

T. J. O'Connor

Python Hacking

- Lernen Sie die Sprache der Hacker, testen Sie Ihr System mit Python und schließen Sie die Lücken!
- Programmierwissen der Hacker mit Quellcode erklärt: Analyse von Netzwerkverkehr, WLANs angreifen, Passwörter knacken, Exploits schreiben, Daten von sozialen Netzwerken abgreifen

T. J. O'Connor
Python Hacking

T. J. O'Connor

Python Hacking

- Lernen Sie die Sprache der Hacker, testen Sie Ihr System mit Python und schließen Sie die Lücken!
- Programmierwissen der Hacker mit Quellcode erklärt: Analyse von Netzwerkverkehr, WLANs angreifen, Passwörter knacken, Exploits schreiben, Daten von sozialen Netzwerken abgreifen

Bibliografische Information der Deutschen Bibliothek

Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte Daten sind im Internet über <http://dnb.ddb.de> abrufbar.

Alle Angaben in diesem Buch wurden vom Autor mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. Der Verlag und der Autor sehen sich deshalb gezwungen, darauf hinzuweisen, dass sie weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen können. Für die Mitteilung etwaiger Fehler sind Verlag und Autor jederzeit dankbar. Internetadressen oder Versionsnummern stellen den bei Redaktionsschluss verfügbaren Informationsstand dar. Verlag und Autor übernehmen keinerlei Verantwortung oder Haftung für Veränderungen, die sich aus nicht von ihnen zu vertretenden Umständen ergeben. Evtl. beigefügte oder zum Download angebotene Dateien und Informationen dienen ausschließlich der nicht gewerblichen Nutzung. Eine gewerbliche Nutzung ist nur mit Zustimmung des Lizenzinhabers möglich.

This edition of **Violent Python** by TJ O'Connor is published by arrangement with **ELSEVIER INC.**, a Delaware corporation having its principal place of business at 360 Park Avenue South, New York, NY 10010, USA
ISBN der englischen Originalausgabe: 978-1597499576

Das Python-Logo ist ein Warenzeichen (TM) der Python Software Foundation (PSF).

© 2015 Franzis Verlag GmbH, 85540 Haar bei München

Alle Rechte vorbehalten, auch die der fotomechanischen Wiedergabe und der Speicherung in elektronischen Medien. Das Erstellen und Verbreiten von Kopien auf Papier, auf Datenträgern oder im Internet, insbesondere als PDF, ist nur mit ausdrücklicher Genehmigung des Verlags gestattet und wird widrigenfalls strafrechtlich verfolgt.

Die meisten Produktbezeichnungen von Hard- und Software sowie Firmennamen und Firmenlogos, die in diesem Werk genannt werden, sind in der Regel gleichzeitig auch eingetragene Warenzeichen und sollten als solche betrachtet werden. Der Verlag folgt bei den Produktbezeichnungen im Wesentlichen den Schreibweisen der Hersteller.

Programmleitung: Dr. Markus Stäuble

Satz und Übersetzung: G&U Language & Publishing Services GmbH

art & design: www.ideehoch2.de

Druck: M.P. Media-Print Informationstechnologie GmbH, 33100 Paderborn

Printed in Germany

ISBN 978-3-645-60415-4

Inhaltsverzeichnis

| | |
|--|-----------|
| Danksagung..... | 15 |
| Der Autor: T. J. O'Connor | 17 |
| Der Coautor: Rob Frost..... | 19 |
| Der Fachgutachter: Mark Baggett | 21 |
| Einleitung | 23 |
| Zielgruppe..... | 23 |
| Der Aufbau dieses Buches | 23 |
| Kapitel 1: Grundlagen | 24 |
| Kapitel 2: Penetrationstests mit Python | 24 |
| Kapitel 3: Forensische Untersuchungen mit Python | 24 |
| Kapitel 4: Analyse des Netzwerkverkehrs mit Python..... | 25 |
| Kapitel 5: Angriffe auf drahtlose Netzwerke mit Python | 25 |
| Kapitel 6: Webaufklärung mit Python | 25 |
| Kapitel 7: Umgehen von Antivirussoftware mit Python..... | 25 |
| Die Begleitwebsite | 26 |
| 1. Grundlagen | 27 |
| 1.1 Einführung: Ein Penetrationstest mit Python | 27 |
| 1.2 Die Entwicklungsumgebung einrichten..... | 29 |
| 1.2.1 Drittanbieterbibliotheken installieren..... | 29 |
| 1.2.2 Interpretiertes und interaktives Python im Vergleich | 33 |

| | | |
|-----------|---|-----------|
| 1.3 | Die Sprache Python | 34 |
| 1.3.1 | Variablen | 35 |
| 1.3.2 | Strings..... | 36 |
| 1.3.3 | Listen..... | 36 |
| 1.3.4 | Dictionarys | 37 |
| 1.3.5 | Netzwerkverbindungen | 38 |
| 1.3.6 | Bedingte Anweisungen..... | 39 |
| 1.3.7 | Ausnahmebehandlung | 40 |
| 1.3.8 | Funktionen | 42 |
| 1.3.9 | Iteration..... | 44 |
| 1.3.10 | Datei-E/A..... | 47 |
| 1.3.11 | Das Modul sys | 48 |
| 1.3.12 | Das Modul os..... | 49 |
| 1.4 | Ihr erstes Python-Programm | 52 |
| 1.4.1 | Kuckucksei: Der Hintergrund für Ihr erstes Python-Programm..... | 52 |
| 1.4.2 | Ein UNIX-Passwortknacker | 53 |
| 1.4.3 | Böse Dinge für einen guten Zweck: Der Hintergrund für Ihr zweites Python-Programm | 57 |
| 1.4.4 | Ein Passwortknacker für Zip-Dateien | 58 |
| 1.5 | Zusammenfassung | 64 |
| 1.6 | Literatur..... | 64 |
| 2. | Penetrationstests mit Python | 67 |
| 2.1 | Einführung: Würde der Morris-Wurm heute noch funktionieren? | 67 |
| 2.2 | Einen Portscanner schreiben | 68 |
| 2.2.1 | Vollständiger TCP-Verbindungsscan | 69 |
| 2.2.2 | Das Anwendungsbanner abrufen..... | 72 |

| | | |
|-------|--|-----|
| 2.2.3 | Den Scan auf Threads aufteilen | 74 |
| 2.2.4 | Den Nmap-Portscanner aufnehmen..... | 78 |
| 2.3 | Ein SSH-Botnetz mit Python aufbauen..... | 80 |
| 2.3.1 | Interaktion mit SSH über Pexpect | 82 |
| 2.3.2 | SSH-Passwörter mit Pssh knacken..... | 86 |
| 2.3.3 | SSH über schwache private Schlüssel angreifen | 90 |
| 2.3.4 | Ein SSH-Botnetz aufbauen | 95 |
| 2.4 | Kombinierter Massenangriff über FTP und das Web | 99 |
| 2.4.1 | Einen Scanner für den anonymen FTP-Zugriff mit Python erstellen | 100 |
| 2.4.2 | FTP-Anmeldeinformationen mit Ftplib ermitteln | 102 |
| 2.4.3 | Auf dem FTP-Server nach Webseiten suchen | 104 |
| 2.4.4 | Schadcode auf den Webseiten injizieren | 105 |
| 2.4.5 | Die Einzelteile zusammenfügen..... | 107 |
| 2.5 | Conficker | 113 |
| 2.5.1 | Den Windows-SMB-Dienst mit Metasploit angreifen | 115 |
| 2.5.2 | Python-Code zur interaktiven Nutzung von Metasploit schreiben | 117 |
| 2.5.3 | Eine Brute-Force-Methode zur Ausführung von Remoteprozessen..... | 119 |
| 2.5.4 | Endmontage: Einen eigenen Conficker schreiben | 120 |
| 2.6 | Eigenen Zero-Day-Angriffscode schreiben | 124 |
| 2.6.1 | Angriffe mithilfe von Stack-Pufferüberläufen..... | 124 |
| 2.6.2 | Die Kernelemente des Exploits | 125 |
| 2.6.3 | Den Exploit senden..... | 127 |
| 2.6.4 | Das Exploit-Skript fertigstellen | 128 |
| 2.7 | Zusammenfassung des Kapitels | 131 |
| 2.8 | Literatur..... | 131 |

| | | |
|-----------|--|------------|
| 3. | Forensische Untersuchungen mit Python..... | 135 |
| 3.1 | Einführung: Wie die BTK-Morde durch forensische Untersuchungen aufgeklärt wurden | 135 |
| 3.2 | Wo bist du gewesen? – Drahtlose Zugriffspunkte in der Registrierung analysieren | 137 |
| 3.2.1 | Die Windows-Registrierung mit WinReg lesen | 138 |
| 3.2.2 | Die MAC-Adresse mit Mechanize an Wigle übertragen... .. | 140 |
| 3.3 | Gelöschte Elemente im Papierkorb mit Python untersuchen | 146 |
| 3.3.1 | Gelöschte Elemente mithilfe des Moduls OS finden | 146 |
| 3.3.2 | SIDs Benutzern zuordnen | 147 |
| 3.4 | Metadaten | 150 |
| 3.4.1 | PDF-Metadaten mit PyPDF analysieren | 151 |
| 3.4.2 | Exif-Metadaten | 154 |
| 3.4.3 | Bilder mit BeautifulSoup herunterladen | 155 |
| 3.4.4 | Exif-Metadaten von Bildern mit der Python-Bibliothek Imaging lesen | 157 |
| 3.5 | Spuren von Anwendungen mit Python untersuchen | 161 |
| 3.5.1 | Die SQLite-Datenbank von Skype | 161 |
| 3.5.2 | Skype-Datenbankabfragen mit Python und Sqlite3 automatisieren | 163 |
| 3.5.3 | SQLite-Datenbanken von Firefox mit Python untersuchen | 171 |
| 3.6 | iTunes-Backups mit Python untersuchen..... | 181 |
| 3.7 | Zusammenfassung des Kapitels | 189 |
| 3.8 | Literatur..... | 189 |

| | | |
|-----------|--|------------|
| 4. | Analyse des Netzwerkverkehrs mit Python | 191 |
| 4.1 | Einführung: Operation Aurora – Das Offensichtliche übersehen | 192 |
| 4.2 | Wohin geht der Datenverkehr? Python antwortet! | 193 |
| 4.2.1 | IP-Adressen mit PyGeolIP physischen Standorten zuordnen | 194 |
| 4.2.2 | Pakete mit Dpkt analysieren | 195 |
| 4.2.3 | Google-Karten mit Python erstellen | 200 |
| 4.3 | Analyse des LOIC-Datenverkehrs: Ist Anonymous wirklich anonym? | 204 |
| 4.3.1 | LOIC-Downloads mit Dpkt finden | 204 |
| 4.3.2 | IRC-Befehle zum Hive analysieren | 206 |
| 4.3.3 | Laufende DDoS-Angriffe erkennen | 209 |
| 4.4 | Wie H. D. Moore das Problem des Pentagon löste | 215 |
| 4.4.1 | Das TTL-Feld | 216 |
| 4.4.2 | TTL-Felder mit Scapy analysieren | 218 |
| 4.5 | Die Fast-Flux- und Domain-Flux-Techniken von Storm und Conficker | 223 |
| 4.5.1 | Weiß Ihr DNS etwas, das Sie nicht wissen? | 224 |
| 4.5.2 | DNS-Datenverkehr mit Scapy analysieren | 225 |
| 4.5.3 | Fast-Flux-Datenverkehr mit Scapy aufspüren | 226 |
| 4.5.4 | Domain-Flux-Datenverkehr mit Scapy aufspüren | 228 |
| 4.6 | Kevin Mitnick: Vorhersage von TCP-Folgenummern | 230 |
| 4.6.1 | Vorhersage von TCP-Folgenummern selbst gemacht | 231 |
| 4.6.2 | Eine SYN-Flood mit Scapy hervorrufen | 232 |
| 4.6.3 | TCP-Folgenummern berechnen | 233 |
| 4.6.4 | Die TCP-Verbindung fälschen | 236 |
| 4.7 | Intrusion-Detection-Systeme mit Scapy unterlaufen | 240 |
| 4.8 | Zusammenfassung des Kapitels | 249 |
| 4.9 | Literatur | 249 |

| | | |
|-----------|--|------------|
| 5. | Angriffe auf drahtlose Netzwerke mit Python | 251 |
| 5.1 | Einführung: (Un-) Sicherheit von WLANs und der Eismann..... | 252 |
| 5.2 | Die Umgebung für WLAN-Angriffe einrichten | 252 |
| 5.2.1 | Die Erfassung von WLAN-Datenverkehr mit Scapy testen..... | 253 |
| 5.2.2 | Bluetooth-Pakete für Python installieren..... | 255 |
| 5.3 | Wall of Sheep – WLAN-Geheimnisse passiv belauschen..... | 256 |
| 5.3.1 | Kreditkarteninformationen mit regulären Ausdrücken ausspähen..... | 256 |
| 5.3.2 | Hotelgäste ausspionieren..... | 261 |
| 5.3.3 | Einen WLAN-Keylogger für Google-Abfragen schreiben..... | 264 |
| 5.3.4 | FTP-Anmeldeinformationen ausspionieren..... | 269 |
| 5.4 | Wo ist Ihr Laptop gewesen? Python antwortet!..... | 271 |
| 5.4.1 | Auf 802.11-Suchanfragen lauschen..... | 272 |
| 5.4.2 | 802.11-Signal von verborgenen Netzwerken finden | 273 |
| 5.4.3 | Verborgene 802.11-Netzwerke enttarnen | 274 |
| 5.5 | Drohnen mit Python übernehmen und ausspionieren... | 275 |
| 5.5.1 | Datenverkehr abfangen und das Protokoll analysieren..... | 276 |
| 5.5.2 | 802.11-Pakete mit Scapy gestalten | 279 |
| 5.5.3 | Die Drohne zum Absturz bringen | 283 |
| 5.6 | Firesheep erkennen | 285 |
| 5.6.1 | Wordpress-Sitzungscookies | 286 |
| 5.6.2 | Schafe hüten: Die Wiederverwendung von Wordpress-Cookies erkennen..... | 288 |
| 5.7 | Spionieren mit Bluetooth und Python..... | 291 |
| 5.7.1 | Drahtlosen Datenverkehr zum Ermitteln von Bluetooth-Adressen abfangen..... | 293 |

| | | |
|-----------|---|------------|
| 5.7.2 | RFComm-Kanäle suchen..... | 297 |
| 5.7.3 | Service Discovery Protocol | 299 |
| 5.7.4 | Einen Drucker mit Python ObexFTP übernehmen..... | 300 |
| 5.7.5 | BlueBug-Angriffe mit Python durchführen | 301 |
| 5.8 | Zusammenfassung des Kapitels | 303 |
| 5.9 | Literatur..... | 304 |
| 6. | Webaufklärung mit Python | 307 |
| 6.1 | Einführung: Social Engineering heute..... | 307 |
| 6.1.1 | Aufklärung vor dem Angriff..... | 308 |
| 6.2 | Mit der Bibliothek Mechanize im Internet surfen | 309 |
| 6.2.1 | Anonymität: Proxys, Benutzeragenten und Cookies | 311 |
| 6.2.2 | Eine Python-Klasse für den anonymen Browser schreiben..... | 315 |
| 6.3 | Webseiten mit anonBrowser untersuchen | 318 |
| 6.3.1 | HREF-Links mit BeautifulSoup abschöpfen..... | 318 |
| 6.3.2 | Bilder mit BeautifulSoup spiegeln | 321 |
| 6.4 | Recherche, Untersuchung und Aufdeckung | 323 |
| 6.4.1 | Mit Python auf die Google-API zugreifen | 324 |
| 6.4.2 | Tweets mit Python analysieren | 328 |
| 6.4.3 | Ortsangaben aus Tweets entnehmen..... | 331 |
| 6.4.4 | Interessen auf Twitter mithilfe regulärer Ausdrücke bestimmen | 334 |
| 6.5 | Anonyme E-Mail..... | 340 |
| 6.6 | Social Engineering als Massenangriff | 341 |
| 6.6.1 | E-Mails mit Smtplib verschicken | 342 |
| 6.6.2 | Spear Phishing mit Smtplib | 344 |
| 6.7 | Zusammenfassung des Kapitels | 348 |
| 6.8 | Literatur..... | 349 |

| | | |
|-----------|---|------------|
| 7. | Umgehen von Antivirussoftware mit Python | 351 |
| 7.1 | Einführung: Flame | 351 |
| 7.2 | Antivirusprogrammen ausweichen | 352 |
| 7.3 | Die Umgehung der Antivirussoftware bestätigen..... | 357 |
| 7.4 | Zusammenfassung | 365 |
| 7.5 | Literatur..... | 365 |
| | Stichwortverzeichnis | 367 |

*Für mein Äffchen und meine Ninja-Prinzessin: Alles ist möglich,
wenn man sich nur stark genug bemüht.*

Der Autor: T. J. O'Connor

T. J. O'Connor ist Experte für Informationssicherheit beim US-Verteidigungsministerium und Fallschirmjäger der US Army. Als Privatdozent an der US-Militärakademie gab er Kurse über Forensik, Exploits und Informationsschutz. Zweimal betreute er als Co-Trainer Teams, die aus den jährlichen Cyber-Defense-Übungen der NSA als Sieger hervorgingen. Außerdem gewann er den ersten jährlichen Cyber Challenge der National Defense University. Er hat schon in mehreren Red Teams gedient, unter anderem zweimal im Northeast Regional Team bei der National Collegiate Cyber Defense Competition.

O'Connor hat einen Mastergrad in Informatik von der North Carolina State University, einen Mastergrad in Informationssicherheit vom SANS Technical Institute sowie einen Bachelorgrad in Informatik von der US-Militärakademie. Er hat wissenschaftliche Forschungsergebnisse in USENIX-Arbeitskreisen, auf ACM- und Sicherheitskonferenzen, im SANS Reading Room, beim Internet Storm Center, im *Army Magazine* und im *Armed Forces Journal* vorgestellt. Zu seinen Auszeichnungen als Experte auf dem Gebiet der Computersicherheit gehören das angesehene Zertifikat als GSE (GIAC Security Expert) und OSCE (Offensive Security Certified Expert). O'Connor ist Mitglied des SANS-Eliteteams Cyber Guardians.

Der Coautor: Rob Frost

Robert Frost erlangte im Jahr 2011 seinen Abschluss an der US-Militärakademie und ging zur Fernmeldetruppe der US Army. Er hat einen Bachelor in Informatik mit Auszeichnung, wobei er sich in seiner Bachelorarbeit auf Informationsbeschaffung aus öffentlich zugänglichen Quellen konzentrierte. Aufgrund seiner Fähigkeit, Regeln zu umgehen, wurde Rob als eines von zwei Mitgliedern des nationalen Meisterschaftsteams für die Cyber Defense Exercise 2011 ausgewählt. Rob hat mehrere Wettbewerbe in Computersicherheit gewonnen.

Der Fachgutachter: Mark Baggett

Mark Baggett ist zertifizierter SANS-Ausbilder und gibt verschiedene Kurse im Rahmen des SANS-Lehrplans über Penetrationstests. Er ist Hauptberater und Gründer der Firma In Depth Defence Inc., die Penetrationstests und Incident-Response-Dienste anbietet. In seiner Rolle als technischer SANS-Berater des US-Verteidigungsministeriums ist Baggett insbesondere mit der praktischen Anwendung von SANS-Ressourcen bei der Entwicklung militärischer Fähigkeiten beschäftigt.

Baggett war in großen internationalen und Fortune-1000-Unternehmen auf verschiedenen Posten für Informationssicherheit verantwortlich. Er hat als Softwareentwickler, als Netzwerk- und Systemingenieur, als Sicherheitsmanager und als CISO gearbeitet. In letzterer Stellung war er für Richtlinien, Konformität, Reaktion auf Störungen und alle anderen Aspekte der Informationssicherheit verantwortlich. Damit kennt er die Herausforderungen, die sich Sicherheitsexperten heute beim Verkaufen, Umsetzen und Unterstützen von Maßnahmen zur Informationssicherheit stellen, aus erster Hand. Baggett ist aktives Mitglied der Informationssicherheit-Community und Gründungspräsident der Greater Augusta ISSA. Er besitzt verschiedene Zertifikate, darunter das angesehene GSE des SANS-Instituts. Sein Blog über verschiedene Sicherheitsthemen finden Sie auf <http://www.pauldotcom.com>.



Forensische Untersuchungen mit Python

Letzten Endes musst du die Technik verdrängen. Je weiter zu voranschreitest, umso weniger Lehren gibt es. Der Große Weg ist in Wirklichkeit KEIN WEG ...

Ueshiba Morihei, Begründer (Kaiso) des Aikido

3.1 Einführung: Wie die BTK-Morde durch forensische Untersuchungen aufgeklärt wurden

Im Februar 2005 entdeckte der Kriminaltechniker Randy Stone von der Polizei in Wichita die entscheidenden Hinweise zur Lösung eines 30 Jahre alten Rätsels. Einige Tage zuvor hatte der Fernsehsender KSAS der Polizei

eine 3,5-Zoll-Diskette überreicht, die ihm von dem berüchtigten BTK-Killer (»bind, torture, kill«) zugespielt worden war. Der Mörder, der zwischen 1974 und 1991 mindestens zehn Menschen umgebracht hatte, war bislang einer Verhaftung entgangen und hatte wiederholt die Polizei und seine Opfer verhöhnt. Am 16. Februar 2005 schickte er dem Sender eine Diskette mit Anweisungen zur Kommunikation. Darunter befand sich auch eine Datei Namens *Test.A.rtf* (Regan, 2006). Neben den Anweisungen des Mörders enthielt sie jedoch noch etwas anderes – nämlich Metadaten. In das proprietäre Rich-Text-Format von Microsoft (RTF) waren der Vorname des Mörders und der Standort eingebettet, an dem die Datei zuletzt gespeichert worden war.

Das engte die Suche auf einen Mann namens Denis bei der evangelischen Kirche von Wichita ein. Stone konnte feststellen, dass ein Mann namens Denis Rader Mitglied des Kirchenvorstands war (Regan, 2006). Aufgrund dieser Informationen forderte die Polizei einen Gerichtsbeschluss an, um eine DNA-Probe aus den medizinischen Unterlagen von Raders Tochter untersuchen zu dürfen (Shapiro, 2007). Diese Probe bestätigte, was Stone bereits wusste: Denis Rader war der BTK-Mörder. Mit Stones Untersuchung der Metadaten endete eine Fahndung, die 31 Jahre gedauert und 100.000 Mannstunden gekostet hatte.

Computerforensik kann immer nur so gut sein wie die Person, die sie durchführt, und die Werkzeuge, die ihr zur Verfügung stehen. Nur zu oft muss sich ein Ermittler mit einer drängenden Frage herumschlagen, ohne das passende Werkzeug zu ihrer Beantwortung zu haben. Hier kommt Python ins Spiel. Wie wir bereits in den vorherigen Kapiteln gesehen haben, besteht eine Stärke von Python darin, komplizierte Probleme mit einem Minimum an Code lösen zu können. In den folgenden Abschnitten werden Sie sehen, dass wir auch sehr komplizierte Fragen mit sehr wenigen Zeilen von Python-Code beantworten können. Als Erstes schauen wir uns an, wie wir einige besondere Windows-Registrierungsschlüssel heranziehen können, um den Standort eines Benutzers zu verfolgen.

3.2 Wo bist du gewesen? – Drahtlose Zugriffspunkte in der Registrierung analysieren

Die Windows-Registrierung enthält eine hierarchische Datenbank, in der die Konfigurationseinstellungen des Betriebssystems gespeichert sind. Seit es drahtlose Netzwerke gibt, werden dort auch Informationen über drahtlose Verbindungen abgelegt. Wenn wir den Speicherort und die Bedeutung dieser Schlüssel kennen, können wir damit herausfinden, wo ein Laptop gewesen ist. Ab Windows Vista werden die einzelnen Netzwerke in Unterschlüsseln von *HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Signatures\Unmanaged* gespeichert. An der Windows-Eingabeaufforderung können wir die einzelnen Netzwerke mit ihrem global eindeutigen Bezeichner (GUID) des Profils, einer Beschreibung und dem Namen des Netzwerks und der MAC-Adresse des Gateways auflisten lassen.

```
C:\Windows\system32>reg query "HKEY_LOCAL_MACHINE\SOFTWARE\
Microsoft\Windows NT\
CurrentVersion\NetworkList\Signatures\Unmanaged" /s
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\
CurrentVersion\NetworkList\Signatures\Unmanaged\
010103000F0000F0080000000F0000F04BCC2360E4B8F7
DC8BDAFAB8AE4DAD862E3960B979A7AD52FA5F70188E103148
ProfileGuid          REG_SZ {3B24CE70-AA79-4C9A-B9CC-83F90C-
2C9C0D}
Description          REG_SZ Hooters_San_Pedro
Source               REG_DWORD 0x8
DnsSuffix            REG_SZ <none>
FirstNetwork         REG_SZ Public_Library
DefaultGatewayMac    REG_BINARY 00115024687F0000
```

3.2.1 Die Windows-Registrierung mit WinReg lesen

Die MAC-Adresse des Gateways wird in der Registrierung als Wert vom Typ REG_BINARY gespeichert. Die Hexbytes `\x00\x11\x50\x24\x68\x7F\x00\x00` im vorstehenden Beispiel stehen für die Adresse `00:11:50:24:68:7F`. Wir wollen nun eine kleine Funktion schreiben, die einen REG_BINARY-Wert in eine MAC-Adresse umwandelt. Wie wir später noch sehen werden, kann es sehr nützlich sein, die MAC-Adresse eines drahtlosen Netzwerks zu kennen.

```
def val2addr(val):
    addr = ""
    for ch in val:
        addr += ("%02x" % ord(ch))
    addr = addr.strip(" ").replace(" ", ":")[0:17]
    return addr
```

Als Nächstes schreiben wir eine Funktion, die den Netzwerknamen und die MAC-Adresse aller aufgeführten Netzwerke aus den entsprechenden Schlüsseln in der Windows-Registrierung gewinnt. Dazu verwenden wir die Bibliothek `_winreg`, die vom Python-Installer für Windows standardmäßig installiert wird. Nachdem wir eine Verbindung zur Registrierung hergestellt haben, können wir den Schlüssel mit der Funktion `OpenKey()` öffnen und die darunter aufgeführten Netzwerkprofile durchlaufen. Für jedes Profil enthält die Registrierung die Unterschlüssel `ProfileGuid`, `Description`, `Source`, `DnsSuffix`, `FirstNetwork` und `DefaultGatewayMac`. Die Schlüssel für den Netzwerknamen und die MAC-Adresse haben den Index 4 bzw. 5 im Array. Damit können wir diese Schlüssel auflisten und auf dem Bildschirm ausgeben.

```
from _winreg import *
def printNets():
    net = "SOFTWARE\Microsoft\Windows NT\CurrentVersion"+
        "\NetworkList\Signatures\Unmanaged"
    key = OpenKey(HKEY_LOCAL_MACHINE, net)
    print '\n[*] Networks You have Joined.'
```

```
for i in range(100):
    try:
        guid = EnumKey(key, i)
        netKey = OpenKey(key, str(guid))
        (n, addr, t) = EnumValue(netKey, 5)
        (n, name, t) = EnumValue(netKey, 4)
        macAddr = val2addr(addr)
        netName = str(name)
        print '[+] ' + netName + ' ' + macAddr
        CloseKey(netKey)
    except:
        break
```

Wenn wir alles zusammennehmen, bekommen wir ein Skript, das alle in der Registrierung gespeicherten drahtlosen Netzwerke ausgibt, mit denen der Computer zuvor verbunden war.

```
from _winreg import *
def val2addr(val):
    addr = ''
    for ch in val:
        addr += '%02x ' % ord(ch)
    addr = addr.strip(' ').replace(' ', ':')[0:17]
    return addr
def printNets():
    net = "SOFTWARE\Microsoft\Windows NT\CurrentVersion"+\
        "\NetworkList\Signatures\Unmanaged"
    key = OpenKey(HKEY_LOCAL_MACHINE, net)
    print '\n[*] Networks You have Joined.'
    for i in range(100):
        try:
            guid = EnumKey(key, i)
            netKey = OpenKey(key, str(guid))
            (n, addr, t) = EnumValue(netKey, 5)
            (n, name, t) = EnumValue(netKey, 4)
```

```
        macAddr = val2addr(addr)
        netName = str(name)
        print '[+] ' + netName + ' ' + macAddr
        CloseKey(netKey)
    except:
        break
def main():
    printNets()
if __name__ == "__main__":
    main()
```

Wenn wir dieses Skript auf unserem Ziel-Laptop ausführen, können wir die drahtlosen Netzwerke, mit denen der Computer verbunden war, und deren MAC-Adressen sehen. Achten Sie beim Testen des Skripts darauf, dass Sie es in einer Administratorkonsole ausführen, da Sie sonst nicht in der Lage sind, die Schlüssel zu lesen.

```
C:\Users\investigator\Desktop\python discoverNetworks.py
[*] Networks You have Joined.
[+] Hooters_San_Pedro, 00:11:50:24:68:7F
[+] LAX Airport, 00:30:65:03:e8:c6
[+] Senate_public_wifi, 00:0b:85:23:23:3e
```

3.2.2 Die MAC-Adresse mit Mechanize an Wigle übertragen

Damit sind wir aber noch nicht fertig. Über die MAC-Adresse eines drahtlosen Zugriffspunkts können wir auch dessen Standort ermitteln. Es gibt eine Reihe von Datenbanken – sowohl quelloffene als auch firmeneigene –, die riesige Listen von drahtlosen Zugriffspunkten und deren Standorten führen. Geräte wie Handys greifen auf diese Datenbanken zu, um auch ohne GPS ihren Standort bestimmen zu können.

Die Datenbank SkyHook, die unter <http://www.skyhookwireless.com/> zur Verfügung steht, bietet ein SDK (Software Developer Kit) für die Standortbestimmung auf der Grundlage der WLAN-Ortung an. Mehrere Jahre lang hat ein von Ian McCracken entwickeltes Open-Source-Projekt Zugriff auf diese Datenbank gegeben (<http://code.google.com/p/maclocate/>). Allerdings hat SkyHook schließlich sein SDK geändert und verwendet jetzt einen API-Schlüssel für die Interaktion mit der Datenbank. Google hat ebenfalls eine umfangreiche Datenbank für den Abgleich von MAC-Adressen mit den Standorten von Zugriffspunkten unterhalten, doch kurz nachdem Gorjan Petrovski ein NMAP-NSE-Skript zu ihrer Nutzung entwickelt hatte, gab Google die Open-Source-Interaktion mit der Datenbank auf (Google, 2009; Petrovski, 2011). Kurz darauf sperrte Microsoft aufgrund von Datenschutzbedenken eine ähnliche WLAN-Ortungsdatenbank (Bright, 2011).

Ein verbliebenes Open-Source-Datenbankprojekt, *wigle.net*, gewährt Benutzern jedoch nach wie vor die Möglichkeit, aus der Adresse eines Zugriffspunkt den Standort zu bestimmen. Nachdem Sie sich bei Wigle registriert haben, können Sie mit ein wenig kreativer Python-Skripterstellung mit der Datenbank arbeiten. Sehen wir uns kurz an, wie Sie ein solches Skript erstellen.

Um ein Ergebnis von Wigle zu erhalten, müssen Sie auf drei verschiedene Seiten zugreifen. Als Erstes öffnen Sie die Startseite auf <http://wigle.net>, dann melden Sie sich auf <http://wigle.net/gps/gps/main/login> an. Schließlich können Sie auf <http://wigle.net/gps/gps/main/confirmquery/> mit einer HTTP-Post-Abfrage den GPS-Standort der MAC-Adresse eines drahtlosen Netzwerkes abfragen. Wenn wir diese Abfrage abfangen, können wir erkennen, dass der Parameter `netid` die MAC-Adresse enthält.

```
POST /gps/gps/main/confirmquery/ HTTP/1.1
Accept-Encoding: identity
Content-Length: 33
Host: wigle.net
User-Agent: AppleWebKit/531.21.10
Connection: close
```

```
Content-Type: application/x-www-form-urlencoded
netid=0A%3A2C%3AEF%3A3D%3A25%3A1B
<... Gekürzt ...>
```

Die Antwort von der Seite enthält in dem String `maplat=47.25264359&maplon=-87.25624084` die geografische Breite und Länge des Zugriffspunkts:

```
<tr class="search"><td>
<a href="/gps/gps/Map/onlinemap2/?maplat=47.25264359&maplon=-
87.25624084&mapzoom=17&ssid=McDonald's
FREE Wifi&netid=0A:2C:EF:3D:25:1B">Get Map</a></td>
<td>0A:2C:EF:3D:25:1B</td><td>McDonald's FREE Wifi</td>
```

Damit wissen wir genug, um eine Funktion zu schreiben, die die Länge und Breite eines Zugriffspunkts aus der Wigle-Datenbank zurückgibt. Dazu verwenden wir die Bibliothek `Mechanize`, die auf <http://wwwsearch.sourceforge.net/mechanize/> zur Verfügung steht und eine zustandserhaltende Webprogrammierung in Python ermöglicht. Das bedeutet, dass das Authentifizierungscookie gespeichert und wiederverwendet wird, nachdem wir uns einmal korrekt beim Wigle-Dienst angemeldet haben.

Da das Skript kompliziert aussieht, wollen wir es gemeinsam durchgehen. Als Erstes erstellen wir eine Instanz des `Mechanize`-Browsers. Danach öffnen wir die Startseite von `wigle.net`. Anschließend übergeben wir unseren Benutzernamen und unser Passwort als Parameter und verlangen eine Anmeldung auf der entsprechenden Seite von Wigle. Nachdem wir erfolgreich angemeldet sind, erstellen wir eine HTTP-Post-Abfrage der Datenbank, wobei wir die gewünschte MAC-Adresse als Parameter `netid` angeben. Dann durchsuchen wir die Antwort auf unsere HTTP-Post-Anfrage nach den Strings `maplat=` und `maplon=` für die geografische Breite bzw. Länge und geben diese Koordinaten als Tupel zurück.

```
import mechanize, urllib, re, urlparse
def wiglePrint(username, password, netid):
    browser = mechanize.Browser()
    browser.open('http://wigle.net')
    reqData = urllib.urlencode({'credential_0': username,
                               'credential_1': password})
    browser.open('https://wigle.net/gps/gps/main/login', reqData)
    params = {}
    params['netid'] = netid
    reqParams = urllib.urlencode(params)
    respURL = 'http://wigle.net/gps/gps/main/confirmquery/'
    resp = browser.open(respURL, reqParams).read()
    mapLat = 'N/A'
    mapLon = 'N/A'
    rLat = re.findall(r'maplat=.*\&', resp)
    if rLat:
        mapLat = rLat[0].split('&')[0].split('=')[1]
    rLon = re.findall(r'maplon=.*\&', resp)
    if rLon:
        mapLon = rLon[0].split
    print '[-] Lat: ' + mapLat + ', Lon: ' + mapLon
```

Wenn wir unser ursprüngliches Skript um die Funktion zur Wigle-Abfrage ergänzen, können wir jetzt die Registrierung nach den Netzwerken durchsuchen, mit denen der Computer verbunden war, und deren Standorte nachschlagen.

```
import os
import optparse
import mechanize
import urllib
import re
import urlparse
from _winreg import *
```

```
def val2addr(val):
    addr = ''
    for ch in val:
        addr += '%02x %ord(ch)
    addr = addr.strip(' ').replace(' ', ':')[0:17]
    return addr

def woglePrint(username, password, netid):
    browser = mechanize.Browser()
    browser.open('http://wogle.net')
    reqData = urllib.urlencode({'credential_0': username,
                               'credential_1': password})
    browser.open('https://wogle.net/gps/gps/main/login',
                reqData)
    params = {}
    params['netid'] = netid
    reqParams = urllib.urlencode(params)
    respURL = 'http://wogle.net/gps/gps/main/confirmquery/'
    resp = browser.open(respURL, reqParams).read()
    mapLat = 'N/A'
    mapLon = 'N/A'
    rLat = re.findall(r'maplat=.*&', resp)
    if rLat:
        mapLat = rLat[0].split('&')[0].split('=')[1]
    rLon = re.findall(r'maplon=.*&', resp)
    if rLon:
        mapLon = rLon[0].split
    print '[-] Lat: ' + mapLat + ', Lon: ' + mapLon

def printNets(username, password):
    net = \
        "SOFTWARE\Microsoft\Windows
        NT\CurrentVersion\NetworkList\Signatures\Unmanaged"
    key = OpenKey(HKEY_LOCAL_MACHINE, net)
    print '\n[*] Networks You have Joined.'
```

```
for i in range(100):
    try:
        guid = EnumKey(key, i)
        netKey = OpenKey(key, str(guid))
        (n, addr, t) = EnumValue(netKey, 5)
        (n, name, t) = EnumValue(netKey, 4)
        macAddr = val2addr(addr)
        netName = str(name)
        print '[+] ' + netName + ' ' + macAddr
        wiglePrint(username, password, macAddr)
        CloseKey(netKey)
    except:
        break

def main():
    parser = \
        optparse.OptionParser("usage%prog "+
            "-u <wigele username> -p <wigele password>")
    parser.add_option('-u', dest='username', type='string',
        help='specify wigele password')
    parser.add_option('-p', dest='password', type='string',
        help='specify wigele username')
    (options, args) = parser.parse_args()
    username = options.username
    password = options.password
    if username == None or password == None:
        print parser.usage
        exit(0)
    else:
        printNets(username, password)

if __name__ == '__main__':
    main()
```

Wenn wir das Skript jetzt mit der neuen Funktion ausführen, sehen wir die Netzwerke, mit denen der Computer zuvor verbunden war, und deren Standorte. Im nächsten Abschnitt schauen wir uns an, wie wir den Papierkorb untersuchen.

Stichwortverzeichnis

Symbole

>>> 33
 802.11-Suchanfragen 272
 _getexif() 157
 !lazor 208
 _winreg 138

A

acquire() 75
 ADMIN\$ 115
 Administrative Netzwerkfreigabe 115
 Aircrack-ng 253
 Amanda 244
 Anonymität
 Anonyme Anmeldung an Social-
 Media-Websites 341
 Anonymer Browser 315
 Anonymer FTP-Zugriff 100
 Benutzeragentenstrings 312
 Cookies anzeigen 314
 E-Mail 340
 Proxyserver 311
 Verzögerung zwischen Anfragen 316
 Virtuelle private Netzwerke 311
 Wiederholter Zugriff auf Webseiten
 als jeweils neuer Besucher 317
 Anonymous
 Botznetz 97
 Hive 206
 LOIC 97, 204
 Metadaten-Panne 152

 Operation Payback 204
 Teilnahme an DDoS-Angriffen nach-
 weisen 210
 Antivirussoftware
 Gefälscht 112
 NoVirusThanks 357
 Umgehen 352
 Umgehung bestätigen 357
 Umgehung durch Flame 352
 Anwendungsbanner 38, 72
 append() 36
 aptitude 32
 Aufklärung
 Anonym 311
 Anonyme Anmeldung an Social-
 Media-Websites 341
 Aufklärungsscans fälschen 244
 Bilder herunterladen 321
 Einführung 308
 Google 324
 HREF-Links abrufen 318
 Interessen von Twitter-Benutzern
 ermitteln 334
 Ortsangaben aus Tweets 331
 Portscan 68
 Social Media 331
 Twitter 328
 Twitter-Benutzer auskundschaften
 338
 Webseiten abrufen 309
 Webseiten untersuchen 318
 Ausführbare Windows-Dateien 355,
 363

- Ausnahmebehandlung 40, 60, 75
- Authentifizierung
 - Adressüberprüfung zur Authentifizierung 231
 - Anmeldung ohne Kenntnis der Authentifizierungsinformationen 173
 - Anonymer FTP-Zugriff 100
 - Cookies 142, 173
 - FTP 100
 - Hotel-WLANs 261
 - Öffentliche Schlüssel 90
 - RFCOMM 297, 301
 - Rlogin 231
 - SSH 83, 90

B

- Beautiful Soup 155, 318
- Bedingte Anweisungen 39, 220
- Benutzeragentenstrings 312
- Benutzernamen ermitteln 147
- Bibliotheken
 - Analyse von XML-Ausgaben 78
 - aptitude 32
 - Bluetooth 255
 - Cookielib 314
 - crypt 53
 - Dpkt 195, 205
 - Drittanbieterbibliotheken 29
 - easy_install 31
 - Eigene Bibliothek zur Paketanalyse anlegen 280

- Fremde Funktionsbibliotheken
 - importieren 354
- Ftplib 101
- hashlib 56
- Httpplib 358
- Imaging 157
- Installieren 29
- IPy 220
- JSON 325
- Mechanize 142, 309
- Namenskonflikte vermeiden 220
- Nmap 117
- Online-Dokumentation 48
- optparse 62, 70
- os 49, 156
- Pexpect 82
- PyBluez 291
- PyGeoIP 194
- PyPDF 152
- Python-Nmap 29, 78
- re 256
- Reguläre Ausdrücke 256
- Scapy 218
- smtplib 342
- socket 38, 70
- Sqlite3 174
- sys 48
- urllib 325
- urllib2 156, 337
- urlparse 156
- _winreg 138
- zipfile 58

- Bilder
 - Exif-Metadaten 154, 157
 - GPS-Metadaten 157
 - Herunterladen 155, 321
 - Bindshell 353
 - BlueBug 297, 301
 - Bluetooth
 - BlueBug 297, 301
 - Bluetooth-Adressen über WLAN-Datenverkehr ermitteln 294
 - Cabir 301
 - Datenverkehr zum Ermitteln von Adressen abfangen 293
 - Dienstermittlungsprotokoll 299
 - Discovery-Modus 291
 - Drucker übernehmen 300
 - Flame 352
 - Gerätenamenanfragen 295
 - Gerätenamen aus MAC-Adressen ermitteln 291
 - MAC-Adressen ermitteln 291
 - Privatmodus 293
 - Python-Pakete 255
 - RFCOMM 297
 - SDP 299
 - Botnetze
 - Anonymous 97
 - SSH 80, 95
 - Storm 223
 - Browser
 - Anonymer Browser 315
 - Benutzeragentenstrings 312
 - Leerzeichenkodierung 266
 - Mechanize-Klasse 309
 - URL-Kodierung 325
 - Verlauf 175
 - Wget 309
 - BTK-Killer 136
 - Butler, Max Ray 252
- ## C
- Cabir 301
 - Castillo, Albert 57
 - Christmas-Scan 79
 - command() 84
 - Conficker
 - Datenverkehr analysieren 228
 - Domain Flux 223
 - Funktionsweise 115
 - Gesamtskript 120
 - Geschichte 113
 - Passwörter 113
 - connect() 38, 83
 - Cookielib 314
 - Cookies
 - Anzeigen 314
 - Authentifizierung 142, 173
 - Wiederholter Zugriff auf Webseiten als jeweils neuer Besucher 317
 - Wiederverwenden 285
 - Wiederverwendung erkennen 288
 - Wordpress-Sitzungsscookies 286
 - create_connection() 70
 - crypt 53
 - Cybercop 244

D

Datei-E/A

- Berechtigungen 49
- Textdateien lesen 47
- Zugriff auf nicht vorhandene Dateien 50

Datenverkehr

- 802.11-Pakete erstellen 279
- 802.11-Suchanfragen 272
- Analysieren 195
- Art des Datenverkehrs aus Port bestimmen 202
- Aufzeichnung 195
- Bluetooth-Adressen abfangen 293
- Conficker 228
- Cookies abfangen 285
- DDoS-Angriffe erkennen 209
- DNS 225
- Domain-Flux 228
- Dpkt 195
- Drohnen 276
- Fast-Flux 226
- FTP-Anmeldeinformationen ausspionieren 269
- Google-Suchabfragen 264
- Hive-Befehle für LOIC 207
- ICMP-Echoanforderungen 220
- In Google Earth darstellen 200
- Kreditkarteninformationen ausspionieren 259
- LOIC 204, 205, 207
- Paketanalyse 198

Pcap 195

Proxyserver 311

- Quell- und Zieladressen ausgeben 196
- Server stummschalten 231
- Social-Media-Konten übernehmen 285
- TCP-Folgeummern berechnen 233
- TCP-Verbindungen fälschen 236
- TTL-Felder 215, 218
- Virtuelle private Netzwerke 311
- WLAN-Datenverkehr 253, 256
- WLAN-Keylogger 264
- WLAN-Sniffer 262
- datetime() 163
- DDoS-Angriffe 209
- Debian 90
- def() 42
- Dictionarys 37, 326
- DisablePayloadHandler 118
- Discovery-Modus 291
- Division durch null 40
- DNS
 - Abfrageeintrag 225
 - Datenverkehr analysieren 225
 - Domain-Flux 228
 - Fast-Flux 226
 - Ressourceneintrag 225
- DNSQR 225
- DNSRR 225
- Domain-Flux 224, 228
- Domänencontroller 28

- Dpkt 195, 205
- Drahtlose Netzwerke. Siehe WLAN
- Drei-Wege-Handshake 69
- Drohnen
 - 802.11-Pakete erstellen 279
 - Datenverkehr abfangen 276
 - Einführung 275
 - Navigationsprotokoll 277
 - Not-Aus-Befehl 282
 - Zum Absturz bringen 283

E

- easy_install 31
- EIP-Register 124
- E-Mail
 - Absenderadresse fälschen 343
 - Anonym 340
 - Maßgeschneiderte Gestaltung des Textes 344
 - Offene Relays 343
 - Port 69
 - Python-E-Mail-Client 342
 - Spear Phishing 344
 - Wegwerfkonten 341
- ENISA 226
- Entwicklungsumgebung 29
- Exif 154, 157
- Exiftool 154
- expect() 83
- Exploits
 - Exploitsignaturen fälschen 241
 - FTP 125

- IDS-Regeln 241
- ms08_067_netapi 115
- ms10_002_aurora 105
- Operation Aurora 192
- psexec 120
- Pufferüberläufe 124
- Remoteausführung von Prozessen 120
- Selbst schreiben 124
- Senden 127
- SMB-Dienst 115
- Zero-Day-Angriffe 124
- extractall() 58

F

- Facebook 285
- Fast-Flux 223, 226
- file 183
- finally 75
- find() 36
- findTags() 117
- finger 68
- FIN-Scan 79
- Firefox
 - Browserverlauf 175
 - Cookies 173
 - Datenbanken 171
 - Datenbankverschlüsselung 174
 - Heruntergeladene Dateien 172
 - WAL-Journalmodus 174
- Firesheep 285
- Flame 352

for 44
FreeFloat 74, 130
FTP
 Anmeldeinformationen ausspionieren 102, 269
 Anonymer Zugriff 100
 Anwendungsbanner 74
 Brute-Force-Angriff 102
 Exploits 125
 FreeFloat 74, 130
 Ftplib 101
 k985ytv 99
 Kombinierter FTP/Web-Angriff 99
 Passwörter 102
 Port 69
 Schwachstellen suchen 39
 Webzugriff über FTP-Server 104
Ftplib 101
Funktionen 42, 55, 61

G

Gefilterte Ports 80
GeoIP 194
GeoLiteCity 193
getDocumentInfo() 152
gethostbyaddr() 70
gethostbyname() 70, 71
Gonzalez, Albert 260
Google
 API 324
 Aufklärung 324
 Google Earth 200

Skripts für die Suche in Google 324
Snifferskript 266
Street View 268
Suchabfragen abfangen 264
Suchparameter 265
Übersichtlichere Darstellung der Suchergebnisse 327
GPS
 Google Street View 268
 GPS-Metadaten von Bildern 157
 Standortspeicherung in iOS 182
GPSInfo 157
Gslite 268

H

Hash 53
hashlib 56
Hello World 33
help() 58
Hilton, Paris 301
Hivemind-Ermittlungsfunktion 208
Hotelgäste ausspionieren 261
HREF-Links 318
Httpplib 358

I

Iceman 252
ICMP-Echoanforderungen 220
if 39
ifconfig 253
IFrame 106

- iKee 89
 - Imaging 157
 - Intrusion-Detection-Systeme
 - Durch Fehlalarme unterlaufen 240
 - SSH-Angriffe 81
 - iOS-Backup 181
 - IP-Adressen
 - Adressüberprüfung zur Authentifizierung 231
 - Aktuelle IP-Adresse anzeigen 311
 - Auflösen 71
 - DNS-Ressourceneintrag 225
 - Domänennamen mit mehreren IP-Adressen 226
 - Fast-Flux-Technik 226
 - In Strings umwandeln 196
 - LOIC-Download 205
 - Nmap-Scan 215
 - Proxyserver 311
 - Quell- und Zieladressen ausgeben 196
 - Scheinadresse 217
 - Standorte in Google Earth darstellen 200
 - Standort ermitteln 193, 194
 - iPhone
 - Drohnensteuerung 276
 - Exif-Daten 154
 - Gerätenamenanfragen 295
 - Gespeicherte Textnachrichten 184
 - iTunes-Backup 182
 - Jailbreak 89
 - Speicherung von GPS-Koordinaten 181
 - Standardpasswort 89
 - IPy 220
 - iscover_devices() 291
 - items() 38
 - Iterationen 44
 - iTunes-Backup 182
- J**
- Jailbreak 89
 - JSON 325
- K**
- k985ytv 99, 108, 112
 - Kali Linux 31
 - Keylogger 264
 - keys() 38
 - Kinderpornografie 57
 - Klassen 95
 - KML-Dateien 200
 - Kommandozeilenargumente 48, 62, 70
 - Kompilierung 355
 - Kreditkarteninformationen 256
 - Kuckucksei 52
- L**
- l0pht 57
 - Leistung 62
 - Linux-Überlaufangriffe 243
 - Listen 36

LOIC

- Angriff erkennen 209
- Botnetz 97
- Download 205
- Download nachweisen 205
- Funktionsweise 204
- Hive 206
- Hivemind-Ermittlungsfunktion 208
- lookup_name() 291
- lower() 36
- Low Orbit Ion Cannon. Siehe LOIC

M

MAC-Adressen

- Bluetooth-Geräte 291
 - Standort ermitteln 140
 - Umwandeln in Gerätenamen 291
 - Verborgene Netzwerke 273
 - WLAN-Gateway 138
- main() 34
- MaxMind 193
- Mechanize 142, 309
- Metadaten
- Anonymous 152
 - Einführung 150
 - Exif 154, 157
 - PDF 151
- Metasploit
- Angriff auf mehrere Ziele 119
 - FileZilla-Anmeldeinformationen 102
 - Interaktive Nutzung 117

Kodierer 363

- Meterpreter 118
- Multihandler 118
- psexec 120
- Schadcode in ausführbare Windows-Dateien umwandeln 363
- Schadseiten erstellen 105
- SMB-Dienst 115
- Meterpreter 118
- Mitnick, Kevin 230
- Moore, H. D. 90, 215
- Morris, Robert Tappan 68
- Morris-Wurm

 - Geschichte 68
 - Passwörter 113
 - RSH-Passwörter 80

- ms08_067_netapi 115
- ms10_002_aurora 105
- Multihandler 118

N

- Netzwerkdatenverkehr. Siehe Datenverkehr
- Netzwerkverbindungen
- Bevorzugte Netzwerke von Laptops feststellen 271
 - Python 38
 - Rückwärtige Verbindung 118
 - Server stummschalten 231
 - Sockets 38
 - TCP/IP-Sockets 69

- TCP-Verbindungen fälschen 236
- Verbindungen herstellen 71
- Virtuelle private Netzwerke 311
- Nlog 216
- nlst 104
- Nmap
 - Angriff auf Pentagon 215
 - IP-Quelladresse eines Nmap-Scans
 - ermitteln 215
 - Portscanner 78
 - Python-Nmap 29, 78
 - Scheinadresse 217
 - Scheinscans 215
 - SMB 117
 - TTL 215
- NOAA 311
- NOP-Schlitten 125
- NoVirusThanks 357
- Null-Scan 79

O

- OBEX 299
- Objektorientierte Programmierung 95
- Offene Relays 343
- OpenKey() 138
- OpenSSH 89
- Operation Aurora 105, 192, 307, 323
- Operation Payback 152, 204
- OptionParser() 71
- optparse 62, 70
- os 49, 156
- OUI-Datenbank 294

P

- Paketanalyse 198
- Papierkorb
 - Benutzernamen zu den Dateien
 - ermitteln 148
 - Einführung 146
 - Gelöschte Elemente finden 146
 - Speicherort 146
- Passwörter
 - Brute-Force-Angriff 53, 60
 - Conficker 113
 - Entschlüsseln 52
 - FTP 102
 - Hash berechnen 53
 - Hash herunterladen 85
 - iPhone-Standardpasswort 89
 - Klartext 102
 - Morris-Wurm 80, 113
 - Passwortknacker für SSH 86
 - Passwortknacker für UNIX 53
 - Passwortknacker für Zip-Dateien 58
 - Pxssh 86
 - RSH 80
 - Salt 53
 - SMB 119
 - Speicherort des Hashes 56
 - SSH 81, 86
 - Wörterbuchangriff 53, 60
 - Zip-Dateien 58
- Pcap 195
- PDF-Metadaten 151
- Peek-A-Boo 256

- Penetrationstests
 - Beispiel 27
 - Domänencontroller 28
 - Kali Linux 31
 - Metasploit 115
 - Python 27, 67
- Pexpect 82
- Pfadnamen 177
- Portscanner
 - Anwendungsbanner abrufen 72
 - Arten von Scans 78
 - Einführung 68
 - Funktionsweise 70
 - Gefilterte Ports 80
 - Kommandozeilenoptionen 70
 - Nmap 78
 - Offene Ports melden 71
 - TCP-Ports 69
 - TCP-Verbindungsscan 69
 - Threads 74
 - Vollständiges Skript 76
- PortScanner() 79
- print() 32
- psexec 120
- Pufferüberläufe 124
- Pxssh 86
- PyBluez 291
- PyGeoIP 194
- Pyinstaller 355
- PyPDF 152
- Python
 - >>> 33
 - 2.x und 3.x 32
 - aptitude 32
 - Ausführbare Windows-Dateien 355
 - Ausnahmebehandlung 40, 60
 - Bedingte Anweisungen 39, 220
 - Bluetooth-Pakete 255
 - Branches 32
 - Datei-E/A 47
 - Dictionarys 37
 - Drittanbieterbibliotheken 29
 - easy_install 31
 - Einführung 34
 - Entwicklungsumgebung einrichten 29
 - Exploits selbst schreiben 124
 - Fremde Funktionsbibliotheken
 - importieren 354
 - Funktionen 42, 55, 61, 315
 - Gute Programmierpraktiken 55
 - Hello World 33
 - Installer 29, 32
 - Interaktive Nutzung 33
 - Interpreter 33
 - Iterationen 44
 - Klassen 95, 315
 - Kommandozeilenargumente 62
 - Leistung steigern 62
 - Lesbarkeit erhöhen 315
 - Listen 36
 - main() 34
 - Netzwerkverbindungen 38
 - Nmap 117

- Objektorientierte Programmierung
 - 95
 - Pakete installieren 29
 - Penetrationstests 27, 67
 - print() 32
 - Pyinstaller 355
 - Python-Code kompilieren 355
 - Reguläre Ausdrücke 176
 - Semaphoren 75
 - Skripts modularisieren 61
 - Strings 36
 - Surfen im Internet 309
 - TCP/IP-Sockets 69
 - TCP-Verbindungsscan 69
 - Threads 62
 - Variablen 35
 - Zustandserhaltende Web-
 - programmierung 142
 - Python-Nmap 29, 78
- Q**
- quote_plus() 325
- R**
- Rader, Denis 136
 - readlines() 48
 - re (Bibliothek) 256
 - recv() 38
 - Redirect 106
 - Reguläre Ausdrücke
 - Hashtags 338
 - HREF-Links 318
 - Interessen von Twitter-Benutzern
 - 334
 - Kreditkarteninformationen 256
 - Twitter-Benutzer 338
 - URLs 176
 - Remote Shell 80
 - replace() 36
 - RFCOMM 297
 - Rlogin 231
 - Root-Zugriff 56
 - Rossum, Guido van 32
 - RSH 80
- S**
- Salt 53
 - scan() 79
 - Scapy 218
 - Schleifen 44, 45
 - Schwachstellen-Scanner 50
 - SDP 299
 - Secure Shell. Siehe SSH
 - SELECT 162
 - Semaphoren 75, 87
 - sendmail 68
 - Server stummschalten 231
 - SHA-512 56
 - ShadowCrew 260

- Shell
 - Bindshell 353
 - Pufferüberlauf 124
 - Umgekehrte TCP-Shell 106
 - Windows-Befehlshell 116
- Shimomura, Tsutomu 230
- Sicherheit durch Verschleierung 317
- SID 147
- Skripts
 - 802.11-Suchanfragen abfangen 272
 - Anonymer Browser 315
 - Benutzeragentenstring ändern 313
 - Bilder herunterladen 322
 - BlueBug 302
 - Bluetooth-Geräte aufspüren 292
 - Bluetooth-Geräte im Privatmodus finden 296
 - Conficker nachstellen 120
 - Cookies anzeigen 314
 - Datenbankabfragen automatisieren 163
 - Daten zu NoVirusThanks hochladen 357
 - DDoS-Angriffe erkennen 210
 - Domain-Flux-Datenverkehr erkennen 228
 - Drohnen zum Absturz bringen 283
 - E-Mails senden 342
 - Exploits selbst schreiben 124
 - Fast-Flux-Datenverkehr erkennen 226
 - Fehlalarme für IDS generieren 241, 244
 - Firefox-Profil untersuchen 178
 - FTP-Anmeldeinformationen ausspionieren 270
 - FTP/Web-Angriff 107
 - Gespeicherte Cookies ermitteln 173
 - Google-Snifferskript 266
 - Google-Suche 325
 - Hivemind-Ermittlungsfunktion 208
 - HREF-Links abrufen 319
 - HTML-Code von Webseiten abrufen 309
 - Interaktive Nutzung von Metasploit 117
 - Interessen von Twitter-Benutzern ermitteln 334
 - iOS-Datenbank mit Textnachrichten ausgeben 186
 - Kreditkarteninformationen ausspionieren 259
 - LOIC-Ermittlungsprogramm 205
 - Maßgeschneiderte E-Mails 345
 - Metadaten analysieren 151
 - Mitnick-Angriff nachstellen 237
 - Navigationsbefehle für Drohnen abfangen 277
 - Nmap-Portscanner 78
 - Ortsangaben aus Tweets 331
 - Paketanalyse 198
 - Passwortknacker für Zip-Dateien 58
 - Portscanner 68
 - Proxyserver festlegen 311
 - Quell- und Zieladressen ausgeben 196

- Shellcode 354
- Skype-Profildatenbanken untersuchen 167
- Skype-Profile ausgeben 163
- Spear Phishing 345
- SSH-Botnetz 80
- SSH-Passwortknacker 86
- SSH-Wurm 82
- Standorte von IP-Adressen in Google Earth darstellen 200
- SYN-Flood 232
- TCP-Folgenummern vorhersagen 237
- Teilnahme an DDoS-Angriffen nachweisen 210
- TTL-Felder analysieren 218
- Tweets abrufen 329
- Twitter-Benutzer auskundschaften 338
- UNIX-Passwortknacker 53
- Verborgene Netzwerke enttarnen 274
- Wiederholter Zugriff auf Webseiten als jeweils neuer Besucher 317
- Windows-Papierkorb untersuchen 146
- WLAN-Informationen aus der Registrierung abrufen 138
- WLAN-Sniffer 262
- SkyGrabber 275
- SkyHook 141
- Skype
 - Datenbankabfragen automatisieren 163
 - Datenbanken 161
 - Kontaktdatenbank 164
 - Kontoinformationen finden 162
 - Konversationen 165
 - Nachrichten abrufen 166
 - Profil ausgeben 163
 - Profildatenbanken untersuchen 167
- SMB
 - Conficker 224
 - Metasploit 115
 - Passwörter 119
 - Port 117
- smtp lib 342
- Sniffer 262, 266
- SNORT 240
- Social Engineering
 - Absenderadresse fälschen 343
 - Aufklärung 309
 - E-Mails senden 342
 - Interessen von Twitter-Benutzern ermitteln 334
 - Massenangriff 341
 - Maßgeschneiderte Gestaltung von E-Mails 344
 - Spear Phishing 344
 - Zielpersonen ausspähen 323
- Social Media
 - Anonyme Anmeldung 341
 - Aufklärung 331

- Facebook 285
 - Firesheep 285
 - Konten übernehmen 285
 - Twitter 329
 - socket() 70
 - socket (Modul) 38, 70
 - Spear Phishing 344
 - spec-Dateien 355
 - SQLite
 - Browserverlaufsdatenbank 175
 - Cookie-Datenbank 173
 - Datenbankabfragen automatisieren 163
 - Einführung 161
 - Firefox 171
 - Informationen aus verknüpften Tabellen abrufen 165
 - iOS-Backup 183
 - Schemadatenbank 184
 - SELECT 162
 - Skype 161, 167
 - Tabellenabfragen 162
 - Verschlüsselung 174
 - WHERE-Klausel 165
 - Sqlite3 161, 174
 - SSH
 - Angriffe 81
 - Befehle senden 84
 - Benutzerinteraktion 82
 - Botnetze 80, 95
 - OpenSSH 89
 - Passwörter knacken 86
 - Pexpect 82
 - Pxssh 86
 - Schwache private Schlüssel 90
 - SSH-Sitzungen interaktiv steuern 84
 - SSH-Wurm 82
 - Verbindung erstellen 83
 - Verschlüsselung 81
 - Zeitüberschreitung 83
 - Stack-Pufferüberläufe 124
 - Stoll, Clifford 52
 - Stone, Randy 135
 - Storm 223
 - str() 35
 - Strings 36, 196
 - strip() 48
 - Stuxnet 67, 308, 323
 - SYN-ACK-Pakete 233
 - SYN-Flood 231, 232
 - SYN-Scan 78
 - sys 48
- ## T
- Tapanaris, Alex 152
 - TCP
 - ACK-Paket 69
 - Art des Datenverkehrs aus Port bestimmen 202
 - Arten von Scans 78
 - DDoS-Angriffe 209
 - Drei-Wege-Handshake 69, 231
 - FIN-Scan 79
 - Folgennummern vorhersagen 230, 233

- Null-Scan 79
 - Ports 69
 - Rückwärtige Verbindung 118
 - Server stummschalten 231
 - SMB-Port 117
 - SYN-Flood 231
 - SYN-Paket 69
 - SYN-Scan 78
 - TCP-Verbindungsscan 69
 - Umgekehrte TCP-Shell 106
 - Verbindungen fälschen 236
 - Weihnachtsbaum-Scan 79
 - Zufälligkeit von Folgenummern 234
 - TCP/IP-Sockets 69
 - TFN-Angriffe 241
 - Threads 62, 74
 - Tor-Netzwerk 311
 - Traffic. Siehe Datenverkehr
 - try/except 40
 - TTL
 - Analysieren 218
 - Betriebssystemabhängige Werte 222
 - Funktionsweise 216
 - Gefälschte TTL-Werte erkennen 220
 - Moore, H. D. 215
 - Nmap-Scheinscans 216
 - Twitter
 - Abgekürzte URLs 337
 - API 329
 - Hashtags 338
 - Interessen von Twitter-Benutzern ermitteln 334
 - Ortsangaben 331
 - Tweets abrufen 329
- ## U
- Übergabe als Referenz 42
 - Unix-Epochenformat 163
 - upper() 36
 - URL-Kodierung 266, 325
 - urllib 325
 - urllib2 156, 337
 - urlparse 156
- ## V
- Variablen 35
 - Vaskovich, Fyodor 78
 - Verschlüsselung
 - Debian-Bug 90
 - Entropie 90
 - Hash 53
 - Passwörter 52
 - Schwache private Schlüssel 90
 - SQLite 174
 - SSH 81, 90
 - Virtuelle private Netzwerke 311
 - VPN 311
- ## W
- W32DownadUp. Siehe Conficker
 - Wagenrücklaufzeichen 48
 - WAL-Journalmodus 174

- Wall of Sheep 256
- Watt, Steven 260
- Webseiten
 - Anmeldung ohne Authentifizierungs-
informationen 173
 - Bilder herunterladen 156, 321
 - Cookie-Authentifizierung 173
 - Cookies abfangen 285
 - Exif-Metadaten von Bildern untersu-
chen 155
 - HREF-Links abrufen 318
 - Links öffnen 337
 - Quellcode abrufen 309
 - Redirect 106
 - Schadcode injizieren 105
 - Schadseiten erstellen 105
 - Social-Media-Konten übernehmen
285
 - Untersuchen 318
 - Wiederholter Zugriff auf Webseiten
als jeweils neuer Besucher 317
 - Wiederverwendung von Cookies
erkennen 288
- Wegwerfkonten 341
- Weihnachtsbaum-Scan 79
- wget 85
- WHERE 165
- Wigle 141
- Windows-Registrierung
 - Benutzer-SIDs 147
 - Einführung 137
 - Lesen 138
 - Netzwerknamen 138
 - WinReg 138
 - WLAN-Informationen 137
- WinReg 138
- WLAN
 - 802.11-Pakete erstellen 279
 - 802.11-Suchanfragen 272
 - Authentifizierung 261
 - Bevorzugte Netzwerke von Laptops
feststellen 271
 - Bluetooth-Adressen 294
 - Cookies wiederverwenden 285
 - Datenverkehr belauschen 253, 256
 - Drohnen-Datenverkehr abfangen
276
 - Firesheep 285
 - FTP-Anmeldeinformationen aus-
spionieren 269
 - Google-Suchabfragen 264
 - Hotels 261
 - Keylogger 264
 - Koordinaten des Zugriffspunkts
ermitteln 142
 - Kreditkarteninformationen ausspi-
onieren 259
 - MAC-Adresse 138
 - Netzwerke ermitteln, mit denen ein
Computer verbunden war 139
 - Netzwerkkarte 252
 - Netzwerknamen abrufen 138
 - Netzwerkprofile 138
 - Sniffer 262

Social-Media-Konten übernehmen
285
Standort aus MAC-Adresse ermitteln
140
Überwachungsmodus 252
Verborgene Netzwerke finden 273
Windows-Registrierung 137
Wordpress 285
Wörterbuchangriff 53

X

Xmas-Scan 79

Z

Zatko, Peiter 57, 114
Zeitüberschreitung 83
Zimmerman, John 57
Zip-Dateien 58
zipfile 58
Zustandserhaltende Web-
programmierung 142

T. J. O'Connor



Python Hacking

Schon einmal selbst gehackt? Na, dann wird es aber Zeit – lernen Sie, wie Hacker mit Python Systeme angreifen und Schwachstellen ausnutzen. Durch Einbindung vorhandener Werkzeuge wie Metasploit und Nmap werden Skripte erschreckend einfach. Nutzen Sie dieses Wissen, um Ihre Systeme auf Lücken zu testen, und schließen Sie diese, bevor andere Ihnen zuvorkommen. Das erlangte Python-Wissen können Sie nicht nur für das Hacken einsetzen, sondern zum Beispiel auch zur Fernsteuerung von Programmen mithilfe von Pexpect.

Bereiten Sie sich vor

Wenn Sie bisher wenig Erfahrung in der Programmierung mit Python haben, dann lernen Sie in Kapitel 1 die notwendigen Grundlagen, um die Beispiele in den darauffolgenden Kapiteln zu verstehen: angefangen bei Variablen und der Behandlung von Ausnahmen über Funktionen bis zu wichtigen Modulen für das Hacking wie sys und os. Diese Grundlagen werden anhand eines Beispiels eingeführt. Schritt für Schritt wird ein Schwachstellenscanner entwickelt. Zum Abschluss des Kapitels bauen Sie ein Skript, um UNIX-Passwörter zu knacken. Danach werden Ihre Passwörter sicher anders aussehen.

Am Quellcode lernen

Hacken und Programmieren lernt man nicht alleine mit Theorie, sondern mit viel Praxis – in diesem Fall mit Quellcode. Alle Hacks, ob SSH-Botnetz, Portscanner oder Analyse des DNS-Datenverkehrs, werden mit aussagekräftigem Quellcode beschrieben. Die relevanten Bestandteile des Quellcodes werden verständlich erklärt. Die Quellcodes der einzelnen Hacks stehen als separater Download zur Verfügung.

Aus dem Inhalt:

- Entwicklungsumgebung aufsetzen
- Programmierung mit Python
- Penetrationstests mit Python
- Einen Portscanner schreiben
- SSH-Botnetz mit Python aufbauen
- SSH-Passwörter mit Pssh knacken
- Kombiniertes Massenangriff über FTP und das Web
- Nutzung von Metasploit und Nmap
- Eigenen Zero-Day-Angriffscode schreiben
- Kernelemente des Exploits
- PDF-Metadaten mit PyPDF analysieren
- SQLite-Datenbanken untersuchen
- Analyse des LOIC-Datenverkehrs
- DNS-Datenverkehr mit Scapy analysieren
- Umgebung für WLAN-Angriffe einrichten
- WLAN-Keylogger für Google-Abfragen schreiben
- Drohnen zum Absturz bringen
- Spionieren mit Bluetooth und Python
- Einführung in Social Engineering
- Webseiten mit anonBrowser untersuchen
- Tweets mit Python analysieren
- Spear Phishing mit Smtplib
- Antivirenprogrammen ausweichen

Der komplette Quellcode aus dem Buch auf www.buch.cd



9 783645 604154

Besuchen Sie
unsere Website
www.franzis.de

FRANZIS