TRABAJO PRÁCTICO FINAL

Laboratorio de sistemas operativos y computadoras

METASPLOIT FRAMEWORK

Axel Lopez Garabal

ÍNDICE

DEFINICIONES PREVIAS

METASPLOIT

Módulos
Interfaces de Metasploit
Requerimientos para la instalación
Instalación de Metasploit(Windows)
Instalación de Metasploit(Linux)

MSFCONSOLE

Obteniendo ayuda.
Completación con tab.
Comando "show".
Comando "search".
Comando "info".
Comando "use".
Comando "connect".
Comando "set".
Comando "check".
Configuración de variables globales.
Comando "exploit/run".
Comando "back".
Comando "resource".
Comando "irb".

Escaneo de Puertos

ERRORES EN METASPLOIT

DEFINICIONES PREVIAS

BUG

Es un fallo de programación durante el proceso de de creación o desarrollo de las aplicaciones(por lo general ocurre durante la etapa de implementación).

EXPLOIT

Es código con el fin de aprovechar un error de programación y obtener diversos privilegios(el atacante busca tomar el control remoto de una máquina y escalar en privilegios)

PAYLOAD

Es la parte de un exploit que tiene como objetivo ejecutarse en la máquina víctima para realizar la acción maliciosa(también puede ser ejecutar desde una máquina remota un conjunto de instrucciones sobre la máquina víctima)

SHELLCODE

Es un conjunto de instrucciones usadas como un payload cuando se produce el proceso de explotación del sistema. Los shellcodes son órdenes, generalmente, escritos en lenguaje Assembler. Por lo general se usa C luego es código se compila y genera se genera la llamado opcode (que es el código máquina). Los shellcodes deben ser de tamaño pequeño para ser inyectados en la pila.

METASPLOIT

Es el nombre que recibe el proyecto, *open source*, sobre la seguridad informática. Este proyecto facilita el trabajo del auditor proporcionando información sobre las vulnerabilidades de seguridad, ayudando a explotarlas en los procesos de pentesting o test de intrusión. El subproyecto más famoso que dispone es ***METASPLOIT framework*** o simplemente *Metasploit*. Originalmente desarrollado en Perl para que con el paso del tiempo fuera escrito de nuevo en Ruby.

Es un conjunto de herramientas con las que el auditor puede desarrollar o ejecutar exploits y lanzarlos contra un maquinas para comprobar la seguridad de estas. Otras de las funcionalidades que aporta es un archivo de shellcodes, herramientas de recolección de información y escanear en búsqueda de

vulnerabilidades .

Módulos

Metasploit dispone de módulos los cuales ayudan a aumentar de manera sencilla las funcionalidades del framework. Un módulo es una pieza o bloque de código que implementa una o varias funcionalidades, como puede ser la ejecución de un exploit concreto o la realización de un escaneo sobre máquinas remotas. Los módulos que componen el framework son el núcleo de Metasploit y los que hacen que sea tan poderoso. Estos módulos pueden ser desarrollados por los usuarios y de esta manera ampliar el framework de manera personalizada(en función de las necesidades del auditor).

Interfaces de Metasploit

Existen varias interfaces con las que se puede utilizar con el framework. El usuario puede interactuar mediante una interfaz gráfica o consola. msfconsole el usuario dispone de una consola desde la cual puede acceder a todas las opciones disponibles de Metasploit. Armitage: el entorno gráfico e intuitivo le proporciona al usuario un entorno gráfico e intuitivo para llevar a cabo los test de intrusión. Metasploit (web UI) : con esta interfaz se puede gestionar el test de intrusión de manera remota, sin necesidad de disponer del framework de forma local.

Requerimientos

MINIMUM HARDWARE

- 2 GHz+ processor
- 4 GB RAM available (8 GB recommended)
- 1 GB available disk space (50 GB recommended)

OPERATING SYSTEMS

64-bit versions of the following platforms are supported.

- Ubuntu Linux 14.04 or 16.04 LTS (RECOMMENDED)

- Microsoft Windows Server 2008 R2
- Microsoft Windows Server 2012 R2
- Microsoft Windows 10
- Microsoft Windows 8.1
- Microsoft Windows 7 SP1+
- Red Hat Enterprise Linux Server 7.1 or later
- Red Hat Enterprise Linux Server 6.5 or later
- Red Hat Enterprise Linux Server 5.10 or later

BROWSERS

- Google Chrome (latest)
- Mozilla Firefox (latest)
- Microsoft Internet Explorer 11

Instalación de Metasploit(Windows)

La instalación en windows se realiza descargando la última versión o alguna de las versiones viejas del instalador. Para instalar simplemente se descarga un paquete de extensión msi, además del paquete se recomienda que antes de ejecutar el paquete se ajuste el antivirus para que ignore la ruta c:\metasploit-framework, luego doble-click para que se instale. La interfaz de msfconsole y todas las herramientas son agregadas como variables de entorno.

Algunas herramientas y exploits que posee el framework son idénticos o muy similares a algunas herramientas maliciosas (que pueden ser usadas con fines nefastos) son usualmente señaladas y removidas automáticamente por los antivirus como los malware que imitan

Instalación de Metasploit(Linux)

The easiest way to get the Metasploit Framework is to download the installer from the Rapid7 site. Visit http://www.rapid7.com/products/metasploit/download.jsp to find and download the installer for your operating system.
The installer provides a self-contained environment for you to run and update the Metasploit Framework. This means that all the necessary dependencies are

installed and configured for you during the installation process. If you prefer to install the dependencies manually, and configure the Metasploit Framework to use those dependencies, read https://kb.help.rapid7.com/docs/installing-the-metasploit-framework-on-ubuntu-linux

The easiest way of installing Metasploit Framework on Ubuntu 18.04 / Debian 9 is from the Metasploit installer. This installer ships with all the dependencies and tools required to run the Metasploit Framework.
Download Metasploit installer by running the commands below in your terminal.

```
curl
https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb > msfinstall && chmod 755 msfinstall && ./msfinstall
```

After the installation completes, open a terminal window and type the following to start msfconsole:

```
$ ./msfconsole
```

The prompt asks you if you want to use and set up a new database. Type y or yes to run the initial configuration script to create the initial database.

```
Creating database at /Users/joesmith/.msf4/db
Starting Postgresql
Creating database users
Creating initial database schema

** Metasploit Framework Initial Setup Complete **

[*] Starting the Metasploit Framework console...-[*] The initial module cache will be built in the background, this can take 2-5 minutes...
/

Metasploit Park, System Security Interface
Versión 4.0.5, Alpha E
Ready...
> access security
access: PERMISSION DENIED.
> access main security grid
access: PERMISSION DENIED....and...
YOU DIDN'T SAY THE MAGIC WORD!
YOU DIDN'T SAY THE MAGIC WORD!
=[ metasploit v4.11.0-dev [core:4.11.0.pre.dev api:1.0.0]]
+ -- --=[ 1454 exploits - 827 auxiliary - 229 post ]
+ -- --=[ 376 payloads - 37 encoders - 8 nops ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
msf >
```

To check to see if the database was set up, run the following command:

```
$ db_status
```

If the Metasploit Framework successfully connected to the database, the following status displays:

[*] postgresql connected to msf

MSFCONSOLE

- Es la única manera soportada para acceder a la mayoría de las funciones dentro de Metasploit.
- Proporciona una interfaz basada en consola al Framework.
- Contiene la mayoría de las características y es la interfaz más estable de MSF.
- Soporte completo de edición de líneas (readline), tabulación, y completación de comandos.
- Es posible la ejecución de comandos externos a través de msfconsole.

```
msf > ping -c 1 192.168.1.2
[*] exec: ping -c 1 192.168.1.2

PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
64 bytes from 192.168.1.2: icmp_seq=1 ttl=128 time=10.3 ms

--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 10.308/10.308/10.308/0.000 ms
msf >
```

Obteniendo ayuda.

Escribiendo "help" o "?" en el command prompt de msf, mostrará una lista de los comandos disponibles junto una descripción de su uso.

```
msf > help

Core Commands
=============

        Command       Description
        -------       -----------
        ?             Help menu
        back          Move back from the current context
        banner        Display an awesome metasploit banner
        cd            Change the current working directory
        connect       Communicate with a host
        exit          Exit the console
        help          Help menu
        info          Displays information about one or more module
        irb           Drop into irb scripting mode
        jobs          Displays and manages jobs
        load          Load a framework plugin
        loadpath      Searches for and loads modules from a path
```

```
    quit      Exit the console
    resource  Run the commands stored in a file
...snip...
```

Completación con tab.

Una de las características más útiles de msfconsole es completación con tab. Con la gran variedad de módulos disponibles, puede ser difícil recordar el nombre exacto y ruta de algún módulo en particular que se quiera usar. Como en la mayoría de los shells, escribiendo lo que sabes y presionando "Tab" le mostrará una lista de las opciones disponibles o se auto completará la palabra si hay una sola opción.

```
msf > use exploit/windows/smb/ms
use exploit/windows/smb/ms03_049_netapi
use exploit/windows/smb/ms04_007_killbill
use exploit/windows/smb/ms04_011_lsass
use exploit/windows/smb/ms04_031_netdde
use exploit/windows/smb/ms05_039_pnp
use exploit/windows/smb/ms06_025_rasmans_reg
use exploit/windows/smb/ms06_025_rras
use exploit/windows/smb/ms06_040_netapi
use exploit/windows/smb/ms06_066_nwapi
use exploit/windows/smb/ms06_066_nwwks
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/smb/msdns_zonename
msf > use exploit/windows/smb/ms08_067_netapi
```

Comando "show".

Introduciendo "show" en el prompt de msfconsole le mostrara cada módulo disponible en Metasploit.

```
msf > show

Encoders
========

   Name              Description
   ----              -----------
   cmd/generic_sh          Generic Shell Variable Substitution Command Encoder
   generic/none      The "none" Encoder
   mipsbe/longxor          XOR Encoder
...snip...
```

Hay una variedad de comandos "show" que puede utilizar, pero los que va a utilizar con más frecuencia son "show auxiliary", "show exploits" y "show payloads".

Ejecutando "show auxiliary" mostrará una lista de todos los módulos auxiliares disponibles en Metasploit. El modules Auxiliary incluye scanners, módulos de denial of service, fuzzers y más.

```
msf > show auxiliary
```

```
Auxiliary
=========
        Name                            Description
        ----                            -----------
        admin/backupexec/dump                   Veritas Backup Exec Windows Remote
File Access
        admin/backupexec/registry               Veritas Backup Exec Server Registry
Access
        admin/cisco/ios_http_auth_bypass        Cisco IOS HTTP Unauthorized
Administrative Access
...snip...
```

Naturalmente, "show exploits" será el comando más interesante de ser ejecutado ya que es el núcleo, Metasploit es todo sobre explotación. Use "show exploits" para obtener una lista de todos los exploits contenidos en el framework.

```
msf > show exploits

Exploits
========
        Name                            Description
        ----                            -----------
        aix/rpc_ttdbserverd_realpath            ToolTalk rpc.ttdbserverd
_tt_internal_realpath Buffer Overflow
        bsdi/softcart/mercantec_softcart        Mercantec SoftCart CGI Overflow

...snip...
```

Ejecutando "show payloads" mostrará todos los diferentes payloads para todas las plataformas disponibles en Metasploit.

```
msf > show payloads

Payloads
========
        Name                        Description
        ----                    -----------
        aix/ppc/shell_bind_tcp      AIX Command Shell, Bind TCP Inline
        aix/ppc/shell_find_port     AIX Command Shell, Find Port Inline
        aix/ppc/shell_reverse_tcp           AIX Command Shell, Reverse TCP Inline
...snip...
```

Así como ve, hay muchos payloads disponibles. Afortunadamente cuando está usando determinado exploit, usando "show payloads" solo mostrará los payloads que son compatibles para ese particular exploit. Por ejemplo, si es un exploit para Windows, no se mostrarán los payloads para Linux.

```
msf exploit(ms08_067_netapi) > show payloads

Compatible payloads
===================

  Name                              Description
  ----                          -----------
  generic/debug_trap                Generic x86 Debug Trap
  generic/debug_trap/bind_ipv6_tcp      Generic x86 Debug Trap, Bind TCP Stager
(IPv6)
```

generic/debug_trap/bind_nonx_tcp Generic x86 Debug Trap, Bind TCP Stager (No NX or Win7)
...snip...

Si ha seleccionado un módulo específico, puede utilizar el comando "show options" para mostrar las opciones disponibles y/o requeridas para ese módulo en específico.

msf exploit(ms08_067_netapi) > show options

Module options:

Name	Current Setting	Required	Description
RHOST		yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	BROWSER	yes	The pipe name to use (BROWSER, SRVSVC)

Exploit target:

Id	Name
0	Automatic Targeting

Si no está seguro si un sistema operativo es vulnerable a un particular exploit, ejecute el comando "show targets" dentro del contexto de un módulo de exploit para ver qué objetivos están soportados.

msf exploit(ms08_067_netapi) > show targets

Exploit targets:

Id	Name
0	Automatic Targeting
1	Windows 2000 Universal
2	Windows XP SP0/SP1 Universal
3	Windows XP SP2 English (NX)
4	Windows XP SP3 English (NX)
5	Windows 2003 SP0 Universal

...snip...

Si desea seguir mejorando un exploit, puede ver las opciones avanzadas ejecutando "show advanced".

msf exploit(ms08_067_netapi) > show advanced

Module advanced options:

Name : CHOST
Current Setting:
Description : The local client address

Name : CPORT
Current Setting:
Description : The local client port

...snip...

Comando "search".

El msfconsole incluye una funcionalidad de búsqueda basada en expresiones regulares. Si tiene alguna idea general de lo que está buscando, puede buscarlo con "search". En la salida de abajo, una búsqueda se hizo para MS Bulletin MS09-011. Esta funcionalidad de búsqueda localizó esta cadena de texto dentro del módulo de referencia.

Tenga en cuenta la nomenclatura que se usa en los módulos de Metasploit, subrayas (underscore) en vez de guiones.

```
msf > search ms09-001
[*] Searching loaded modules for pattern 'ms09-001'...

Auxiliary
=========

  Name                  Description
  ----                  -----------
  dos/windows/smb/ms09_001_write  Microsoft SRV.SYS WriteAndX Invalid DataOffset
```

Comando "info".

EL comando "info" proporciona información detallada sobre un determinado módulo incluyendo todas las opciones, objetivos, y otras informaciones.

```
msf > info dos/windows/smb/ms09_001_write

       Name: Microsoft SRV.SYS WriteAndX Invalid DataOffset
       Versión: 6890
       License: Metasploit Framework License (BSD)

Provided by:
  j.v.vallejo
```

Comando "use".

Cuando se ha decidido por un módulo en particular, utilice el comando "use" para seleccionarlo.

```
msf > use dos/windows/smb/ms09_001_write
msf auxiliary(ms09_001_write) > show options

Module options:

  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST                   yes       The target address
  RPORT  445              yes       Set the SMB service port

msf auxiliary(ms09_001_write) >
```

Comando "connect".

Utilizando el comando "connect" con una dirección IP y un número de puerto, se puede conectar a un host remoto desde msfconsole igual como si usara netcat o telnet.

```
msf > connect 192.168.1.1 23
[*] Connected to 192.168.1.1:23
ÿýÿýÿýÿý!ÿûÿû
DD-WRT v24 std (c) 2008 NewMedia-NET GmbH
Release: 07/27/08 (SVN revision: 10011)
ÿ
DD-WRT login:
```

Comando "set".

El comando "set" es usado para configurar las opciones del módulo que actualmente se está utilizando.

```
msf auxiliary(ms09_001_write) > set RHOST 192.168.1.1
RHOST => 192.168.1.1
msf auxiliary(ms09_001_write) > show options

Module options:

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.1.1      yes       The target address
   RPORT  445              yes       Set the SMB service port
```

Una reciente funcionalidad añadida a Metasploit es la habilidad de establecer un encoder o codificador para ser usado. Esto particularmente es útil en el desarrollo de un exploit cuando no está muy seguro que método de codificación de payload funcionara con un exploit.

```
msf exploit(ms08_067_netapi) > show encoders

Compatible encoders
===================

   Name                Description
   ----                -----------
   cmd/generic_sh              Generic Shell Variable Substitution Command Encoder
   generic/none        The "none" Encoder
   mipsbe/longxor              XOR Encoder
   mipsle/longxor      XOR Encoder
   php/base64          PHP Base64 encoder
   ppc/longxor         PPC LongXOR Encoder
   ppc/longxor_tag             PPC LongXOR Encoder
   sparc/longxor_tag           SPARC DWORD XOR Encoder
   x64/xor             XOR Encoder
   x86/alpha_mixed             Alpha2 Alphanumeric Mixedcase Encoder
   x86/alpha_upper             Alpha2 Alphanumeric Uppercase Encoder
   x86/avoid_utf8_tolower      Avoid UTF8/tolower
```

```
  x86/call4_dword_xor       Call+4 Dword XOR Encoder
  x86/countdown             Single-byte XOR Countdown Encoder
  x86/fnstenv_mov           Variable-length Fnstenv/mov Dword XOR Encoder
  x86/jmp_call_additive     Polymorphic Jump/Call XOR Additive Feedback Encoder
  x86/nonalpha      Non-Alpha Encoder
  x86/nonupper      Non-Upper Encoder
  x86/shikata_ga_nai        Polymorphic XOR Additive Feedback Encoder
  x86/unicode_mixed         Alpha2 Alphanumeric Unicode Mixedcase Encoder
  x86/unicode_upper         Alpha2 Alphanumeric Unicode Uppercase Encoder

msf exploit(ms08_067_netapi) > set encoder x86/shikata_ga_nai
encoder => x86/shikata_ga_nai
```

Comando "check".

No hay muchos exploits que lo soporten, pero hay una opción de "check" la cual comprueba si el objetivo es vulnerable a un exploit en particular en lugar de explotarla.

```
msf exploit(ms04_045_wins) > show options

Module options:

  Name   Current Setting  Required  Description
  ----   ---------------  --------  -----------
  RHOST  192.168.1.114 yes       The target address
  RPORT  42              yes       The target port


Exploit target:

  Id  Name
  --  ----
  0   Windows 2000 English


msf exploit(ms04_045_wins) > check
[-] Check failed: The connection was refused by the remote host (192.168.1.114:42)
```

Configuración de variables globales.

Con el fin de no escribir mucho durante un pentest, puede establecer variables globales dentro de msfconsole. Puede hacer esto con el comando "set". Una vez que se a establecido, lo puede usar en muchos exploits y módulos auxiliares como le guste. También lo puede guardar para usarlo la próxima vez cuando ejecute msfconsole. Sin embargo, ciertos datos no son guardados, así que siempre revise sus opciones antes de usar "run" o "exploit". Por el contrario puede usar el comando "unsetg" para eliminar una variable global. En los ejemplos siguientes, las variables se registran todas en mayúsculas (Ejem: LHOST), pero Metasploit no es sensible a las mayúsculas así que no es necesario hacerlo.

```
msf > setg LHOST 192.168.1.101
LHOST => 192.168.1.101
msf > setg RHOSTS 192.168.1.0/24
```

```
RHOSTS => 192.168.1.0/24
msf > setg RHOST 192.168.1.136
RHOST => 192.168.1.136
msf > save
Saved configuration to: /root/.msf3/config
msf >
```

Comando "exploit/run".

Cuando lanza un exploit, puede usar el comando "exploit" mientras que si usa un módulo auxiliar, el uso correcto es "run" aunque "exploit" funciona tan bien.

```
msf auxiliary(ms09_001_write) > run

Attempting to crash the remote host...
datalenlow=65535 dataoffset=65535 fillersize=72
rescue
datalenlow=55535 dataoffset=65535 fillersize=72
rescue
datalenlow=45535 dataoffset=65535 fillersize=72
rescue
datalenlow=35535 dataoffset=65535 fillersize=72
rescue
datalenlow=25535 dataoffset=65535 fillersize=72
rescue
...snip...
```

Comando "back".

Cuando haya terminado de usar un módulo, o si selecciona el módulo equivocado, puede usar el comando "back" para salir de ese contexto. Esto, sin embargo no es requerido. Así como con los routers comerciales, que puede cambiar módulos dentro de otros. Como recordatorio, las variables son cambiadas si son establecidas globalmente.

```
msf auxiliary(ms09_001_write) > back
msf >
```

Comando "resource".

Algunos ataques como Karmetasploit usan archivos adicionales que se pueden cargar a través de msfconsole usando el comando "resource". Estos archivos son unos scripts básicos para msfconsole. Ejecutan los comandos del archivo en secuencia. Más adelante lo discutiremos, fuera de Karmetasploit, esto puede ser muy útil.

```
msf > resource karma.rc
resource> load db_sqlite3
[-]
[-] The functionality previously provided by this plugin has been
[-] integrated into the core command set. Use the new 'db_driver'
[-] command to use a database driver other than sqlite3 (which
[-] is now the default). All of the old commands are the same.
[-]
[-] Failed to load plugin from /pentest/exploits/framework3/plugins/db_sqlite3:
```

```
Deprecated plugin
resource> db_create /root/karma.db
[*] The specified database already exists, connecting
[*] Successfully connected to the database
[*] File: /root/karma.db
resource> use auxiliary/server/browser_autopwn
resource> setg AUTOPWN_HOST 10.0.0.1
AUTOPWN_HOST => 10.0.0.1
...snip...
```

Comando "irb".

Si ejecutamos el comando "irb", cambiará al modo de edición de scripts en ruby, donde podrá usar comandos y crear scripts en el momento.

```
msf > irb
[*] Starting IRB shell...

>> puts "Hello, metasploit!"
Hello, metasploit!
```

Escaneo de Puertos

Aunque tenemos listo y configurado dradis para guardar nuestras notas y resultados, es buena práctica crear una nueva base de datos dentro de Metasploit los datos pueden ser útiles para una rápida recuperación y para ser usado en ciertos escenarios de ataque.

```
msf > db_create
[*] Creating a new database instance...
[*] Successfully connected to the database
[*] File: /root/.msf3/sqlite3.db
msf > load db_tracker
[*] Successfully loaded plugin: db_tracker
msf > help
...snip...
Database Backend Commands
=========================

    Command                 Description
    -------                 -----------
    db_add_host             Add one or more hosts to the database
    db_add_note             Add a note to host
    db_add_port             Add a port to host
    db_autopwn              Automatically exploit everything
    db_connect              Connect to an existing database
    db_create               Create a brand new database
    db_del_host             Delete one or more hosts from the database
    db_del_port             Delete one port from the database
    db_destroy              Drop an existing database
    db_disconnect           Disconnect from the current database instance
    db_driver               Specify a database driver
    db_hosts                List all hosts in the database
    db_import_amap_mlog     Import a THC-Amap scan results file (-o -m)
    db_import_nessus_nbe    Import a Nessus scan result file (NBE)
```

```
    db_import_nessus_xml  Import a Nessus scan result file (NESSUS)
    db_import_nmap_xml    Import a Nmap scan results file (-oX)
    db_nmap               Executes nmap and records the output automatically
    db_notes              List all notes in the database
    db_services           List all services in the database
    db_vulns      List all vulnerabilities in the database

msf >
```

Podemos usar el comando "db_nmap" para hacer un escaneo con Nmap contra nuestros objetivos y tener el resultado de exploracion guardado en la base de datos creada recientemente sin embargo, Metasploit solo creará el archivo de salida en XML que es el formato usado para la base de datos mientras que dradis puede importar cualquier salida grep o normal. Siempre es bueno tener los tres formatos de salida de Nmap (XML, grep y normal) así que escaneamos con Nmap usando el flag o parámetro "-oA" para generar los tres archivos de salida luego usar el comando "db_import_nmap_xml" para rellenar la base de datos Metasploit.

Si no desea importar los resultados dentro de dradis, simplemente ejecute Nmap usando "db_nmap" con las opciones que normalmente usa, omitiendo el parámetro (flag) de salida. El ejemplo de abajo seria "db_nmap -v -sV 192.168.1.0/24".

```
msf > nmap -v -sV 192.168.1.0/24 -oA subnet_1
[*] exec: nmap -v -sV 192.168.1.0/24 -oA subnet_1

Starting Nmap 5.00 ( http://nmap.org ) at 2009-08-13 19:29 MDT
NSE: Loaded 3 scripts for scanning.
Initiating ARP Ping Scan at 19:29
Scanning 101 hosts [1 port/host]
...
Nmap done: 256 IP addresses (16 hosts up) scanned in 499.41 seconds
Raw packets sent: 19973 (877.822KB) | Rcvd: 15125 (609.512KB)
```

Con el escaneo finalizado, usaremos el comando "db_import_nmap_xml" para importar el archivo xml de Nmap.

```
msf > db_import_nmap_xml subnet_1.xml
```

El resultado importado del escaneo de Nmap se puede ver a través de los comandos "db_hosts" y "db_services":

```
msf > db_hosts
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Host: 192.168.1.1 Status: alive OS:
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Host: 192.168.1.2 Status: alive OS:
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Host: 192.168.1.10 Status: alive OS:
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Host: 192.168.1.100 Status: alive OS:
...

msf > db_services
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Service: host=192.168.1.1 port=22 proto=tcp
state=up name=ssh
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Service: host=192.168.1.1 port=23 proto=tcp
state=up name=telnet
```

[*] Time: Thu Aug 13 19:39:05 -0600 2009 Service: host=192.168.1.1 port=80 proto=tcp state=up name=http
[*] Time: Thu Aug 13 19:39:05 -0600 2009 Service: host=192.168.1.2 port=23 proto=tcp state=up name=telnet
...

ERRORES EN METASPLOIT

Dado al sistema de módulos que conforman al framework usualmente los errores de funcionamiento se producen por la incorporación de nuevos módulos que cometen alguno de los errores de la siguiente colección

1. No revisar el valor que retorna la API de Metasploit
2. Ruby 1.9.3 vs 1.8.7
3. No revisar el valor que que retorna el valor de la función match()
4. No revisar por el valor nil antes de utilizar un método
5. Usar un manejador de errores para “callar” un error
6. No aprovechando el bloque 'asegurar'.
7. Agregar la opción VERBOSE
8. Evitar usar 'vars_post' para send_request_cgi() cuando se elabora un POST request
9. Malos nombres de variables
10. Uso de variables globales
11. Modificar la datastore durante la ejecucion