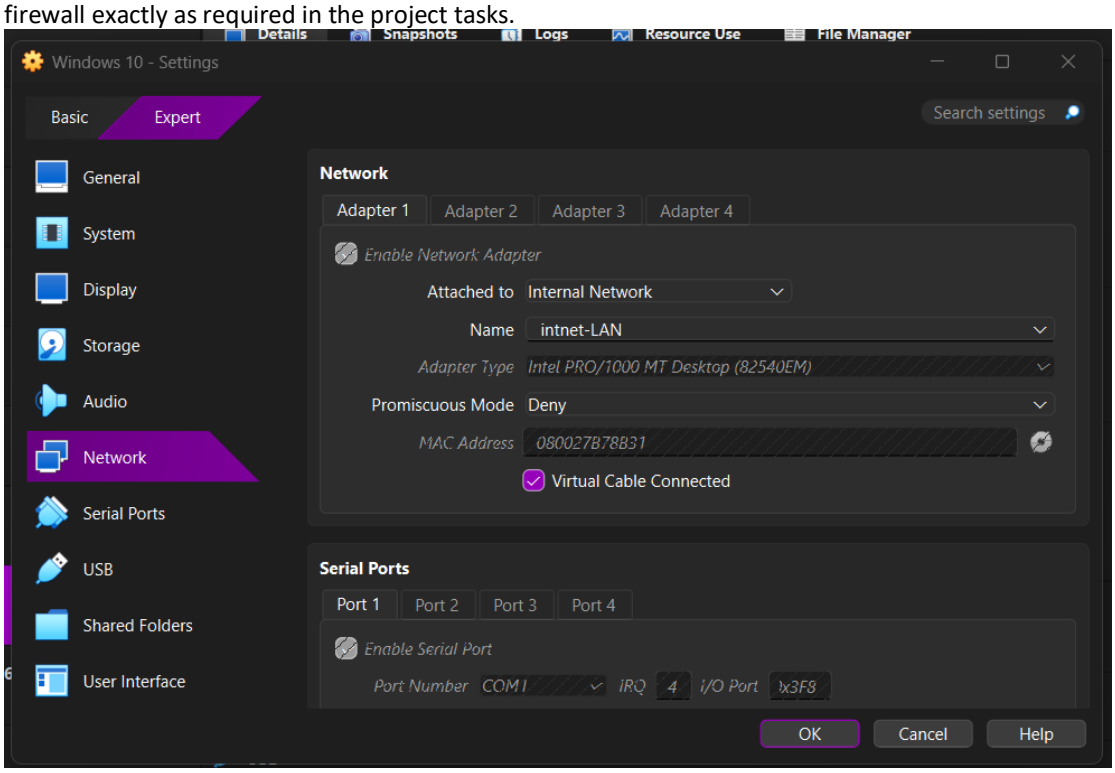


The image captured here shows the pfSense Adapter 3 that has been set up for the "Internal Network" which is the netint-DMZ named network and is a network dedicated to the DMZ branch (192.168.8.0/24). By means of this adapter, the DMZ turned into a last-hop router, which made it

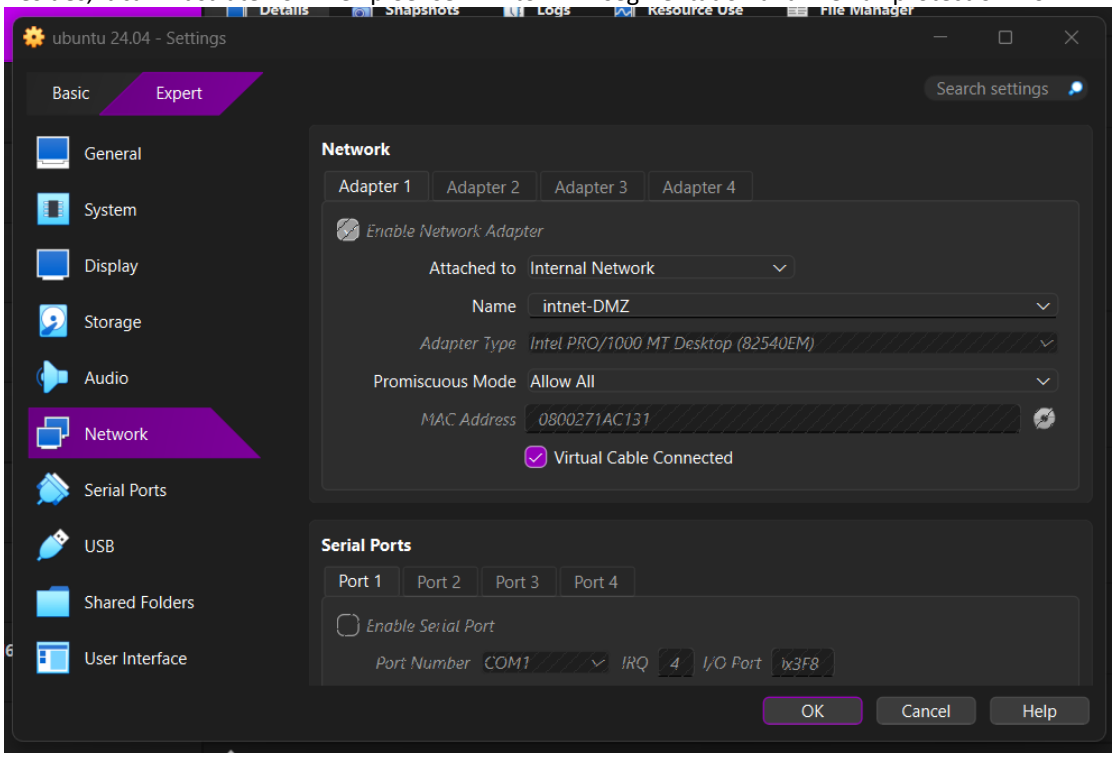
possible to separate the public-facing or the most vulnerable hosts from the main LAN. For promiscuous mode, the setting is at "Allow All" and, therefore, pfSense is capable of checking all packets originating from the DMZ and enforcing the firewall policies. Through this adapter, which is the interface, the DMZ hosts (like Ubuntu Server) can access the outside world whilst being separated from the LAN for safety purposes.



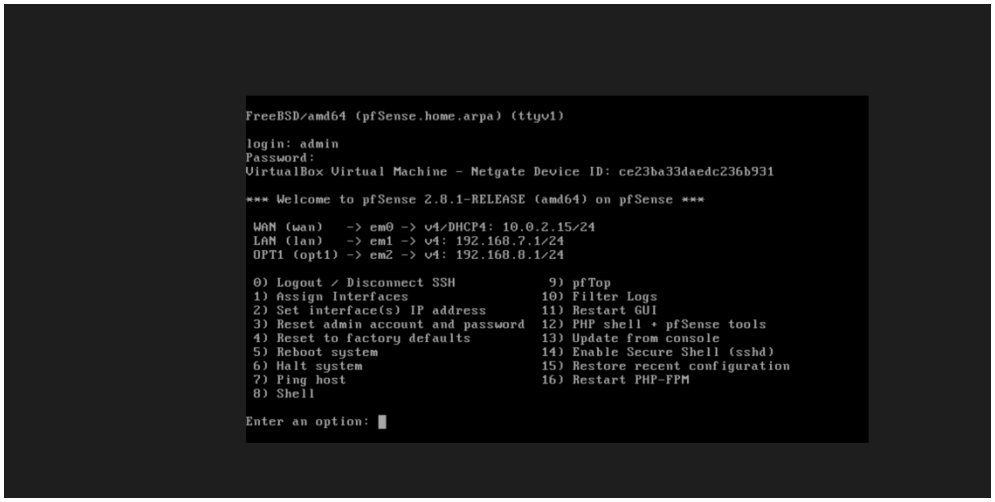
This screenshot shows Kali Linux connected to the **intnet-LAN** internal network. As a penetration testing machine within the LAN, Kali must remain fully isolated from the DMZ by pfSense but still allowed to communicate internally. Promiscuous mode is disabled to ensure Kali can only see its own traffic, providing a realistic LAN workstation environment. This connection allows Kali to perform vulnerability scanning, brute-forcing, and DoS attacks against DMZ systems through the pfSense firewall exactly as required in the project tasks.



This image is a visual representation of the Windows 10 virtual machine which is similarlyconnected to intnet-LAN
hence, it becomes a part of the secured internal network. The situation is the same as with Kali; it gets DHCP configuration from pfSense and runs under the LAN firewall rules. By having Windows on the LAN segment, one can still carry out a genuine scenario of a typical user workstation which needs to be safe from any kind of attack coming from the DMZ or services that are publicly exposed. Besides, it can illustrate how well pfSense LAN-to-DMZ segmentation and firewall protection work..

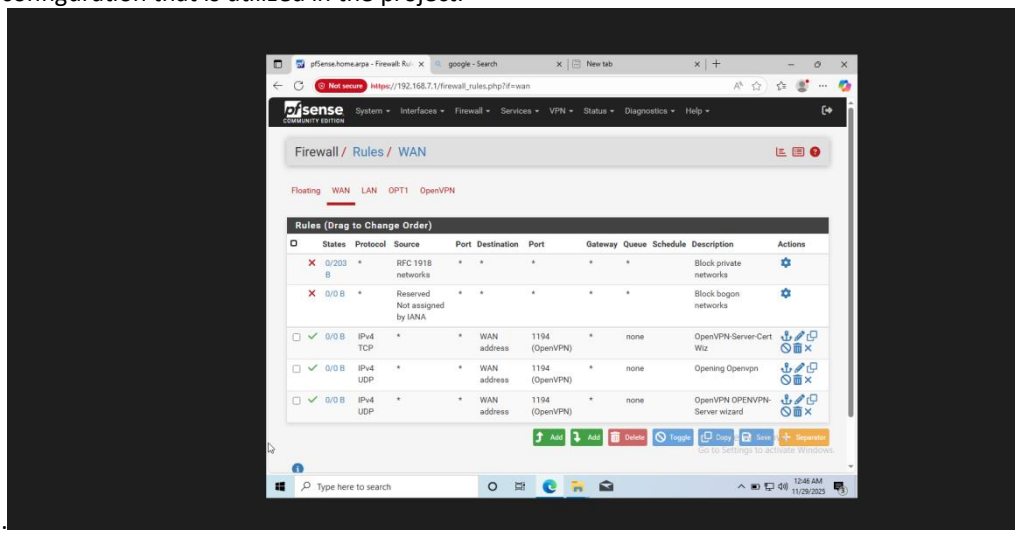


This screenshot shows the Ubuntu VM connected to the **intnet-DMZ** network thus defining it as the DMZ server. Ubuntu is getting its IP from pfSense's DMZ DHCP scope and is deliberately set outside the LAN in order to host services that are vulnerable or exposed. Promiscuous mode is turned on so that Ubuntu can grab the attack traffic that is coming in for analysis by Wireshark or tcpdump. With this positioning, the testing of firewall hardening, DoS mitigation, brute-force detection, and IDS/IPS alert generation is possible.

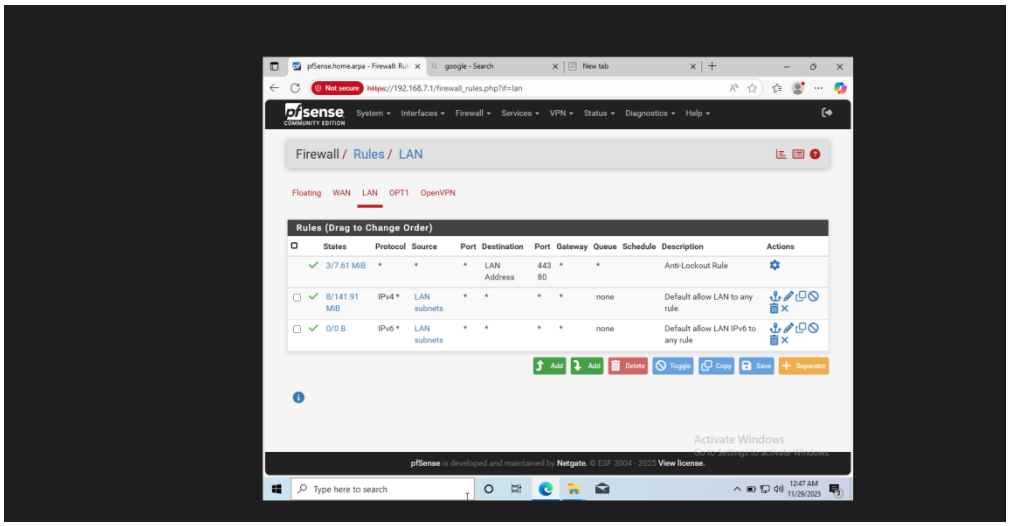


This screenshot shows the pfSense console menu immediately after booting, where the system displays WAN, LAN, and OPT1 interface assignments and their corresponding IP addresses. From here, administrative tasks such as assigning interfaces, configuring IP addresses, enabling SSH, resetting It is

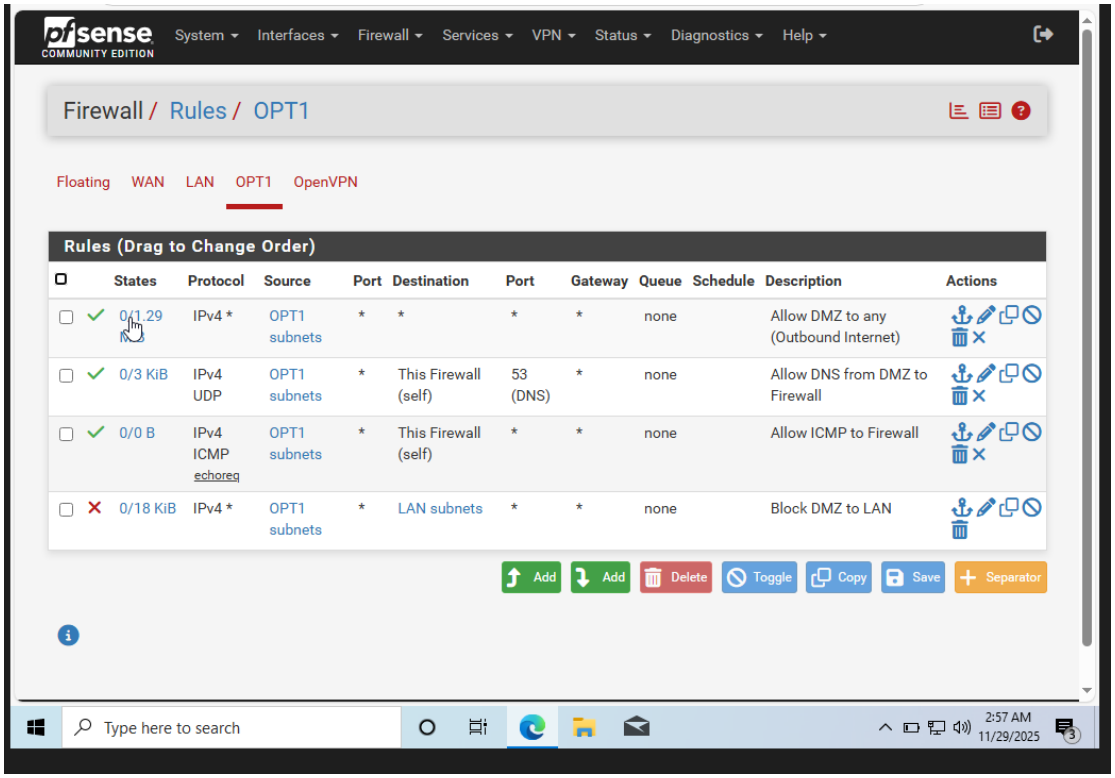
possible to do a reset on the firewall after a change of the admin credentials. This is an indication that pfSense has recognized all the VirtualBox network adapters and has properly associated them with WAN(em0), LAN(em1), and DMZ/OPT1(em2), thus creating the base of the three-network firewall configuration that is utilized in the project.



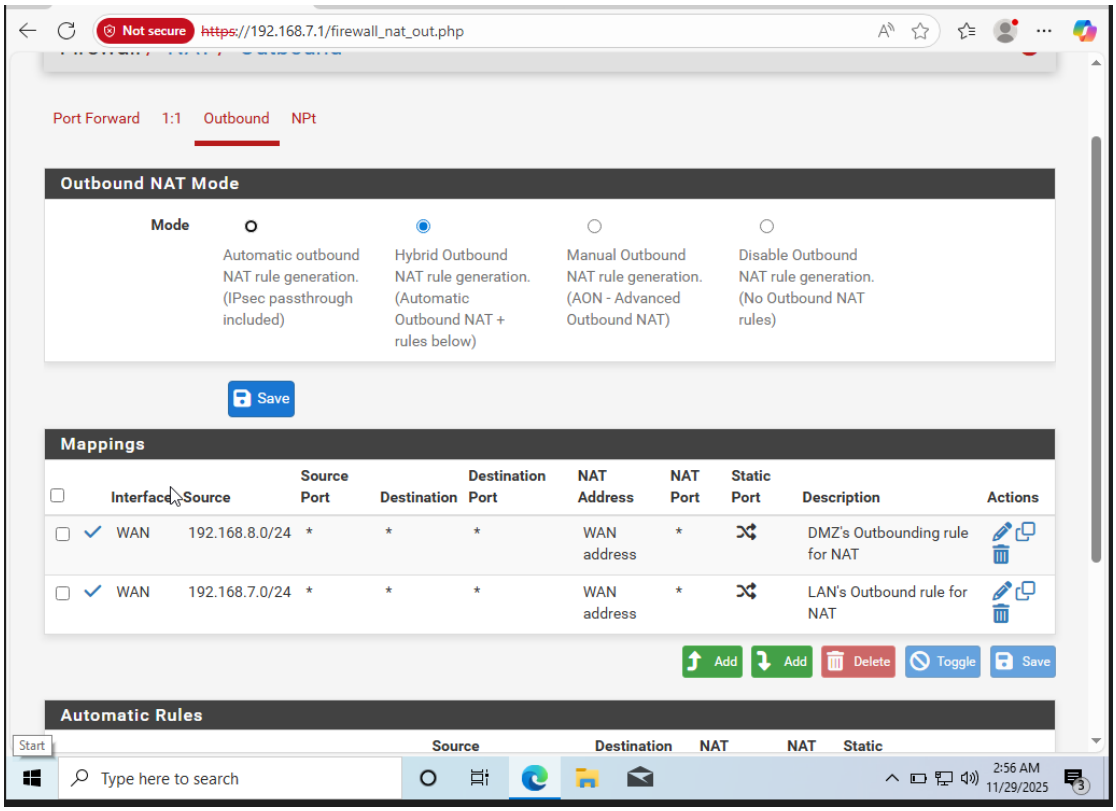
The image here is showing the pfSense WAN interface firewall rules. We can see the default "block private networks" and "block bogon networks" rules which prevent the WAN interface from receiving any unwanted traffic from RFC1918 and unassigned address ranges. Moreover, there are custom OpenVPN access rules that permit inbound UDP traffic on port 1194. These are the only allowed incoming traffic rules on the firewall which provide VPN access while the security measures on the WAN side are still in place.



The image depicts the firewall regulations for the local area network, featuring the automatically created anti-lockout regulation that permits HTTPS access to the pfSense GUI from the LAN. The other rules permit all IPv4 and IPv6 traffic from the LAN subnets to go to any destination without restrictions, which is normal for a trusted internal network. With this setup, devices on the LAN will be able to access the DMZ as well as the internet and at the same time, the admin will be able to access the firewall.

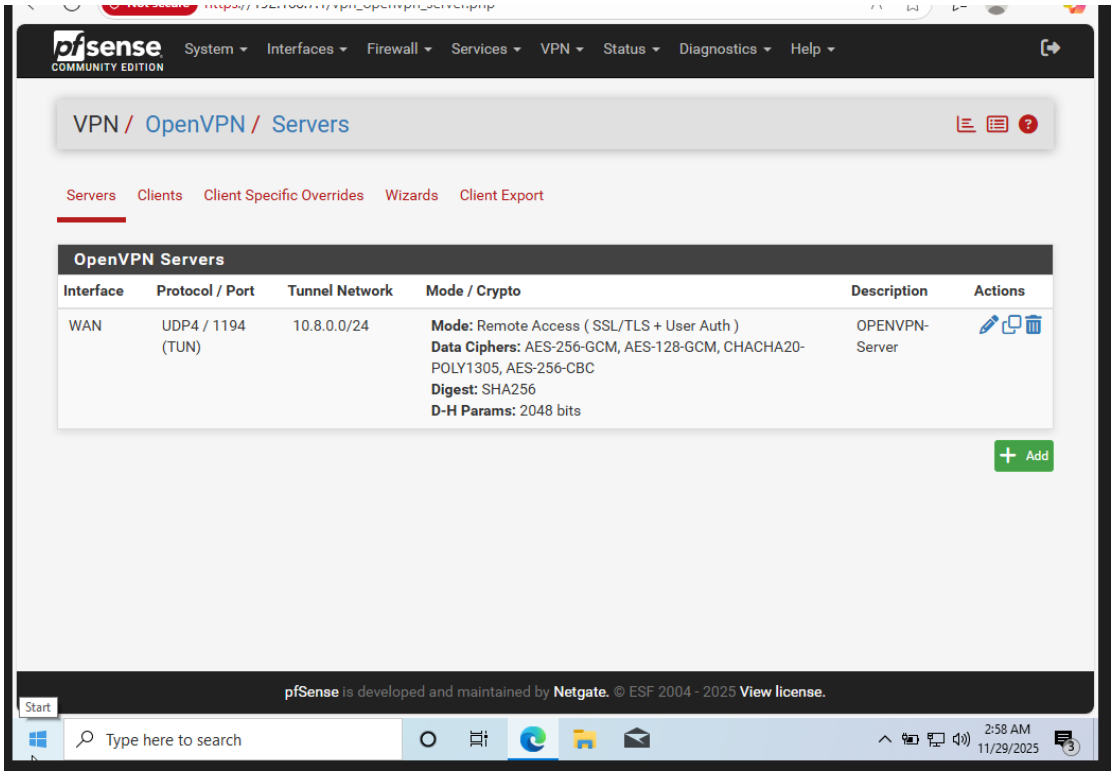


The screenshot displays the firewall rules set up for the OPT1 interface that acts as the DMZ zone. The regulations have permitted outbound Internet access, DNS queries to the firewall, and ICMP echo requests for health checks, but they have also explicitly prohibited any DMZ traffic that tries to reach the LAN network. Such an arrangement is part of a tough network segmentation policy that stops servers in the DMZ which are accessible to the public from being able to hack into the internal LAN, thereby sustaining a safe perimeter-style architecture.

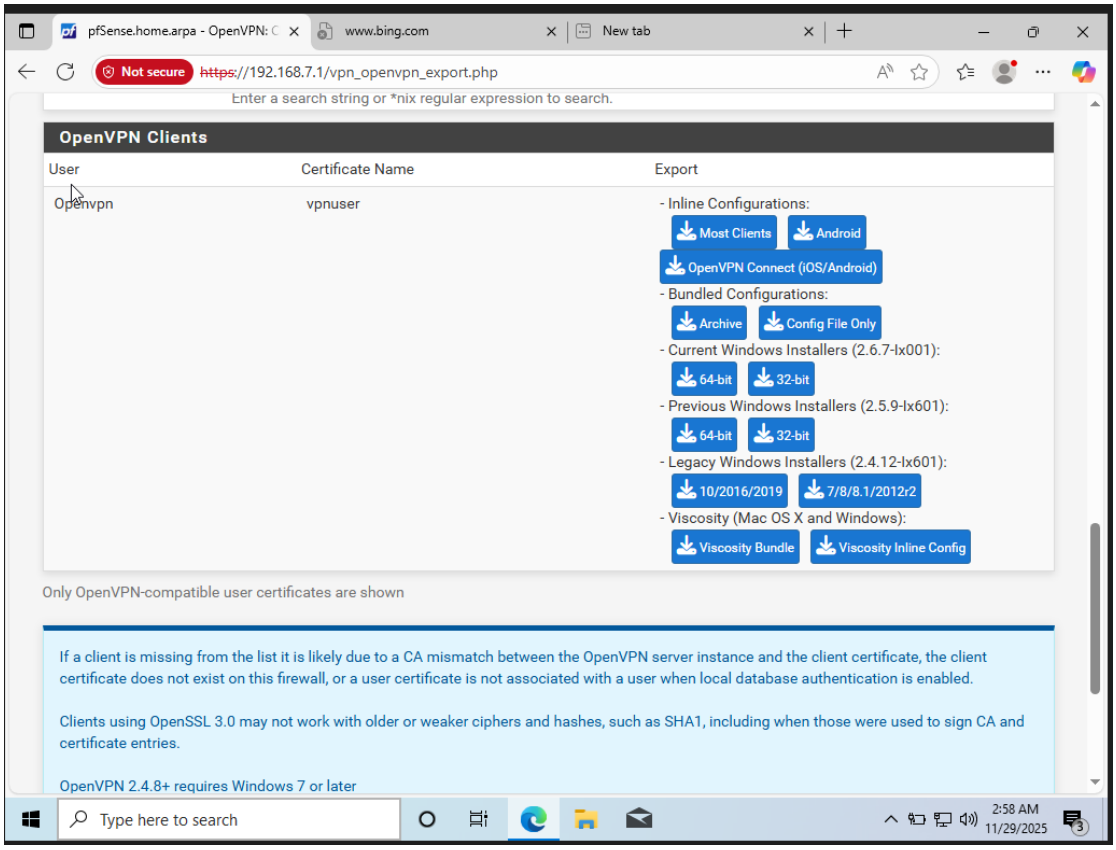


The screenshot demonstrates pfSense setup in Hybrid Outbound NAT mode, which enables automatic NAT rules and at the same time allows custom NAT entries for the LAN and DMZ networks. Different outbound rules for the LAN (192.168.7.0/24) and DMZ (192.168.8.0/24) networks are shown, where each one is changing the internal private addresses to the WAN address before the traffic is going out of the firewall. Consequently, the LAN as well as the DMZ networks are guaranteed to have Internet access, and at the same time, the traffic isolation is properly maintained.

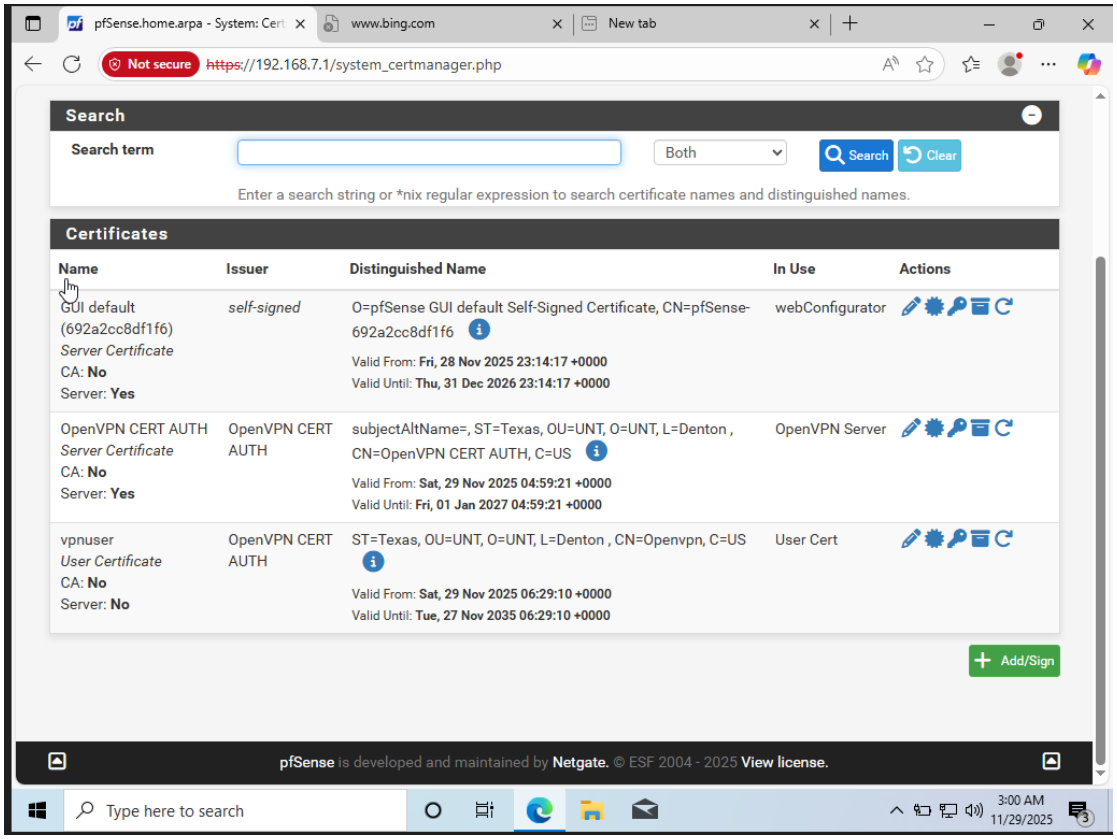
.



The OpenVPN server configuration is evident in this screenshot. A VPN server is depicted to operate on the WAN interface through UDP port 1194, while the tunnel network is 10.8.0.0/24. To ensure the security of remote access, the server makes use of SSL/TLS authentication and employs robust encryption ciphers like AES-256-GCM as well as SHA256 for integrity. Hence, this installation is a secure solution for authorized users coming from outside to obtain access into the network via a confidential connection.



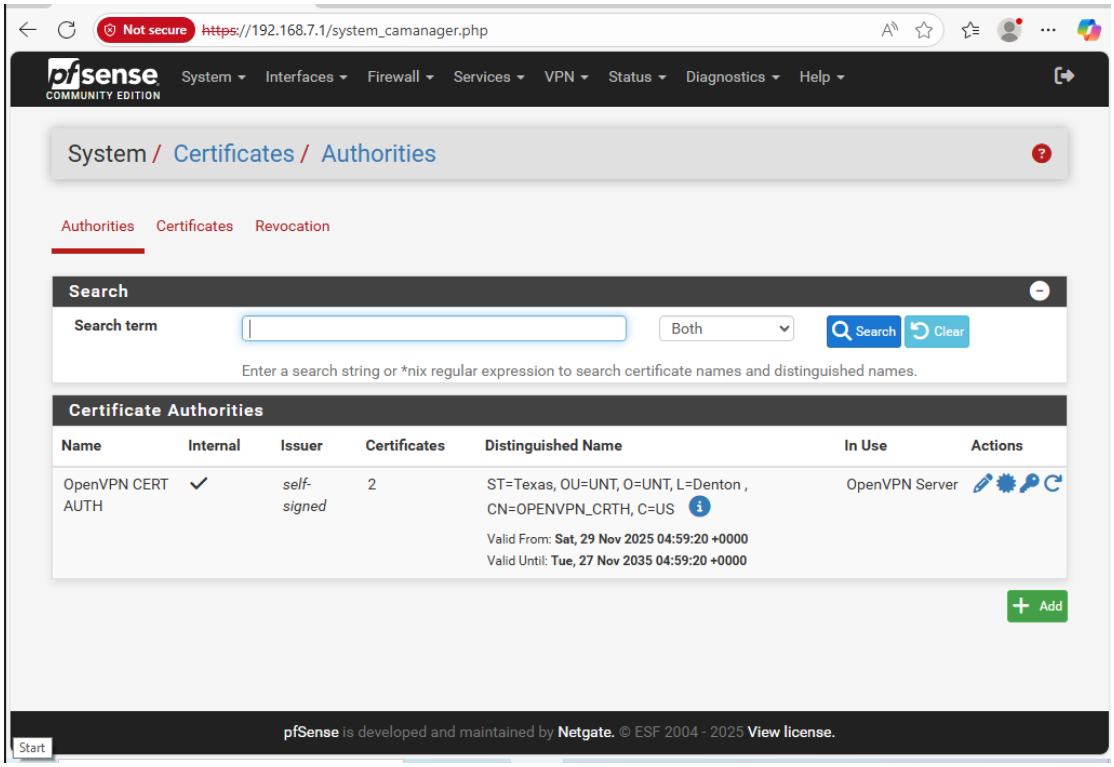
This webpage shows the OpenVPN Client Export Utility, which is able to export VPN client profiles that are already configured for different types of devices as well as operating systems. In this setup, a user account named *vpnuser* was created, and a certificate issued to the user was assigned as the means of VPN authentication. pfSense automatically produces quite a few packages that can be downloaded, among which there are inline configuration files, bundled installers for Windows (both 32-bit and 64-bit), and Viscosity profiles for macOS. The present facility makes VPN distribution less complicated as it is able to produce clients in file formats which can be directly used without the need for manual editing and thus, client-side security is ensured with the same encryption settings.



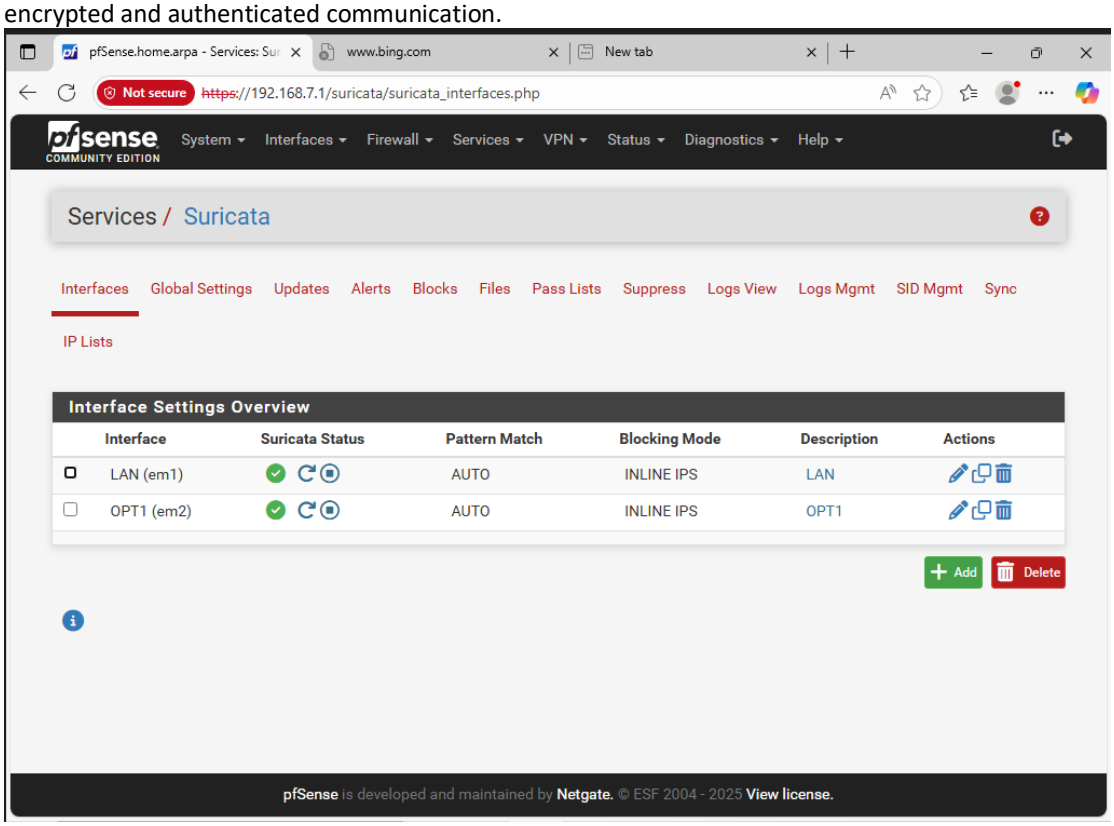
So, this page is kind of like a certificate album displaying all the certificates that have been created and installed on pfSense. The “GUI default” certificate can be considered as the system’s self-signed certificate which is used to secure the login interface via HTTPS.

We can also see the OpenVPN server certificate which is the one that allows the encryption of tunnel connections between VPN users and the firewall. As well as that, the user certificate *vpnuser* can be spotted which is the VPN client account that is used for authentication.

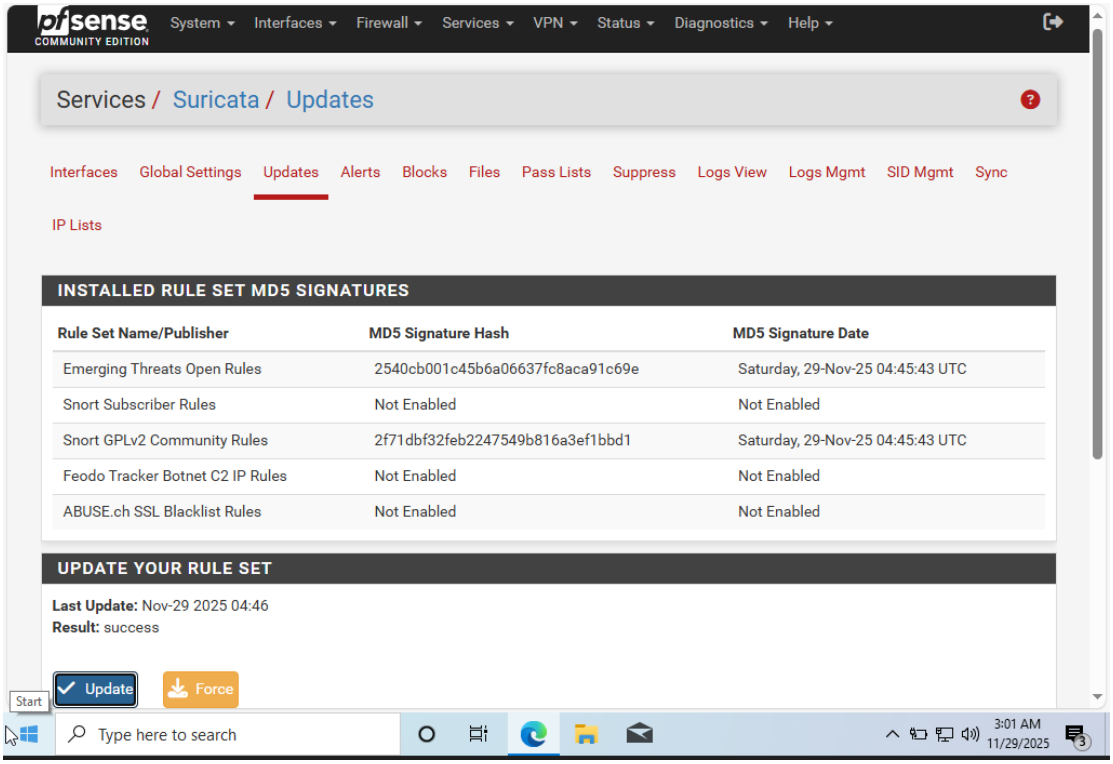
In summary, these certificates are the trust chain that is necessary for the encrypted OpenVPN sessions to be established while the web administration is accessed securely.



The image captured here powerfully illustrates the location of the Certificate Authorities (CA), where the root CA for OpenVPN is under the supervision. The CA instance *OpenVPN CERT AUTH* has been deliberately produced for the purpose of creating and signing certificates server and user which are used for VPN setup. As it is self-signed, pfSense is the certificate authority for itself, thereby making the arrangement in a lab environment very easy without having to depend on external PKI systems. This CA is the key to the chain of trust between the OpenVPN server and clients, thus guaranteeing encrypted and authenticated communication.



On the LAN and OPT1 (DMZ) interfaces Suricata is set up in *Inline IPS* mode as can be seen from this screen. Inline IPS enables Suricata to block the offending traffic on the fly apart from just detection. Automatic pattern matching is turned on for both interfaces where Suricata is also shown as alive and running. The firewall thereby makes sure by individually securing the LAN and DMZ segments that inbound as well as lateral movement attempts are checked and intercepted basing on the active IDS/IPS rulesets and thus the network security posture is enhanced.



This screenshot illustrates the Suricata rules update page, a place where the latest threat detection signatures are pulled and synchronized. The device went through a successful update of its rulesets from several malware and intrusion signature sources like Emerging Threats, Snort GPLv2, and others. Continuously updating signature databases is a must if the IDS/IPS engine is to be capable of detecting the newest threats, malware patterns, exploitation attempts, and anomalous traffic behaviors. The present update is indicative of correct upkeep of the security monitoring layer in the network.

11/29/2025 08:28:45	3	TCP	Generic Protocol Command Decode	192.168.7.101	57182	135.224.68.108	443	1:2210042	SURICATA STREAM TIMEWAIT ACK with wrong seq
11/29/2025 08:17:14	3	TCP	Generic Protocol Command Decode	192.168.7.101	63401	20.50.80.210	443	1:2210038	SURICATA STREAM FIN out of window
11/29/2025 08:10:38	3	TCP	Generic Protocol Command Decode	192.168.7.101	63394	150.171.28.11	80	1:2210016	SURICATA STREAM CLOSEWAIT FIN out of window
11/29/2025 07:58:38	3	TCP	Generic Protocol Command Decode	23.198.7.179	443	192.168.7.101	63379	1:2210038	SURICATA STREAM FIN out of window
11/29/2025 06:32:33	3	TCP	Generic Protocol Command Decode	23.40.70.176	80	192.168.7.101	63256	1:2210042	SURICATA STREAM TIMEWAIT ACK with wrong seq
11/29/2025 06:32:32	3	TCP	Generic Protocol Command Decode	23.40.70.176	80	192.168.7.101	63256	1:2210042	SURICATA STREAM TIMEWAIT ACK with wrong seq
11/29/2025 06:32:31	3	TCP	Generic Protocol Command Decode	192.168.7.101	63256	23.40.70.176	80	1:2210042	SURICATA STREAM TIMEWAIT ACK with wrong seq

This is a list of the currently active alert messages that were generated by Suricata as it kept track of the traffic. The alerts point to TCP anomalies like STREAM TIMEWAIT ACK and STREAM FIN errors, which most of the time, signal that there is something not quite right with the session or that the traffic is trying to evade the detection. The alerts indicate that Suricata is actually looking at real network packets from both the LAN and the outside world and is finding that there may be some suspicious behavior going on. Together with the timestamps, source/destination IPs, and ports, as well as the particular rule identifiers, each entry in the log serves as a demonstration of an efficient intrusion detection system operating in real-time.

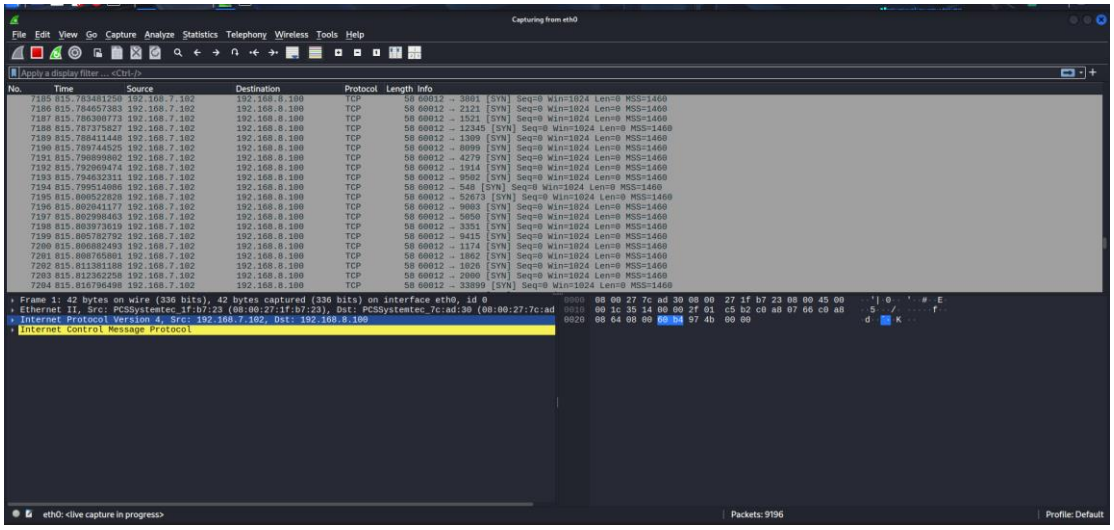
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -sS -sV -O 192.168.8.100

Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-29 03:52 EST
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Stats: 0:01:46 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 50.54% done; ETC: 03:55 (0:01:42 remaining)
Stats: 0:04:12 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 66.26% done; ETC: 03:58 (0:02:08 remaining)
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
RTTVAR has grown to over 2.3 seconds, decreasing to 2.0
Nmap scan report for 192.168.8.100
Host is up (0.048s latency).
Not shown: 999 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
80/tcp open  http    nginx 1.24.0 (Ubuntu)
Device type: general purpose
Running: Linux 4.X
OS CPE: cpe:/o:linux:linux_kernel:4
OS details: Linux 4.19 - 5.15
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/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 619.17 seconds

┌──(kali㉿kali)-[~]
└─$ 
```

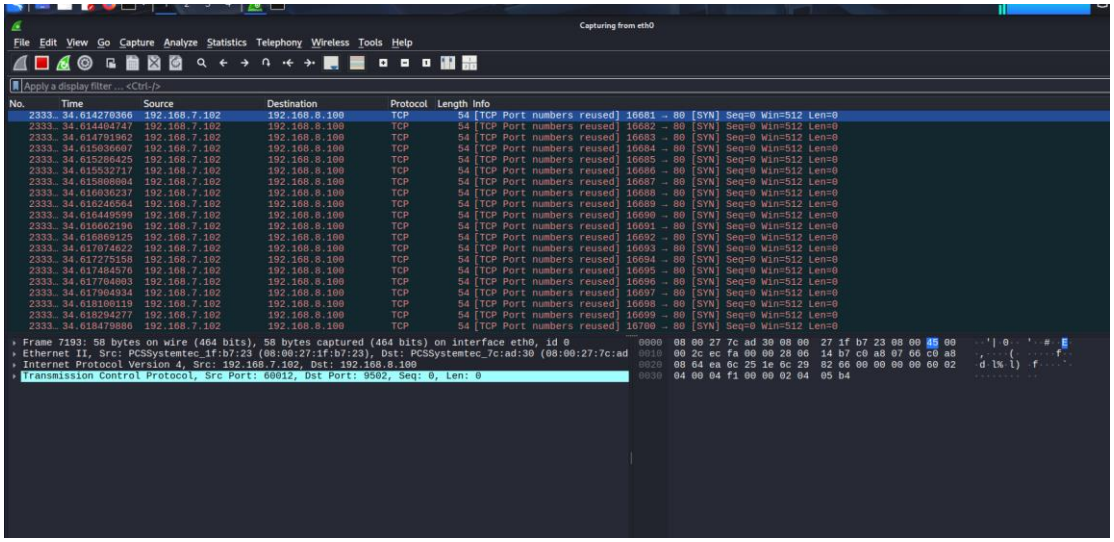
This screenshot depicts an in-depth Nmap SYN Stealth scan of the Ubuntu server in the DMZ (192.168.8.100). The scanning activity reveals that port 80 is open and that an nginx web server is running there. The operating system fingerprinting results point to Linux kernel 4.x-5.x as the closest match. The extended scan time suggests that IDS/IPS are detecting and throttling the scan attempts, which would be normal in a security-hardened environment. Hence, this scan serves as a confirmation of both the DMZ service being reachable and the firewall's IPS working.



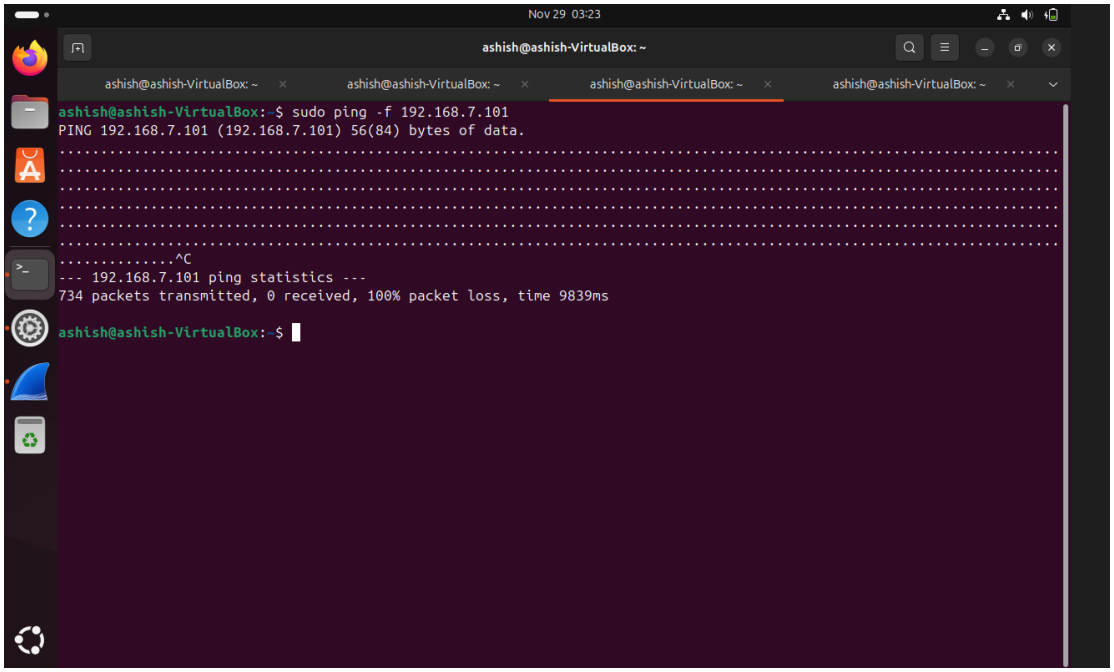
During the hping3 DoS attack simulation, a large number of SYN packets were found in this Wireshark capture. The repetitive SYN fragments verify the flood operation thereby visualizing the harmful traffic of the packet-level which is what the targeted web server in the DMZ is receiving.

.

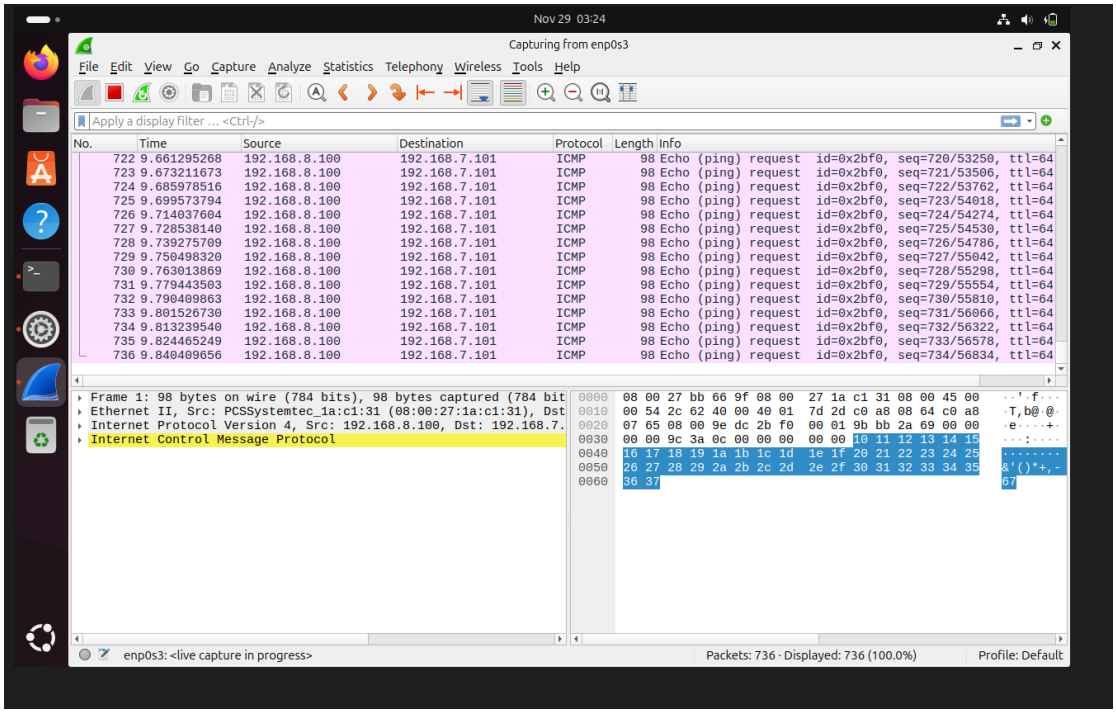
The screenshot illustrates the hping3 command that was used to create the SYN flood(hping3 -S --flood -p 80). The command issues a high-rate of spoofed SYN packets to the target server. This is a DoS test which is done to see if pfSense and Suricata can recognize irregular traffic patterns as a security test.



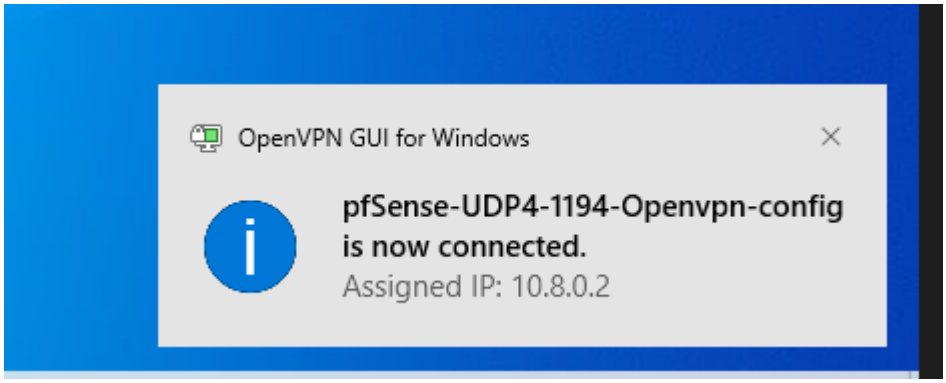
During the denial of service attack, the Wireshark screenshot is filled with repeated messages of “TCP Port numbers reused” and abnormal ACK sequences, which suggest that the attacked system is crashing. The presence of such a disturbance in the network traffic acts as evidence of the project’s IDS features in action.



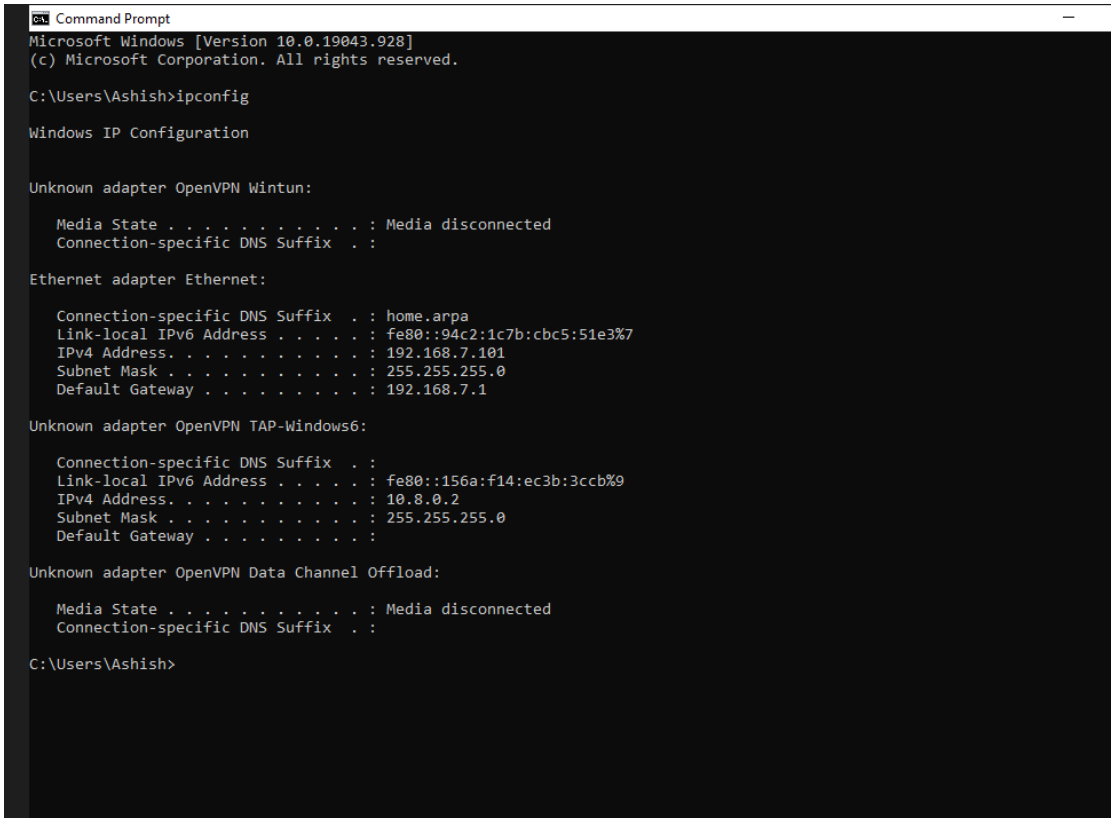
This screen capture displays the flood-ping (ping -f) that was utilized to create ICMP exhaustion traffic aimed at the LAN gateway. The packet loss and congestion that have been detected confirm the device's response to ICMP-based DoS attacks and serve as evidence of a deliberately controlled stress test in the local network.



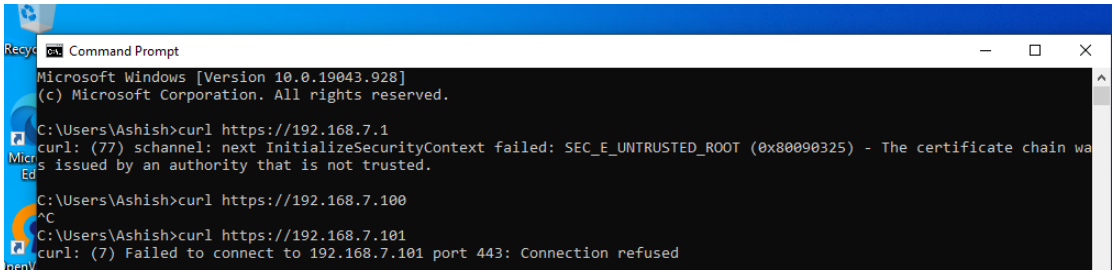
The repeated ICMP Echo Requests being sent at a very fast rate are clearly visible from this perspective. The recording serves as a confirmation that pfSense and Suricata are in compliance with their functions as they not only log but also respond to a high-volume ICMP flood situation. Thus, the DoS analysis part of the project has been completed.



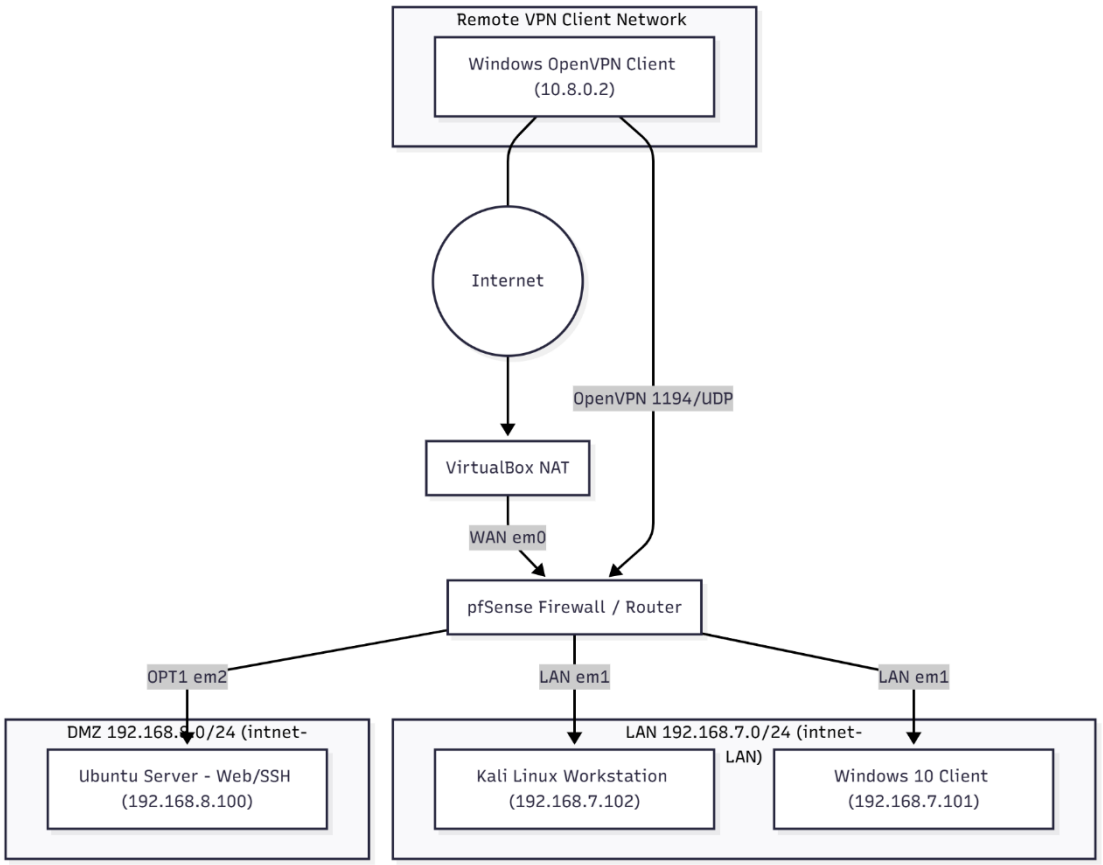
This alert indicates that the Windows OpenVPN client has made a connection by utilizing the configuration generated by pfSense. The tunnel IP given (10.8.0.2) is the definite proof of VPN functionality from source to destination, hence, the requirement of secure access for the remote user has been accomplished.



The command output of ipconfig shows the LAN adapter as well as the OpenVPN TAP interface. The tunnel network (10.8.0.0/24) is clearly allocated, thus illustrating the successful VPN routing and the creation of an interface on the Windows client.



The screenshot illustrates HTTPS curl tries to LAN and DMZ web servers. The untrusted certificate notification is due to the fact that pfSense is utilizing a self-signed CA which is not installed on the Windows machine. The connection refusal errors are the reasons why pfSense firewall rules are tightly guarding against unauthorized access, thus, security enforcement of the configured architecture is confirmed.



The network topology for this project is the result of a design to simulate an enterprise security environment implemented pfSense as the central firewall and the router. The whole system divides into three functional zones: the LAN (192.168.7.0/24), DMZ (192.168.8.0/24) and an externally connected Remote VPN Client Network (10.8.0.0/24). pfSense is the focus of the layout, and it gets its upstream connection via VirtualBox NAT on its WAN (em0) interface. At the same time, the LAN (em1) and OPT1 (em2) interfaces are used to extend internal connectivity. Two internal user machines are hosted in the LAN, i.e., Kali Linux (192.168.7.102) and Windows 10 (192.168.7.101) that are the user simulators for legitimate traffic generation as well as penetration testing activities. The DMZ hosts a hardened Ubuntu Server (192.168.8.100) that models an exposed perimeter service where web and SSH services are available for the attack, the monitoring, and the traffic analysis. Remote clients could be connected to pfSense through a dedicated OpenVPN tunnel that utilizes UDP port 1194, thereby permitting a Windows VPN client to join the network securely with a given address of 10.8.0.2. This is a replica of the remote-access scenarios that happen in the real world where external users or administrators can connect to the internal network. Besides this, the topology exhibits the usage of the next level of security by leveraging the Suricata IDS/IPS feature in the inline mode that is operational on both LAN and OPT1 interfaces, therefore, allowing the identification and blocking of the enemy traffic that is hallmarked by internal, DMZ, or VPN-connected sources. The organized layout of this design aids the implementation and accomplishing of the NAT performance, firewall rules, intrusion detection, VPN access, and simulated attack scenarios objectives that the project entails.