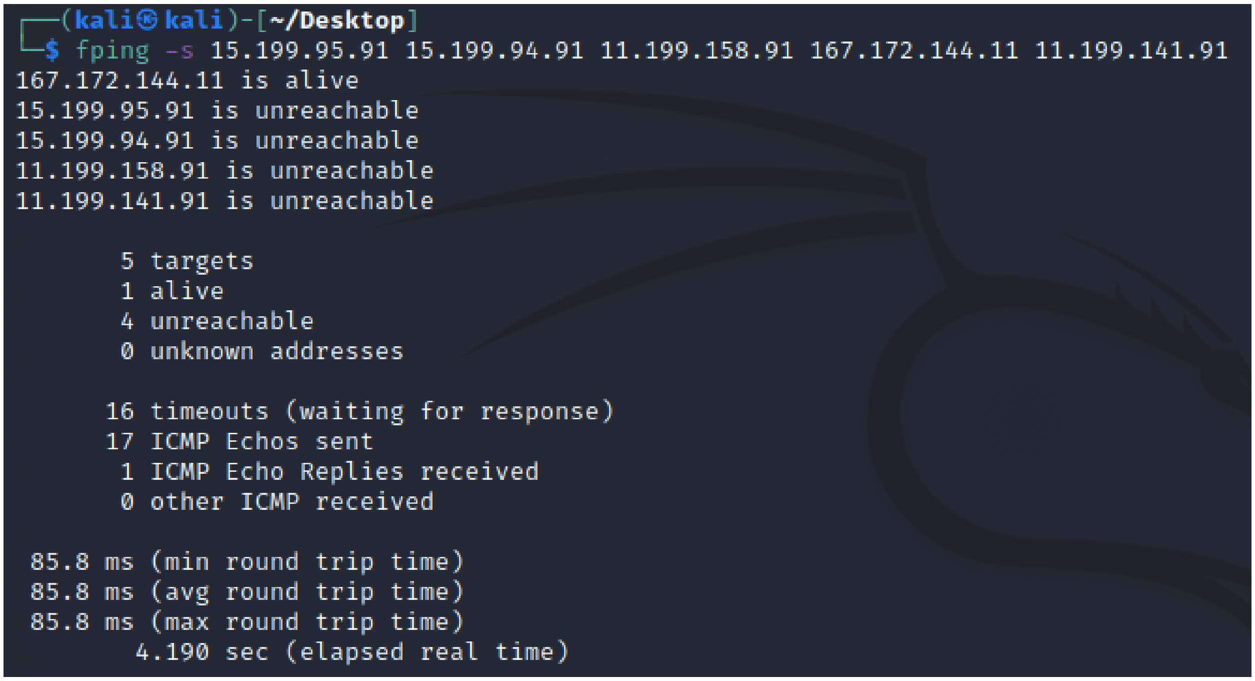
You will follow instructions to work through four phases of the network assessment. For each phase, include the following:

- The steps and commands used to complete the tasks.
- A summary of your findings for each testing phase.
- Any network vulnerabilities discovered.
- Findings associated with a hacker.
- Recommended mitigation strategy.
- Document the OSI layer where the findings were found.

Phase 1: *"I'd like to Teach the World to Ping"*

Determine the IPs for the Hollywood office and run fping against the IP ranges in order to determine which IP is accepting connections.

The related IP’s are listed in the excel sheet attached to assignment directions. We are going to utilize the “fping” command with the “s” flag (to pull up statistics) on all related IP’s that are associated with **Hollywood.** There are five IP addresses total.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ fping -s 15.199.95.91 15.199.94.91 11.199.158.91 167.172.144.11 11.199.141.91
167.172.144.11 is alive
15.199.95.91 is unreachable
15.199.94.91 is unreachable
11.199.158.91 is unreachable
11.199.141.91 is unreachable

       5 targets
       1 alive
       4 unreachable
       0 unknown addresses

      16 timeouts (waiting for response)
      17 ICMP Echos sent
       1 ICMP Echo Replies received
       0 other ICMP received

 85.8 ms (min round trip time)
 85.8 ms (avg round trip time)
 85.8 ms (max round trip time)
        4.190 sec (elapsed real time)
```

It looks like the ip 167.172.144.11 is the only one “alive” and accepting connections. This is a **vulnerability** since RockStar doesn’t want any IP accepting connections. Since “fping” uses ICMP type echoes to tell us if information is arriving at our destination IP. The caveat here is that anyone who can issue an ICMP request can know that the IP is responding and transmitting data.

The recommendation here then is to restrict ICMP echo requests at the IP address 167.172.144.11. Outright disabling ICMP can cause hindrances in how the network functions so that is not a viable option. Every other one of the Hollywood offices are not responding, other than 167.172.144.11, which is what we want

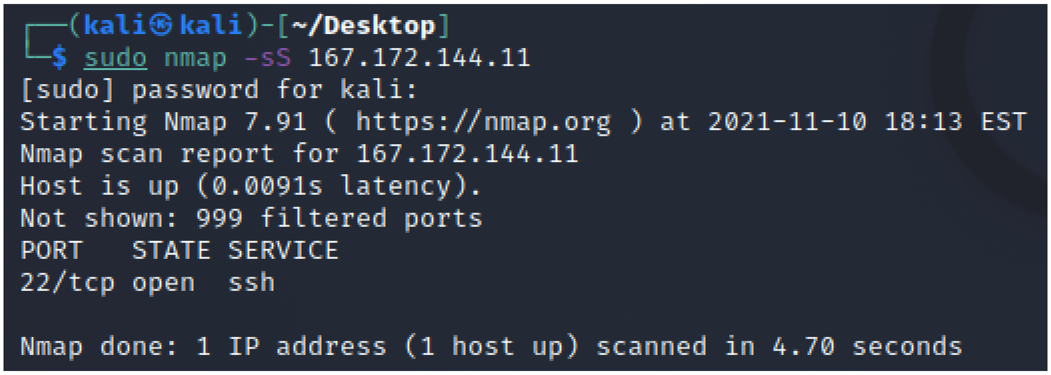
Since this is dealing with IP’s and Networks, this falls under OSI level 3: Network

Phase 2: *"Some Syn for Nothin`"*

With the IP(s) found from Phase 1, determine which ports are open:

- You will run a SYN SCAN against the IP accepting connections. See **SYN SCAN Instructions** below.
- Using the results of the SYN SCAN, determine which ports are accepting connections.
- Add these findings to the summary and be sure to indicate at which OSI layer your findings were found.

So now we’re going to identify open ports on our “167” address by running a SYN scan since SYN/ACK acts upon TCP ports. The actual command for this will be “nmap” with the flag “sS” which runs a specific TCP SYN scan.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo nmap -sS 167.172.144.11
[sudo] password for kali:
Starting Nmap 7.91 ( https://nmap.org ) at 2021-11-10 18:13 EST
Nmap scan report for 167.172.144.11
Host is up (0.0091s latency).
Not shown: 999 filtered ports
PORT   STATE SERVICE
22/tcp open  ssh

Nmap done: 1 IP address (1 host up) scanned in 4.70 seconds
```

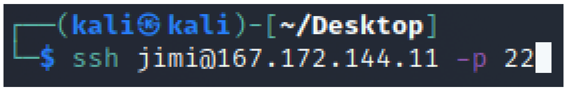
Looks like Port 22 TCP is open and accepting connections from ssh services. This is **not** good and a blaring vulnerability. Since TCP is connection focused, this is in the realms of **Transport**, **Layer 4 of the OSI Model.**

Phase 3: *"I Feel a DNS Change Comin' On"*

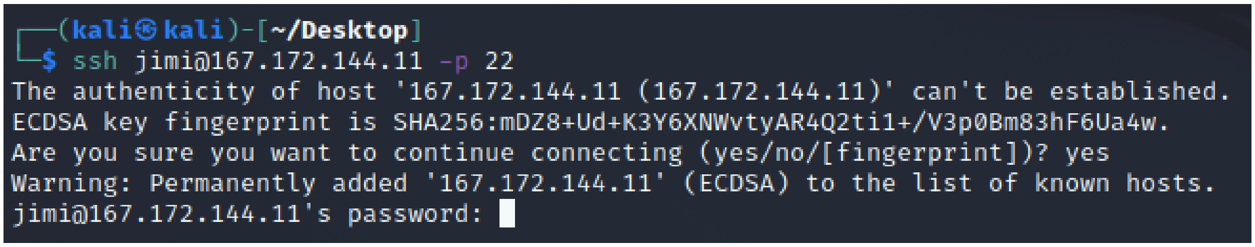
With your findings from Phase 2, determine if you can access the server that is accepting connections.

- RockStar typically uses the same default username and password for most of their servers, so try this first:
  - **Username:** jimi
  - **Password:** hendrix
- Try to figure out which port/service would be used for remote system administration, and then using these credentials, attempt to log into the IP that responded to pings from **Phase 1**.

We said the “ssh” service is a pretty significant vulnerability so we’re going to test that right now to see if we can access the server. The username provided is “jimi” so we’ll be using that as part of our ssh login with the associated IP and on port 22.

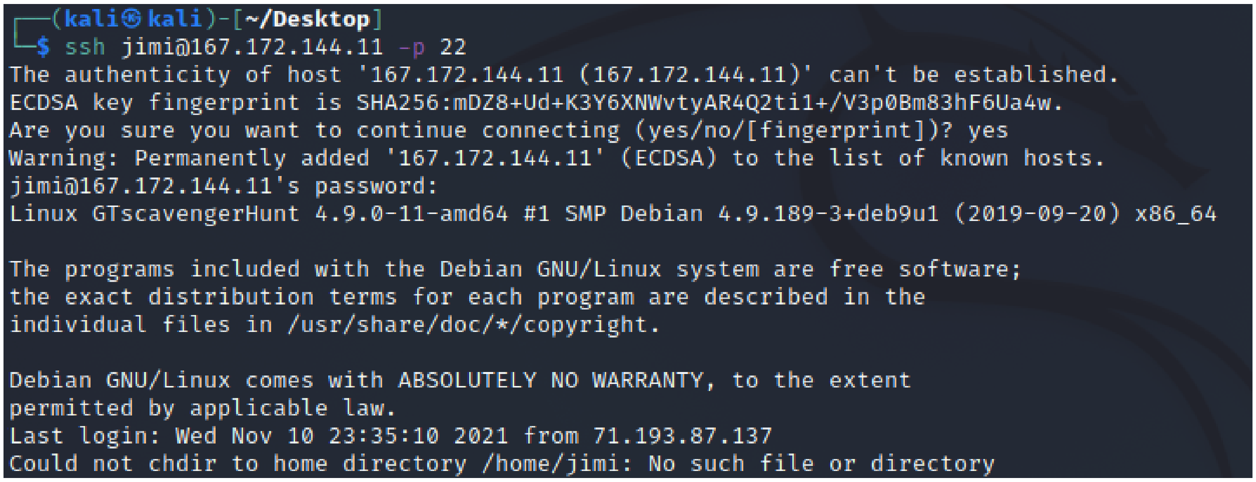


```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh jimi@167.172.144.11 -p 22
```



```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh jimi@167.172.144.11 -p 22
The authenticity of host '167.172.144.11 (167.172.144.11)' can't be established.
ECDSA key fingerprint is SHA256:mDZ8+Ud+K3Y6XNWvtyAR4Q2ti1+/V3p0Bm83hF6Ua4w.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '167.172.144.11' (ECDSA) to the list of known hosts.
jimi@167.172.144.11's password:
```

Here we use the password “hendrix” provided to us.

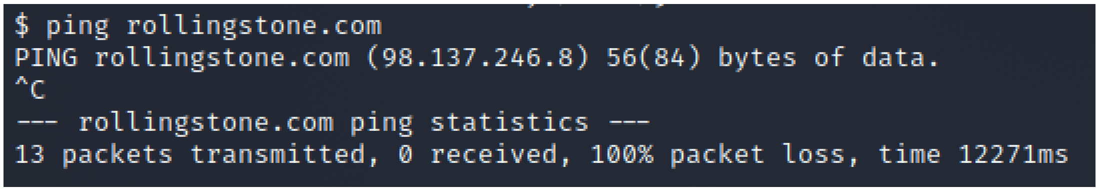


```
┌──(kali㉿kali)-[~/Desktop]
└─$ ssh jimi@167.172.144.11 -p 22
The authenticity of host '167.172.144.11 (167.172.144.11)' can't be established.
ECDSA key fingerprint is SHA256:mDZ8+Ud+K3Y6XNWvtyAR4Q2ti1+/V3p0Bm83hF6Ua4w.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '167.172.144.11' (ECDSA) to the list of known hosts.
jimi@167.172.144.11's password:
Linux GTscavengerHunt 4.9.0-11-amd64 #1 SMP Debian 4.9.189-3+deb9u1 (2019-09-20) x86_64

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Nov 10 23:35:10 2021 from 71.193.87.137
Could not chdir to home directory /home/jimi: No such file or directory
```

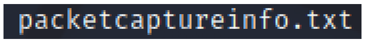
And we're in.



```
$ ping rollingstone.com
PING rollingstone.com (98.137.246.8) 56(84) bytes of data.
^C
--- rollingstone.com ping statistics ---
13 packets transmitted, 0 received, 100% packet loss, time 12271ms
```

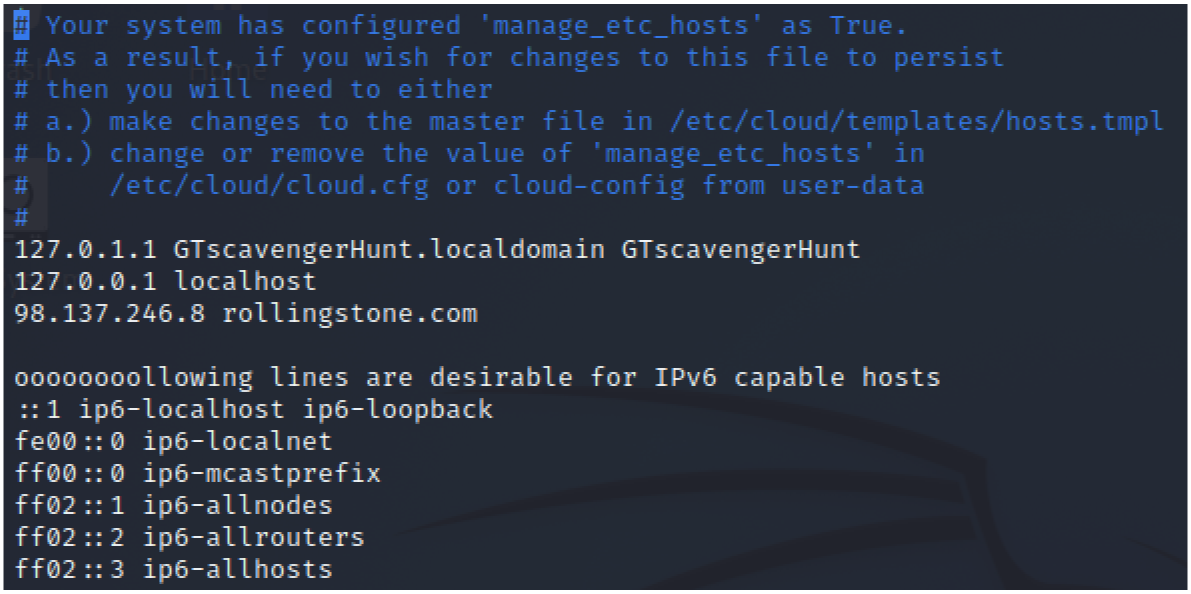
Next, we're going to ping the website "rollingstone.com" as the corporation is having difficulties accessing the website. Interestingly, no packets were received. Ping uses an ICMP echo which, as we discussed earlier, tells us if our destination is responding and transmitting data.

Now, the information we're looking for to find this IP information will be under etc/hosts and we're going to read that file. But there's something interesting in the etc directory as well.



```
packetcaptureinfo.txt
```

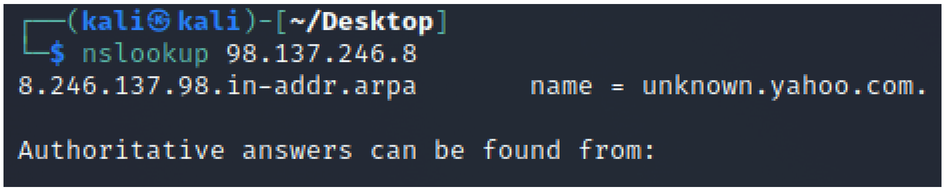
This isn't something commonly found in the etc directory. Let's hold onto its location for now.



```
# Your system has configured 'manage_etc_hosts' as True.
# As a result, if you wish for changes to this file to persist
# then you will need to either
# a.) make changes to the master file in /etc/cloud/templates/hosts.tmpl
# b.) change or remove the value of 'manage_etc_hosts' in
#     /etc/cloud/cloud.cfg or cloud-config from user-data
#
127.0.1.1 GTscavengerHunt.localdomain GTscavengerHunt
127.0.0.1 localhost
98.137.246.8 rollingstone.com

ooooooooollowing lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
ff02::3 ip6-allhosts
```

There is the IP entry for “rollingstone.com”, interesting. Now for the next part we’re going to be using the “nslookup” command which will help us identify the real domain of the IP address we found for rollingstone.com: 98.137.246.8



```
┌──(kali㉿kali)-[~/Desktop]
└─$ nslookup 98.137.246.8
8.246.137.98.in-addr.arpa       name = unknown.yahoo.com.

Authoritative answers can be found from:
```

Since we’re working in the realms of web addresses that the user can put in as well as DNS poisoning techniques, this interaction works at **Layer 7: Application** of the OSI model

Phase 4: *"ShARP Dressed Man"*

Within the RockStar server that you SSH'd into, and in the same directory as the configuration file from **Phase 3**, the hacker left a note as to where he stored away some packet captures.

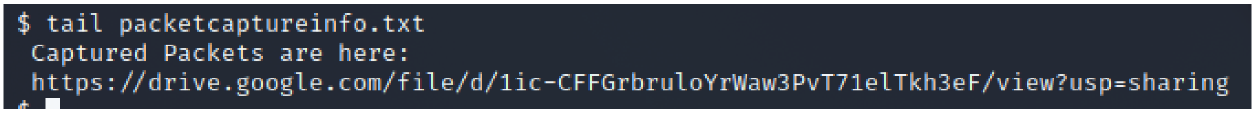
- View the file to find where to recover the packet captures.
- These are packets that were captured from the activity in the Hollywood Office.
- Use Wireshark to analyze this pcap file and determine if there was any suspicious activity that could be attributed to a hacker.
    - **Hint**: Focus on the ARP and HTTP protocols. Recall the different types of HTTP request methods and be sure to thoroughly examine the contents of these packets.

- Add your findings in your summary and be sure to indicate at which OSI layer they were found.

Let’s ssh back into the user we were previously at. Now, we’re looking for a specific txt file that was left behind. Luckily we caught one in the etc directory so let’s start there.

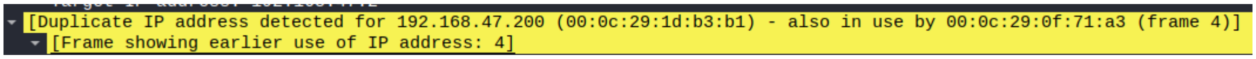
```
packetcaptureinfo.txt
```

We’re going to read this with a tail command.



```
$ tail packetcaptureinfo.txt
 Captured Packets are here:
 https://drive.google.com/file/d/1ic-CFFGrbruloYrWaw3PvT71elTkh3eF/view?usp=sharing
```

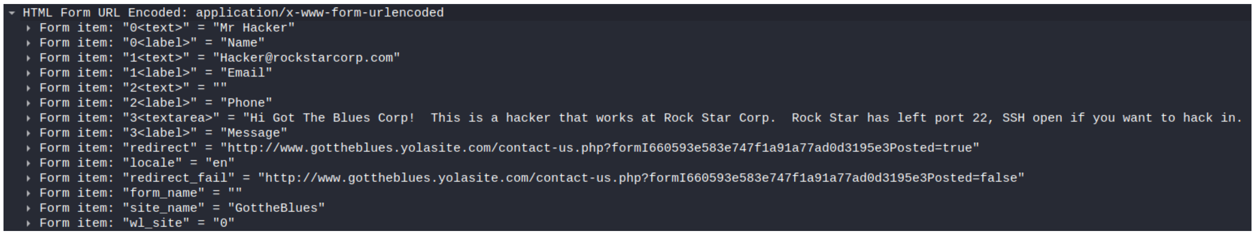
And look at that. Now let’s grab this file. Kali lets us open the link directly from the command line. Downloading the file lets us open it directly with wireshark.



```
▾ [Duplicate IP address detected for 192.168.47.200 (00:0c:29:1d:b3:b1) - also in use by 00:0c:29:0f:71:a3 (frame 4)]
  ▾ [Frame showing earlier use of IP address: 4]
```

Scanning through the ARP logs. We find this. Duplicate IP addresses for the Mac address ending in b1. This means that this specific Mac address is likely spoofed. What about the original? Wireshark tells us that it is also used by the MAC address ending in a3 which appears to be our hacker’s non-spoofed MAC address.

Now to examine our HTTP packets. Here’s something interesting. There’s a “POST” packet usually meaning some kind of online posting so let’s start there. As we dig a little deeper into the HTML form we can see the messages.



```
▾ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▸ Form item: "0<text>" = "Mr Hacker"
  ▸ Form item: "0<label>" = "Name"
  ▸ Form item: "1<text>" = "Hacker@rockstarcorp.com"
  ▸ Form item: "1<label>" = "Email"
  ▸ Form item: "2<text>" = ""
  ▸ Form item: "2<label>" = "Phone"
  ▸ Form item: "3<textarea>" = "Hi Got The Blues Corp!  This is a hacker that works at Rock Star Corp.  Rock Star has left port 22, SSH open if you want to hack in.
  ▸ Form item: "3<label>" = "Message"
  ▸ Form item: "redirect" = "http://www.gottheblues.yolasite.com/contact-us.php?formI660593e583e747f1a91a77ad0d3195e3Posted=true"
  ▸ Form item: "locale" = "en"
  ▸ Form item: "redirect_fail" = "http://www.gottheblues.yolasite.com/contact-us.php?formI660593e583e747f1a91a77ad0d3195e3Posted=false"
  ▸ Form item: "form_name" = ""
  ▸ Form item: "site_name" = "GottheBlues"
  ▸ Form item: "wl_site" = "0"
```

Well well well, Looks like we found whoever left port 22 open. And they’re an insider. We found our overall vulnerability and evidence against them.

Since the input for all this occurs at the website level, this is a **Layer 7: Application** usage of the OSI model.