LABORATOIRE V – CEG 4399

CEG 4399- Design of Secure Computer Sys.

Université d'Ottawa

Professeur : Amir H. Razavi

Noms et numéros des étudiants :
Gbegbe Decaho Jacques 300094197

Date de soumission: 24 October 2024

Using nmap for network discovery

Introduction

Overview

This Labtainer exercise explores the use of the nmap utility to discover computers and services on networks.

Begining the lab

The lab is started from the labtainer working directory on your Linux host, e.g. , a Linux VM. From there, issue the command:

```
labtainer nmap-discovery
```

The resulting virtual terminal will include a bash shell. The nmap utility is pre-installed on the computer connected to the termial.

We loaded the "**nmap-discovery**" labtainer into the virtualbox in order to launch our working environment.

1. Task

Your boss Randall wants you to prepare for a meeting on a project you have not worked on in months. You have a summary file on the "friedshrimp" server that you previously accessed via ssh; however, you cannot remember the IP address of "friedshrimp", and you also forgot which port the pesky IT staff assigned for ssh on that server. You know it's somewhere in between 2000 and 3000. The one thing you most certainly know is that your username and password are both "ubuntu". You are left with only one option: use the nmap command to find the IP address and and port number used by the ssh service. After finding that information review the contents of the "friedshrimp.txt" file from an ssh session.

If you need any help with the nmap commands, you can use "`man nmap`" to view the manual. Note that in order to ssh to a host via a port other than the default one, use "`ssh -p <port> <host>`".

```
ubuntu@mycomputer:~$ nmap -p 2000-3000 -sV 172.25.0.5

Starting Nmap 7.01 ( https://nmap.org ) at 2024-10-08 19:35 UTC
Nmap scan report for nmap-discovery.friedshrimp.student.intranet (172.25.0.5)
Host is up (0.00021s latency).
Not shown: 1000 closed ports
PORT     STATE SERVICE VERSION
2831/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
Service Info: OS: 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 0.57 seconds
ubuntu@mycomputer:~$
```

Thanks to the command **"ip addr show"** we were able to find the IP address of our local network.
Then following the lab instructions, we added the IP in the command **"nmap -p 2000-3000 -sV <IP_Address>"**.

```
ubuntu@mycomputer:~$ ssh -p 2831 ubuntu@172.25.0.5
The authenticity of host '[172.25.0.5]:2831 ([172.25.0.5]:2831)' can't be established.
ECDSA key fingerprint is SHA256:nFDnpYXdisAGpF1ZxOBv8Xc83CDp5qYU2frYQvB7Pt8.
Are you sure you want to continue connecting (yes/no)? yes
yes
Warning: Permanently added '[172.25.0.5]:2831' (ECDSA) to the list of known hosts.
ubuntu@172.25.0.5's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 6.8.0-36-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.
```

Thanks to the previous command we were able to obtain the name and number of active hosts on the network and also the ports used in this conversation.
After which, we executed the command "**ssh -p <port><host>**", then, we were able to observe the results on the screen below.

```
ubuntu@friedshrimp:~$ cat friedshrimp.txt
My summary notes from the fried shrimp project:

Fried Shrimp Project: We concluded it is better to
buy than to build.

==================================================

Congratulations! You managed to find the summary file
for "fired shrimp"and impress Randall.
```

We were able to confirm that our approach was the right one since we were able to obtain the hidden message thanks to the command "**cat friedshrimp.txt**" which displayed the congratulatory message shown in the screen above.

2. **Stop the labtainer**

When the lab is completed, or you'd like to stop working for a while, run:

```
stoplab
```

from the host Labtainer working directory (VM). You can always restart the Labtainer and continue your work. When the Labtainer is stopped, a zip file is created and copied to a location displayed by the "`stoplab`" command. When the lab is completed, send that zip file to the instructor.

```
ubuntu@friedshrimp:~$ exit
logout
Connection to 172.25.0.5 closed.
ubuntu@mycomputer:~$
```

We finally closed the used labtainer in order to continue our study on network discovery.

Snort – Intrusion Detection

Introduction

1. Overview

This exercise introduces the use of the snort system to provide intrusion detection within a Linux environment. Students will configure simple snort rules and experiment with a network intrusion detection system, (IDS).

2. Lab Environment

This lab runs in the Labtainer framework, available at http://my.nps.edu/web/c3o/labtainers. That site includes links to a pre-built virtual machine that has Labtainers installed, however Labtainers can be run on any Linux host that supports Docker containers.

From your labtainer-student directory start the lab using:

```
labtainer snort
```

A link to this lab manual will be displayed.

3. Network Configuration

This lab includes several networked computers as shown in Figure 1. When the lab starts, you will get virtual terminals, one connected to each component. The gateway is configured with `iptables` to use NAT to translate sources addresses of traffic from internal IP addresses, e.g., 192.168.2.1, to our external address, i.e., 203.0.113.10. The `iptables` in the gateway also routes web traffic (ports 80 and 443) to the web_server component by translating the externally visible destination address to the internal web server address.

The gateway is also configured to mirror traffic that enters the gateway via either the 203.0.113.10 link, or the link to the web server. This mirrored traffic is routed to the `snort` component. This mirroring allows the snort component to reconstruct TCP sessions between the web server and external addresses.

The snort component includes the Snort IDS utility. It also includes Wireshark to help you observe traffic being mirrored to the snort component.

The web server runs Apache and is configured to support SSL for web pages in the `www.example.com` domain.

The `remote_ws` component includes the Firefox browser, and a local `/etc/hosts` file that maps www.example.com to the external address of the gateway, i.e., 203.0.113.10. The internal workstation (ws2) also includes Firefox and an entry in `/etc/hosts` for www.example.com. Both workstations also include the nmap utility.

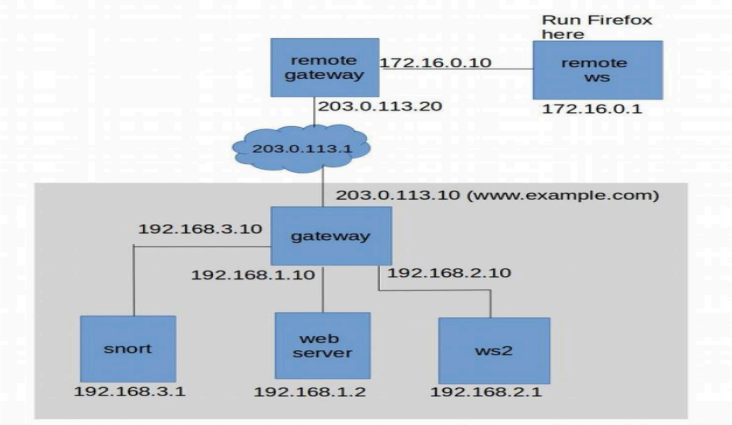


Figure 1: Network topology for the snort lab

4. Lab Tasks

It is assumed that the student has received instruction or independent study on the basic operation of Snort, and the general goals and mechanics of network intrusion detection.

Review the network topology. In particular, consider the `iptables` settings on the gateway. These can be seen by reviewing the commands in `/etc/rc.local`, which are used to define the NAT translations and, critically for this lab, mirror traffic to the snort component.

```
The lab manual is at
   file:///home/student/labtainer/trunk/labs/snort/docs/snort.pdf

You may open these by right clicking
and select "Open Link".


Press <enter> to start the lab
```

We loaded the "**snort**" labtainer into the virtualbox in order to launch our working environment.

1) Starting and stopping snort

The Snort utility is installed on the snort component. The home directory includes a `start_snort.sh` script that will start the utility in *Network Intrustion Dection Mode*, and display alerts to the console. For this lab, you are required to start snort with:

```
./start_snort.sh
```

When it comes time to stop snort, e.g., to add rules, simply use `CTL-C`.

```
tom@snort:~$ ./start_snort.sh
10/08-20:05:26.434061  [**] [1:469:3] ICMP PING NMAP [**] [Classification: Attempted Information Leak] [Priority: 2] {IC
MP} 203.0.113.20 -> 203.0.113.10
10/08-20:05:26.434061  [**] [1:384:5] ICMP PING [**] [Classification: Misc activity] [Priority: 3] {ICMP} 203.0.113.20 -
> 203.0.113.10
10/08-20:05:26.434300  [**] [1:453:5] ICMP Timestamp Request [**] [Classification: Misc activity] [Priority: 3] {ICMP} 2
03.0.113.20 -> 203.0.113.10
10/08-20:05:27.781873  [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Prior
ity: 2] {TCP} 203.0.113.20:57008 -> 203.0.113.10:705
10/08-20:05:27.842518  [**] [1:1418:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2]
 {TCP} 203.0.113.20:57008 -> 203.0.113.10:161
^C0tom@snort:~$ *** Caught Int-Signal
```

We loaded the script "**start_snort.sh**" in the window of "**tom@snort**" in order to run our snort environment in a network intrusion detection mode.
We were able to observe a list of instructions presented in the screen above.

2) Pre-configured Snort rules

The Snort utility includes a set of pre-configured rules that create alerts for known suspicious network activity. The configuration on the snort component is largely as it exists after initial installation of the snort utility. To see an example of some of the preconfigured rules, perform an nmap scan of www.example.com from the remote workstation:

```
sudo nmap www.example.com
```

Note the alerts displayed at the snort console. The rules that generate these alerts can be seen, along with all rules, in `/etc/snort/rules/`

```
hank@remote_ws:~$ sudo nmap www.example.com

Starting Nmap 7.01 ( https://nmap.org ) at 2024-10-08 20:05 UTC
Nmap scan report for www.example.com (203.0.113.10)
Host is up (0.000086s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 3.27 seconds
hank@remote_ws:~$
```

Then with the command "**sudo nmap www.example.com**" we were able to observe the different active ports on this network and also the number of hosts.
To do this this command was used in the workstation window "**hank@remote_ws**".

3) Write a simple (bad) rule

Custom rules are typically added to the file at `/etc/snort/rules/local.rules` Stop snort and add a rule that generates an alert for each packet within a TCP stream. For example:

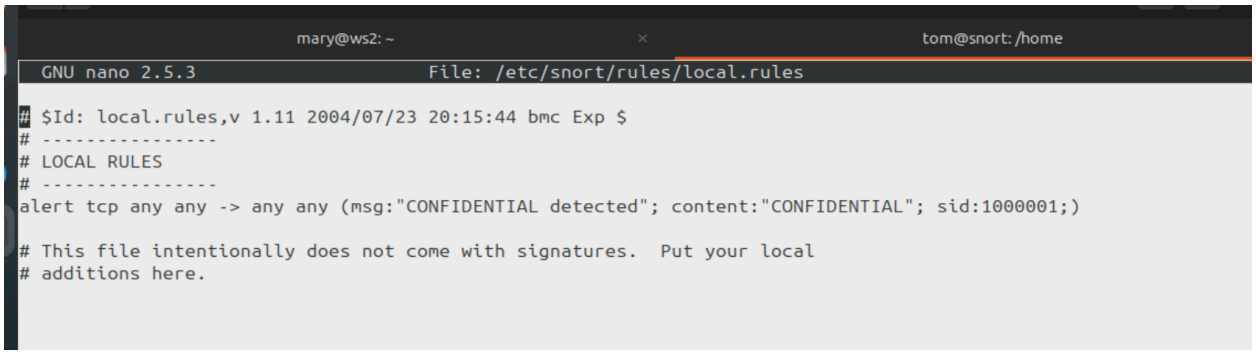
```
alert tcp any any -> any any (msg:"TCP detected"; sid:00002;)
```

That rule can be read as: "Generate an alert whenever a TCP packet from any address on any port is sent to any address on any port, and include the message tagged as `msg:`, and give the rule an identifier of 00002." Then restart snort. Test this rule by starting Firefox on the remote workstation:

```
firefox www.example.com
```

As you can see, the rule you wrote will overwhelm you with useless information. So, stop snort and delete the rule.

```
tom@snort:/home$ sudo nano /etc/snort/rules/local.rules
tom@snort:/home$
```



Then, with the **"sudo**" command, we were able to add a rule in the **local.rules** file that sends alerts for each TCP packet.

Afterwards, we displayed a web page using the command "**firefox www.example.com**". We were able to observe the result in the screen above.

4) Custom rule for CONFIDENTIAL traffic

At the Firefox browser, which should be displaying the webpage from www.example.com, we will display an unpublished web page that we know exists on the website. In particular, we have heard that the keen minds at the startup company have placed their confidential business plan at www.example.com/plan.html. Take a look at it.

Now add a rule to your local.rules file on snort that will generate an alert whenever the text "CONFIDENTIAL" is sent out to the internet. Reference the snort manual `https://www.snort.org/downloads/snortplus/snort_manual.pdf` or existing rules to understand how to qualify alerts based on *content*. Be sure to include the word "CONFIDENTIAL" in the alert message, and give the rule its own unique sid. After adding the rule, restart snort.

On the browser at the remote workstation, clear your history (Menu / Preferences Security & Privacy), and then refresh the plan.html page. You should see an alert at the snort console.

```
tom@snort:~$ ./start_snort.sh
10/08-20:17:04.729582  [**] [1:1000001:0] CONFIDENTIAL detected [**] [Priority: 0] {TCP} 192.168.1.2
:80 -> 192.168.1.10:43154
^C0tom@snort:~$ *** Caught Int-Signal

tom@snort:~$
```

```
tom@snort:~$ sudo nano /etc/snort/rules/local.rules
tom@snort:~$ sudo nano /etc/snort/rules/local.rules
```

```
  GNU nano 2.5.3             File: /etc/snort/rules/local.rules

# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------

# This file intentionally does not come with signatures.  Put your local
# additions here.
```

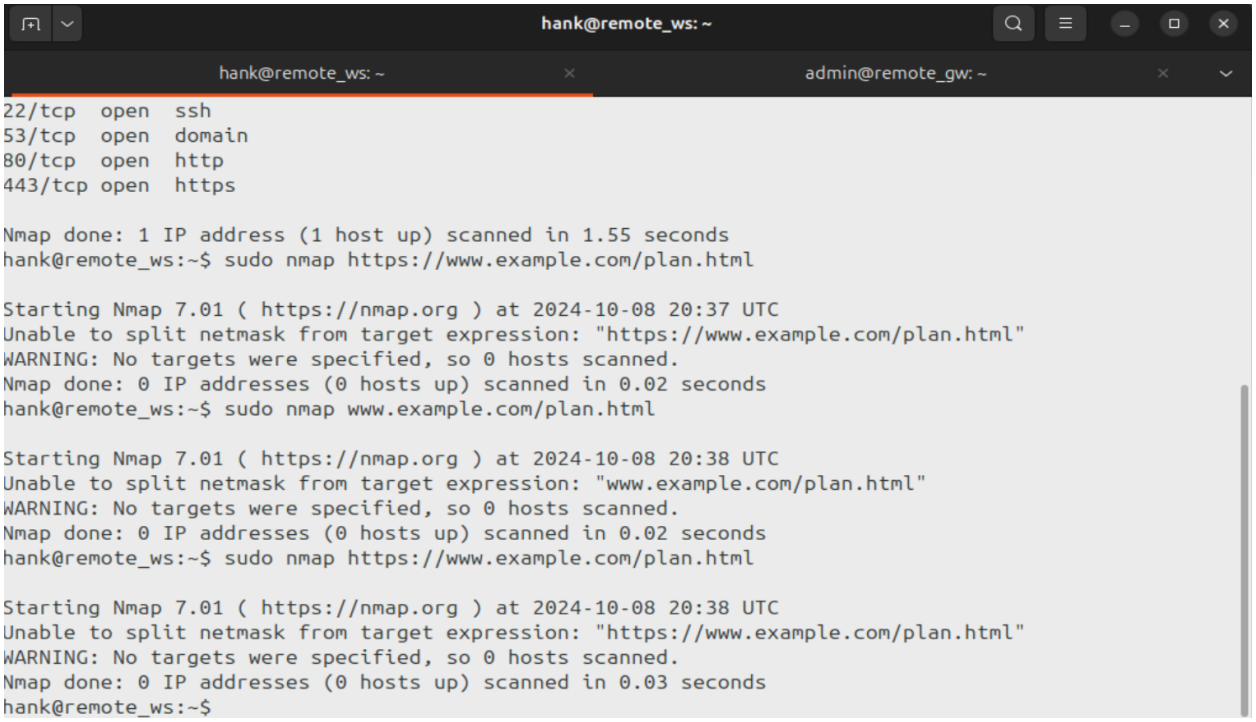
```
  GNU nano 2.5.3             File: /etc/snort/rules/local.rules                  Modified

# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
alert tcp any any -> any any (msg:"CONFIDENTIAL traffic detected"; content:"CONFIDENTIAL"; sid:0000$

# This file intentionally does not come with signatures.  Put your local
# additions here.
```

We replaced the existing rule with a new one, which now alerts whenever the word '**CONFIDENTIAL**' is found in the traffic.
Which will display the alert message on the "**firefox www.example.com**" when the page containing the word '**CONFIDENTIAL**' has been loaded.

5) Effects of encryption

Back at the Firefox browser, again clear the browser history. Now alter the URL to make use of the web server SSL function. Change the url to https://www.example.com/plan.html. Do you see a new snort alert? Why?

One solution to this problem is to use a reverse proxy in front of the web server. This reverse proxy would handle the incoming web traffic and manage the SSL connections. The web server would then receive only clear-text HTTP traffic, and outgoing traffic from the web server could then be mirrored to the IDS. We will not pursue that solution in this lab.



In the "**hank@remote_ws**", when we go to the page via HTTPS "**https://www.example.com/plan.html**" , we do not see any alert message because the traffic is encrypted and Snort cannot inspect the content. This is due to the SSL encryption of the traffic between the browser and the server.

6) Watching internal traffic

Go to the ws2 (mary) component and run nmap:

```
sudo nmap www.example.com
```

What do you see on the snort component? Does it include the `ICMP PING NMAP` alert that you saw when the remote workstation ran nmap? Why not?

Go to the gateway component and edit the `/etc/rc.local` script so that traffic from Mary's workstation is mirrored to the snort component. You can do this by adding this line to the section of that file that defines the packet mirroring:

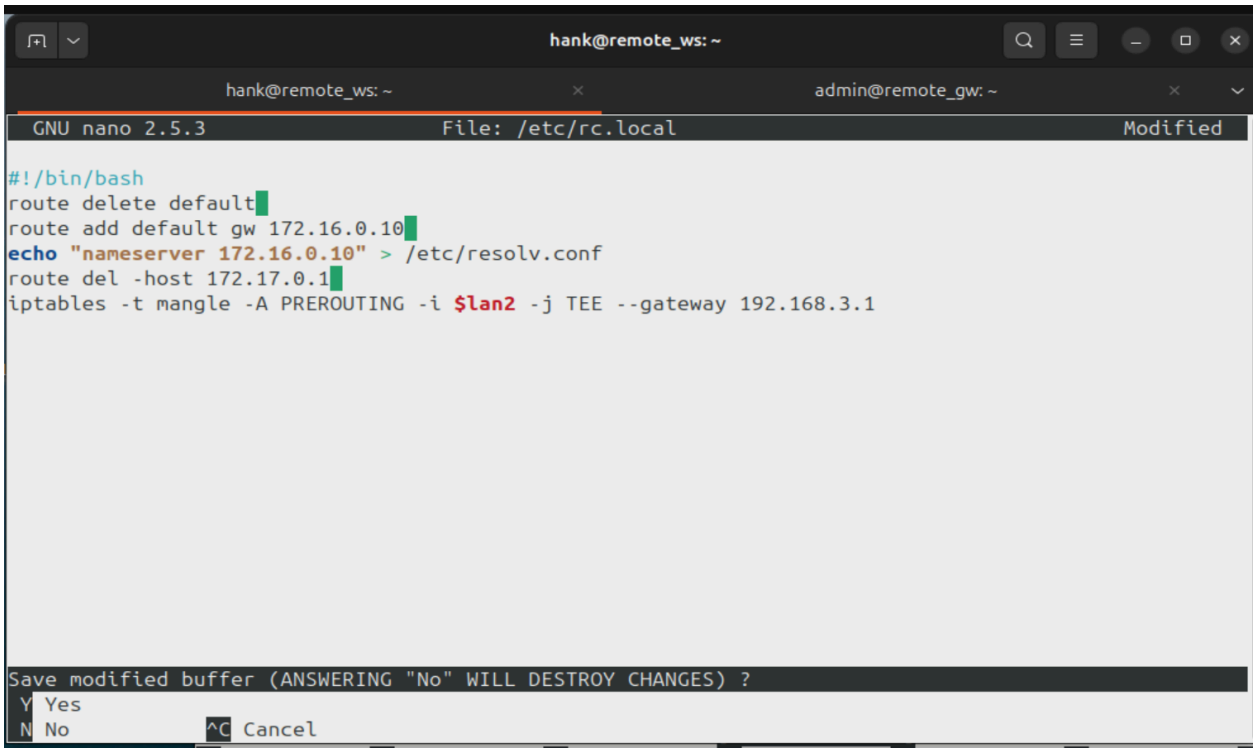
```
iptables -t mangle -A PREROUTING -i $lan2 -j TEE --gateway 192.168.3.1
```

Then run the script to replace the `iptables` rules with your new rules:

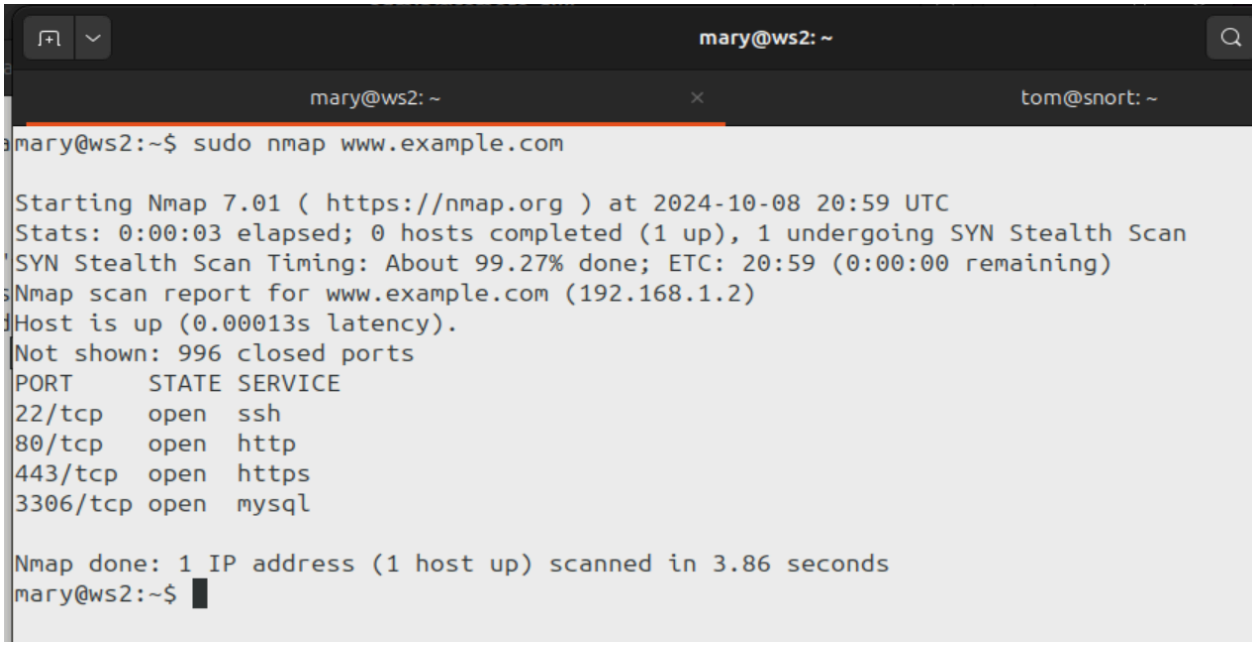
```
sudo /etc/rc.local
```

Now restart snort and again run nmap from mary's ws2 computer.

```
hank@remote_ws:~$ sudo nano /etc/rc.local
hank@remote_ws:~$ sudo nmap www.example.com

Starting Nmap 7.01 ( https://nmap.org ) at 2024-10-08 20:41 UTC
Nmap scan report for www.example.com (203.0.113.10)
Host is up (0.000036s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 1.58 seconds
```




```
hank@remote_ws:~$ sudo nano /etc/rc.local
hank@remote_ws:~$ sudo nmap www.example.com

Starting Nmap 7.01 ( https://nmap.org ) at 2024-10-08 20:41 UTC
Nmap scan report for www.example.com (203.0.113.10)
Host is up (0.000036s latency).
Not shown: 996 closed ports
PORT    STATE SERVICE
22/tcp  open  ssh
53/tcp  open  domain
80/tcp  open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 1.58 seconds
```



To duplicate traffic from the internal workstation (**ws2**), we modify the **iptables rules** on the gateway by editing the "**/etc/rc.local**" file. We add the rule to duplicate traffic from **ws2** using the **TEE target**: "**iptables -t mangle -A PREROUTING -i $lan2 -j TEE --gateway 192.168.3.1**".

After saving the file, we run the script with **"sudo /etc/rc.local"** to apply the changes. Finally, we restart Snort and run **nmap from ws2** to see the traffic alerts on the Snort console.

7) Distinguishing traffic by address

Start Firefox on mary's ws2 computer to view the confidential business plan:

```
firefox www.example.com/plan.html
```

Then observe the snort console. This will not do! The keen minds at the startup need to view their confidential business plan without IDS alerts firing off. But they do want to monitor internal computers for suspicious traffic, e.g., nmap scans. In this task, you will adjust your snort rule so that the CONFIDENTIAL alert only fires when the plan is accessed by addresses outside of the site.

If you review rules found in the `/etc/snort/rules` directory, you will see that rules have the general form of:

```
alert <protocol> <source_addr> <src_port> ->  \
      <dest_addr> <dest_port> <rule options in parens>
```

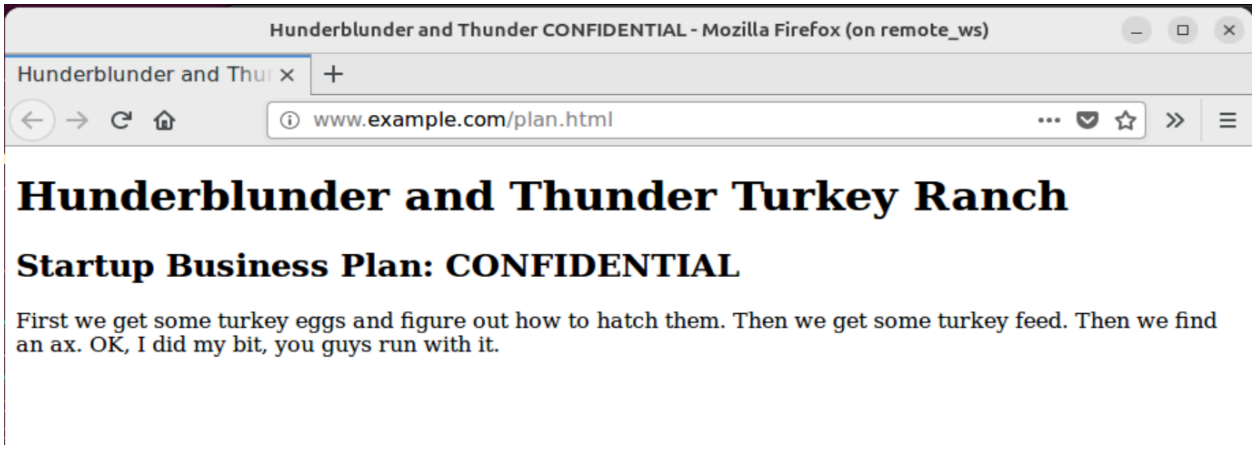
The snort rules include two address fields: `source_addr` and `dest_addr`. These addresses are used to check the source from which the packet originated and the destination of the packet. The address may be a single IP address or a network address. You likely have used the *any* keyword to apply a rule on all addresses. For network addresses, the address is followed by a slash character and number of bits in the netmask. For example, a network address of 192.168.2.0/24 represents C class network 192.168.2.0 with 24 bits in the network mask.

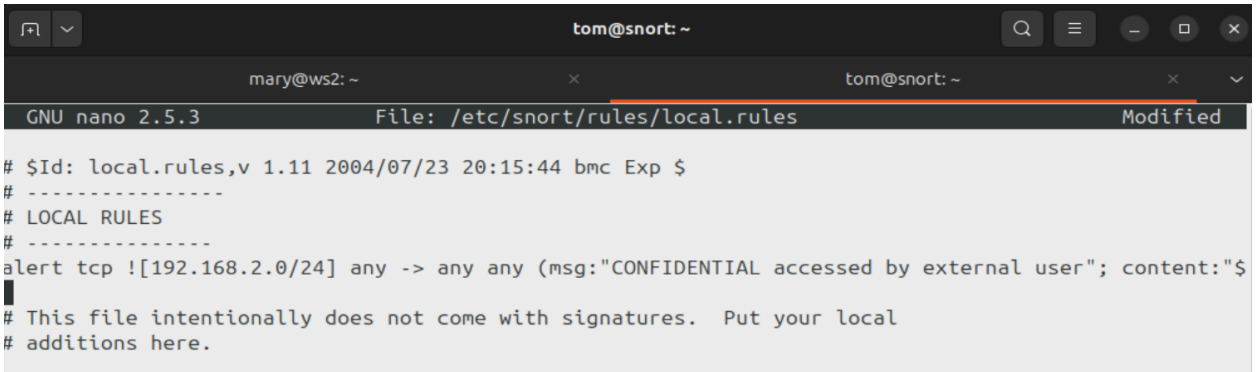
Note that as a result of our use of NAT, all traffic from the web server destined for an external address will have a destination address of the gateway, (i.e., 192.168.1.10), while web traffic destined for internal users will have destination addresses that match the internal user.

For this task, you must set your snort rules and traffic mirroring such that:

1. External access to the business plan generates an alert;
2. Internal access to the business plan does not generate an alert;
3. External or internal use of nmap will generate an ICMP NMAP PING alert.

Your must test each of these criteria during a single snort session, i.e., if you change a snort rule, or port mirroring, you must restart your tests.





We then adjusted our Snort rules to differentiate between internal and external access to the business plan by modifying the rule in “local.rules” to generate alerts only for external traffic. For example:
“**alert tcp ![192.168.2.0/24] any -> any any (msg:"CONFIDENTIAL accessed by external user"; content:"CONFIDENTIAL"; sid:1000002;)**”. This rule triggers an alert only when external users (**outside the 192.168.2.0/24 network**) access the confidential plan. We then tested the rule by accessing the plan from both internal and external stations, ensuring that only external connections trigger an alert.

5. Submission

After finishing the lab, go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab snort
```

When you stop the lab, the system will display a path to the zipped lab results on your Linux system. Provide that file to your instructor, e.g., via the Sakai site.

We finally stopped the labtainer with the “**stoplab**” command