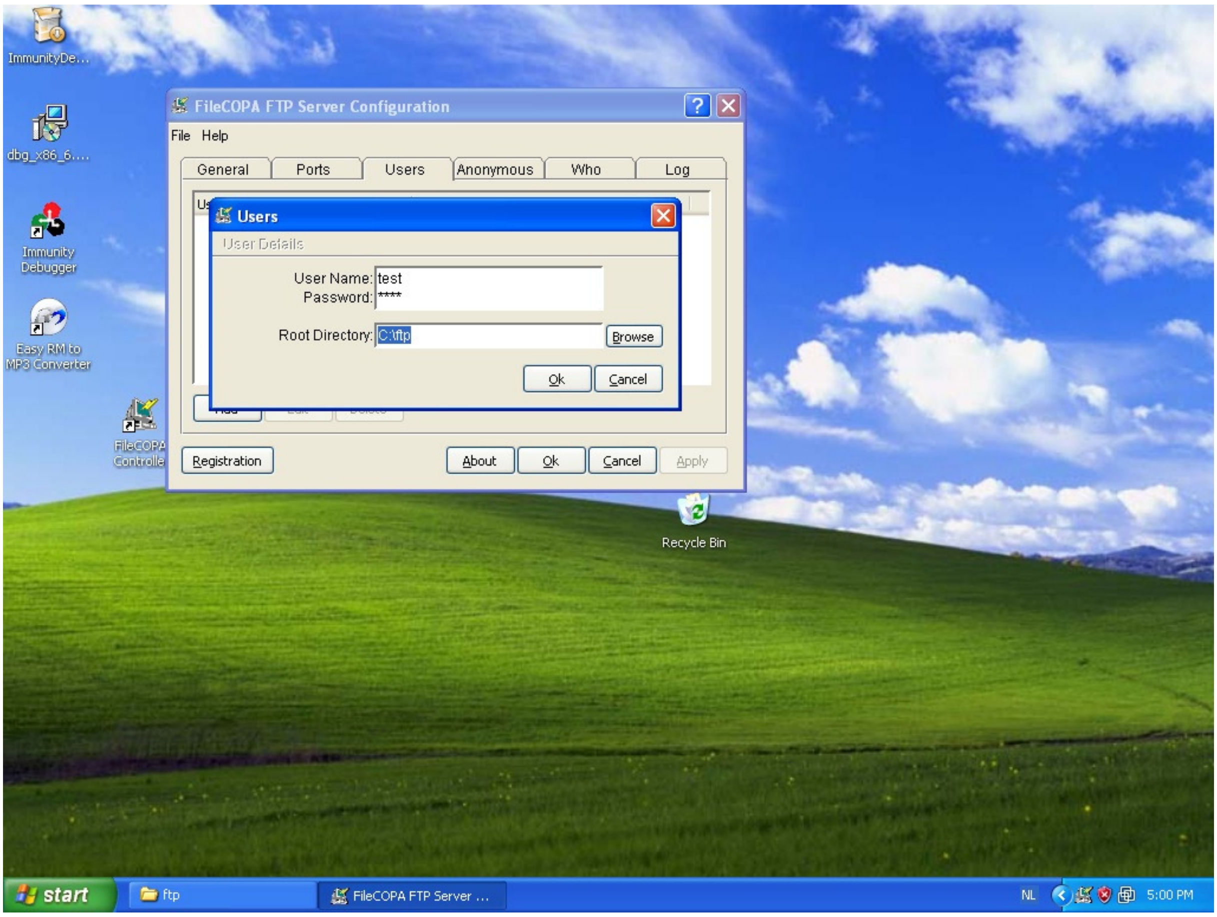
Discovering and exploiting a remote buffer overflow vulnerability in an FTP server – PART 1

Hello all, in this tutorial we will learn how to identify a vulnerability in an FTP server through the process of "**Fuzzing**" which could lead to a **DoS** or **Buffer Overflow** vulnerability identification. In this specific part we will use FTPFuzz to crash FileCOPA and identify a vulnerability in the LIST command.

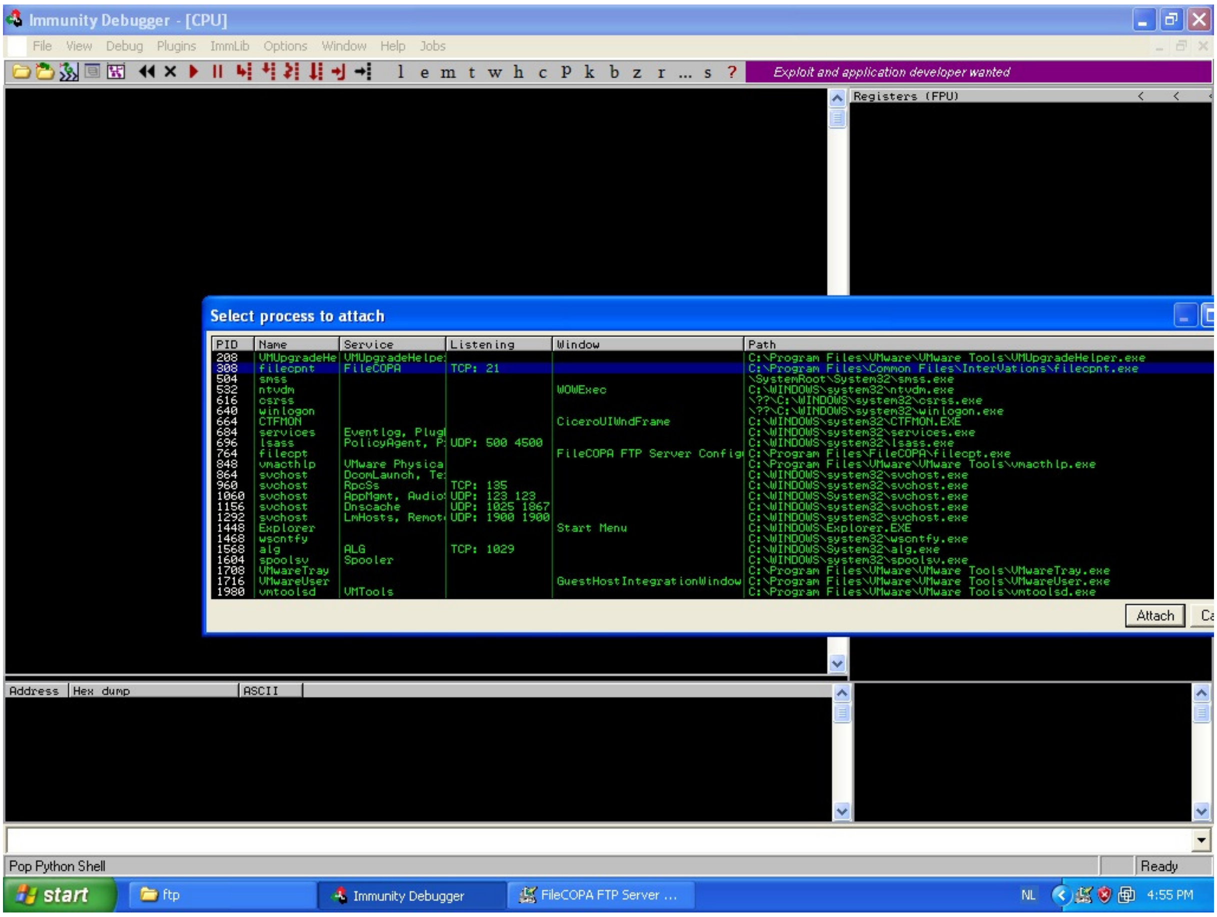
First of all to get started, everything you will need to follow along this tutorial is downloadable in this .rar: http://sharingmatrix.com/file/713096/FileCOPAtutorial.rar

1) Set up **FileCOPA** and add a user with username "test" and password "test" (or whatever you want the username and password to be) and set the root folder to anything you like (here I used c:\ftp).



User Name: test
Password: ****
Root Directory: C:\ftp

2) Just leave the server running, open **Immunity Debugger** and attach it to the process and press the **play** button to let it run.

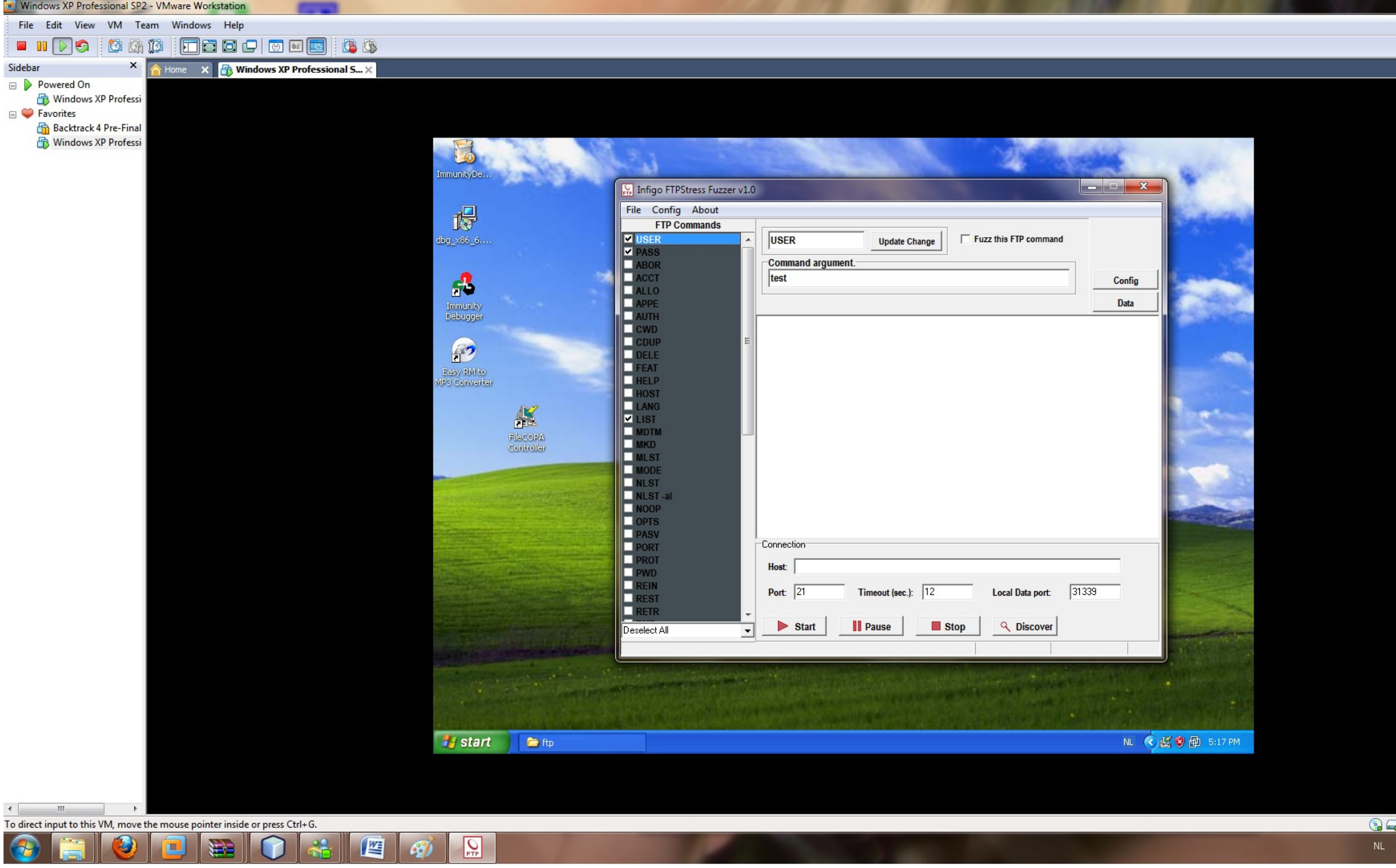

Immunity Debugger - [CPU]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Exploit and application developer wanted
Registers (FPU)
Select process to attach
Attach
Ca
Address
Hex dump
ASCII
Pop Python Shell
Ready
start
ftp
Immunity Debugger
FileCOPA FTP Server ...
NL
4:55 PM

PID	Name	Service	Listening	Window	Path
208	VMUpgradeHe	VMUpgradeHelpe:			C:\Program Files\VMware\VMware Tools\VMUpgradeHelper.exe
308	filecpnt	FileCOPA	TCP: 21		C:\Program Files\Common Files\InterVations\filecpnt.exe
504	smss				\SystemRoot\System32\smss.exe
532	ntvdm			WOWExec	C:\WINDOWS\system32\ntvdm.exe
616	csrss				\??\C:\WINDOWS\system32\csrss.exe
640	winlogon				\??\C:\WINDOWS\system32\winlogon.exe
664	CTFMON			CiceroUIWndFrame	C:\WINDOWS\system32\CTFMON.EXE
684	services	Eventlog, Plugl			C:\WINDOWS\system32\services.exe
696	lsass	PolicyAgent, P:	UDP: 500 4500		C:\WINDOWS\system32\lsass.exe
764	filecpt			FileCOPA FTP Server Config	C:\Program Files\FileCOPA\filecpt.exe
848	vmacthlp	VMware Physica			C:\Program Files\VMware\VMware Tools\vmacthlp.exe
864	svchost	DcomLaunch, Te:			C:\WINDOWS\system32\svchost.exe
960	svchost	RpcSs	TCP: 135		C:\WINDOWS\system32\svchost.exe
1060	svchost	AppMgmt, Audio:	UDP: 123 123		C:\WINDOWS\System32\svchost.exe
1156	svchost	Dnscache	UDP: 1025 1867		C:\WINDOWS\system32\svchost.exe
1292	svchost	LmHosts, Remot	UDP: 1900 1900		C:\WINDOWS\system32\svchost.exe
1448	Explorer			Start Menu	C:\WINDOWS\Explorer.EXE
1468	wscntfy				C:\WINDOWS\system32\wscntfy.exe
1568	alg	ALG	TCP: 1029		C:\WINDOWS\System32\alg.exe
1604	spoolsv	Spooler			C:\WINDOWS\system32\spoolsv.exe
1708	VMwareTray				C:\Program Files\VMware\VMware Tools\VMwareTray.exe
1716	VMwareUser			GuestHostIntegrationWindow	C:\Program Files\VMware\VMware Tools\VMwareUser.exe
1980	vmtoolsd	VMTools			C:\Program Files\VMware\VMware Tools\vmtoolsd.exe

3) Open **FTPFuzz** and enter the host. Now press “Discover” to discover what FTP commands are available on this server. FTPFuzz will automaticly edit this.

*Note: if this were a real-life test on a server with no known vulnerabilities, we would leave it as it is after FTPfuzz discovered the commands for us, but for this example I’ll only leave on “**USER**” “**PASS**” and “**LIST**” (the vulnerability is in **LIST**. **USER** and **PASS** are just to login on the test account).*

4) Go to **USER** and **PASS** and edit the arguments to fit the username and pass you specified when setting up the server (in my case this is just test:test).



FTP Commands

☑ USER
☑ PASS
□ ABOR
□ ACCT
□ ALLO
□ APPE
□ AUTH
□ CWD
□ CDUP
□ DELE
□ FEAT
□ HELP
□ HOST
□ LANG
☑ LIST
□ MDTM
□ MKD
□ MLST
□ MODE
□ NLST
□ NLST -al
□ NOOP
□ OPTS
□ PASV
□ PORT
□ PROT
□ PWD
□ REIN
□ REST
□ RETR

Deselect All

USER

□ Fuzz this FTP command

Command argument: test

Host:

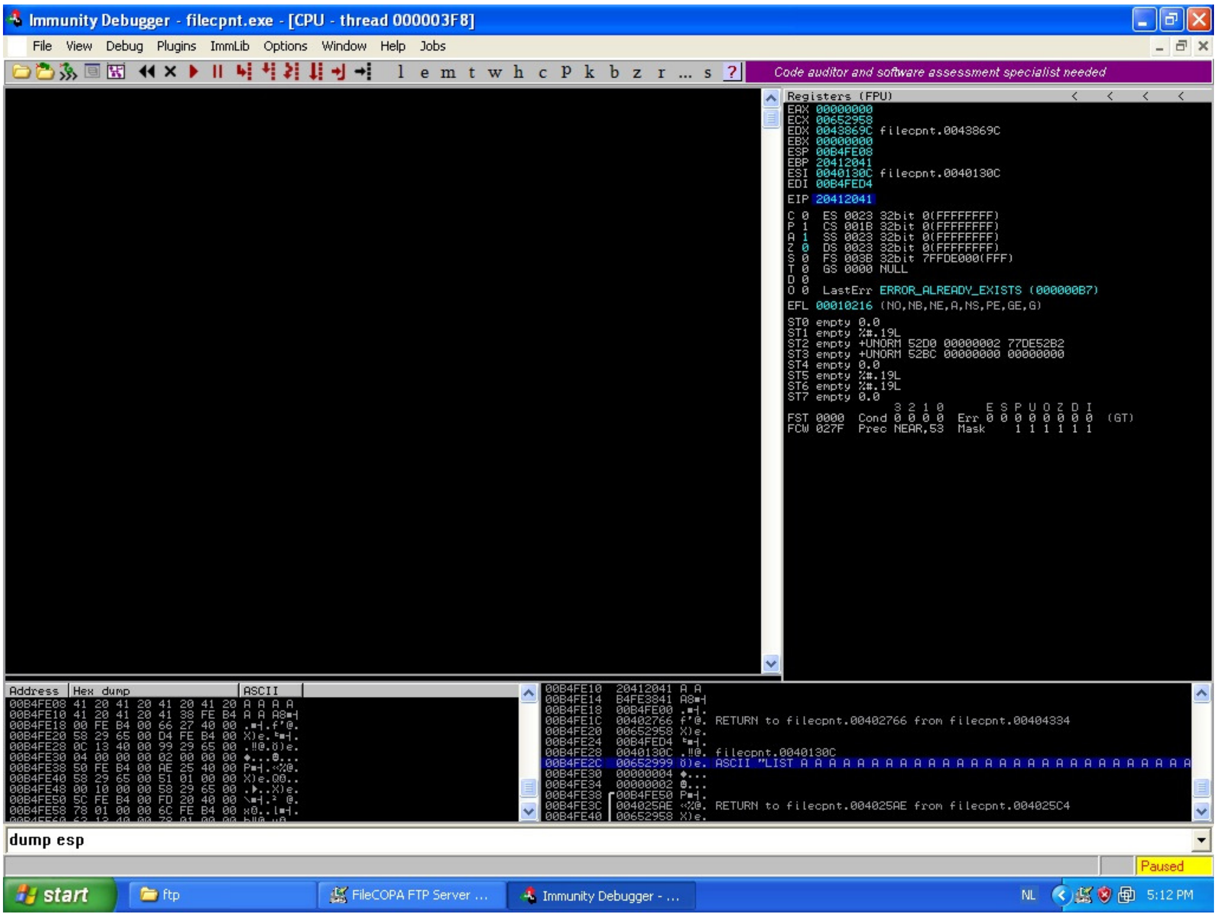
Port: 21 Timeout (sec.): 12 Local Data port: 31339

5) Let **FTPFuzz** run and wait for **FileCOPA** to crash… and jup… it crashes!

6) Inside Immunity Debugger we can now see the **EIP** and **EBP** have been overwritten with **0×20412041** (which is "**A A** " — note the spaces — in reverse).

We have now identified a vulnerability when using the LIST command, followed many times "A " (\x20\x41)

Take a moment to examine the following screenshot closely:


Immunity Debugger - filecpnt.exe - [CPU - thread 000003F8]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Code auditor and software assessment specialist needed
Paused
start
ftp
FileCOPA FTP Server ...
Immunity Debugger - ...
NL
5:12 PM

```
Registers (FPU)
EAX 00000000
ECX 00652958
EDX 0043869C filecpnt.0043869C
EBX 00000000
ESP 00B4FE08
EBP 20412041
ESI 0040130C filecpnt.0040130C
EDI 00B4FED4

EIP 20412041

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 003B 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_ALREADY_EXISTS (000000B7)

EFL 00010216 (NO,NB,NE,A,NS,PE,GE,G)

ST0 empty 0.0
ST1 empty %#.19L
ST2 empty +UNORM 52D0 00000002 77DE52B2
ST3 empty +UNORM 52BC 00000000 00000000
ST4 empty 0.0
ST5 empty %#.19L
ST6 empty %#.19L
ST7 empty 0.0
               3 2 1 0      E S P U O Z D I
FST 0000  Cond 0 0 0 0  Err 0 0 0 0 0 0 0 0  (GT)
FCW 027F  Prec NEAR,53  Mask    1 1 1 1 1 1
```

```
Address  Hex dump                 ASCII
00B4FE08 41 20 41 20 41 20 41 20  A A A A
00B4FE10 41 20 41 20 41 38 FE B4  A A A8■┤
00B4FE18 00 FE B4 00 66 27 40 00  .■┤.f'@.
00B4FE20 58 29 65 00 D4 FE B4 00  X)e.╘■┤.
00B4FE28 0C 13 40 00 99 29 65 00  .‼@.Ö)e.
00B4FE30 04 00 00 00 02 00 00 00  ♦...☻...
00B4FE38 50 FE B4 00 AE 25 40 00  P■┤.«%@.
00B4FE40 58 29 65 00 51 01 00 00  X)e.Q☺..
00B4FE48 00 10 00 00 58 29 65 00  .►..X)e.
00B4FE50 5C FE B4 00 FD 20 40 00  \■┤.² @.
00B4FE58 78 01 00 00 6C FE B4 00  x☺..l■┤.
```

```
00B4FE10   20412041 A A
00B4FE14   B4FE3841 A8■┤
00B4FE18   00B4FE00 .■┤.
00B4FE1C   00402766 f'@.  RETURN to filecpnt.00402766 from filecpnt.00404334
00B4FE20   00652958 X)e.
00B4FE24   00B4FED4 ╘■┤.
00B4FE28   0040130C .‼@.  filecpnt.0040130C
00B4FE2C   00652999 Ö)e.  ASCII "LIST A A A A A A A A A A A A A A A A A A A A A A A A A A A A A
00B4FE30   00000004 ♦...
00B4FE34   00000002 ☻...
00B4FE38  ┌00B4FE50 P■┤.
00B4FE3C  │004025AE «%@.  RETURN to filecpnt.004025AE from filecpnt.004025C4
00B4FE40  │00652958 X)e.
```

```
dump esp
```

I hope you enjoyed this tutorial and that it has given you some insight on how to discover DoS or buffer overflow vulnerabilities.
In the next part, we will look at how we can exploit this vulnerability to execute remote shellcode.

Diggs, StumbleUpon's, bookmarks at Delicious, … are always appreciated!

Many thanks go out to corelanc0d3r for his great tips.

Don't forget to check out my twitter for more updates: http://twitter.com/Raykoid666

Original tutorial at my blog: http://raykoid666.wordpress.com