3 Das Metasploit-Framework

Nachdem wir uns im vorigen Kapitel ausgiebig mit der Erstellung unserer Testumgebung befasst haben, kommen wir nun zum eigentlichen Anliegen des Buches. In diesem Kapitel werden die einzelnen Komponenten des Metasploit-Frameworks vorgestellt.

3.1 Die Metasploit-Framework-Architektur

Als Einführung in Metasploit beschäftigen wir uns zunächst ein wenig mit der Architektur des Frameworks. Wir werden später das Metasploit-Framework ausschließlich aus der Sicht des Nutzers betrachten. Trotzdem ist es notwendig, bestimmte Zusammenhänge zu erfassen und den grundlegenden Aufbau zu kennen. Abbildung 3–1 zeigt die Hauptkomponenten des Frameworks.

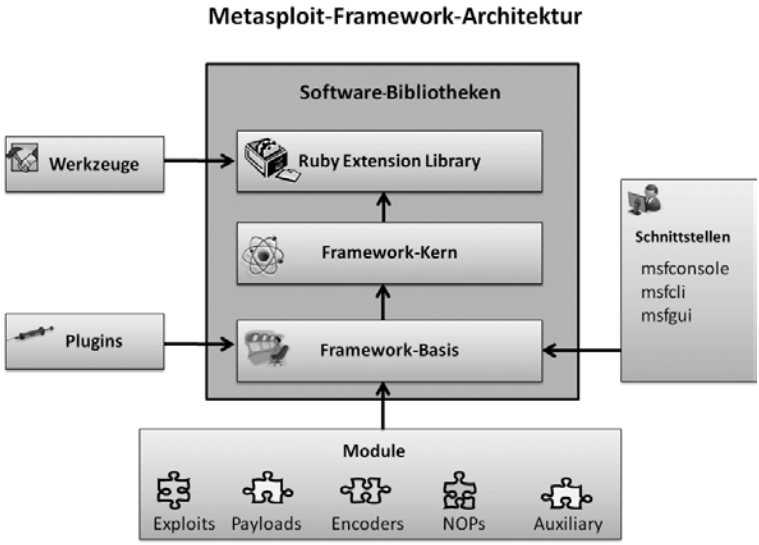

Abb. 3–1 *Metasploit-Framework-Architektur*

Leser, die sich später mit der Weiterentwicklung bzw. der Erweiterung des Frameworks beschäftigen wollen, werden sich besonders für die Softwarebibliotheken interessieren. Die *Ruby Extension Library* (auch Rex genannt) ist die fundamentale Komponente des gesamten Frameworks. Sie ist im Grunde genommen eine Sammlung aus Klassen und Modulen.

Zusätzlich zu den in Rex bereitgestellten Bibliotheken können die Entwickler auf weitere *Werkzeuge*, z.B für die Arbeit mit verschiedenen Protokollen (HTTP, SMB oder Sun RPC), zurückgreifen.

Der Framework-Kern besteht aus verschiedenen Untersystemen, die u.a. für das Module- und Sessionmanagement bzw. die Ereignisbehandlung benötigt werden. Er besteht aus:

- DataStore
- Event Notifications
- Framework Managers
- Utility Classes

Die Frameworkbasis soll die Arbeit mit dem Kern erleichtern und dient daher als Schnittstelle. Wichtige Komponenten sind:

- Configuration
- Logging
- Serialization
- Sessions
- Simplified Framework

Plug-ins dienen dazu, neue Features und Funktionalitäten in das Framework einfließen zu lassen. Unter anderem ist es somit möglich, neue Kommandos in ein Interface (z.B. msfconsole) einzufügen. Das Plug-in-Konzept wurde ab der Version 3.0 in das Framework aufgenommen und trägt damit wesentlich zur Flexibilität und Erweiterbarkeit bei.

Für die meisten Nutzer werden wohl die zur Verfügung gestellten Module und Schnittstellen des Metasploit-Frameworks besonders interessant sein. Dies ist ein Grund mehr, sie in den folgenden Abschnitten etwas ausführlicher zu betrachtet.

3.2 Metasploit-Module

Die Metasploit-Module sind eine wesentlicher Bestandteil des Frameworks. Die einzelnen Bestandteile werden hier etwas genauer beschrieben.

3.2.1 Exploits

Wie der englische Begriff »to exploit« (ausnutzen) bereits ausdrückt, sind in diesem Modul kleine Schadprogramme und Skripte abgelegt, die die einzelnen

Sicherheitslücken in den Betriebssystemen und Anwenderprogrammen ausnutzen. Dieses Modul ist sicher der bekannteste Bestandteil des Frameworks. Leider wird es von unbedarften Nutzern auf diese eine Komponente reduziert. Somit wird Metasploit auch oftmals als Angriffstool für sogenannte »Skriptkiddies« eingestuft. Dass Metasploit viel mehr kann, soll dieses Buch u.a. zeigen.

Als Nächstes werden wir uns anschauen, wo die einzelnen Exploits im Dateisystem von Backtrack 5 abgelegt sind. In unserem Beispiel suchen wir Exploits für den Microsoft SQL Server. Da die Ablage nachvollziehbar aufgebaut ist, werden wir in der Verzeichnisstruktur relativ schnell fündig. Wir wechseln zunächst in das Modulverzeichnis, rufen dann das Modul *Exploits* auf, gehen von dort in das *Windows*-Verzeichnis und finden hier im Verzeichnis *mssql* die gewünschten Exploits.

```
root@bt:/# cd /pentest/exploits/framework/modules

root@bt:/pentest/exploits/framework/modules# ls
auxiliary  encoders  exploits  modules.rb.ts.rb  nops  payloads

root@bt:/pentest/exploits/framework/modules# cd exploits/
root@bt:/pentest/exploits/framework/modules/exploits# ls
aix   dialup   hpux  linux  netware  solaris  unix
bsdi  freebsd  irix  multi  osx      test     windows

root@bt:/pentest/exploits/framework/modules/exploits# cd windows
root@bt:/pentest/exploits/framework/modules/exploits/windows# ls
antivirus   dcerpc      ftp    ldap     motorola  oracle  smtp       vnc
arkeia      driver      games  license  mssql     pop3    ssh        vpn
backdoor    email       http   lotus    mysql     proxy   ssl        wins
backupexec  emc         iis    lpd      nfs       scada   telnet
brightstor  fileformat  imap   misc     nntp      sip     tftp
browser     firewall    isapi  mmsp     novell    smb     unicenter

root@bt:/pentest/exploits/framework/modules/exploits/windows# cd mssql
root@bt:/pentest/exploits/framework/modules/exploits/windows/mssql# ls
lyris_listmanager_weak_pass.rb  ms09_004_sp_replwritetovarbin.rb
ms02_039_slammer.rb             mssql_payload.rb
ms02_056_hello.rb
```

Listing 3–1 *Auffinden der Exploits in der Verzeichnisstruktur*

3.2.2 Payloads

War ein Exploit erfolgreich, benötigt der Angreifer Code, der auf das infiltrierte System übertragen werden kann. Ziel wird es sein, einen erweiterten oder auch dauerhaften Zugriff auf das Zielsystem zu erlangen. Dies wird u.a. dadurch geschehen, dass der Angreifer neue Nutzer-Accounts anlegt, sich mittels Kommandozeile Zugriff auf das System verschafft oder auf dem System installierte Programme (z.B. VNC, Remote Desktop) für spätere Fernzugriffe ausnutzt.

Auch hier ist der Angreifer natürlich bestrebt, unentdeckt zu bleiben und Intrusion-Detection-Systeme (IDS) zu umgehen. Somit können entsprechende Mechanismen in den Payload integriert sein.

Einer der bekanntesten Payloads des Frameworks ist sicherlich der Meta-Interpreter (kurz Meterpreter genannt). Der Meterpreter bringt seine eigene »Command Shell« mit und unterstützt die Penetrierung des Zielsystems mit einer breiten Palette an Befehlen. Schauen wir uns auch hier an, wo wir die Dateien finden können. In diesem Fall interessieren wir uns für den Meterpreter-Payload zur Penetrierung eines Linux-Systems (x86).

```
root@bt:/pentest/exploits/framework/modules# cd  payloads/

root@bt:/pentest/exploits/framework/modules/payloads# ls
singles  stagers  stages

root@bt:/pentest/exploits/framework/modules/payloads# cd singles/
root@bt:/pentest/exploits/framework/modules/payloads/singles# ls
aix  bsd  bsdi  cmd  generic  java  linux  osx  php  solaris  tty  windows

root@bt:/pentest/exploits/framework/modules/payloads/singles# cd linux
root@bt:/pentest/exploits/framework/modules/payloads/singles/linux# ls
mipsbe  mipsle  ppc  ppc64  x86

root@bt:/pentest/exploits/framework/modules/payloads/singles/linux# cd x86
root@bt:/pentest/exploits/framework/modules/payloads/singles/linux/x86# ls
adduser.rb  metsvc_bind_tcp.rb      shell_bind_tcp.rb   shell_reverse_tcp.rb
chmod.rb    metsvc_reverse_tcp.rb   shell_find_port.rb  shell_reverse_tcp2.rb
exec.rb     shell_bind_ipv6_tcp.rb  shell_find_tag.rb
```

Listing 3–2 *Auffinden der Payloads in der Verzeichnisstruktur*

3.2.3 Encoder

Das englische Wort »Encoder« steht im Deutschen für »Codierer« und bezeichnet im Metasploit-Framework die Möglichkeit, Payload so zu verschleiern, dass dieser für IDS-Systeme oder Virenschutzprogramme schwer zu erkennen ist.

Hier hat sich insbesondere der Encoder *shikata_ga_nai* als sehr wirksam erwiesen. Wir werden ihn daher in einem unserer nächsten Tests verwenden, um Virenschutzprogramme zu umgehen. Er lässt sich im Dateiverzeichnis wie folgt finden:

```
root@bt:/pentest/exploits/framework/modules# cd encoders/

root@bt:/pentest/exploits/framework/modules/encoders# ls
cmd  encoder_test.rb.ut.rb  generic  mipsbe  mipsle  php  ppc  sparc  x64  x86

root@bt:/pentest/exploits/framework/modules/encoders# cd x86
root@bt:/pentest/exploits/framework/modules/encoders/x86# ls
```

```
alpha_mixed.rb                  context_time.rb             nonupper.rb
alpha_upper.rb                  countdown.rb                shikata_ga_nai.rb
avoid_utf8_tolower.rb           countdown.rb.ut.rb          single_static_bit.rb
call4_dword_xor.rb              fnstenv_mov.rb              unicode_mixed.rb
call4_dword_xor.rb.ut.rb        fnstenv_mov.rb.ut.rb        unicode_upper.rb
context_cpuid.rb                jmp_call_additive.rb
context_stat.rb                 nonalpha.rb
```

Listing 3–3 *Auffinden der Encoder in der Verzeichnisstruktur*

3.2.4 NOPs

Die sogenannten NOP-Generatoren werden zur Verschleierung des Shellcodes eingesetzt und sollen die Erkennung von Angriffen mittels Intrusion-Detection-Systemen (IDS) erschweren. Diese Systeme sind in der Lage, Netzwerkpakete nach bestimmten Bytesequenzen zu durchsuchen. Dabei sind die sogenannten »NOP sleds« durch eine Folge von 0x90-Maschinebefehlen zu erkennen und eindeutig zu identifizieren. Mittels der NOP-Generatoren ist der Angreifer in der Lage, verschiedene Sequenzen zu erzeugen und somit den auf Signaturbasis arbeitenden IDS ein »Schnippchen zu schlagen«.

Listing 3–4 zeigt, wo Sie die verschiedenen NOP-Generatoren finden können.

```
oot@bt:/pentest/exploits/framework/modules# cd nops/

root@bt:/pentest/exploits/framework/modules/nops# ls
armle  nop_test.rb.ut.rb  php  ppc  sparc  tty  x64  x86

root@bt:/pentest/exploits/framework/modules/nops# cd x86
root@bt:/pentest/exploits/framework/modules/nops/x86# ls
opty2.rb  single_byte.rb
```

Listing 3–4 *Auffinden der NOP-Generatoren in der Verzeichnisstruktur*

3.2.5 Auxiliary

Die sogenannten Helfer (engl. Auxiliaries) sind eine weitere Errungenschaft im Metasploit-Framework und seit der Version 3.0 ein fester Bestandteil. Hinter dieser Bezeichnung verbergen sich kleine Tools, die das Fingerprinting und Vulnerability Scanning in Computernetzwerken verbessern sollen. Mit diesen Hilfsmitteln ist der Penetrationstester u.a. in der Lage, Scans nach angebotenen Diensten im Netzwerk durchzuführen oder das Betriebssystem eines Zielrechners eindeutig zu bestimmen. In Abschnitt 4.1.3 werden wir u.a. nach Passwörtern im Netzwerk sniffen. Hier lässt sich *psnuffle* recht gut einsetzen. Auch dieses Hilfsmittel ist dank der durchdachten Verzeichnisstruktur schnell zu finden.

```
root@bt:~# cd /pentest/exploits/framework/modules/auxiliary/

root@bt:/pentest/exploits/framework/modules/auxiliary# ls
admin   dos      gather  scanner  sniffer  sqli  voip
client  fuzzers  pdf     server   spoof    test

root@bt:/pentest/exploits/framework/modules/auxiliary# cd sniffer/
root@bt:/pentest/exploits/framework/modules/auxiliary/sniffer# ls
psnuffle.rb
```

Listing 3–5 *Auffinden der Auxiliaries in der Verzeichnisstruktur*

3.3 Metasploit-Konsole (msfconsole)

Die Metasploit-Konsole ist das populärste Werkzeug im Framework. Wir werden sie in unseren Tests ausgiebig nutzen. Sie ist für alle Betriebssysteme verfügbar und stellt alle notwendigen Funktionen für eine effektive Arbeit bereit. In der Metasploit-Konsole werden die meisten Komponenten unterstützt. Gleichzeitig gilt sie als sehr stabil in der Arbeit und lässt keinen Komfort in der Bedienung vermissen. Features wie Tabbing, Command Completion und die Ausführung externer Kommandos werden unterstützt.

Abb. 3–2 *Metasploit-Konsole in Backtrack 5 aufrufen*

Im Folgenden werden die einzelnen Features anhand unserer in Abschnitt 2.2.2 aufgesetzten virtuellen Maschine (Backtrack 5) erläutert. Bevor Sie mit der eigentlichen Arbeit beginnen, wird dringend empfohlen, die neusten Updates zu installieren. Öffnen Sie die Metasploit-Konsole (msfconsole) gemäß Abbildung 3–2 und führen Sie den Befehl `svn update` aus.

Als Erstes sollten Sie sich mithilfe des Kommandos `help` einen Überblick verschaffen. Neben den verfügbaren Kommandos finden Sie eine kurze Beschreibung.

```
msf > help

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

    Command       Description
    -------       -----------
    ?             Help menu
    back          Move back from the current context
    banner        Display an awesome metasploit banner
    cd            Change the current working directory
    color         Toggle color
    connect       Communicate with a host
    exit          Exit the console
    help          Help menu
    info          Displays information about one or more module
    irb           Drop into irb scripting mode
    jobs          Displays and manages jobs
    kill          kill a job
    load          Load a framework plugin
    loadpath      Searches for and loads modules from a path
    quit          Exit the console
    resource      Run the commands stored in a file
    route         Route traffic through a session
    save          Saves the active datastores
    …
    …

Database Backend Commands
=========================
    Command        Description
    -------        -----------
    creds          List all credentials in the database
    db_connect     Connect to an existing database
    db_disconnect  Disconnect from the current database instance
    db_driver      Specify a database driver
    db_export      Export a file containing the contents of the database
    db_import      Import a scan result file (filetype will be auto-detected)
    db_nmap        Executes nmap and records the output automatically
    db_status      Show the current database status
    hosts          List all hosts in the database
```

```
    loot          List all loot in the database
    notes         List all notes in the database
    services      List all services in the database
    vulns         List all vulnerabilities in the database
    workspace     Switch between database workspaces
```

Listing 3–6 *msfconsole-Hilfe (gekürzte Ausgabe)*

Das Kommando `show` zeigt alle im Metasploit befindlichen Module an. Die Anzeige ist in folgende Bereiche unterteilt:

- Encoders
- NOPs
- Exploits
- Payloads
- Auxiliary

Folgendes Listing stellt einen kurzen Ausschnitt der verfügbaren Komponenten dar:

```
msf > show encoders

Encoders
========

Name                    Rank       Description
----                    ----       -----------
cmd/generic_sh          good       Generic Shell Variable Substitution Command
                                   Encoder
cmd/ifs                 low        Generic ${IFS} Substitution Command Encoder
generic/none            normal     The "none" Encoder
mipsbe/longxor          normal     XOR Encoder
mipsle/longxor          normal     XOR Encoder
php/base64              normal     PHP Base64 encoder
ppc/longxor             normal     PPC LongXOR Encoder
ppc/longxor_tag         normal     PPC LongXOR Encoder


msf > show auxiliary

Auxiliary
=========

Name                         Rank      Description
----                         ----      -----------
admin/backupexec/dump        normal     Veritas Backup Exec Windows Remote File
                                        Access
admin/backupexec/registry    normal     Veritas Backup Exec Server Registry Access
```

```
admin/cisco/ios_http_auth_bypass  normal     Cisco IOS HTTP Unauthorized
                                             Administrative Access

msf > show payloads

Payloads
========

Name                              Rank    Description
----                              ----    -----------
aix/ppc/shell_bind_tcp            normal  AIX Command Shell, Bind TCP Inline
aix/ppc/shell_find_port           normal  AIX Command Shell, Find Port Inline
aix/ppc/shell_interact            normal  AIX execve shell for inetd
aix/ppc/shell_reverse_tcp         normal  AIX Command Shell, Reverse TCP Inline
bsd/sparc/shell_bind_tcp          normal  BSD Command Shell, Bind TCP Inline
bsd/sparc/shell_reverse_tcp       normal  BSD Command Shell, Reverse TCP Inline
bsd/x86/exec                      normal  BSD Execute Command
```

Listing 3–7 *Das show-Kommando (gekürzte Ausgabe)*

Wie man sicher leicht sehen kann, verliert man bei der großen Anzahl an Exploits, Payloads und Scanner leicht den Überblick. Aus diesem Grund sollte man sich mit dem Kommando `search` gezielt nach verfügbaren Komponenten umschauen und dann mittels der folgenden Kommandos weitere Informationen einholen:

- `show payloads`
- `show options`
- `show target`
- `show advanced`

In diesem Beispiel interessieren wir uns für den Exploit `ms08_067_netapi`. Mit dem Kommando `use` wählen wir es für weitere Abfragen aus:

```
msf > search ms08_067
[*] Searching loaded modules for pattern 'ms08_067'...

Exploits
========

Name                         Rank   Description
----                         ----   -----------
windows/smb/ms08_067_netapi  great  Microsoft Server Service Relative Path Stack
Corruption

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

```
Compatible Payloads
===================

Name                                Rank    Description
----                                ----    -----------
generic/debug_trap                  normal  Generic x86 Debug Trap
generic/shell_bind_tcp              normal  Generic Command Shell, Bind TCP
                                            Inline
generic/shell_reverse_tcp           normal  Generic Command Shell, Reverse TCP
                                            Inline
generic/tight_loop                  normal  Generic x86 Tight Loop
windows/adduser                     normal  Windows Execute net user /ADD

msf exploit(ms08_067_netapi) > show options

Module options:

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

Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting

msf exploit(ms08_067_netapi) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic Targeting
   1   Windows 2000 Universal
   2   Windows XP SP0/SP1 Universal
   3   Windows XP SP2 English (NX)
   4   Windows XP SP3 English (NX)

msf exploit(ms08_067_netapi) > show advanced

Module advanced options:

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

   Name           : CPORT
```

Listing 3–8 *Informationen zum Exploit ms08_067_netapi (gekürzte Ausgabe)*

Das Kommando `info` listet eine detaillierte Beschreibung des Exploits einschließlich der verfügbaren Ziele, Optionen und Referenzen auf:

```
msf exploit(ms08_067_netapi) > info

       Name: Microsoft Server Service Relative Path Stack Corruption
    Version: 9396
   Platform: Windows
 Privileged: Yes
    License: Metasploit Framework License (BSD)
       Rank: Great

Provided by:
  hdm <hdm@metasploit.com>
  Brett Moore <brett.moore@insomniasec.com>

Available targets:
  Id  Name
  --  ----
  0   Automatic Targeting
  1   Windows 2000 Universal
  2   Windows XP SP0/SP1 Universal
  3   Windows XP SP2 English (NX)
  4   Windows XP SP3 English (NX)
  5   Windows 2003 SP0 Universal
  6   Windows 2003 SP1 English (NO NX)
  7   Windows 2003 SP1 English (NX)
  8   Windows 2003 SP2 English (NO NX)
  9   Windows 2003 SP2 English (NX)
  10  Windows 2003 SP2 German (NO NX)
  11  Windows 2003 SP2 German (NX)
  .
  .
  .
Basic options:
  Name     Current Setting  Required  Description
  ----     ---------------  --------  -----------
  RHOST                     yes       The target address
  RPORT    445              yes       Set the SMB service port
  SMBPIPE  BROWSER          yes       The pipe name to use (BROWSER, SRVSVC)

Payload information:
  Space: 400
  Avoid: 8 characters

Description:
  This module exploits a parsing flaw in the path canonicalization
  code of NetAPI32.dll through the Server Service. This module is
  capable of bypassing NX on some operating systems and service packs.
```

```
  The correct target must be used to prevent the Server Service (along
  with a dozen others in the same process) from crashing. Windows XP
  targets seem to handle multiple successful exploitation events, but
  2003 targets will often crash or hang on subsequent attempts. This
  is just the first version of this module, full support for NX bypass
  on 2003, along with other platforms, is still in development.

References:
  http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-4250
  http://www.osvdb.org/49243
  http://www.microsoft.com/technet/security/bulletin/MS08-067.mspx
  NEXPOSE (dcerpc-ms-netapi-netpathcanonicalize-dos)
```

Listing 3–9 *Das info-Kommando (gekürzte Ausgabe)*

Mit dem Kommando `connect` lassen sich Verbindungen zu anderen Servern, Clients, Routern oder mittels Netcat geöffneten Ports herstellen. Man braucht dazu also nicht die Metasploit-Konsole zu verlassen. Geben Sie einfach die IP-Adresse und den entsprechenden Port an. Mit der Option –z lässt sich z.B. einfach überprüfen, ob zum entsprechenden Server eine Verbindung über den angegebenen Port aufgebaut werden kann. Danach wird die Verbindung gleich wieder geschlossen.

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

Communicate with a host, similar to interacting via netcat.

OPTIONS:

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

msf > connect -z 192.168.222.66 23
[*] Connected to 192.168.222.66:23
msf >
```

Listing 3–10 *Die Hilfe zum connect-Kommando*

Wie bereits in Listing 3–8 mittels des Kommando `show options` gezeigt, sind die entsprechenden Module mit verschiedenen Variablen versehen. Diese können

nun mit den Kommandos set bzw. setg (Setting Globals) gesetzt werden, um z.B. Exploits zu konfigurieren. Im Unterschied zum Kommando set werden die mittels setg konfigurierten Variablen »global« festgelegt und können mit dem Kommando save gespeichert und beim nächsten Start von msfconsole weiterverwendet werden. Das Setzen von globalen Variablen kann mit dem Befehl unsetg wieder rückgängig gemacht werden.

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

Module options:

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


Exploit target:

   Id  Name
   --  ----
   0   Automatic Targeting


msf exploit(ms08_067_netapi) > set RHOST 192.168.222.66
RHOST => 192.168.222.66
msf exploit(ms08_067_netapi) > set RPORT 445
RPORT => 445
msf exploit(ms08_067_netapi) >

msf exploit(ms08_067_netapi) > setg RHOST 192.168.222.66
RHOST => 192.168.222.66
msf exploit(ms08_067_netapi) > setg RPORT 445
RPORT => 445
msf exploit(ms08_067_netapi) > save
Saved configuration to: /root/.msf4/config
msf exploit(ms08_067_netapi) >

msf exploit(ms08_067_netapi) > unsetg RHOST
Unsetting RHOST...
msf exploit(ms08_067_netapi) >
```

Listing 3–11 *Das set- und setg-Kommando*

Mit dem Kommando run werden die Exploits oder z.B. Scanner aus dem Auxiliary-Modul gestartet. In diesem Beispiel möchten wir in unserem Testnetzwerk

nach Servern suchen, die den SSH-Service nutzen. Wir interessieren uns für das Betriebssystem bzw. die laufende SSH-Version.

```
msf > use scanner/ssh/ssh_version
msf auxiliary(ssh_version) > show options

Module options:

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   RHOSTS                    yes       The target address range or CIDR identifier
   RPORT    22               yes       The target port
   THREADS  1                yes       The number of concurrent threads

msf auxiliary(ssh_version) > set RHOSTS 192.168.222.0/24
RHOSTS => 192.168.222.0/24

msf auxiliary(ssh_version) > run

[*] 192.168.222.10:22, SSH server version: SSH-2.0-OpenSSH_5.1p1 Debian-3ubuntu1
[*] Scanned 026 of 256 hosts (010% complete)
[*] Scanned 052 of 256 hosts (020% complete)
[*] 192.168.222.66:22, SSH server version: SSH-2.0-OpenSSH_5.2
[*] Scanned 077 of 256 hosts (030% complete)
```

Listing 3–12 *Das run-Kommando*

Mit dem Kommando `back` werden ausgewählte Exploits oder andere Module wieder abgewählt. Alternativ dazu kann man den Befehl `use` wählen, um auf andere Module/Exploits zu wechseln.

```
msf > use scanner/snmp/community
msf auxiliary(community) > back
msf >
msf > use scanner/snmp/community
msf auxiliary(community) > use scanner/telnet/telnet_login
msf auxiliary(telnet_login) >
```

Listing 3–13 *Das back-Kommando*

In der Konsole können externe Befehle (entsprechend dem installierten Betriebssystem) ausgeführt werden. In Listing 3–14 wird beispielhaft die Netzwerkkonfiguration mithilfe des Befehls `ifconfig` angezeigt bzw. mittels `nslookup` die aktuelle IP-Adresse eines Domainnamens ermittelt.

```
msf > ifconfig eth0
[*] exec: ifconfig eth0

eth0      Link encap:Ethernet  HWaddr 00:0c:29:9d:48:12
```

```
          inet addr:192.168.222.10  Bcast:192.168.222.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe9d:4812/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:103 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:136133 (136.1 KB)  TX bytes:19435 (19.4 KB)
          Interrupt:19 Base address:0x2000

msf > nslookup remotexploit.com
[*] exec: nslookup remotexploit.com

Server:         192.168.222.2
Address:        192.168.222.2#53

Non-authoritative answer:
Name:   remotexploit.com
Address: 174.137.132.45
```

Listing 3–14 *Externe Befehle ausführen*

3.4 Metasploit-Client (msfcli)

Mit dem Metasploit-Client (msfcli) steht im Framework eine mächtige Kommandozeilen-Umgebung zur Verfügung. Leider unterscheiden sich die Bedienung und die Optionen ein wenig von der Metasploit-Konsole (msfconsole). Die wichtigsten Unterschiede werden hier vorgestellt. Verschaffen Sie sich zunächst einen Überblick durch Aufrufen der Hilfefunktion wie in Abbildung 3–3 dargestellt.

Die Option `(S)ummary` entspricht dem Kommano `info` in der msfconsole. Es werden Informationen über das verfügbare Modul, Targets und Referenzen angezeigt.

```
root@bt:/pentest/exploits/framework# ./msfcli windows/smb/ms08_067_netapi S
[*] Please wait while we load the module tree...
      Name: Microsoft Server Service Relative Path Stack Corruption
      Module: exploit/windows/smb/ms08_067_netapi
      Version: 14319
      Platform: Windows
      Privileged: Yes
      License: Metasploit Framework License (BSD)
      Rank: Great

   Provided by:
      hdm hdm@metasploit.com
      Brett Moore brett.moore@insomniasec.com
      staylor
      jduck <jduck@metasploit.com>
```