python™

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

FRANZIS

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.

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

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

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*.

3

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 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
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 <wigle username> -p <wigle password>")
   parser.add_option('-u', dest='username', type='string',
      help='specify wigle password')
   parser.add_option('-p', dest='password', type='string',
      help='specify wigle 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

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

X

Z

T. J. O'Connor

python™

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:

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

9 783645 604154