CS 646 – Network Protocols Security Project 2

Kurudunje Deekshith Shetty – ks2378

Carlos Jimenez Joaquin – cmj26

04/12/2024

Objective: - Design and Implementation of defensive technologies to prevent attacks against systems, services and protocols.

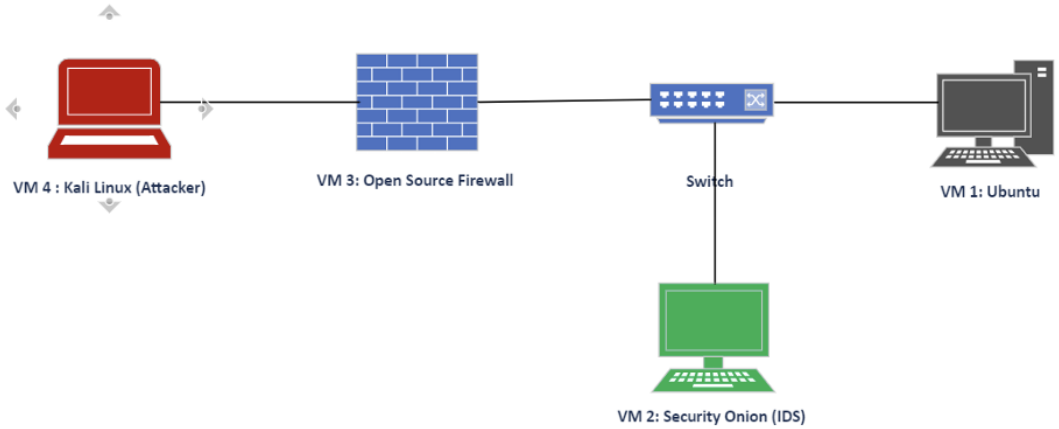
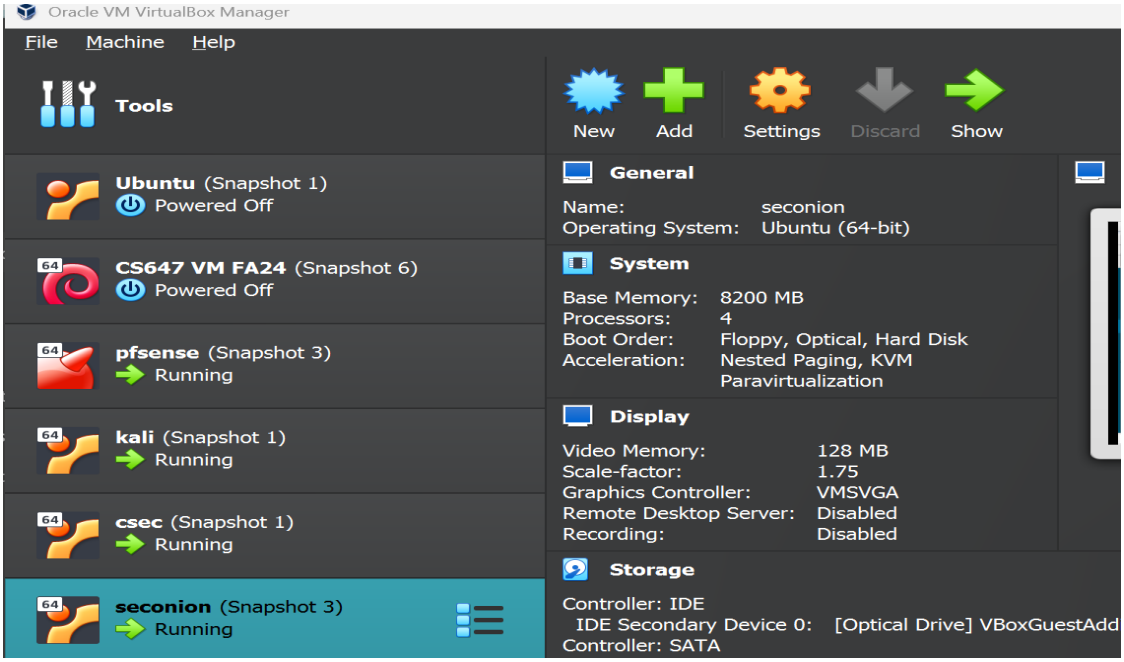


Figure 1: Network Design

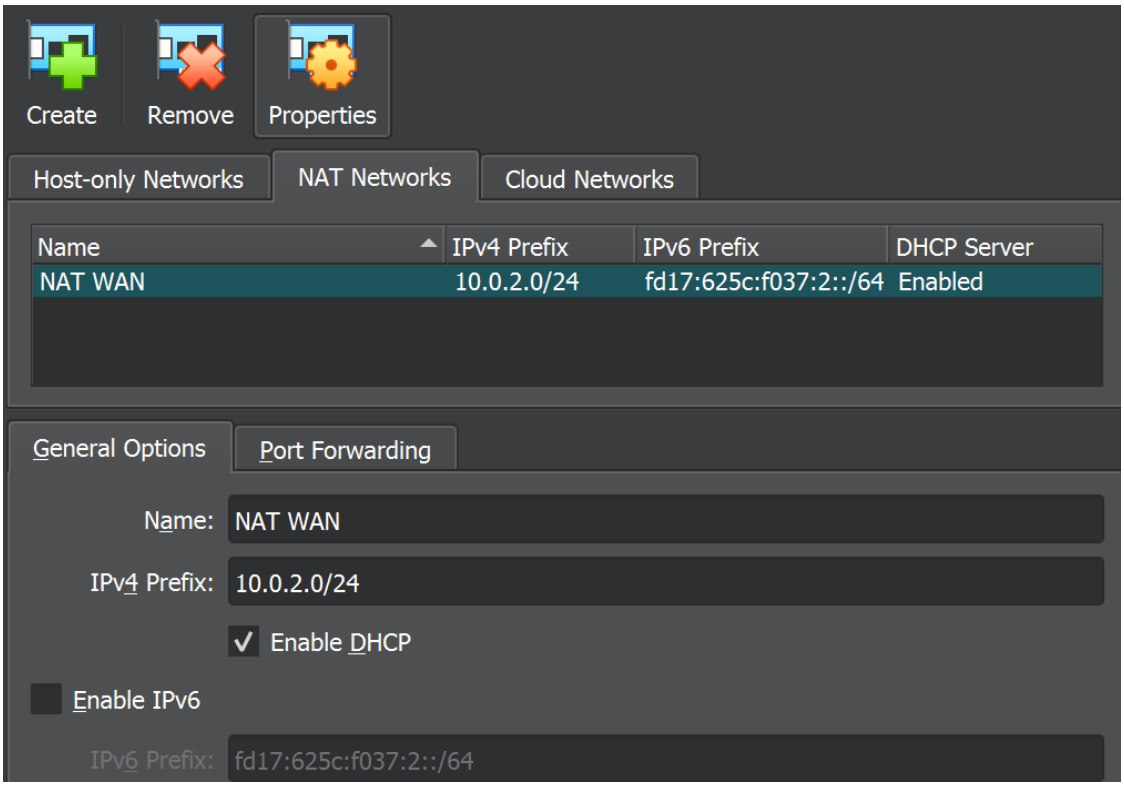
To design this network, we are required to create 4 VMs. Below is the list of VM's that we implemented on Virtual Box for this project:

Virtual Machine Name	OS/Type	IP Address	Network
Pfsense	Firewall	WAN – 10.0.2.15 LAN – 192.168.213.100	WAN – NAT WAN LAN – Host Only
Security Onion	Ubuntu (NIDS)	192.168.213.105	Host Only
Kali (Attacker)	Kali Linux	10.0.2.20	NAT WAN
CSEC (Host)	Ubuntu	192.168.213.109	Host Only

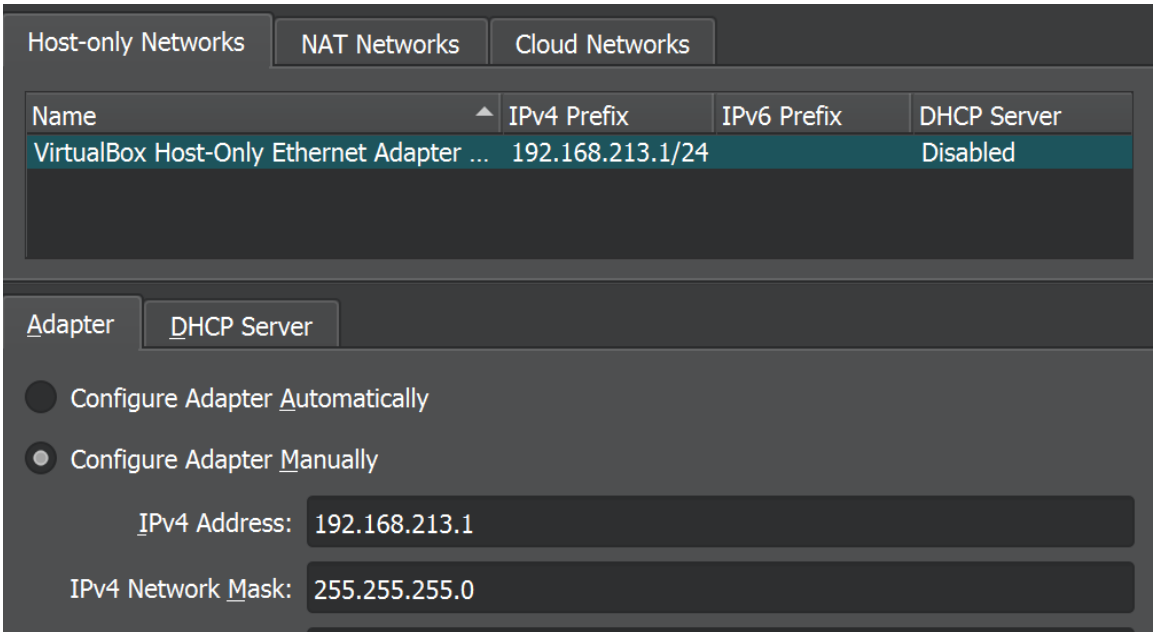
Below are the images of our VirtualBox dashboard and the Networks we created for reference:



F1: VirtualBox Dashboard

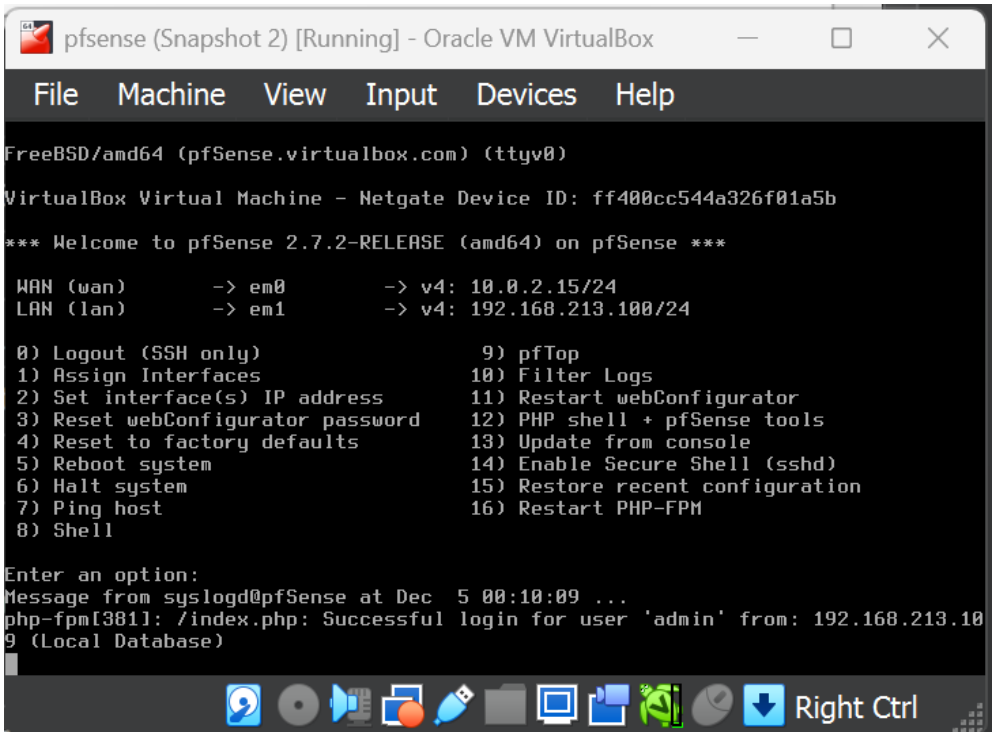


F2: NAT WAN network for WAN side

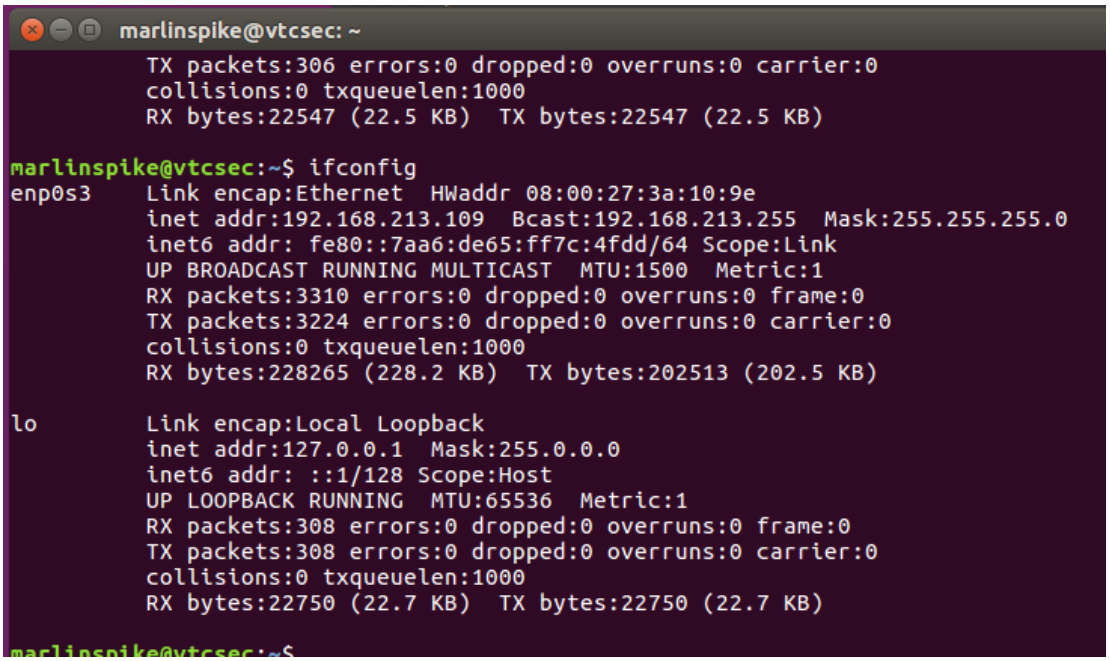


F3: Host Only Network for LAN side

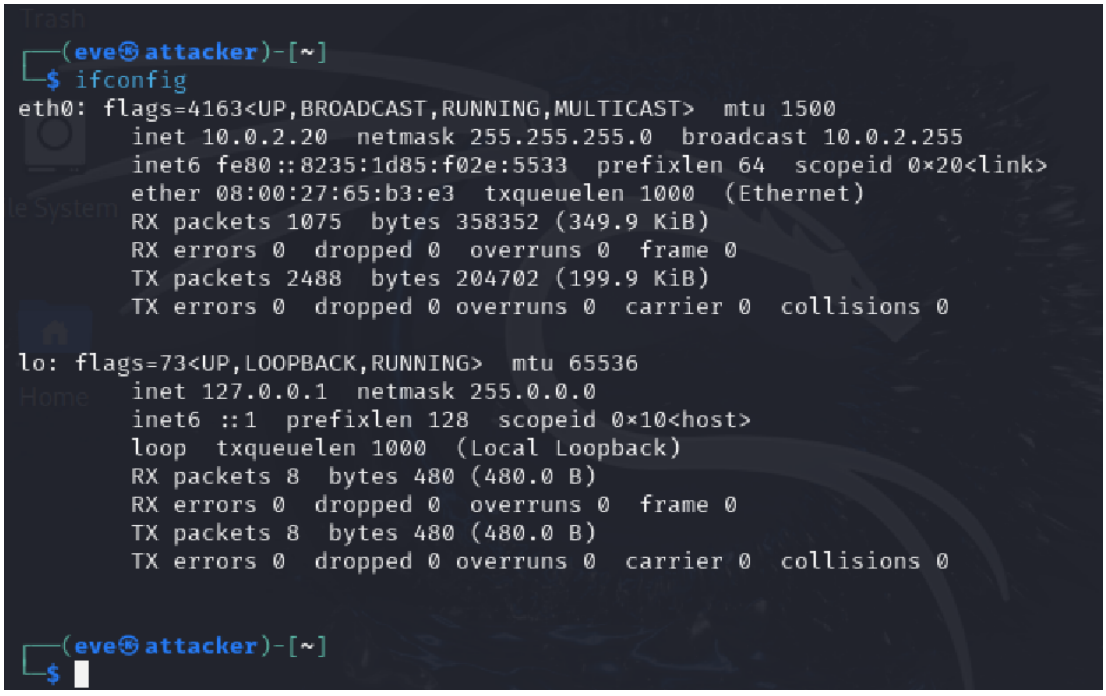
For further information, we've attached the screenshots referring to our VM deployments and their network configuration:



F4: pfSense Firewall



F5: Host Machine



F6: Attacker Machine

Next, we are going to demonstrate the attacks as per our objectives.

Attack 1: NMAP service scan

```
┌──(eve㉿attacker)-[~]
└─$ nmap -sV 192.168.213.109
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 19:02 EST
Nmap scan report for 192.168.213.109
Host is up (0.0017s latency).
Not shown: 997 filtered tcp ports (no-response)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3c
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.2 (Ubuntu Linux; protocol
2.0)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.96 seconds
```

F7: Network mapping scan

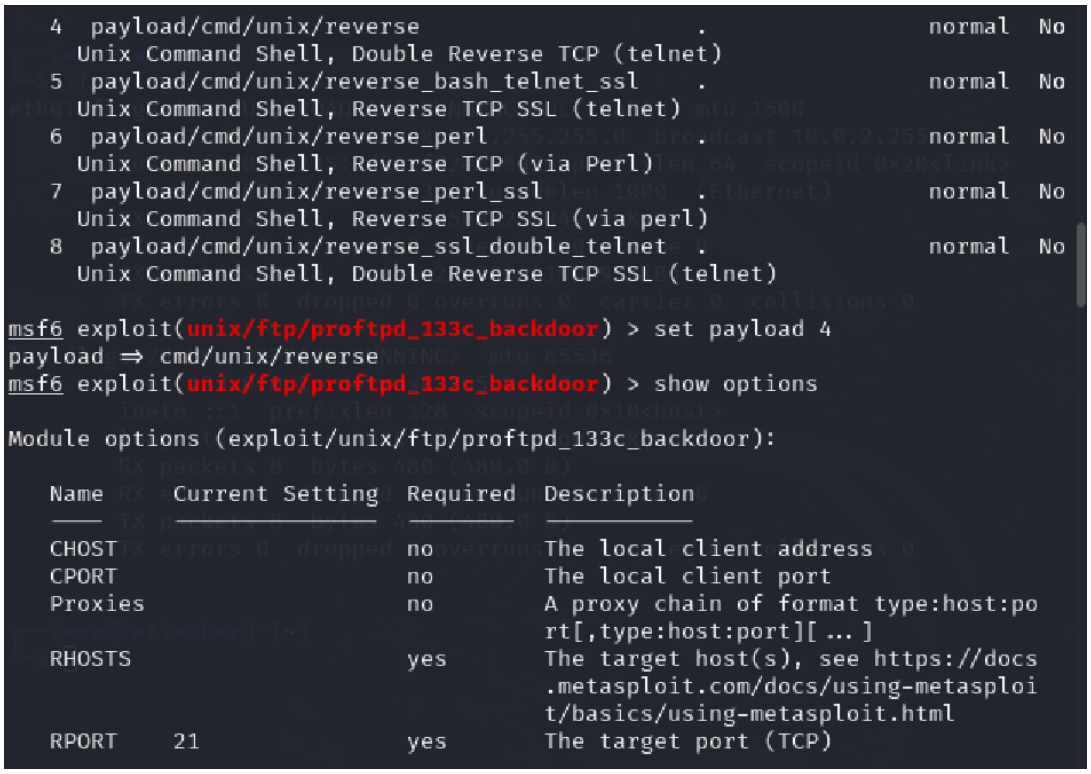
By using this attack, we've done our preliminary reconnaissance. We've deduced that the host machine has ports 21, 22 and 80 open. We are going to focus on the FTP and the HTTP port for this project. The host has a vulnerable version of FTP that we can exploit which is what we are going to do in our next attack.

ATTACK 2: Exploitation of ProFTPD 1.3.3.c using Metasploit:

```
eve@attacker: ~
File  Actions  Edit  View  Help
       =[ metasploit v6.4.18-dev                          ]
+ -- --=[ 2437 exploits - 1255 auxiliary - 429 post       ]
+ -- --=[ 1468 payloads - 47 encoders - 11 nops           ]
+ -- --=[ 9 evasion                                       ]

Metasploit Documentation: https://docs.metasploit.com/

msf6 > use exploit/unix/ftp/proftpd_133c_backdoor
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show payloads

Compatible Payloads
===================

   #  Name                                Disclosure Date  Rank    Ch
eck  Description
   -  ----                                ---------------  ----    --
---  -----------
   0  payload/cmd/unix/adduser                            .        normal  No
     Add user with useradd
   1  payload/cmd/unix/bind_perl                          .        normal  No
     Unix Command Shell, Bind TCP (via Perl)
   2  payload/cmd/unix/bind_perl_ipv6                     .        normal  No
     Unix Command Shell, Bind TCP (via perl) IPv6
   3  payload/cmd/unix/generic                            .        normal  No
     Unix Command, Generic Command Execution
   4  payload/cmd/unix/reverse                            .        normal  No
     Unix Command Shell, Double Reverse TCP (telnet)
```

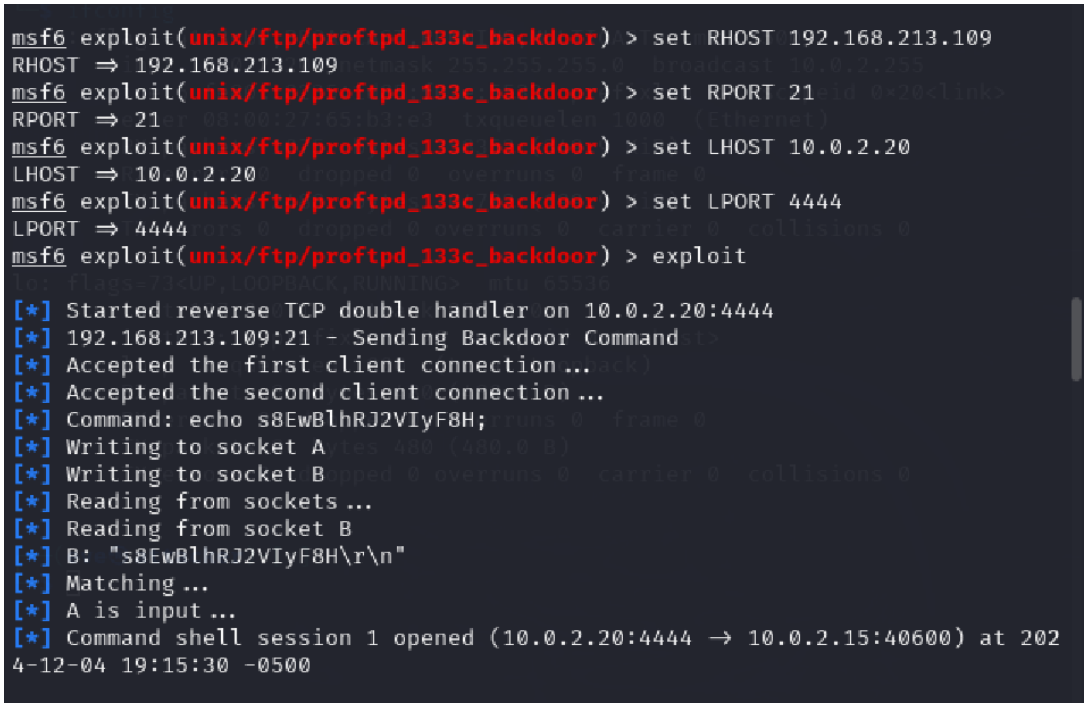
F8: Using exploit for proftpd 1.3.3c

ProFTPD 1.3.3c has a critical vulnerability which we could exploit to create a connection via a backdoor.



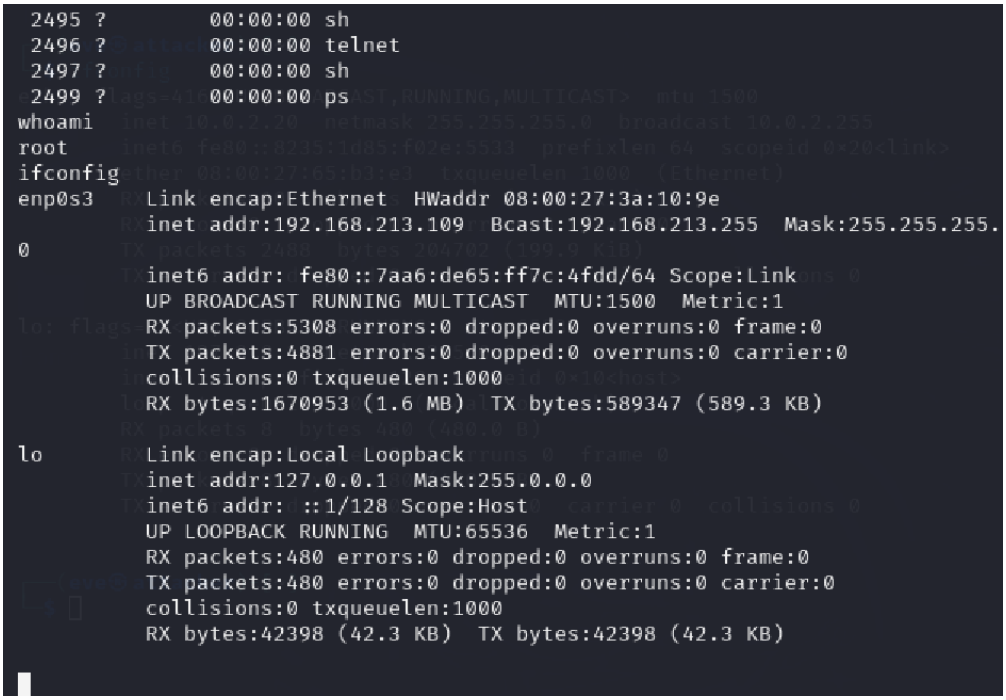
F9: Selecting the payload

We loaded the double reverse TCP payload. There are a few variables we had to set. Below is the screenshot for that:



F10: Setting the variables

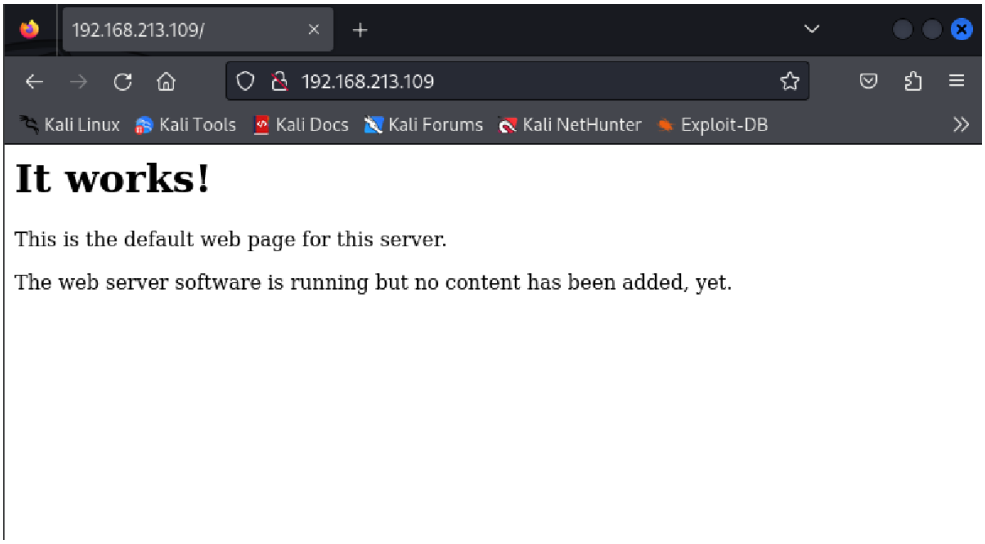
We set the variables for the victim IP(RHOST), victim port (RPORT), attacker IP(LHOST), attacker port (LPORT) and ran the exploit to create a command shell session on the victim's machine with root level privileges. Below is the proof of the success of the attack:



F11: Access shell to victims' machine

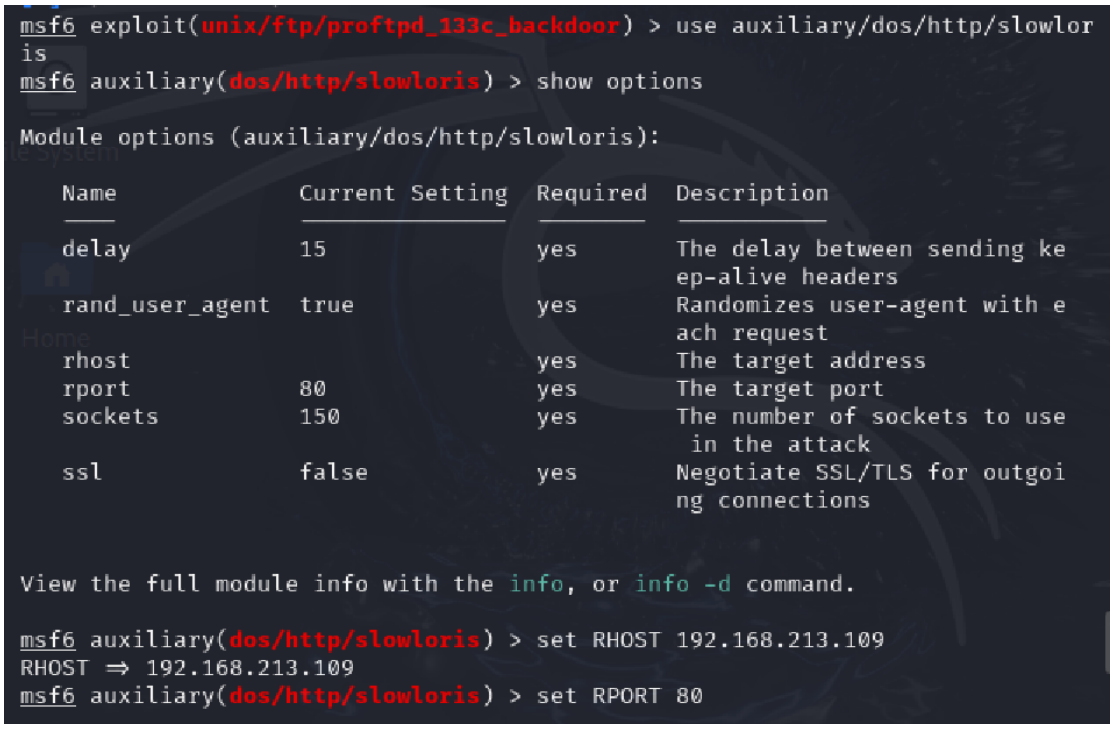
ATTACK 3: DOS attack using SlowLoris:

Moving on to the final attack. We are going to exploit the next open port, the HTTP port. The host machine has a HTTP page that its hosting, lets try to open that first:



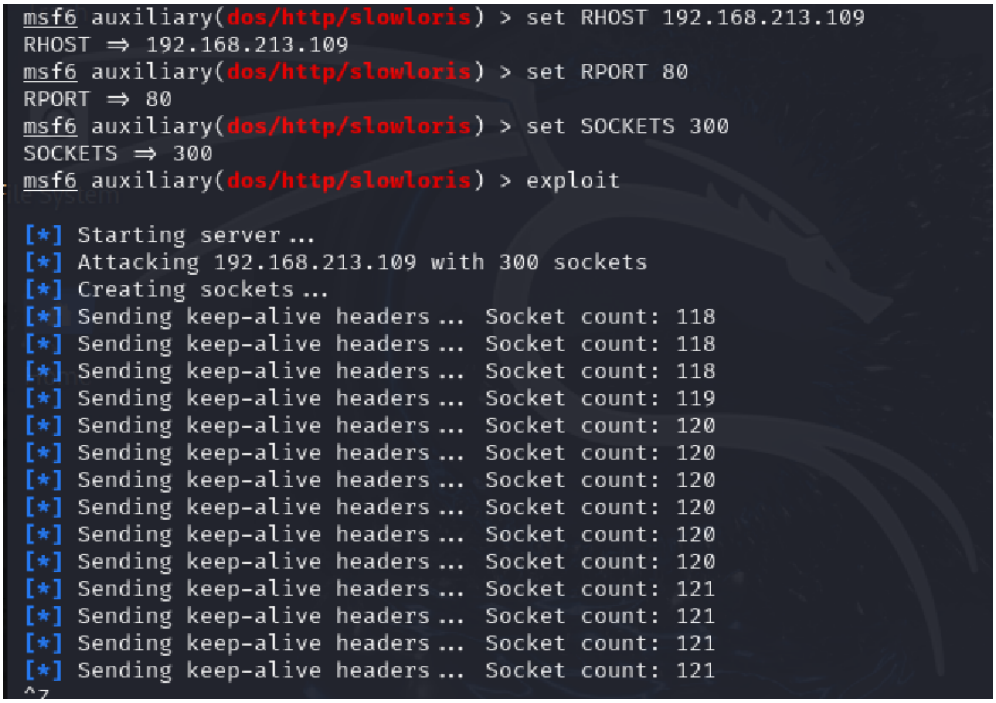
F12: Working HTTP page

Using the Metasploit, we loaded the slowloris auxiliary for our attack. We had to configure a few variables prior to the DOS attack



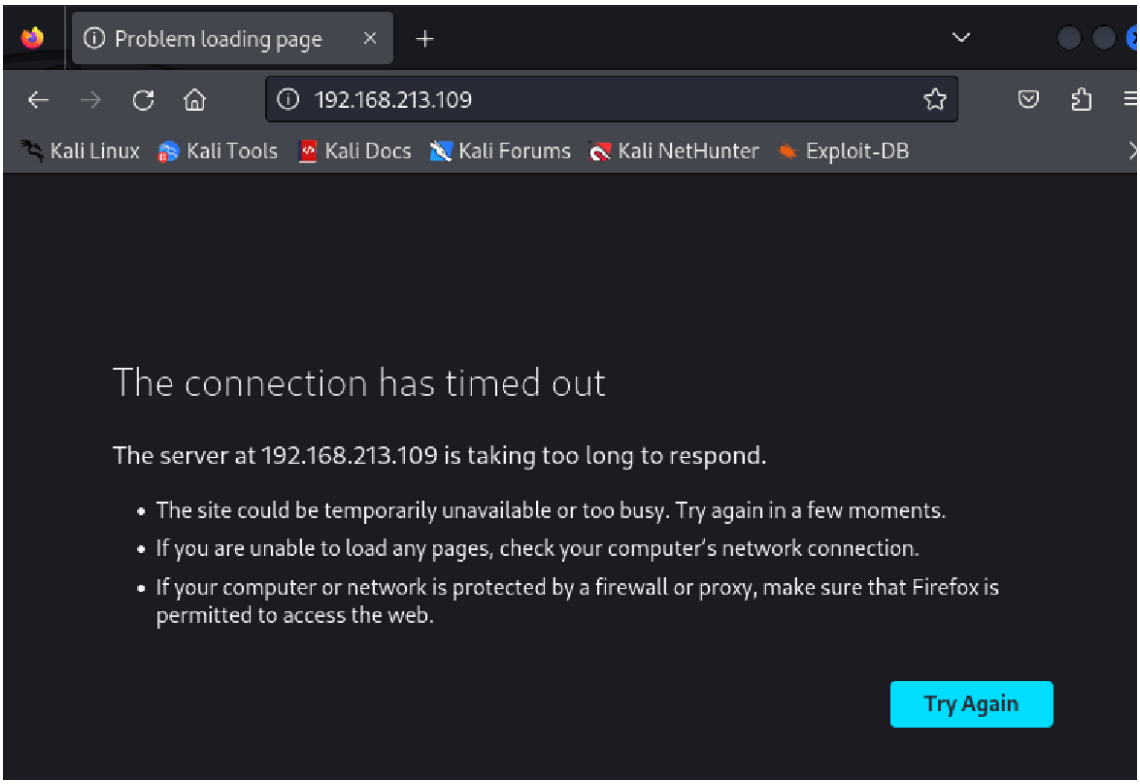
F13: SlowLoris Module

After setting the variables for the victim IP, host and the number of sockets, we launched the attack.



F14: DOS attack

As a result of this, the HTTP page crashed and could not service us. Below are the screenshots of proof of the DOS attack.

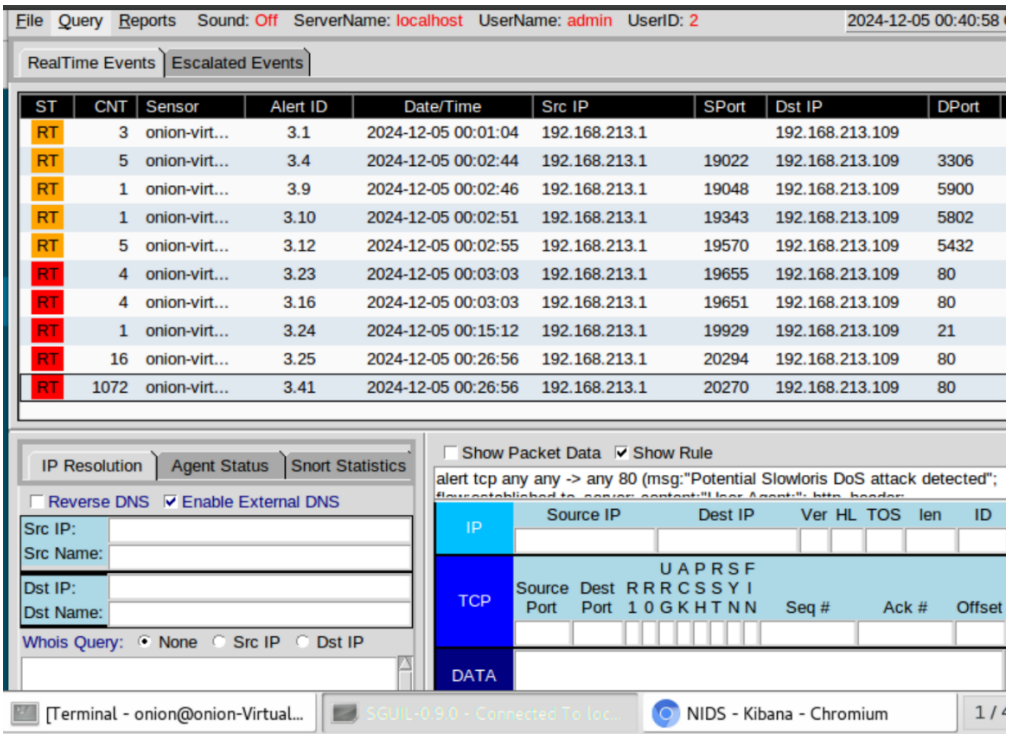


F15: Website not working after the attack

```
marlinspike@vtcsec:~$ sudo tcpdump -i enp0s3 port 80 -nn
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), capture size 262144 bytes
19:31:40.075010 IP 192.168.213.109.48320 > 192.168.213.100.80: Flags [P.], seq 2
389392714:2389393278, ack 2302908960, win 1447, options [nop,nop,TS val 33342629
72 ecr 279400031], length 564: HTTP: POST /getstats.php HTTP/1.1
19:31:40.076083 IP 192.168.213.100.80 > 192.168.213.109.48320: Flags [.], ack 56
4, win 510, options [nop,nop,TS val 279401010 ecr 3334262972], length 0
19:31:40.195247 IP 192.168.213.100.80 > 192.168.213.109.48320: Flags [P.], seq 1
:505, ack 564, win 514, options [nop,nop,TS val 279401122 ecr 3334262972], lengt
h 504: HTTP: HTTP/1.1 200 OK
19:31:40.195270 IP 192.168.213.109.48320 > 192.168.213.100.80: Flags [.], ack 50
5, win 1447, options [nop,nop,TS val 3334263002 ecr 279401122], length 0
19:31:40.642618 IP 192.168.213.109.41688 > 199.60.103.226.80: Flags [S], seq 175
5671891, win 29200, options [mss 1460,sackOK,TS val 676824094 ecr 0,nop,wscale 7
], length 0
19:31:42.080476 IP 192.168.213.109.48320 > 192.168.213.100.80: Flags [P.], seq 5
64:1149, ack 505, win 1447, options [nop,nop,TS val 3334263473 ecr 279401122], l
ength 585: HTTP: POST /widgets/widgets/disks.widget.php HTTP/1.1
19:31:42.081343 IP 192.168.213.100.80 > 192.168.213.109.48320: Flags [.], ack 11
```

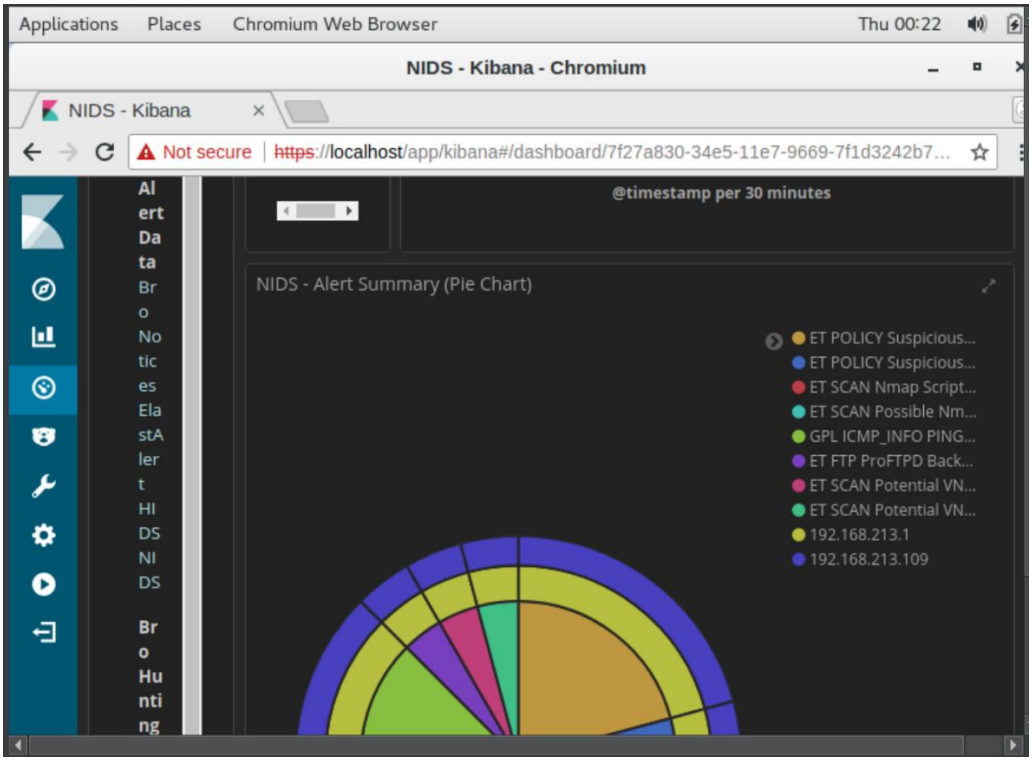
F16: Tcpdump on victim's machine port 80 showing the HTTP requests

Now that we are done with the attacks part, below we have attached the alerts that were generated because of the attacks on our NIDS (Security Onion) as seen on SGUILL:



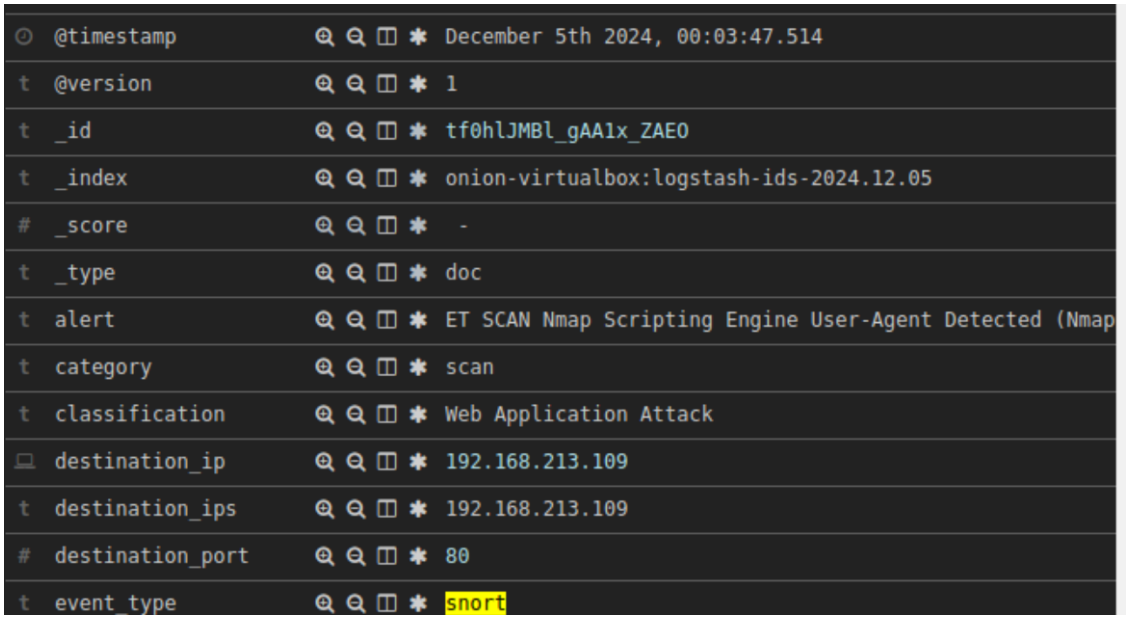
F17: SGUIL Alerts

These Alerts can be seen on Kibana, which is much more visually pleasing and clear.



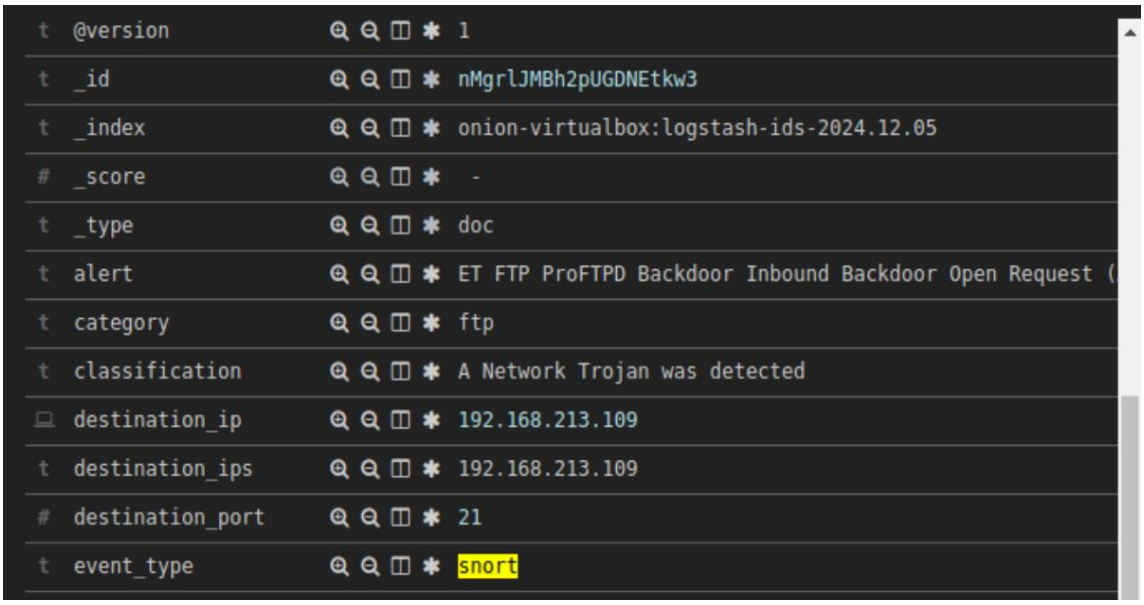
F18: Security Onion Dashboard using Kibana – NIDS dashboard

Below we've attached the screenshots of alerts on every attack on Kibana:



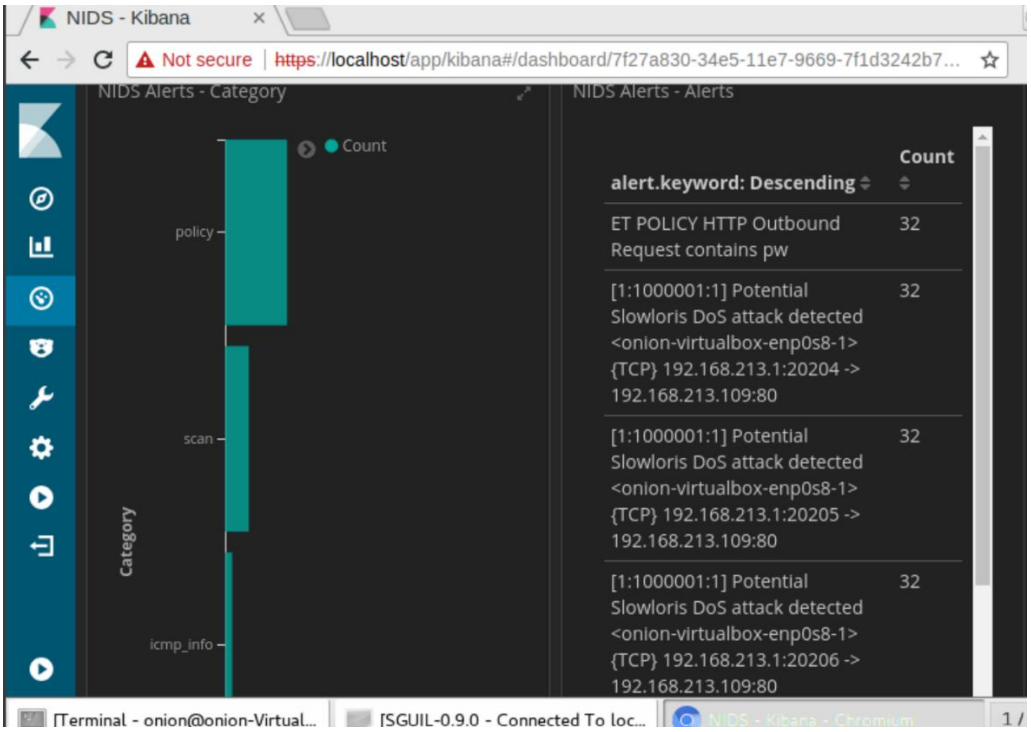
@timestamp		December 5th 2024, 00:03:47.514
@version		1
_id		tf0hlJMBl_gAA1x_ZAEO
_index		onion-virtualbox:logstash-ids-2024.12.05
_score		-
_type		doc
alert		ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap
category		scan
classification		Web Application Attack
destination_ip		192.168.213.109
destination_ips		192.168.213.109
destination_port		80
event_type		snort

F19: NMAP Scan Alert



@version		1
_id		nMgrlJMBh2pUGDNEtkw3
_index		onion-virtualbox:logstash-ids-2024.12.05
_score		-
_type		doc
alert		ET FTP ProFTPD Backdoor Inbound Backdoor Open Request (
category		ftp
classification		A Network Trojan was detected
destination_ip		192.168.213.109
destination_ips		192.168.213.109
destination_port		21
event_type		snort

F20: ProFTPD Backdoor Alert

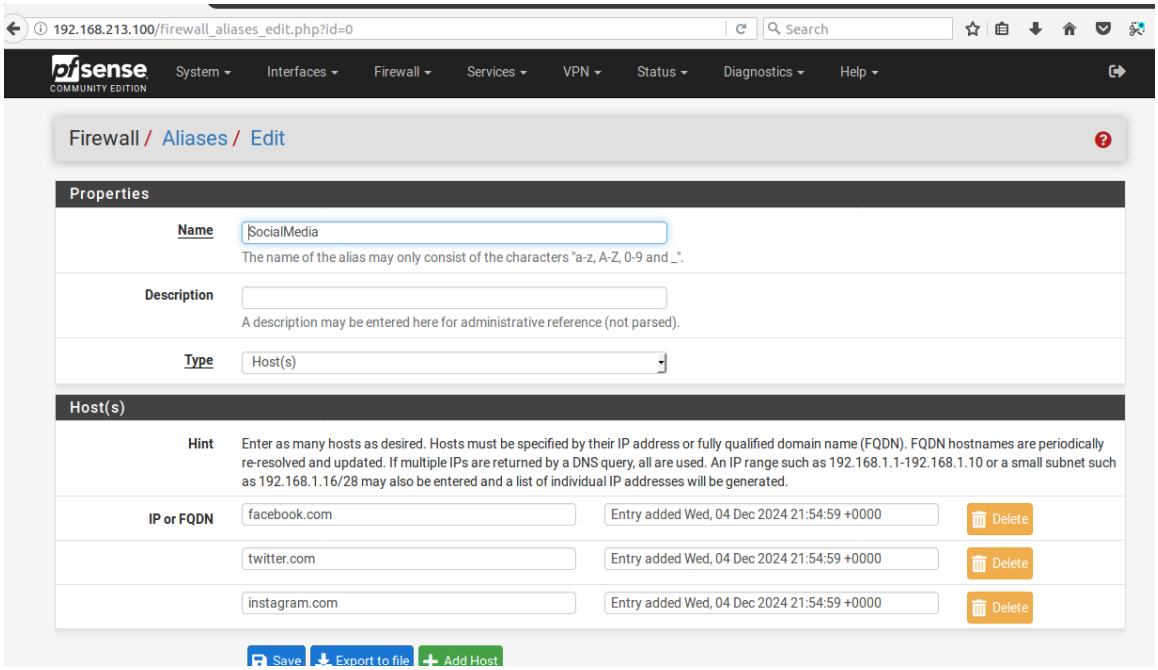


F21: SlowLoris DOS Alert

Now let's try to configure the Firewall rules as per the requirements:

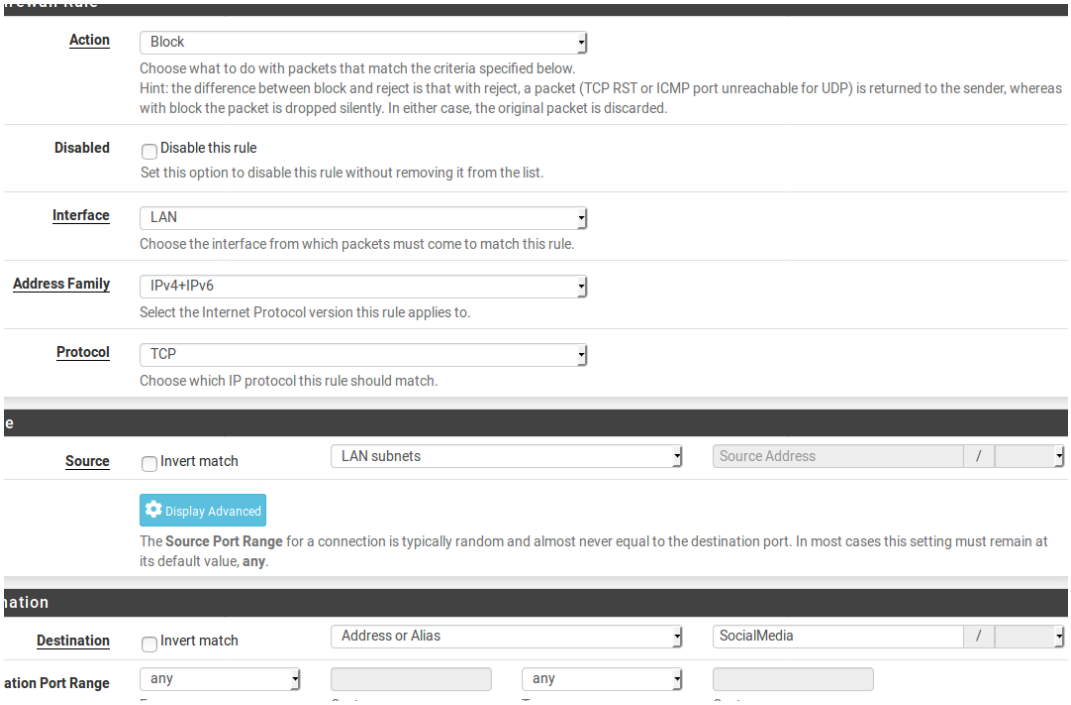
Rule 1: Internal Users should not be able to visit any Social Media sites like Facebook, Instagram and Twitter:

First, we created a list of the sites we wanted to block.



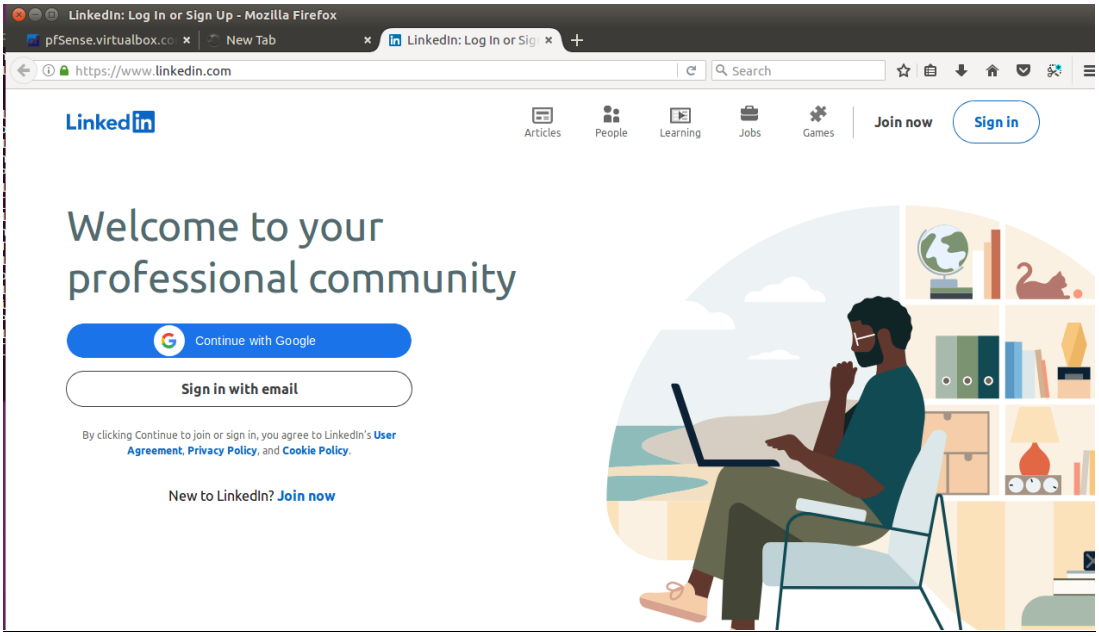
F22: Aliases for the sites we need to block

Using this we configured the block rule for social media



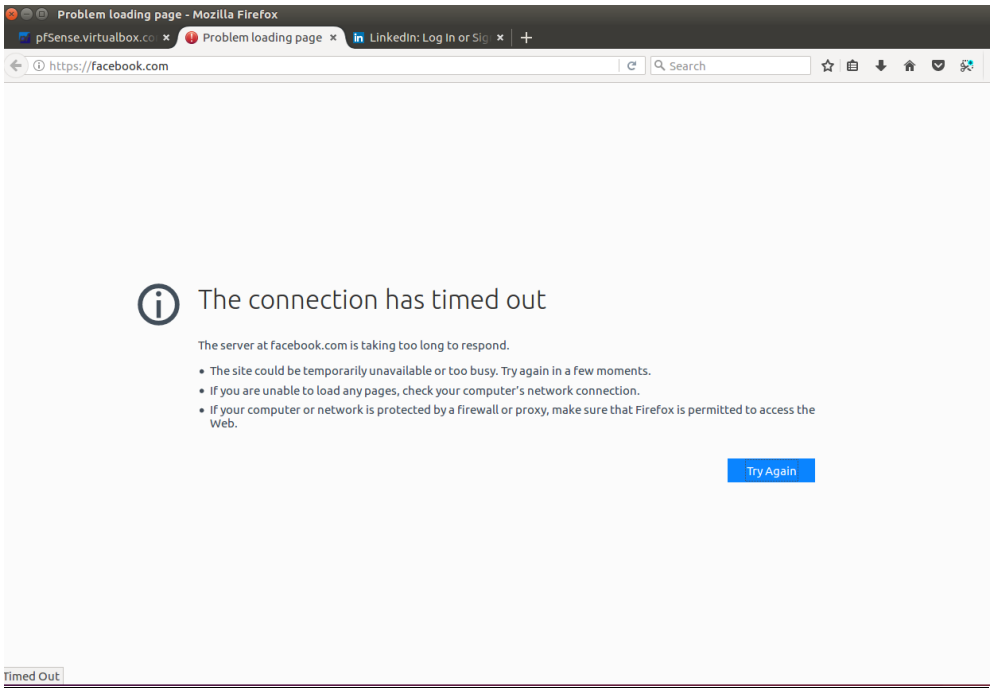
F23: Social media block rule

To see if this rule works, we did not add another networking site – LinkedIn which was not in the list of required sites to block to the list. Let's try to access LinkedIn:



F24: Sites not part of the rule

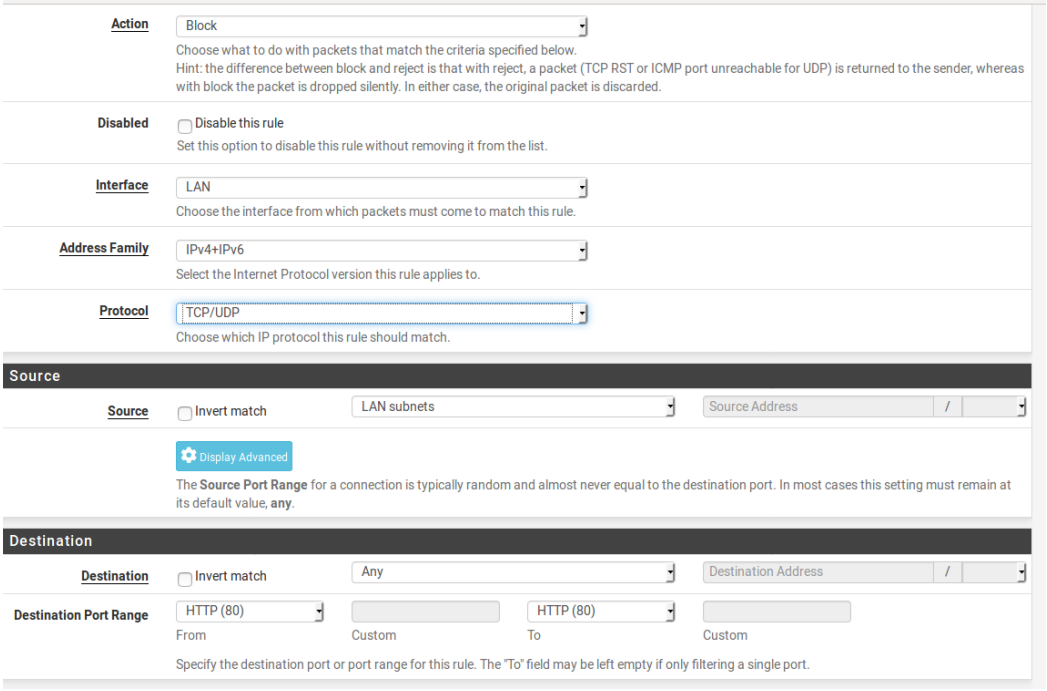
Now we tried to access Facebook, which we were asked to block. We could prove that our rule was working



F25: Facebook not working

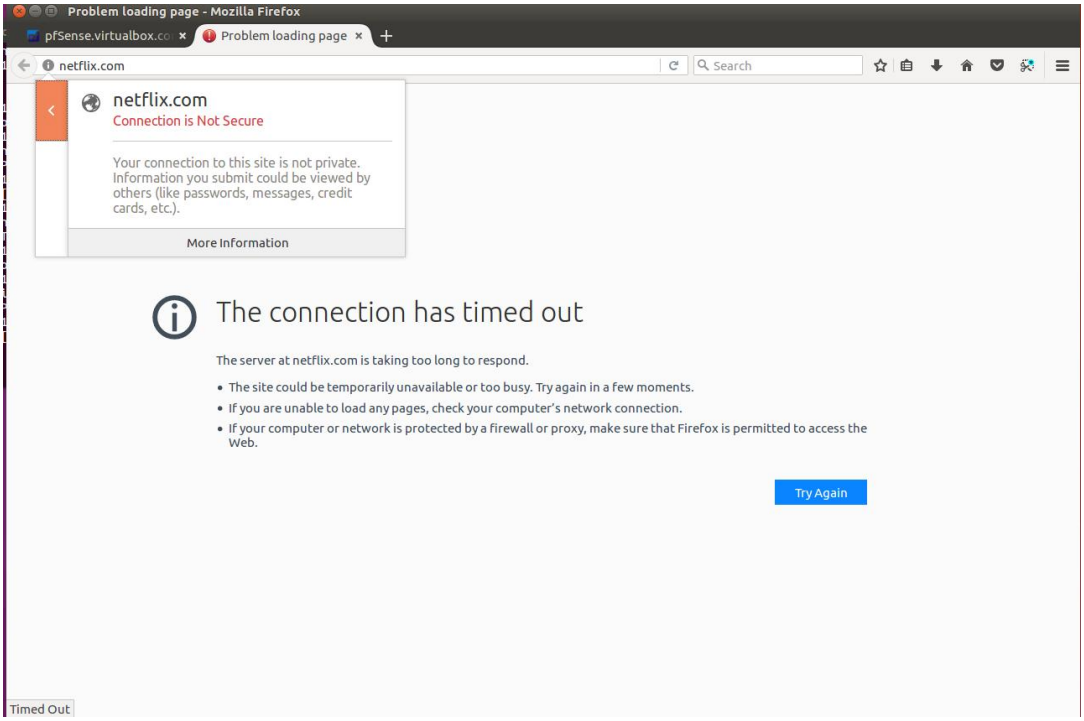
Rule 2: Internal Users should not be able to visit any sites with HTTP:

We created a rule to block internal networks from using HTTP



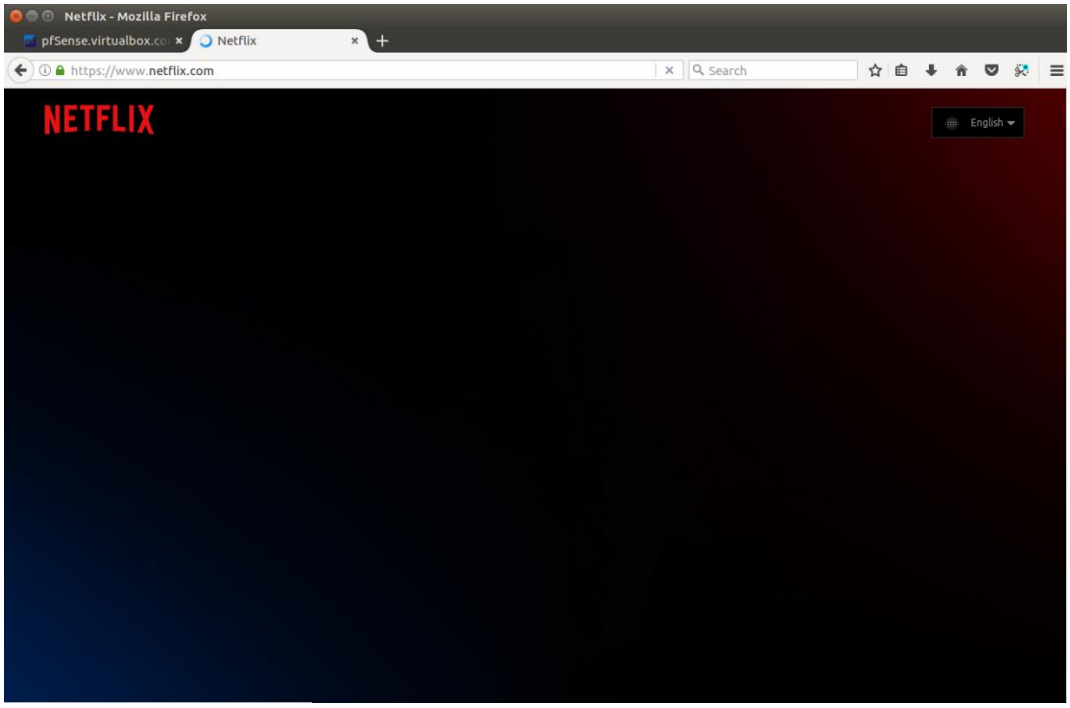
F26: HTTP block rule – Internal users

Now we tried to access the HTTP website. It was blocked proving that our rule worked:



F27: HTTP blocked

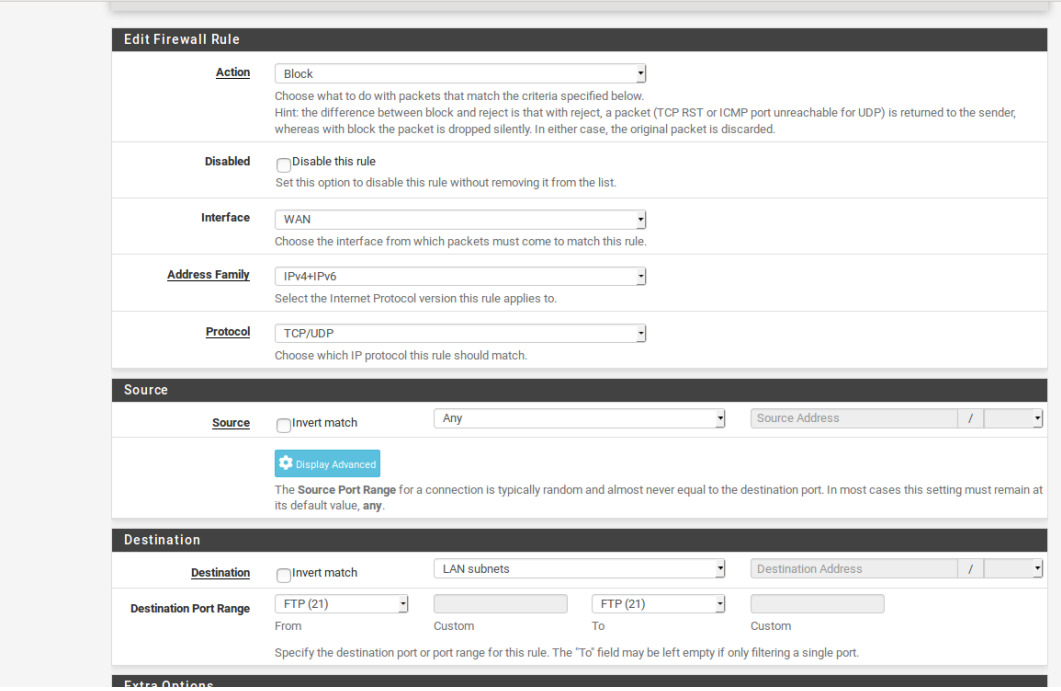
Just to be sure it's only limited to HTTP; we tried changing this to HTTPS and it started working.



F28: HTTPS working

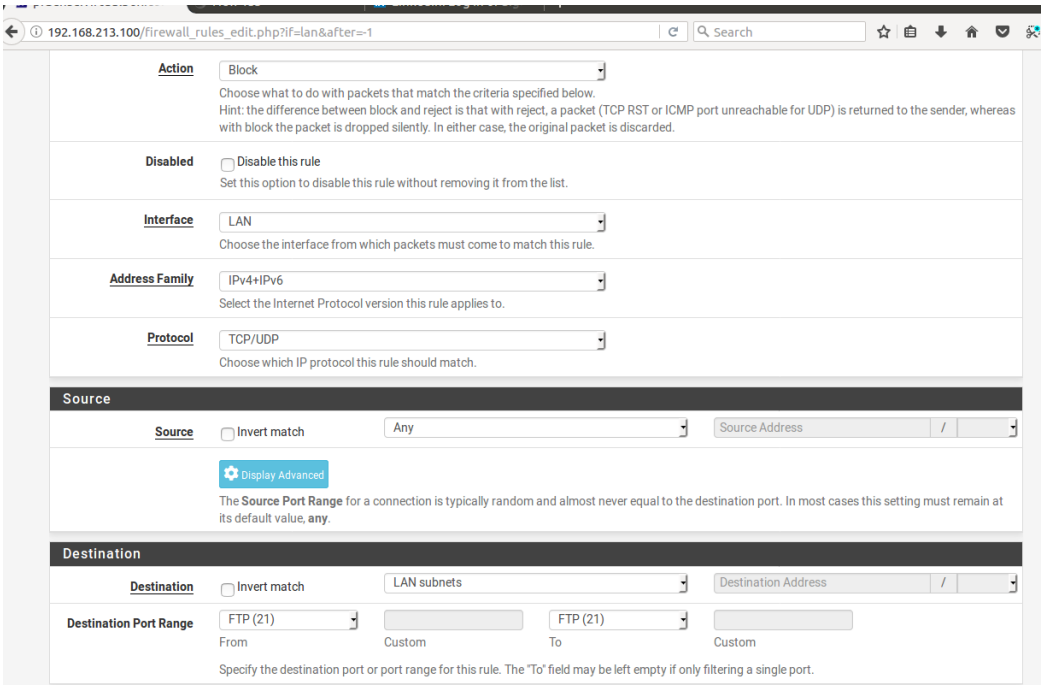
Rule 3: Block Incoming FTP:

Now we created a rule to block FTP incoming connections on the WAN interface



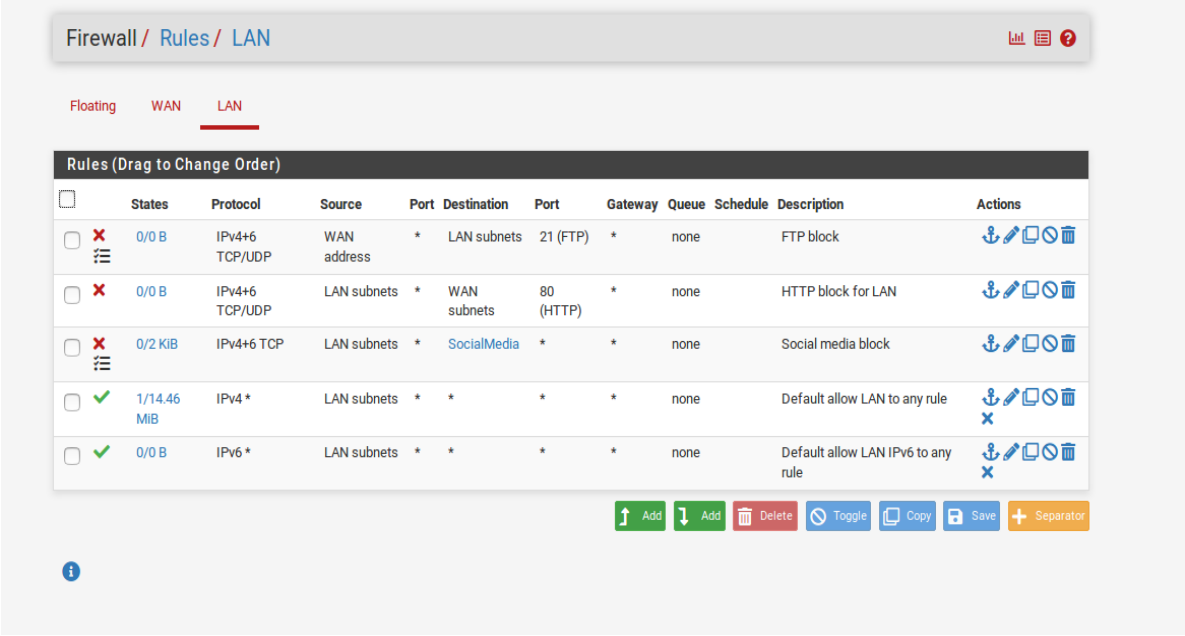
F29: FTP block on WAN interface

Next, we created another FTP block rule on the LAN interface, to block any FTP traffic if a device is connected to the LAN interface of the Firewall



F30: FTP block on LAN Interface

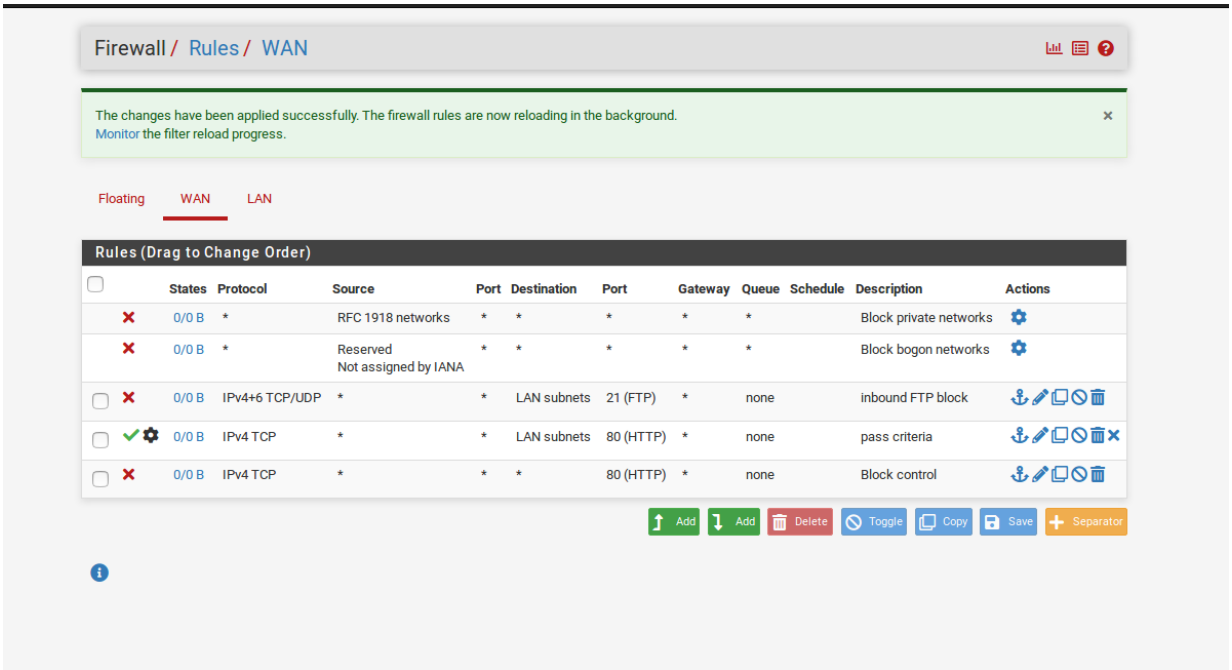
Below is a screenshot for all the LAN rules configured:



F31: ALL Configured LAN rules

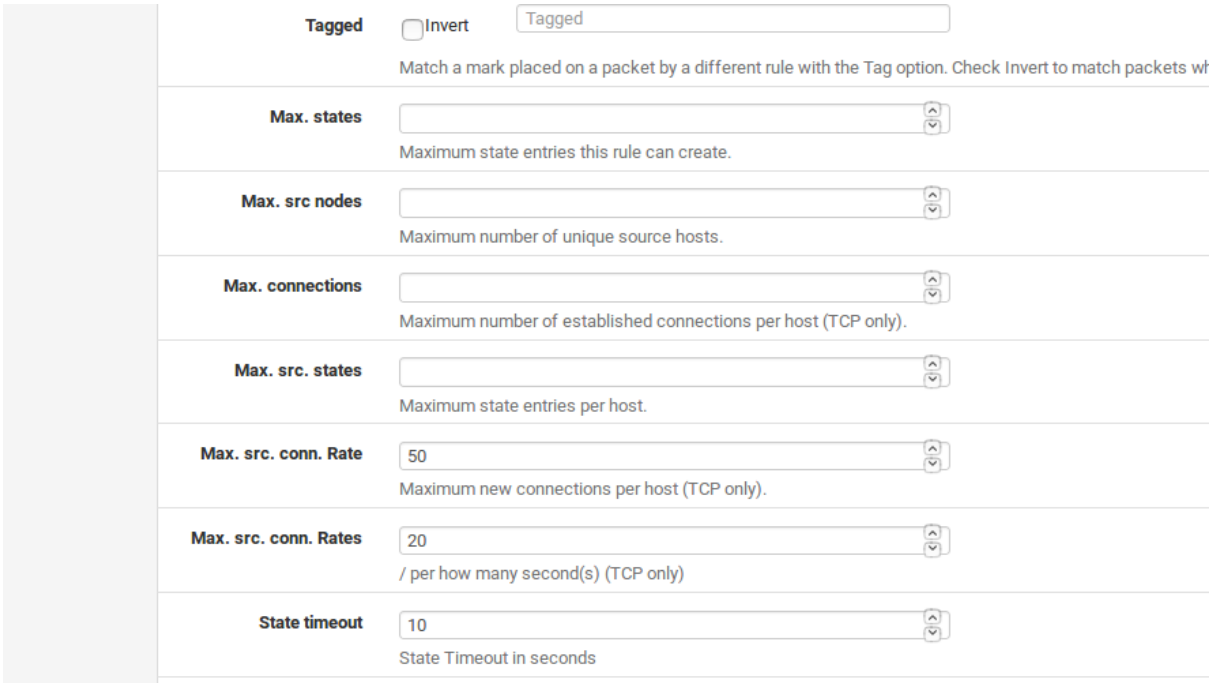
Rule 4: Block DOS attacks:

There are many ways to implement this rule, we created a block generalized block HTTP traffic and then created a HTTP pass rule with advanced controls to limit the maximum number of states created by one user and the state timeout. The screenshot us attached below for the WAN rules:



F32: HTTP block control and HTTP pass rule with advanced controls

Below is a screenshot for the HTTP pass criteria that we've used. We have set maximum connections per host to be 50, maximum connections per second to be 20 and state timeout to be 10 seconds.



F33: Pass control Criteria for HTTP traffic

Any HTTP traffic not matching this would be blocked thus preventing DOS attack.