CySA+ Lab Series

Lab 14: Packet Crafting and Scanner Honeypots with Scapy

Document Version: 2022-10-10

Material in this Lab Aligns to the Following	
CompTIA CySA+ (CS0-002) Exam Objectives	2.1 - Explain software assurance best practices 3.1 - Given a scenario, analyze data as part of security monitoring activities 3.3 - Explain the importance of proactive threat hunting 3.4 - Compare and Contrast automation concepts and technologies
All-In-One CompTIA CySA+ Second Edition ISBN-13: 978-1260464306 Chapters	8: Security Solutions for Infrastructure Management 11: Data Analysis in Security Monitoring Activities 13: The Importance of Proactive Threat Hunting 14: Automation Concepts and Technologies

Contents

Introduction

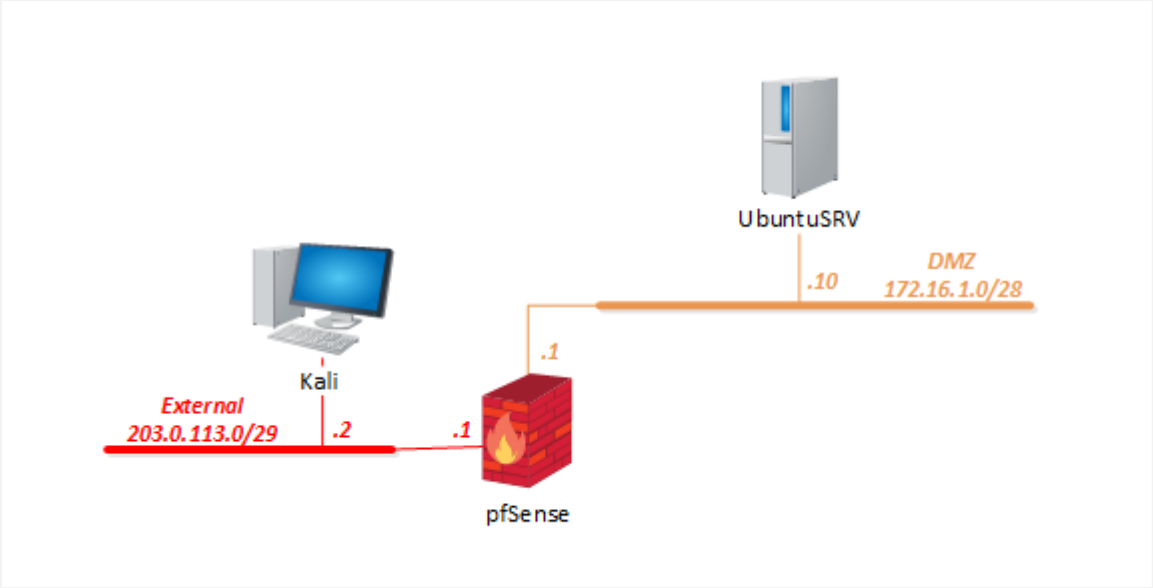
Attacks and defenses of an organization's network begin with packets. Hackers will craft and compromise packets and try to inject them into a network to do their nefarious deeds. The security analyst needs to be able to detect these evil packets and send them to the "bit-bucket" or reflect them back to the hacker in hopes of tripping them up. This means that packet manipulation is an important part of the security analyst's arsenal. One of the most commonly used tools is *Scapy*.

"Scapy is a powerful interactive packet manipulation program. It is able to forge or decode packets of a wide number of protocols, send them on the wire, capture them, match requests and replies, and much more." [https://scapy.net]

Objectives

- Create a honeypot
- Sniffing the packets using Scapy
- Write a Python script that will answer every SYN with a SYN/ACK

Lab Topology



Lab Settings

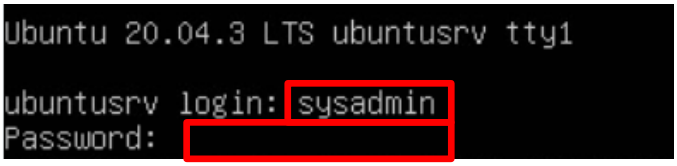
The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address	Account	Password
WinOS (Server 2019)	192.168.0.50	Administrator	NDGlabpass123!
MintOS (Linux Mint)	192.168.0.60	sysadmin	NDGlabpass123!
OSSIM (AlienVault)	172.16.1.2	root	NDGlabpass123!
UbuntuSRV (Ubuntu Server)	172.16.1.10	sysadmin	NDGlabpass123!
Kali	203.0.113.2	sysadmin	NDGlabpass123!
pfSense	203.0.113.1 172.16.1.1 192.168.0.1	admin	NDGlabpass123!

1 Crafting Packets with Scapy

In the first part of this lab, you will be crafting some basic IP and TCP packets to see how *Scapy* builds packets.

1. Set the focus to the **UbuntuSRV** computer.
2. Log in as `sysadmin` using the password: `NDGlabpass123!`



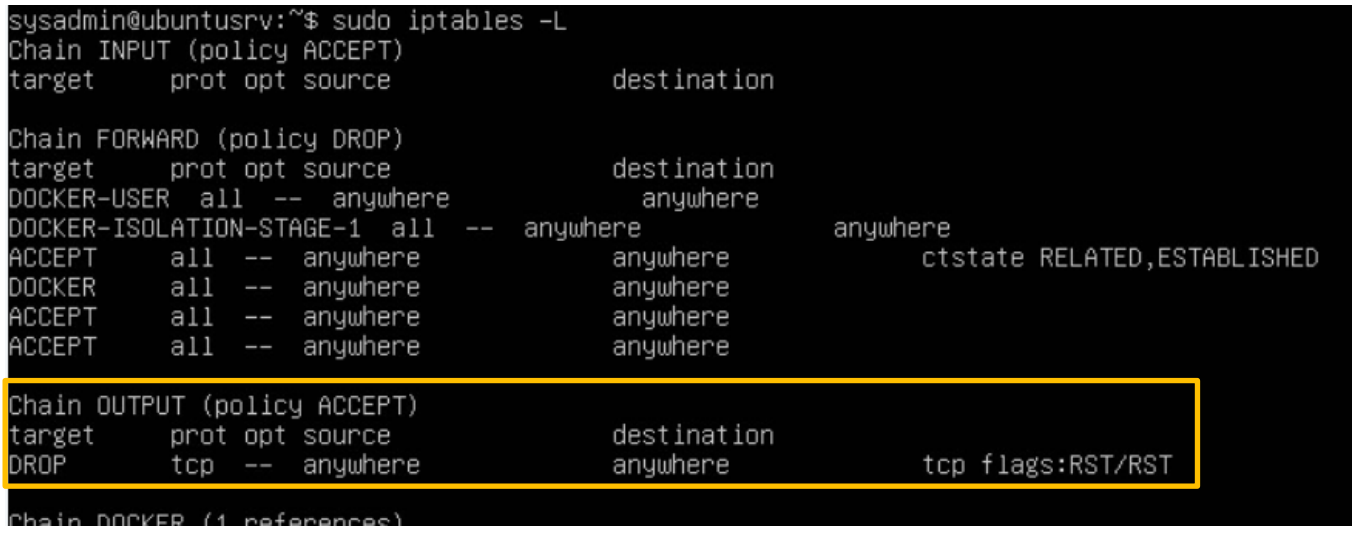
3. On the *UbuntuSRV* computer, add the following rule to **iptables** to block all outbound **RST** packets, or else the *UbuntuSRV* computer will cancel all connections with an *RST* packet.

```
sudo iptables -A OUTPUT -p tcp --tcp-flags RST RST -j DROP
```

If asked for the **sysadmin** password, type: `NDGlabpass123!`

4. Check the **iptables** rules by using the following command:

```
sudo iptables -L
```



Confirm you see a rule in the **OUTPUT** section that drops **RST** packets.

5. Type the command below to start the **scapy** application, followed by the **Enter** key.

```
sudo scapy
```

If asked for the **sysadmin** password, type: `NDGlabpass123!`

```
sysadmin@ubuntusrv:~$ sudo scapy
INFO: Can't import matplotlib. Won't be able to plot.
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)

                     aSPY//YASa
             apyyyyCY//////////YCa       |
            sY//////YSpcs  scpCY//Pp     | Welcome to Scapy
 ayp ayyyyyyySCP//Pp           syY//C    | Version 2.4.3
 AYAsAYYYYYYYY///Ps              cY//S   |
         pCCCCY//p          cSSps y//Y   | https://github.com/secdev/scapy
         SPPPP///a          pP///AC//Y   |
              A//A            cyP////C   | Have fun!
              p///Ac            sC///a   |
              P////YCpc           A//A   | Craft packets before they craft
       scccccp///pSP///p          p//Y   | you.
      sY/////////y  caa           S//P   |                     -- Socrate
       cayCyayP//Ya              pY/Ya   |
        sY/PsY////YCc           aC//Yp
         sc  sccaCY//PCypaapyCP//YSs
                  spCPY//////YPSps
                       ccaacs
                                       using IPython 7.13.0
>>> _
```

6. Change the appearance of the *Scapy* window to make the output easier to read. At the **>>>** prompt, execute the command, as shown below.

```
conf.color_theme = RastaTheme()
```

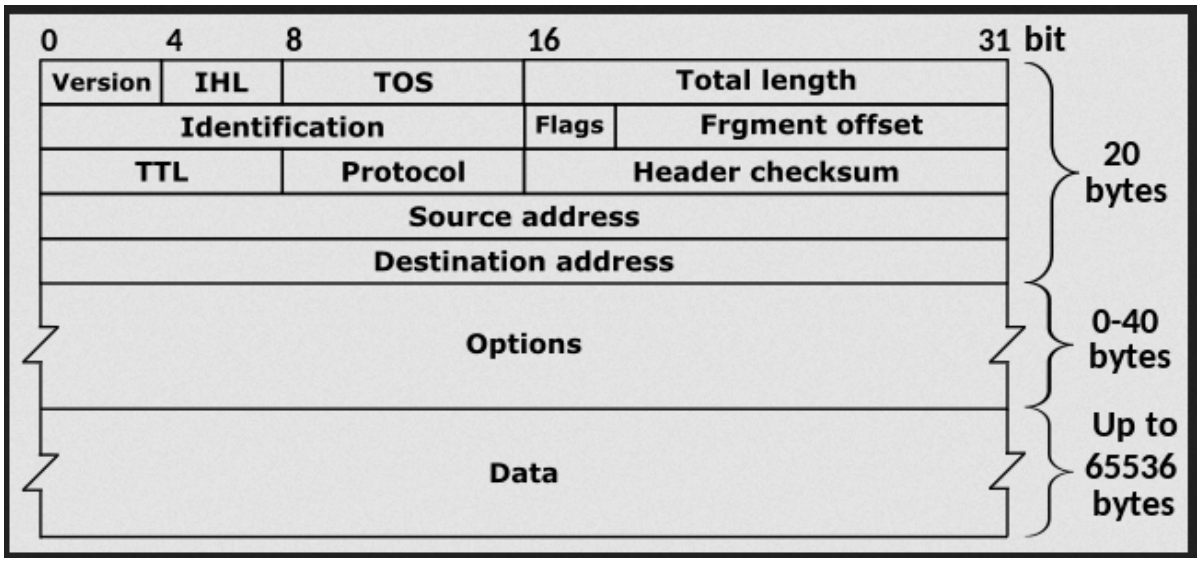
```
>>> conf.color_theme = RastaTheme()
```

7. In the *Scapy* window, at the **>>>** prompt, execute the two commands, as shown below. They are required for traffic to be accessed by the local host.

```
conf.L3socket
conf.L3socket=L3RawSocket
```

```
>>> conf.L3socket
<L3PacketSocket: read/write packets at layer 3 using Linux PF_PACKET sockets>
>>> conf.L3socket=L3RawSocket
```


To build an *IP Packet*, use RFC 79. The diagram below lists the fields in an IP packet header.



Michel Bakni - Postel, J. (Septemper 1981) RFC 791, Internet Protocol, DARPA Internet Program Protocol Specification, The Internet Society, p. 11 DOI: 10.17487/RFC0791., CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=79949694

8. You enter the header information into the *Scapy* interface interactively. First, you need to create an **IP Packet** object. Type the following command at the *scapy* **>>>** prompt (the statements are case sensitive):

```
i=IP()
```

```
>>> i=IP()
```

9. Type the following command to display the attributes of the IP packet object.

```
i.display()
```

```
>>> i.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 127.0.0.1
  dst= 127.0.0.1
  \options\
```


10. Add a source and destination IP address by typing the following commands:

```
i.src="172.16.1.10"
i.dst="172.16.1.1"
```

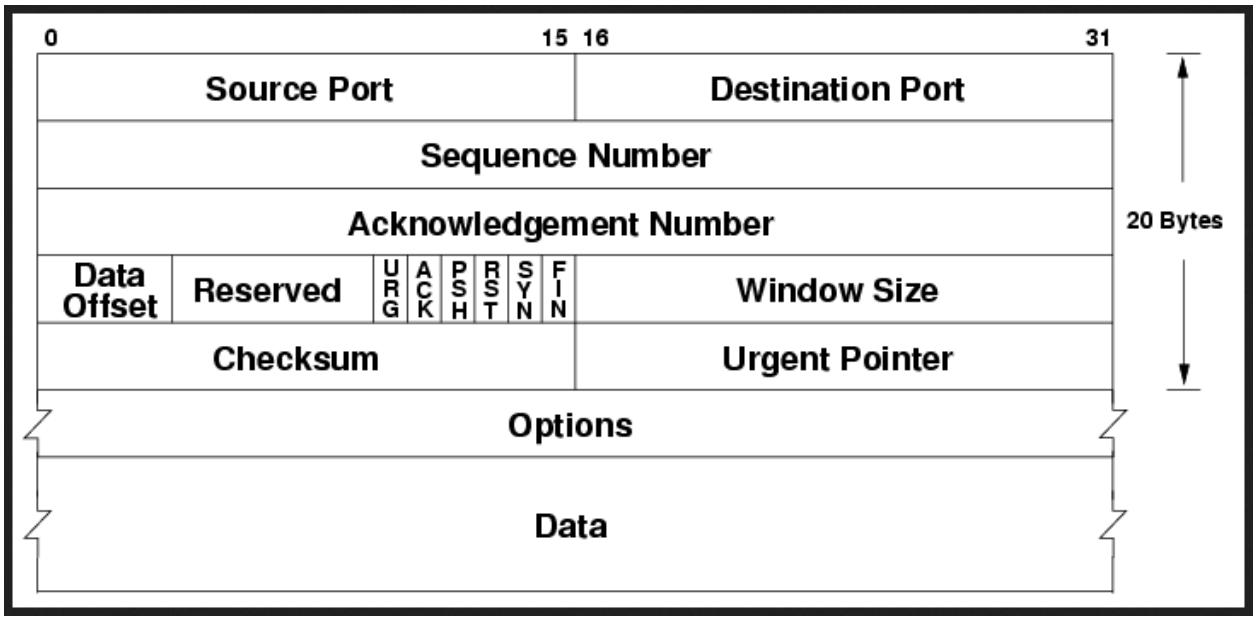
```
>>> i.src="172.16.1.10"
>>> i.dst="172.16.1.1"
```

11. Display the IP packet header's attributes again to make sure the addresses are correct by typing the command:

```
i.display()
```

```
>>> i.display()
###[ IP ]###
  version= 4
  ihl= None
  tos= 0x0
  len= None
  id= 1
  flags=
  frag= 0
  ttl= 64
  proto= hopopt
  chksum= None
  src= 172.16.1.10
  dst= 172.16.1.1
  \options\
```

To build a *TCP Packet*, use RFC 793. The diagram below lists the fields in a TCP packet header.



https://commons.wikimedia.org/wiki/File:TCP_header.png

12. Create a **TCP Packet** object by typing the following command at the *scapy* **>>>** prompt:

```
t=TCP()
```

```
>>> t=TCP()
```

13. Type the following command to display the attributes of the TCP packet object:

```
t.display()
```

```
>>> t.display()
###[ TCP ]###
  sport= ftp_data
  dport= http
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```


14. Add source and destination **TCP Ports** by typing the following commands:

```
t.sport=30000
i.dport=80
```

```
>>> t.sport=30000
>>> t.dport=80
```

15. Display the *TCP Packet's* attributes again to make sure the ports are correct by typing the command:

```
t.display()
```

```
>>> t.display()
###[ TCP ]###
  sport= 30000
  dport= http
  seq= 0
  ack= 0
  dataofs= None
  reserved= 0
  flags= S
  window= 8192
  chksum= None
  urgptr= 0
  options= []
```

16. Type the following command to send the crafted packet onto the network and listen to a single packet in response:

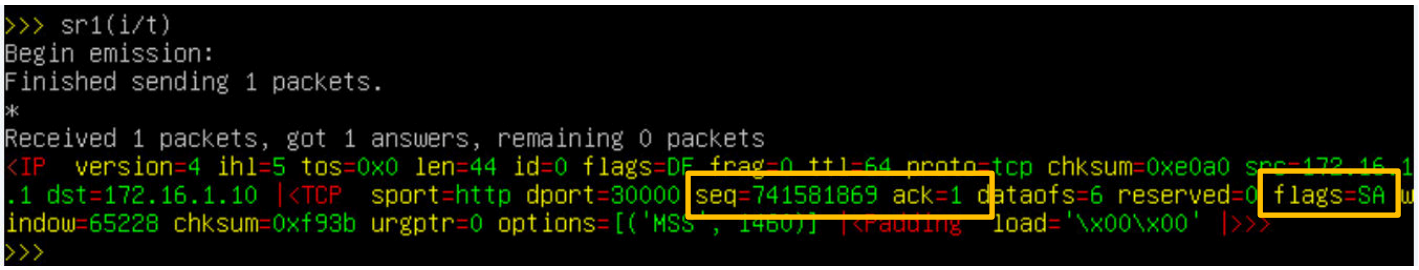
```
sr1(i/t)
```

Note that the third character is the numeral 1, not a lowercase L

This command sends a **SYN** packet and receives a **SYN/ACK** packet using the crafted **IP** and **TCP** headers that were just created.

17. In the image below, the response from *pfSense* (172.16.1.1) is shown with **TCP flags=SA**, which is the **SYN/ACK** reply. The **TCP ack=1**, indicates this is a reply to your **SYN** packet with **seq=2307383688**.

    Find the **TCP Seq** number in the **SYN/ACK** reply. It's highlighted in the figure below (your number might be different).



18. Leave *Scapy* open on the *UbuntuSRV* computer and continue to the next task.

2 Scanner Honeypot with Scapy

So, what is a honeypot, and what would a security analyst use it for?

"A honeypot is a network-attached system set up as a decoy to lure cyber attackers and detect, deflect and study hacking attempts to gain unauthorized access to information systems. The function of a honeypot is to represent itself on the internet as a potential target for attackers -- usually, a server or other high-value asset -- and to gather information and notify defenders of any attempts to access the honeypot by unauthorized users."

[https://www.techtarget.com/searchsecurity/definition/honey-pot]

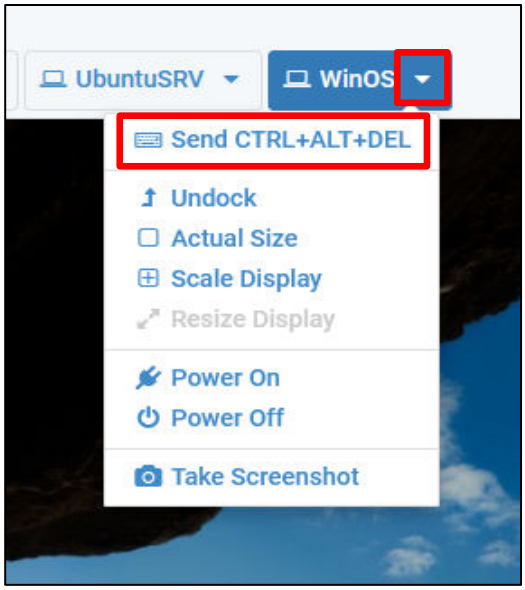
In this lab, you will create a "honeypot" and a simple script using *Scapy* that will answer every **TCP SYN** with a **TCP SYN/ACK,** making port scans useless as every port appears open.

In this part of the lab, the *UbuntuSRV* will act as the honeypot and will "lure" the *Kali* computer into the trap.

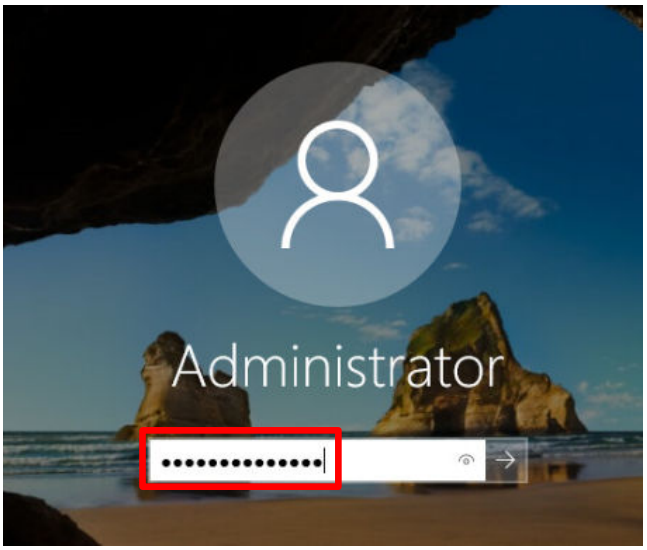
2.1 Permit all TCP SYN Packets Through the Firewall

The firewall is configured to only allow packets through to specific devices for specific services, for example, Port 443 (HTTPS:) to 172.16.1.10 (UbuntuSRV). But, to set up a cyber honeypot, which is intended to attract cyberattacks by baiting a trap, a security analyst will need to allow packets in. It would be easy enough to just turn off the firewall; however, that would expose the entire network to cyberattacks. Therefore, a firewall rule will need to be added that will allow all TCP packets that have the destination IP address of 172.16.1.10 (UbuntuSRV).

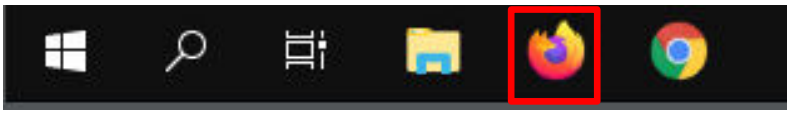
1. Change the focus to the **WinOS** computer.
2. Bring up the login window by sending a Ctrl + Alt + Delete. To do this, click the **WinOS** dropdown menu and click **Send CTRL+ALT+DEL**.



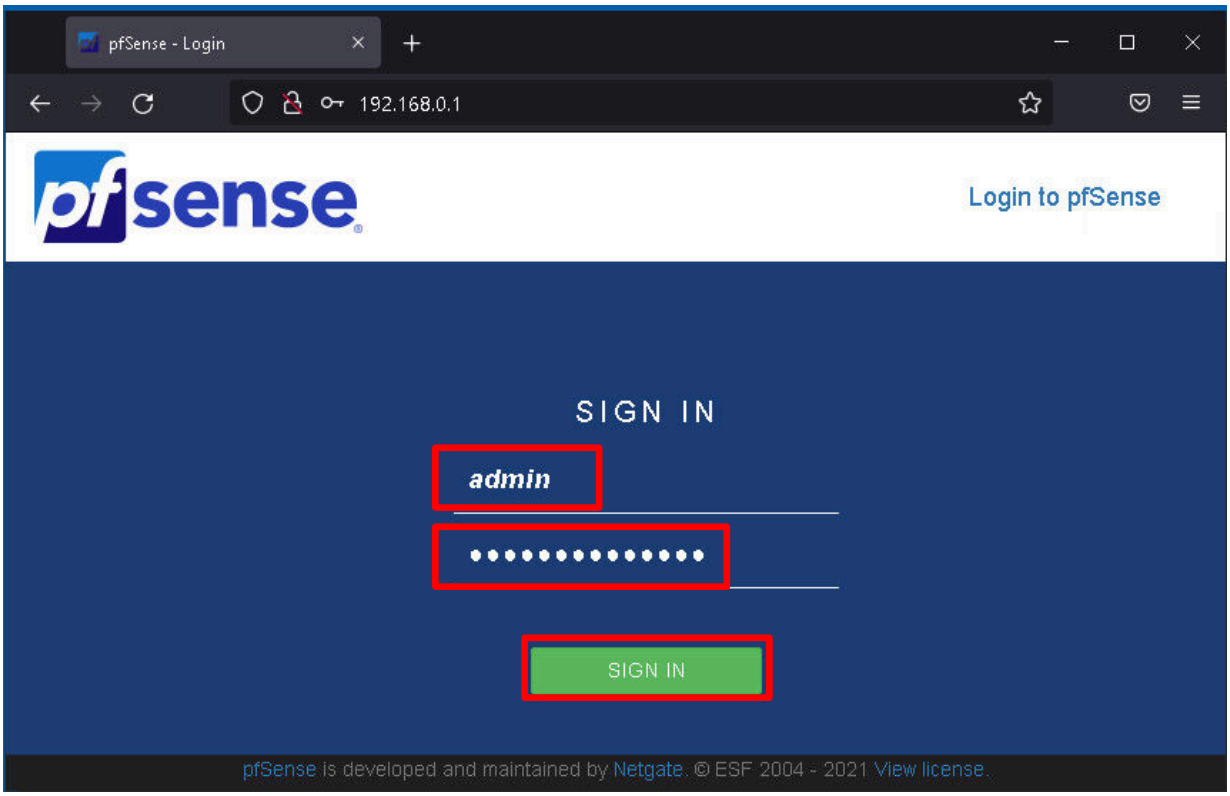
3. Log in as *Administrator* using the password: `NDGlabpass123!`



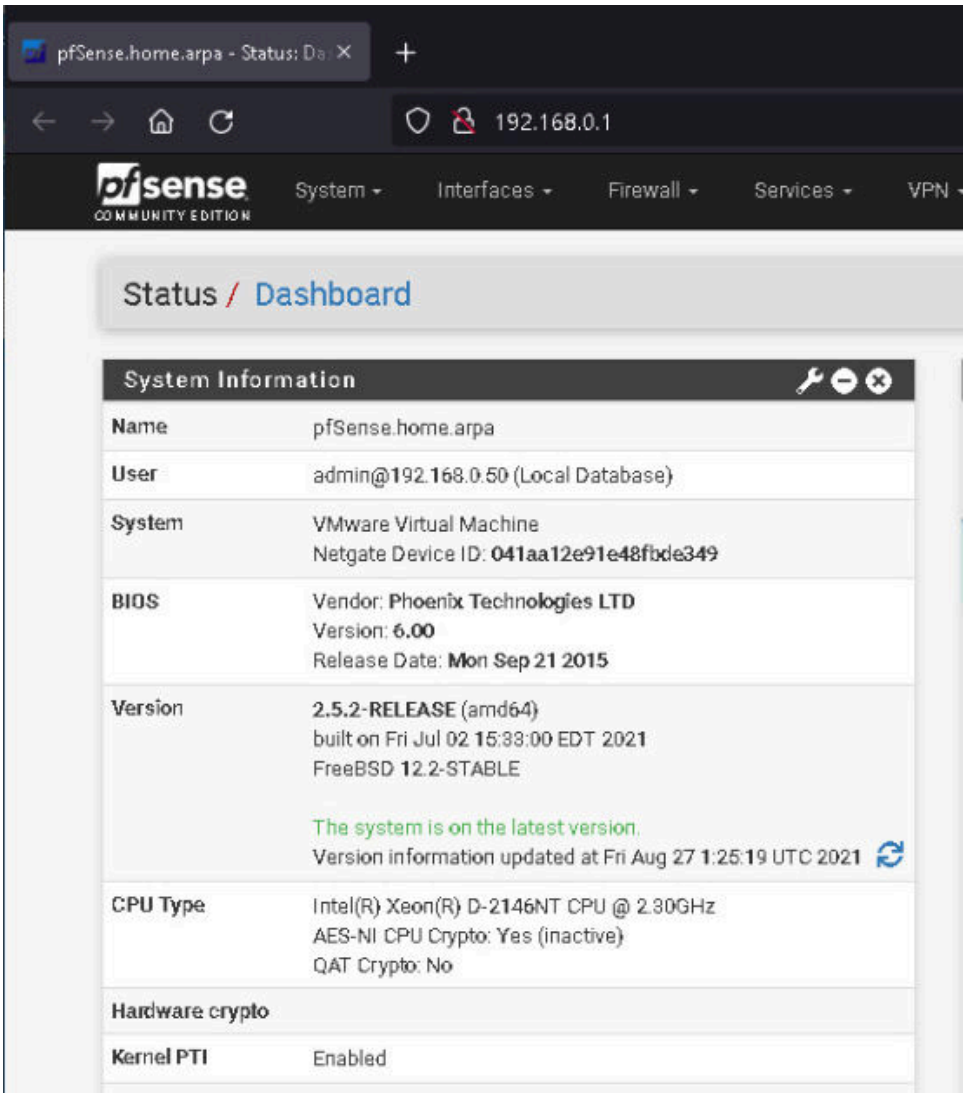
4. Click on the **Firefox** browser icon in the taskbar to open a web browser.



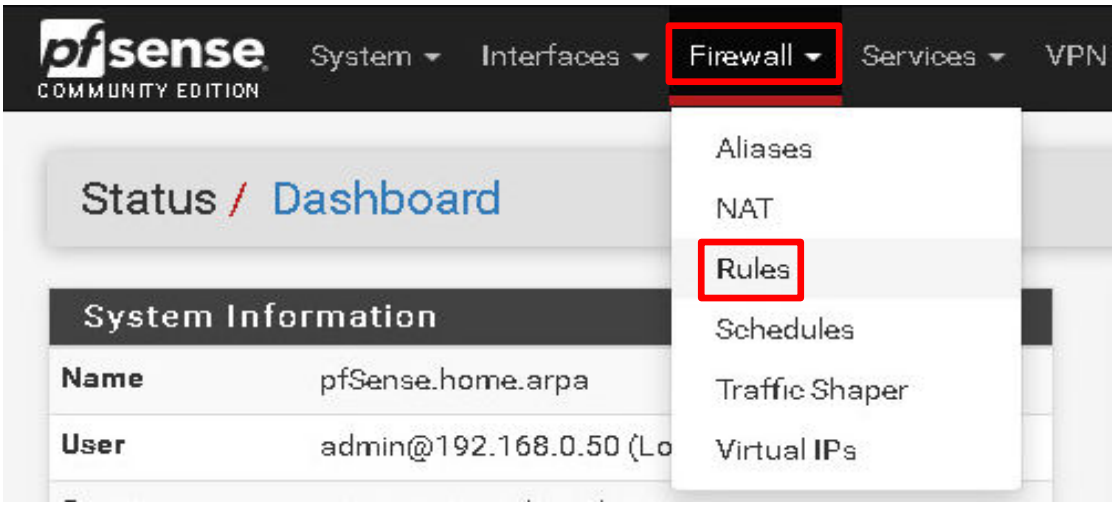
5. In the address bar of the browser, type `192.168.0.1`, the IP address of the *pfSense* server.
6. Log in as `admin` using the password `NDGlabpass123!` and click the **SIGN IN** button.



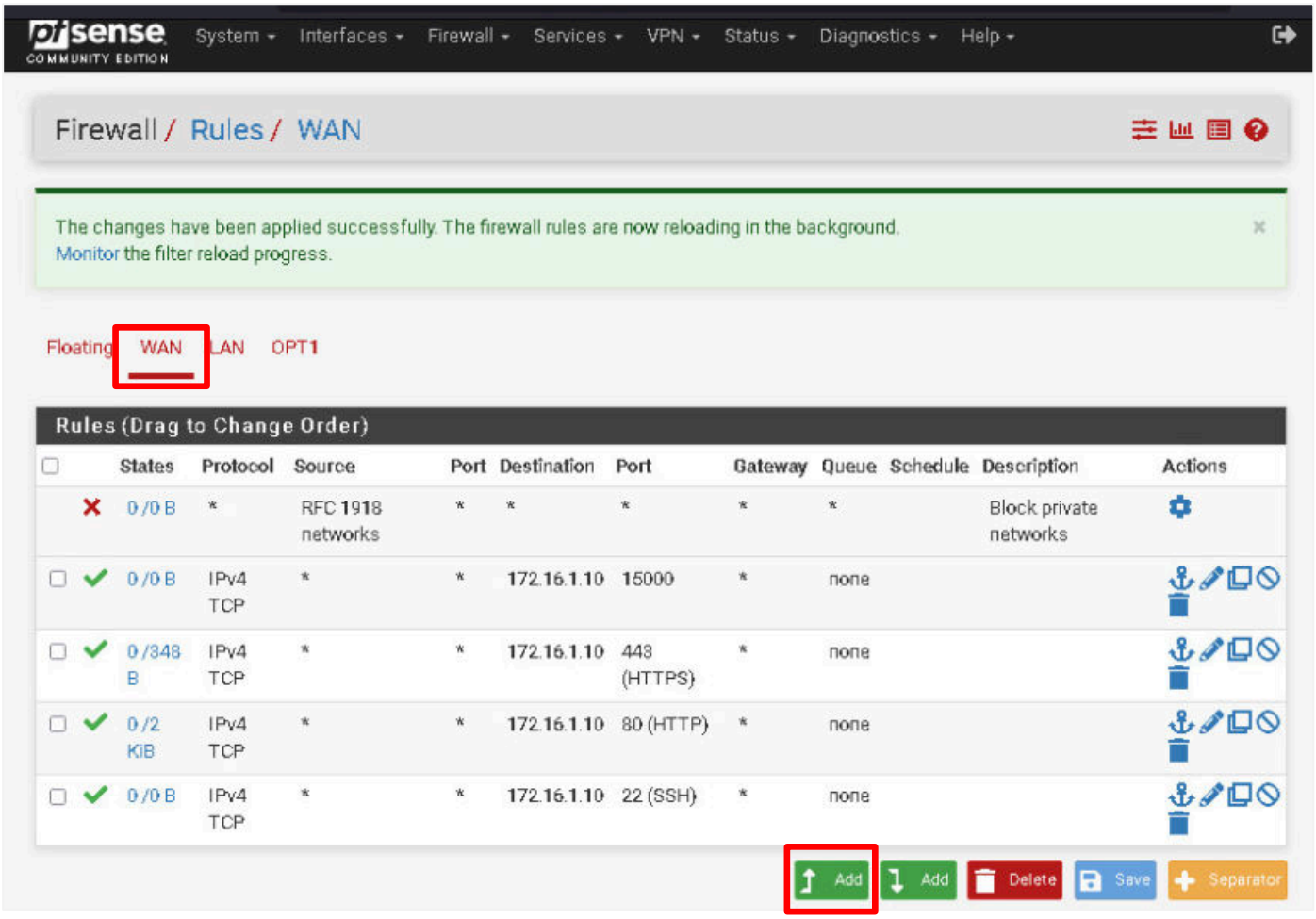
7. The *Dashboard* for the *pfSense* firewall should be shown.



8. Add a rule to allow through TCP SYN packets addressed to 172.16.1.10 (the UbuntuSRV computer) from the external network. Click on the **Firewall** menu item and then click on **Rules**.

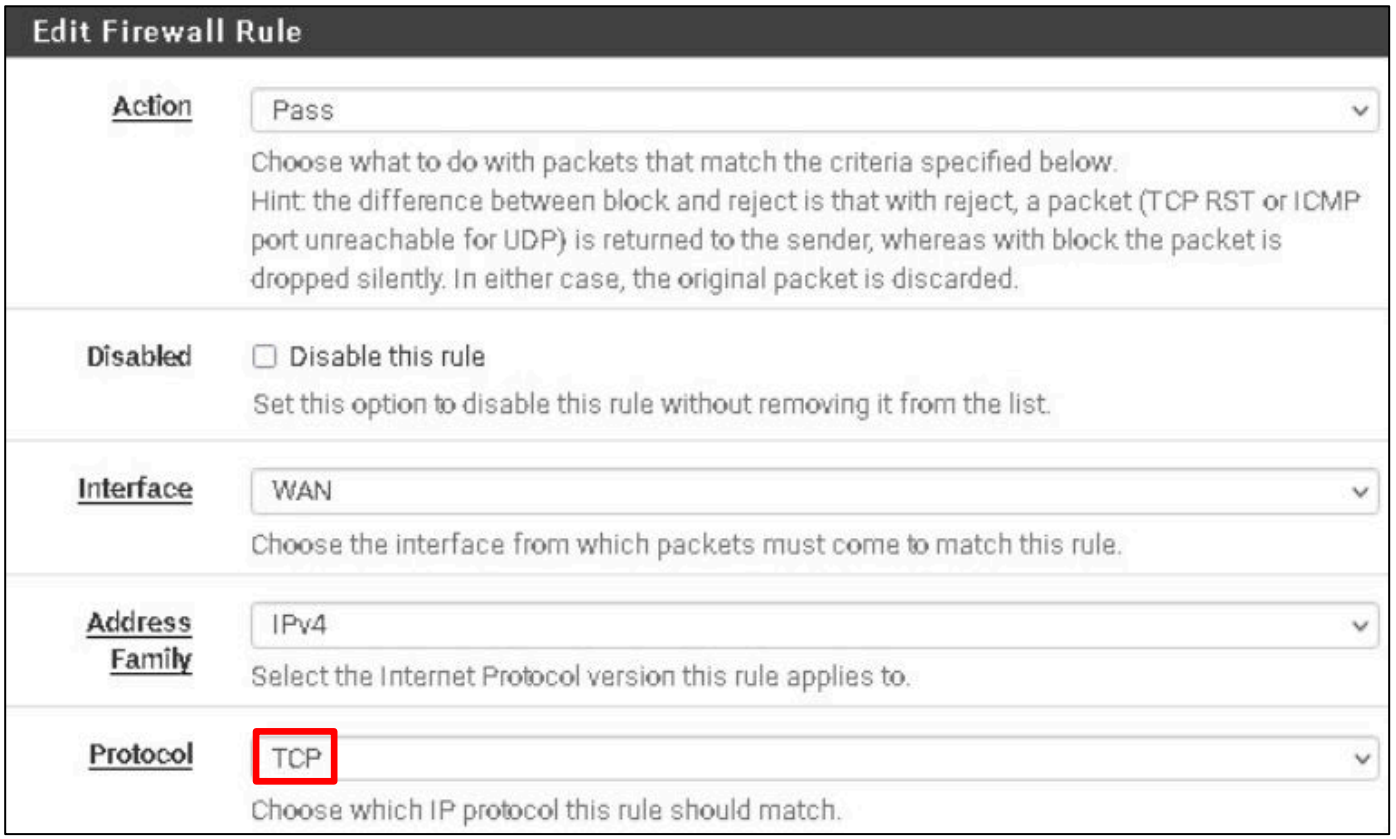


9. Make sure the **WAN** item is selected and click the **Add to Top** button.

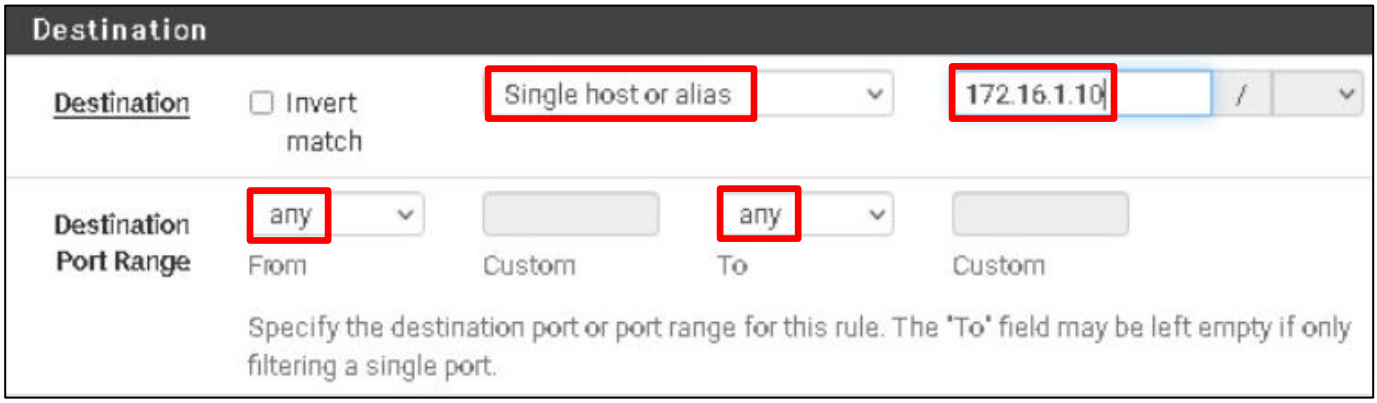


You will see in the list of firewall rules the four TCP ports that are being forwarded to the *UbuntuSRV* computer.

10. In the *Edit Firewall Rule* section, use the list arrow to change the *Protocol* to **TCP.**

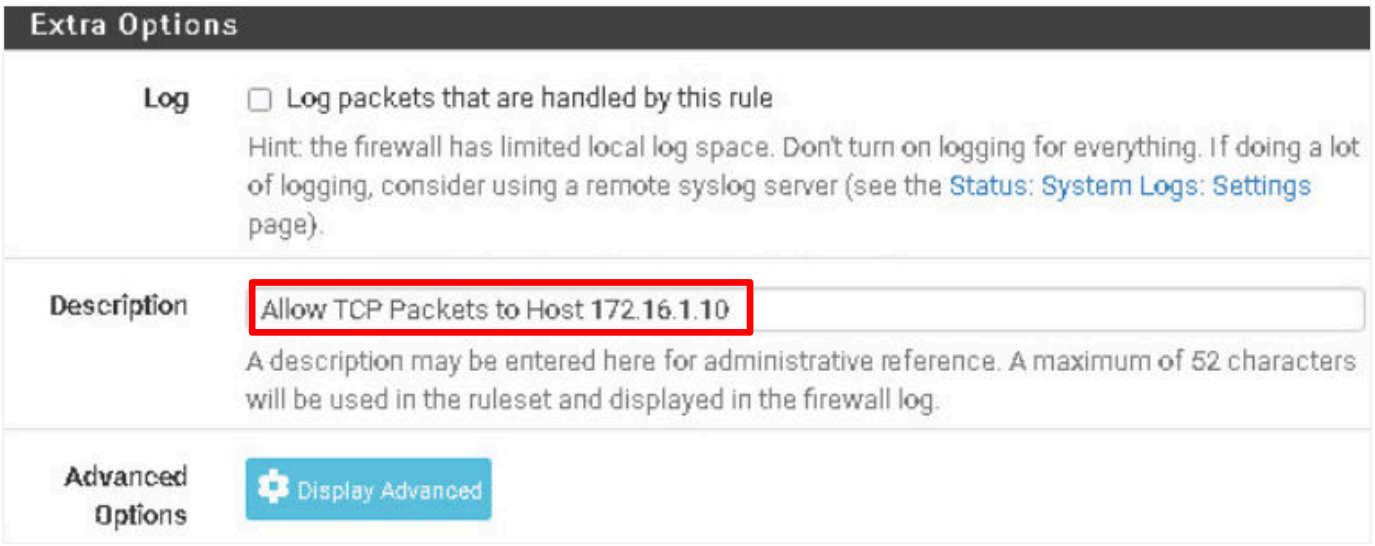


11. In the *Destination* section, use the list arrow to change the *Destination* to **Single host or alias,** change the *Destination Address* to `172.16.1.10` and the *Destination Port Range* to **any.**

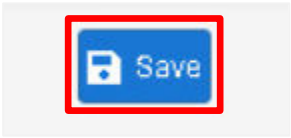


12. Scroll down the page.

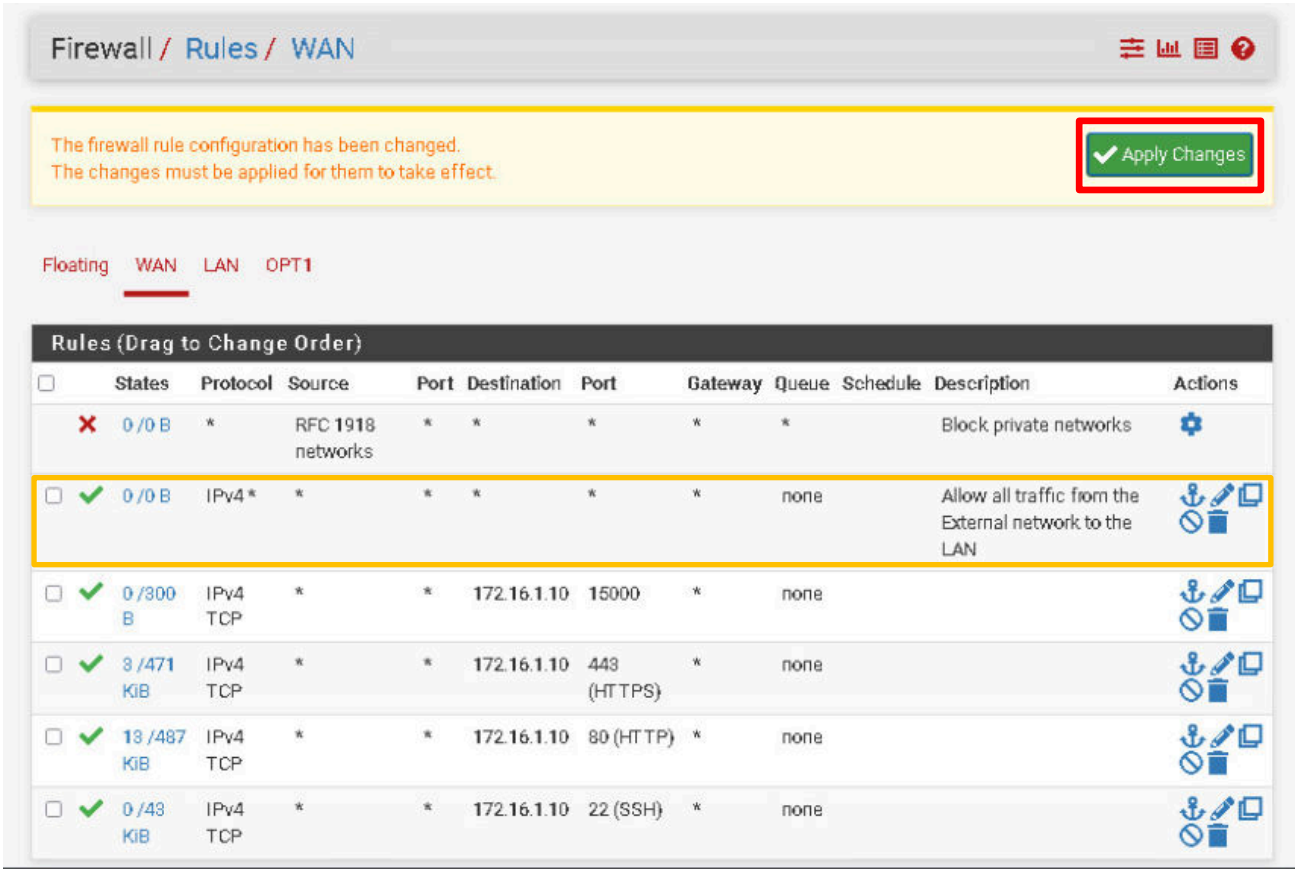
13. In the *Extra Options* section, change the description to `Allow TCP SYN Packets to Host 172.16.1.10`.



14. Scroll to the bottom of the page and click the **Save** button.

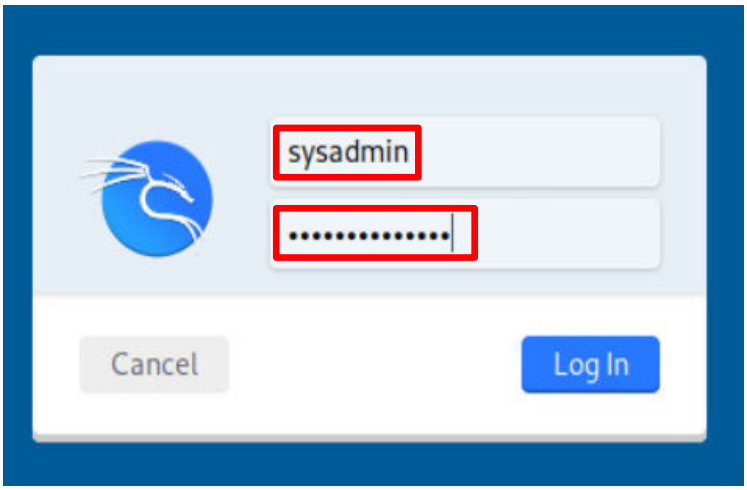


15. Note the new rule has been added to the list. Click the **Apply Changes** button.

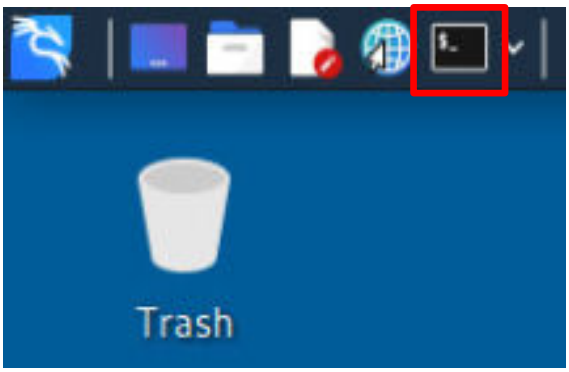


2.2 Sniffing Packets in Scapy

1. Set focus on the **Kali** computer.
2. Log in as `sysadmin` using the password: `NDGlabpass123!`



3. Click the **Terminal** icon to open a terminal window.



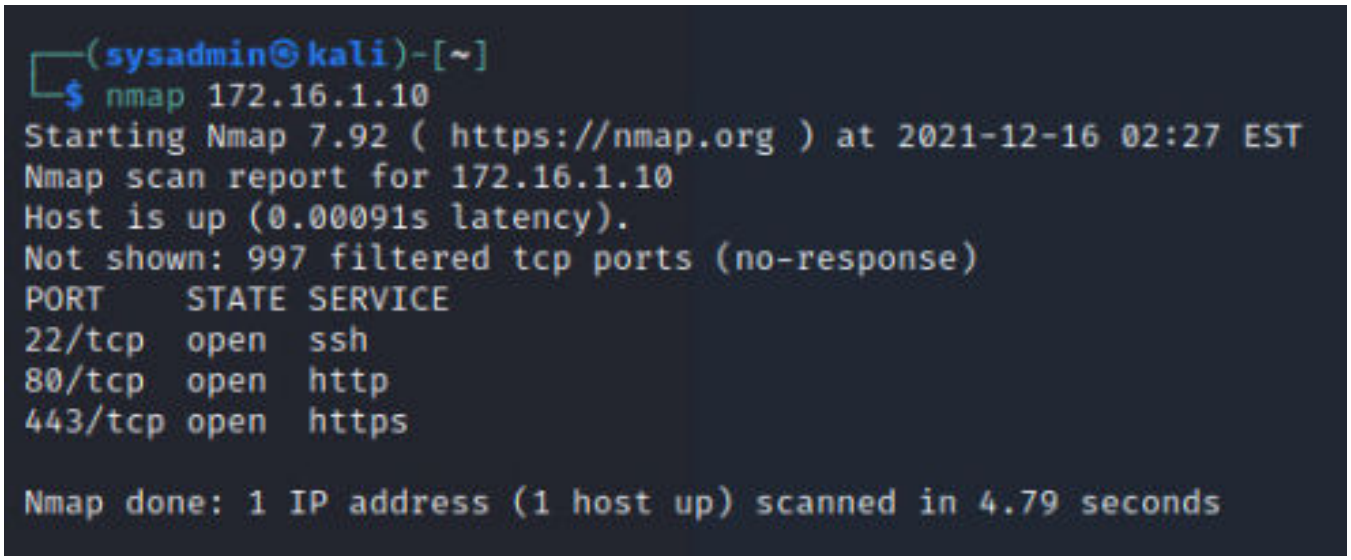
4. Set the focus on the **UbuntuSRV** computer.
5. The *Scapy* screen should still be open and at a **>>>** prompt. Tell *Scapy* to sniff 100 packets and save the results in the variable **p** by typing the following command at the >>> prompt:

```
p=sniff(count=100)
```

```
>>> p=sniff(count=100)
```

6. Set the focus back on the **Kali** computer and run *nmap* against the *UbuntuSRV* computer:

```
nmap 172.16.1.10
```



7. When *nmap* completes, return to the **UbuntuSRV** computer and type the command **p** to show the summary of how many packets of each type:

```
p
```

If you do not have many TCP packets, repeat the p=sniff(count=100) on the *UbuntuSRV* computer and the nmap on the *Kali* computer.

8. Type the command **p[50:70].summary()** to print out **20** packets starting at captured packet **50.**

```
p[50:70].summary()
```

```
>>> p[50:70].summary()
Ether / IP / TCP 203.0.113.2:52866 > 172.16.1.10:smux S
Ether / IP / TCP 203.0.113.2:32818 > 172.16.1.10:sunrpc S
Ether / IP / TCP 203.0.113.2:47170 > 172.16.1.10:mysql S
Ether / IP / TCP 203.0.113.2:48448 > 172.16.1.10:ftp S
Ether / IP / TCP 203.0.113.2:44360 > 172.16.1.10:1720 S
Ether / IP / TCP 203.0.113.2:50500 > 172.16.1.10:telnet S
Ether / IP / TCP 203.0.113.2:32832 > 172.16.1.10:sunrpc S
Ether / IP / TCP 203.0.113.2:47184 > 172.16.1.10:mysql S
Ether / IP / TCP 203.0.113.2:34750 > 172.16.1.10:1025 S
Ether / IP / TCP 203.0.113.2:48506 > 172.16.1.10:pop3 S
Ether / IP / TCP 203.0.113.2:48566 > 172.16.1.10:auth S
Ether / IP / TCP 203.0.113.2:48464 > 172.16.1.10:ftp S
Ether / IP / TCP 203.0.113.2:52878 > 172.16.1.10:smux S
Ether / IP / TCP 203.0.113.2:58770 > 172.16.1.10:ms_wbt_server S
Ether / IP / TCP 203.0.113.2:57716 > 172.16.1.10:imaps S
Ether / IP / TCP 203.0.113.2:56168 > 172.16.1.10:smtp S
Ether / IP / TCP 203.0.113.2:34048 > 172.16.1.10:5900 S
Ether / IP / TCP 203.0.113.2:34934 > 172.16.1.10:8888 S
Ether / IP / TCP 203.0.113.2:46368 > 172.16.1.10:netbios_ssn S
Ether / IP / TCP 203.0.113.2:38674 > 172.16.1.10:imap2 S
```

Look at the right side of the line summarizing your **TCP** packet. If it's a **SYN** from *nmap*, it will show an **S**; an **SA** is a **SYN/ACK** and an **A** is an **ACK**. To proceed, you need to find a **TCP SYN** packet from *203.0.113.2* (*Kali*) to examine. Look in the list that was produced and find the first packet from **203.0.0113.2** that has an **S** at the end. In the above example, it is the first packet, which would be packet 50.

9. Let's dive in and take a look at that **SYN TCP** packet by executing the following command:

```
p[50]
```

```
>>> p[50]
<Ether  dst=00:50:56:99:ce:0c src=00:50:56:99:8e:48 type=IPv4 |<IP  version=4 ihl=5 tos=0x0 len=60 i
d=20722 flags=DF frag=0 ttl=63 proto=tcp chksum=0x1ad src=203.0.113.2 dst=172.16.1.10 |<TCP  sport=4
8106 dport=http_alt seq=4084384083 ack=0 dataofs=10 reserved=0 flags=S window=64240 chksum=0xa72d ur
gptr=0 options=[('MSS', 1460), ('SAckOK', b''), ('Timestamp', (480382945, 0)), ('NOP', None), ('WSca
le', 7)] |>>>
```

Make sure the **IP SRC** field has the IP address of **Kali** (203.0.113.2), the **IP DST** field is **UbuntuSRV** (172.16.1.10), the **TCP SEQ** field has a number that is not 0, and the **TCP ACK** field is 0.

10. Exit *Scapy* by typing **quit** at the **>>>** prompt.

```
quit
```

11. Remain on the *UbuntuSRV* computer at the command prompt and continue to the next task.

2.3 Write a Python Script, spoofopen.py, to Setup the Honeypot

The **spoofopen.py** script will just answer every **SYN** packet with a **SYN/ACK,** which will confuse port scanners and make every port appear to be open.

1. On the **UbuntuSRV** computer, use the **nano** editor to create the script using this command:

```
sudo nano spoofopen.py
```

If asked for the **sysadmin** password, type: `NDGlabpass123!`

2. In the editor, type this script:

```
  GNU nano 4.8                                          spoofopen.py
#!/usr/bin/env python
import sys
from scapy.all import *

def findSYN(p):
        flags = p.sprintf("%TCP.flags%")
        if flags == "S":                # Only respond to SYN packets
                ip = p[IP]              # Received IP packet
                tcp = p[TCP]            # Received TCP packet
                i = IP()                # Outgoing IP packet
                i.dst = ip.sprintf("%IP.src%")
                i.src = ip.sprintf("%IP.dst%")
                t = TCP()               # Outgoing TCP segment
                t.flags = "SA"
                t.dport = tcp.sport
                t.sport = tcp.dport
                t.seq = tcp.ack
                t.ack = tcp.seq + 1
                print "SYN/ACK sent to ",i.dst,":",t.dport
                send(i/t)

sniff(prn=findSYN)
```

3. Save the file with **Ctrl+O,** then press **Enter** to save with the same name and **Ctrl+X** to exit.
4. The *spoofopen.py* file won't run until you give it Execute permission. To do that, on the *UbuntuSRV* computer, execute this command:

```
sudo chmod a+x spoofopen.py
```

If asked for the **sysadmin** password, type: **NDGlabpass123!**

```
sysadmin@ubuntusrv:~$ sudo chmod a+x spoofopen.py
```

5. Confirm the permissions have been changed by typing the command **ls –l** and check the output for the **x** permission for Owner, Group, and Everyone.

```
ls -l
```

```
sysadmin@ubuntusrv:~$ ls -l
total 4
-rwxr-xr-x 1 root root 617 Dec 16 21:44 spoofopen.py
```

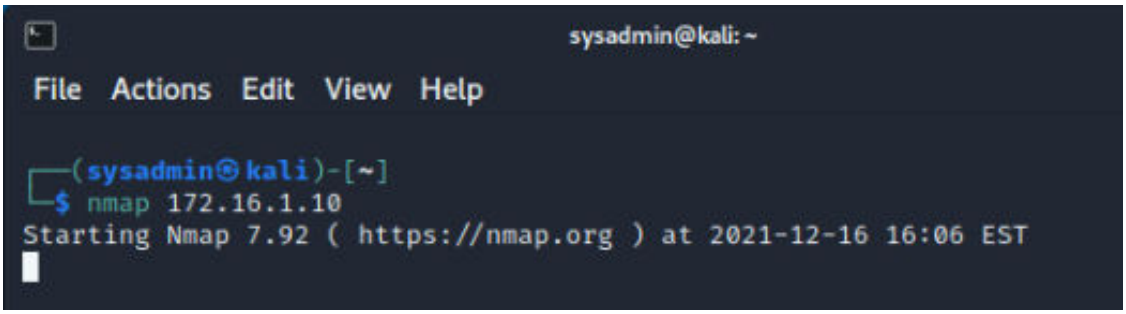

6. Start the **spoofopen.py** script by executing the following command:

```
sudo /usr/bin/python2.7 spoofopen.py
```

```
sysadmin@ubuntusrv:~$ sudo /usr/bin/python2.7 spoofopen.py
```

The script will be running in the background and you will not see any response on the screen until you run an *nmap* scan from the *Kali* computer.

7. Return to the **Kali** computer and open a terminal session if one is not already open.
8. Run *nmap* against the **UbuntuSRV** computer again.

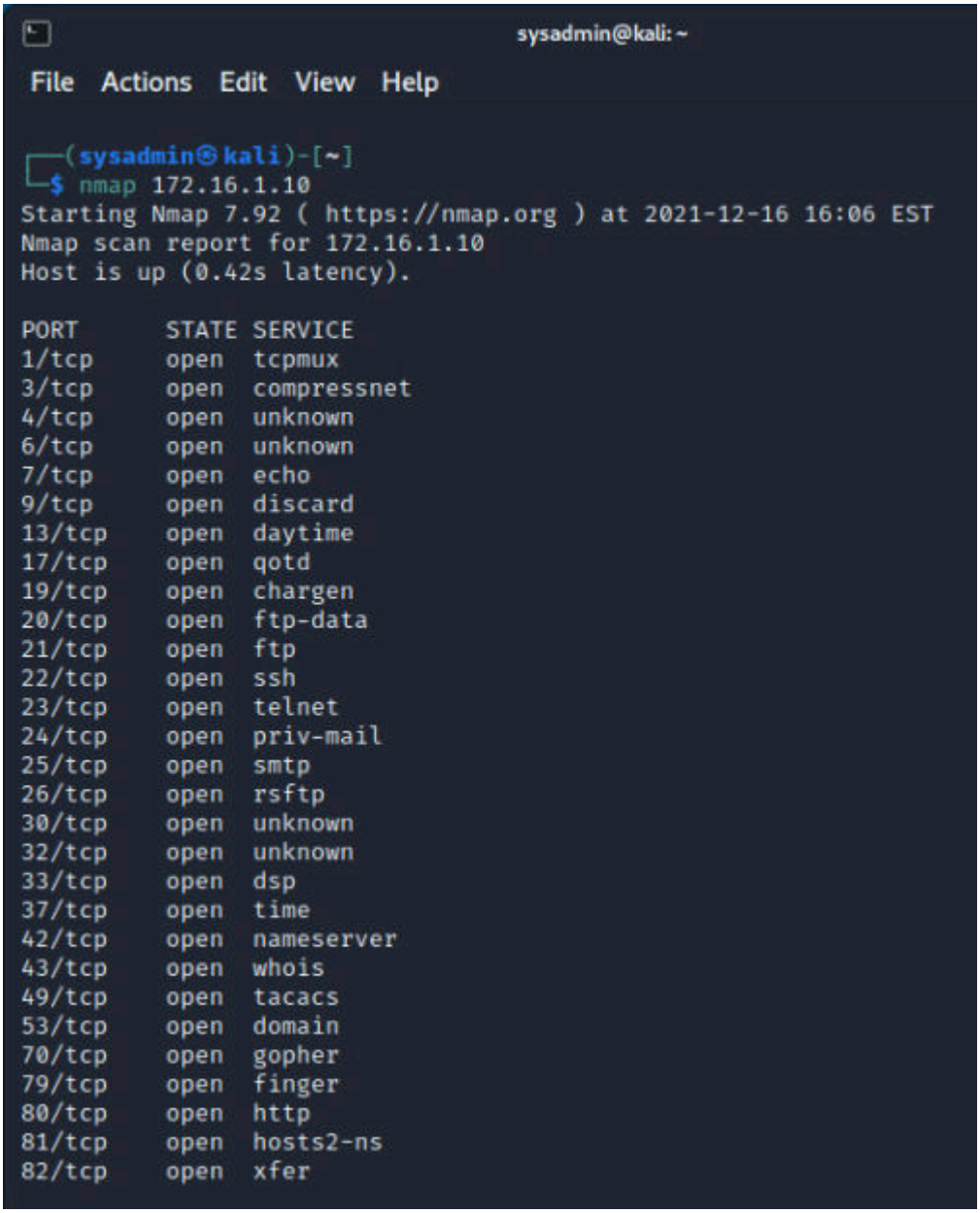


The *nmap* scan can take upwards of 3 minutes to complete.

9. Set the focus on the **UbuntuSRV** computer, and you will see all of the messages showing the **SYN/ACK** packages were sent back to *Kali*. Click on **Ctrl+C** to exit from the **spoofopen.py** script.

Included in the messages is the IP address of the "attacker", which could be used to track down the source of the port scan. In addition, the script could be modified to display any other information that is kept in the *IP* and *TCP* headers of the attacking computer.

10. Return focus to the **Kali** computer.
11. When the scan is complete, you should see that, according to *nmap*, every port on the *UbuntuSRV* is open (which, of course, they are not). Scroll up the terminal window, and you will see the ports starting at 1.



This task was adapted from Dr. Sam Bowne
Professor, City College of San Francisco.

12. This concludes the lab. You may now end the reservation.