Zertifikatsprogramm - Z203

Python 2 - Penetration Testing

- Netzwerkforensik mit Python
- Penetrationstests mit Python
- Python-Hacks

Prof. Dr. Martin Rieger
Patrick Eisoldt, M.Eng.
David Schlichtenberger, M.Sc.
Benjamin Welte, M.Eng.
Christian Schneider, B.Eng.

Open C³S
Open Competence Center for Cyber Security

Hochschule
Albstadt-Sigmaringen
Albstadt-Sigmaringen University

Prof. Dr. Martin Rieger
rieger@hs-albsig.de
07571 / 732 - 9124

Benjamin Welte, M.Eng.
welte@hs-albsig.de
07571 / 732 - 9554

Python 2 – Penetration Testing

Studienbrief 1: Netzwerkforensik mit Python

Studienbrief 2: Penetrationstests mit Python

Studienbrief 3: Python-Hacks

Autoren:

Prof. Dr. Martin Rieger

Patrick Eisoldt, M.Eng.

David Schlichtenberger, M.Sc.

Benjamin Welte, B.Eng.

Christian Schneider, B.Eng.

---

2. Auflage

Hochschule Albstadt-Sigmaringen

Institut für wissenschaftliche Weiterbildung
Open C^{3}S | Zertifikatsprogramm
Steinachstraße 11
72336 Balingen

2. Auflage (31. August 2017)

Um die Lesbarkeit zu vereinfachen, wird auf die zusätzliche Formulierung der weiblichen Form bei Personenbezeichnungen verzichtet. Wir weisen deshalb darauf hin, dass die Verwendung der männlichen Form explizit als geschlechtsunabhängig verstanden werden soll.

Inhaltsverzeichnis

Einleitung zu den Studienbriefen

I. Abkürzungen der Randsymbole und Farbkodierungen

Beispiel	B
Definition	D
Exkurs	E
Kontrollaufgabe	K
Quelltext	Q
Übung	Ü

II. Zu den Autoren

Prof. Dr. Martin Rieger studierte Elektro- und Informationstechnik an der Technischen Universität München und schloss an derselben Hochschule die Promotion mit Auszeichnung ab. Ein Schwerpunkt seiner Forschungsarbeit lag in der Erstellung von Methoden und Werkzeugen zur Modellierung sowie Analyse und Optimierung elektrischer Schaltungen. Er war fünf Jahre Leiter des Labors für schnelle Analog-ICs in der IC-Entwicklungsabteilung des Forschungs- und Entwicklungszentrums der Firma Thomson Multimedia in Villingen. In der Zeit bei Thomson Multimedia war er Erfinder bzw. Miterfinder an 15 deutschen, 12 europäischen und 8 weltweiten Patenten.

Seit 1993 ist er als Professor an der Fakultät Engineering der Hochschule Albstadt-Sigmaringen auf den Gebieten Informatik und Informationstechnik tätig. Im Labor für Eingebettete Systeme und IT-Sicherheit betreibt er anwendungsnahe Forschung auf den Gebieten Embedded Systems und IT-Sicherheit. Er hatte über viele Jahre an der Hochschule Albstadt-Sigmaringen Positionen als Studiendekan, Prodekan, Prorektor und Rechenzentrumsleiter inne.

Prof. Dr. Rieger ist Initiator und Gründungs-Studiendekan des Master-Studiengangs Digitale Forensik, der in Kooperation mit der Friedrich-Alexander Universität Erlangen-Nürnberg und der Ludwig-Maximilians-Universität München betrieben wird.

Er leitet das vom BMBF geförderte Zertifikatsprogramm Open-C^3S, das 35 Hochschul-Zertifikatsmodule auf dem Gebiet Cybersicherheit anbietet und das kooperativ von der Hochschule Albstadt-Sigmaringen, der Friedrich-Alexander-Universität Erlangen-Nürnberg, der Freien Universität Berlin, und der Ludwig-Maximilians-Universität München, getragen wird.

Patrick Eisoldt, M. Eng. hat an der Hochschule Albstadt-Sigmaringen und der Glyndŵr University in Wales studiert. 2012 schloss er erfolgreich sein Masterstudium Systems Engineering ab. Im Rahmen seiner Master-Thesis konzipierte und realisierte er einen prototypischen Editor zur Projektierung von Prozessleitsystemen der Firma Siemens nach dem Ursache-Wirkung-Prinzip. Von November 2010 bis August 2011 unterstützte er das Institut für Wissenschaftliche Weiterbildung bei der Erstellung von Studienbriefen für den Studiengang Digitale Forensik.

Seit 2012 ist er für das Open Competence Center for Cyber Security als Modulentwickler tätig.

David Schlichtenberger studierte Medien- und Kommunikationsinformatik an der Hochschule Reutlingen. Nach seinem Masterstudium arbeitete er einige Jahre als Webentwickler und Kundenberater für Internetservices. Seit November 2014 ist er als akademischer Mitarbeiter an der Hochschule Albstadt-Sigmaringen am Institut für Wissenschaftliche Weiterbildung beschäftigt.

Benjamin Welte absolvierte den Bachelor-Studiengang Kommunikations- und Softwaretechnik an der Hochschule Albstadt-Sigmaringen.
Während des Bachelors arbeitete er als Werkstudent im Bereich der Softwareentwicklung.
Seit Februar 2016 arbeitet er als Modulentwickler und Tutor im Zertifikatsprogramm des Open Competence Center for Cyber Security.

Christian Schneider hat an der Hochschule Albstadt-Sigmaringen Technische Informatik studiert. Während seines Bachelorstudiums arbeitete er für einen Antivirenhersteller und als Webentwickler. Weiter betreute er, im Rahmen von Vorlesungen, Studenten als Tutor.
Seit Februar 2016 unterstützt er als wissenschaftlicher Mitarbeiter das Institut für wissenschaftliche Weiterbildung bei der Erstellung von Studienbriefen.

III. Modullehrziele

Penetrationstests werden durchgeführt, um Schwachstellen in IT-Systemen (logisch oder physikalisch) oder bezüglich des Faktors Mensch aufzudecken und sollen somit unmittelbar dazu führen, die Sicherheit eines Systems zu erhöhen.
Dies kann durch Schließen der Sicherheitslücken bzw. Schulung der Mitarbeiter geschehen. Zur Durchführung von Penetrationstests hat sich eine spezielle Methodik bewährt, die ein strukturiertes Vorgehen ermöglicht.
Für Penetrationstest werden Angriffe ausgeführt, die mittels von Python-Programmen flexibel und kofigurierbar gestaltet werden können.

Ziel dieses Moduls ist es, aus abstrakten Aufgabenstellungen zu Penetrationstests ablauffähige Programme zu entwickeln.
Als Programmiersprache wird Python verwendet. Python ist eine leistungsfähige Skriptsprache, die im Forensik- und Pentest-Umfeld häufig verwendet wird.
Nach Bearbeitung dieses Moduls soll der Student Netzwerkprotokolle analysieren und deren Inhalt aufschlüsseln können. Weiter soll die Implementierung von Penetrationstests das Verständnis über Angriffe auf IT-Strukturen erweitern. Die Implementierung und Anwendung von Proxy-Diensten und die Fertigkeit des Python-gestützten Mailen und Browsens runden das Wissensspektrum ab.

Der Studienbrief 1 behandelt das Thema Netzwerkanalyse.

Der Studienbrief 2 befasst sich mit dem Penetrationstesten von Systemen.

Der Studienbrief 3 befasst sich mit allgemeinen Pythonhacks.

Hinweise zur Typographie

Dieses Modul enthält zahlreiche Programmbeispiele, die stets in einer `Monospace Schriftart` gehalten sind. Zusätzlich wird zwischen Beispielen ohne gelbe Kästen und Quelltext in gelben Kästen unterschieden. Die Beispiele ohne gelbe Kästen sollen in der Python-Shell der IDLE-Umgebung realisiert werden, Quelltexte in gelben Kästen in einer Skriptdatei. Zusätzlich können die Beispiele für die Shell an der sogenannten Prompt (>>>) erkannt werden. An einigen Stellen sind die Codezeilen zu lang, weshalb sie mit einem Backslash (\) umgebrochen wurden. Der Quellcode ist somit korrekt und kann wie abgebildet ausgeführt werden.

Hinweise zur Installation

Als Editor kommt in diesem Modul PyCharm (Community) von JetBrains zum Einsatz. Soweit möglich werden die einzelnen Pythonmodule über dieses Programm verwaltet. Module können der globalen Pythoninstallation mittels folgender Anleitung hinzugefügt werden `https://www.jetbrains.com/help/pycharm/2016.1/installing-uninstalling-and-upgrading-packages.html?origin=old_help`.

Ist ein Compiling eines Moduls erforderlich wird der Microsoft Visual C++ Compiler for Python 2.7 (`https://www.microsoft.com/en-us/download/details.aspx?id=44266`) verwendet. Dieser wird ab dem zweiten Studienbrief benötigt.

Studienbrief 1 Netzwerkforensik mit Python

Im Zusammenhang mit diesem Studienbrief wird das Betriebssystem Kali Linux kurz vorgestellt. Die Programmbeispiele wurden ausschließlich unter dieser Umgebung getestet. Um diesen Studienbrief vollständig nachvollziehen zu können empfiehlt sich beispielsweise das VMware Image des Kali-Betriebssystems herunterzuladen[1] und die Beispiele unter dieser Umgebung zu testen. Gleiches gilt für die Kontrollaufgaben und die abschließende Übung. Kali Linux enthält bereits Python in der Version 2.7. Da in diesem Studienbrief ausschließlich mit der Version 2.7 gearbeitet wird, ist somit eine Installation von Python nicht notwendig. Dieser Studienbrief orientiert sich zu großen Teilen an der Literatur „Violent Python“ (O'Connor [2012])

1.1 Lernergebnisse

Sie können unter der Umgebung Kali Linux Python-Skripte erstellen, die dem Zweck der Netzwerkforensik dienen. Tiefer gehende Kenntnisse über Linux sind kein Lernziel - Kali Linux dient viel mehr als Mittel zum Zweck. Sie haben gelernt, wie Sie eine IP-Adresse einem physikalischen Standort zuordnen können. Sie können die Herkunft und das Ziel von Datenpaketen aus einer pcap-Datei auslesen und die IP-Adressen folglich als konkrete Standorte darstellen. Die Standorte können Sie zudem auch als Grafiken in Google Earth oder Google Maps visualisieren. Abschließend haben Sie gelernt, wie man den HTTP-Verkehr mit Python mitschneiden und analysieren kann.

1.2 Advance Organizer

Die Anzahl der Dienstleistungen, die über das Internet bereitgestellt werden, steigt stetig an. Viele Nutzer sehen nur die Vorteile der bequemen Internetdienste und sind sich selten bewusst, welche Informationen sie unweigerlich preisgeben.

1.3 Einführung in die Netzwerkforensik

Netzwerkforensik ist ein Teilgebiet der Digitalen Forensik. Dabei geht es darum, aus Netzwerkmitschnitten und -komponenten netzwerkbasierte Beweise sicherzustellen. Nicht zuletzt lassen sich so auch Eindringversuche in Netzwerke nachvollziehen. Informationsquellen für netzwerkbasierte Beweise können zum einen Netzwerkmitschnitte und zum anderen Informationen aus Netzwerkkomponenten sein. Wir konzentrieren uns in diesem Studienbrief auf die Anfertigung und Auswertung von Netzwerkmitschnitten.

1.3.1 Leitungsgebundene Netzwerkmitschnitte

Physische Netzwerkverkabelung wird verwendet, um Netzwerkverbindungen zwischen Clients sowie Routern und Switches herzustellen. Über Kupfer- oder Glasfaserleitungen werden digitale Signale in Form von Informationen ausgetauscht.

1.3.2 Forensischer Informationsgehalt

Ermittler können diesen Netzwerkverkehr replizieren und passiv mitschneiden. Hierfür gibt es Softwarelösungen, die wir näher betrachten werden, aber auch hardwarebasierte Netzwerk-Taps sind möglich.

[1] `http://www.kali.org/downloads/[Stand:14.2.2014]` Architecture: i386 - Windows manager: Gnome - Image type: ISO

Studienbrief 2 Penetrationstests mit Python

In diesem Studienbrief werden eine Reihe von Werkzeugen entwickelt, die bei einem Penetrationstest zum Einsatz kommen. Bei diesem Test werden nach Möglichkeit alle Systembestandteile auf potentielle Schwachstellen getestet. Es werden somit Hackerangriffe simuliert, um die Empfindlichkeit des Systems einschätzen zu können. Dieser Studienbrief orientiert sich zu großen Teilen an der Literatur „Violent Python“ (O'Connor [2012])

2.1 Lernergebnisse

Hauptziel dieses Studienbriefes ist es, dass Sie anhand konkreter Beispiele lernen, wie Sie Werkzeuge für Penetrationstests aus bestehenden Ansätzen weiterentwickeln oder, bei nicht zu komplexen Gegebenheiten, selbst entwickeln können. Sie haben den im Studienbrief behandelten TCP-Portscanner verstanden und können geringfügige Änderungen am Quellcode vornehmen. Außerdem kennen Sie das de facto Standard-Toolkit für Portscanner und haben einen SSH-Wurm mit dem Modul *paramiko* entwickelt. Da das Modul nur für Python 2.7 verfügbar ist, haben Sie auch in diesem Zusammenhang wesentliche Unterschiede zu der bislang verwendeten Version 3.3 kennengelernt. Abschließend haben Sie sich mit dem File-Transfer-Protokoll beschäftigt und können einen FTP Scanner in Python realisieren.

2.2 Advance Organizer

Ein Penetrationstest liefert Aussagen, inwieweit das untersuchte System gegen bestimmte Angriffe gehärtet ist. Auf der Basis dieser Erkenntnis können dann Maßnahmen zur Verbesserung der IT-Sicherheit des Systems entwickelt werden.

2.3 Einführung in Penetrationstests

Penetrationstests werden durchgeführt, um Schwachstellen in IT-Systemen (logisch oder physikalisch) oder bezüglich des Faktors Mensch aufzudecken und sollen somit unmittelbar dazu führen, die Sicherheit eines Systems zu erhöhen.
Dies kann durch Schließen der Sicherheitslücken bzw. Schulung der Mitarbeiter geschehen. Zur Durchführung von Penetrationstests hat sich eine spezielle Methodik bewährt, die ein strukturiertes Vorgehen ermöglicht.

2.3.1 Vorgehensweise beim Pentrationstesting

Die Qualität und der Nutzen eines Penetrationstests wird stark davon bestimmt, inwieweit dieser auf die individuelle Situation des Auftraggebers eingeht, d. h. wie viel Zeit und Ressourcen der Dienstleister auf die Ausforschung von Schwachstellen, die die konkrete IT-Infrastruktur betreffen, verwendet und wie kreativ er dabei vorgeht. Dieser Ablauf kann nicht vollkommen mit allgemein gültiger Beschreibung dargestellt werden. Die folgende Vorgehensweise ist typisch, die tatsächliche Vorgehensweise kann aber im konkreten Fall abweichen. Die Vorgehensweise zur Durchführung eines Penetrationstests sollte nach dem folgenden Schema aufgebaut sein.

1. Recherche nach Informationen über das Zielsystem
   Im Internet erreichbare Rechner müssen über eine offizielle IP-Adresse verfügen. Frei zugängliche Datenbanken liefern Informationen über IP-Adressblöcke, die einer Organisation zugewiesen sind.

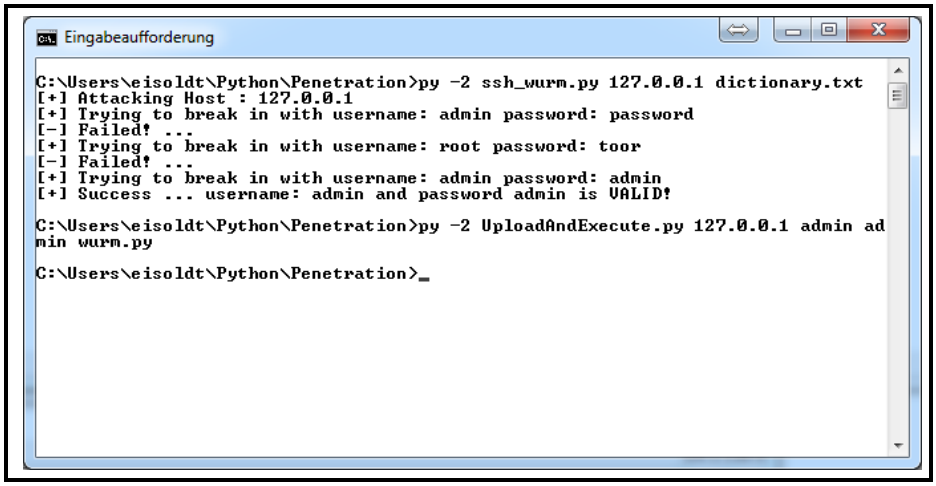

Abb. 2.1: SSH-Wurm angewendet auf einem lokalen Server

Nachdem die drei Python-Skripte erstellt wurden und wie in Abb. 2.1 gezeigt ausgeführt wurden, sollte sich auf dem Rechner ein neuer Ordner im Verzeichnis *C:\temp* befinden. Ein rekursiver Aufruf eines in Python geschriebenen Wurms ist an dieser Stelle nicht gewünscht, da eine Implementierung relativ aufwändig wäre.

Voraussetzung für einen erfolgreichen Ablauf des Quelltextes ist, dass in der Serveranwendung für SFTP und SSH der selbe Pfad hinterlegt wurde. Ist dies nicht der Fall, funktioniert zwar der Upload aber die Datei kann beim Versuch diese auszuführen nicht gefunden werden. Das liegt daran, dass die SSH-Shell in einem anderen Pfad startet. Da der Pfad des SFTP-Dienstes von außen nicht zu ermitteln ist, muss vor dem Ausführen der Datei ermittelt werden, wo diese sich befindet. Folgender Quelltext wurde um diese Funktionalität erweitert. Weiterhin werden

Studienbrief 3 Python-Hacks

In diesem Studienbrief werden verschiedene Angriffe auf Systeme mittels Python demonstriert. Explizit eingegangen wird auf verschiedene Verbindungsprotokolle, das Erstellen von Ransomware, das Extrahieren von Windowspasswörtern mit Schwerpunkt Hashangriffe und das Ausnutzen der Bash Sicherheitslücken Shellshock.

3.1 Lernergebnisse

Hauptziele des Studienbriefes sollen erweiterte Erkenntnisse über Kommunikationsprotokolle, das Arbeiten mit Penetrationstools und die daraus abgeleiteten Angriffsmechanismen sein.
Sie haben sich nach Abschluss dieses Briefes mit dem SMTP und HTTP Protokoll auseinandergesetzt und können deren Ablauf beschreiben. Neben automatisierten Anmeldungen an einem Internetformular, dem Versand von automatischen E-Mails und der Extraktion von Windowspassworthashes mit anschließendem Angriff sind Sie in der Lage, die Bashsicherheitslücke Shellshock auszunutzen. Als Abschluss wird Ihnen die Erstellung eines Keyloggers mittels Python erläutert.

3.2 Advance Organizer

3.2.1 IT-Bedrohungen

IT-Bedrohungen sind allgegenwärtig. Kein Tag vergeht, an dem keine neue Malware entwickelt wird. Zudem wird in den Medien subjektiv und in den letzten Jahren verstärkt über IT-Angriffe berichtet. Die Anzahl von Betrugsfällen ist zudem stark gestiegen.

Schadsoftware

Spezielle Klassen von Bedrohungen eines IT-Systems gehen von Viren,Würmern und Trojanischen Pferden aus, die wir als Schadsoftware (engl. malware oder malicious code) bezeichnen. Es handelt sich hierbei um Programme, die festgelegte, dem Benutzer jedoch verborgene, Funktionen ausführen. Die Festlegung der zusätzlichen und beabsichtigten (i. d. R. böswilligen) Funktionalität unterscheidet diese Klasse von Programmen von den so genannten Wanzen (engl. bugs), deren Fehlverhalten meist von nicht beabsichtigten Programmierfehlern verursacht wird.

3.2.2 Bedrohungen im Rechnernetz

Bei einer hierarchisch gegliederten Kommunikationsarchitektur wird stets eine genau definierte Funktionalität in der nächsttieferen Schicht gekapselt. Ein Dienst einer höheren Schicht, der einen Dienst der nächsttieferen Schicht in Anspruch nimmt, verlässt sich darauf, dass dieser seine Aufgabe korrekt erledigt.

Schichtenmodelle

Wird nun ein Dienst an einer bestimmten Stelle kompromittiert, so sind Daten und Steuerinformationen höherer Schichten, ebenfalls davon betroffen. Sind die Daten die auf den höheren Schichten übermittelt werden nicht verschlüsselt oder anderweitig gegen Manipulationen und unerwünschter Einsichtnahme gesichert, ist eine zuverlässige Kommunikation nicht mehr möglich. Es ist somit offensichtlich, dass das Gesamtsystem, das nicht mit Verschlüsselung oder vergleichbaren Schutzmechanismen arbeitet, nie sicherer ist als die einzelnen Teilkomponenten.

3.2.3 Bedrohungen aus dem Internet

Man muss davon ausgehen, dass prinzipiell alle Anwendungen Sicherheitsschwachstellen besitzen. Obwohl Sicherheitslücken zum Großteil durch gezielte

Nach Ausführung enthält das Empfängerpostfach eine neue E-Mail. Die Absenderemailadresse wird entsprechend der Variablenangabe *From* angezeigt.

Kontrollaufgabe 3.2

Wenden Sie den Quelltext 3.1 auf Ihnen zugängliche Mail-Server an, um anmeldefreie Server zu sammeln.
Legen Sie dann eine Mailliste in Form einer Datei *adressen.txt* an, die gültige Mail-Adressen enthält.
Verwenden Sie *adressen.txt*, indem Sie Quelltext 3.2 so ändern, dass alle Adressen der Liste eine Mail mit identischem Absender und identischem Betreff erhalten.

K

3.4 Webbrowsing

3.4.1 HTTP Protokoll

Allgemeines

Das Hypertext Transfer Protokoll (HTTP) [3] dient der Übertragung von Daten auf Anwendungsschicht. Größtenteils wird es zum Abrufen von Hypertext-Dokumenten (Webseiten) aus dem World Wide Web verwendet. Bei HTTP 1.0[4] handelt es sich um ein zustandsloses Protokoll. Die Verbindung wird also nach einer Übertragung nicht aufrechterhalten. Sollen weitere Daten übertragen werden, muss eine neue Verbindung aufgebaut werden. Dies bringt eine große Belastung für Webserver mit sich. Deshalb wurde in Protokollversion 1.1 eine Möglichkeit geschaffen, die Verbindung für eine gewisse Zeitspanne offen zu halten. Dadurch können Anfragen schneller und effizienter bearbeitet werden.[5]

HTTP 1.0

Anwender kommen in der Regel mit HTTP nur über Webbrowser wie https://www.google.de/chrome/browser/desktop/, http://windows.microsoft.com/de-de/internet-explorer/download-ie oder https://www.mozilla.org/de/firefox/new/ in Kontakt. Peterson and Davie [2007] (S.626 ff.)

Anfragemethoden

Der HTTP-Standard stellt eine Vielzahl von Anfragemethoden zur Verfügung. *GET* ist eine der einfachsten Methoden. Sie ermöglicht es dem Client Daten vom Server abzurufen. Beim Abruf der Datei mittels dem URL Schema *http://<servername>/<dateiname>*, verbindet sich der Client mit dem Host <servername> und sendet diesem eine GET Anfrage, in der er die Datei <dateiname> anfordert. Weiter können der Anfrageparameter mitgegeben werden. Diese werden dabei an die URL angehängt.

GET

Als Beispiel wird im Folgenden eine Suchanfrage an die Suchmaschine *https://duckduckgo.com/* gestellt. Der Webserver hängt den zu suchenden Begriff entsprechend an die URL an und verweist den Browser anschließend auf diese.

```
https://duckduckgo.com/?q=google&ia=about
```

[3] https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol
[4] http://www.netplanet.org/www/http.shtml
[5] http://www.mediaevent.de/tutorial/http-request.html

```
            return stopAt // 8, binaryToString(new_array)
          else:
            messageLen = binaryToInt(new_array)
            new_array = []
            n = 0
            stopAt = messageLen
            lenComplete = True
        new_array.append(0 if pixel[i] & 1 == 0 else 1)
        n += 1

length, msg2 = ExtractMessage('out_image.png')
print("Nachricht:", msg2)
print("Länge der Nachricht in Byte:", length)
```

Das Vorgehen unterscheidet sich nicht wesentlich von dem des Speicherns. Das Bild wird geöffnet, Pixelweise durchlaufen und je Farbe das letzte Bit ausgelesen und im Array new_array gespeichert. Zuerst werden 32 Bit ausgelesen, welche die Länge (X) der eigentlichen Nachricht darstellen. Diese werden in einen Integer umgewandelt. Danach werden X Bit ausgelesen und in einen String umgewandelt, welcher die Nachricht enthällt.

Beispiel mit einer langen Nachricht:
Nachricht:
hallohallohallohallohallohallohallohallohallohallohallohallohallo<gekürzt>
Länge der ungekürzten Nachricht:
40950 Byte
Abbildung 3.5 zeigt das Originalbild (oben) und das Trägerbild (unten) im Vergleich.

Abb. 3.5: Vergleich von Oginial und Träger

Im Vergleich dazu wird in Abbildung 3.6 ersichtlich, welche Folgen die Nutzung des höchstwertigen Bits (unten) hat. Relativ gut zu erkennen ist die Grafik dagegen bei Nutzung des vierthöchstwertigen Bits (oben).

Abb. 3.6: Nutzung höherwertiger Bits

Kontrollaufgabe 3.8

Erweitern Sie die vorgestellten Quelltexte so, dass für die Speicherung der Daten zusätzlich zum niederwertigsten das zweitniederwertigste Bit verwendet wird.

K

3.6 Erstellung eines Keyloggers

3.6.1 Installationshinweise

Die benötigten Module für die folgenden Quelltexte setzen die Installation der Module pyHook und pywin32 voraus. Für das Modul pyHook muss die 32 bit Version von Python 2.7 installiert werden. Da hier kein Setup für die 64 bit Version zur Verfügung steht. Dabei muss bei der Installation darauf geachtet werden, dass falls bereits die 64 bit Version von Python 2.7 installiert ist, ein anderer Pfad angegeben wird. Die Module sind am einfachsten über ein Setup zu installieren, da eine manuelle Installation an dieser Stelle deutlich aufwändiger wäre.
Das jeweilige Setup kann unter den angegebenen Links heruntergeladen werden.

- pyhook `https://sourceforge.net/projects/pyhook/files/pyhook/1.5.1/`
- pywin32 `https://sourceforge.net/projects/pywin32/files/pywin32/Build%20220/`

3.6.2 Einführung

In diesem Kapitel wird ein einfacher Keylogger in Python erstellt. Hierbei werden mittels Python-Modulen Funktionen aus der Windows-API verwendet. Auf die Funktionen aus der Windows-API soll jedoch nicht weiter eingegangen werden. Grundlage bildet der Programmansatz, beschrieben im Buch *Black Hat Python* Seitz [2014].

Ü

Übung 3.2

Die folgende Übungsaufgabe soll als Erweiterung der im Unterkapitel 3.4.2 gewonnenen Erkenntnisse dienen.
Im Kapitelbeispiel mussten zum Prüfen der Verzeichnisse einige Parameter über das installierte System im Ziel bekannt sein. So war es nötig, Erkenntnisse über das dort verwendete CMS System (Joomla, Wordpress, etc.) zu besitzen.
Ziel dieser Übung ist es, ein allgemeingültiges Skript zu schreiben, welches auf Grundlage von Wort- bzw. Dateilisten eine Webserververzeichnisstruktur durchsucht. Anhand den Statuscodes wird erkennbar, ob eine Datei vorhanden ist.
Als Wörterbuch soll das Ziparchiv *SVNDigger.zip* unter `https://www.netsparker.com/blog/web-security/svn-digger-better-lists-for-forced-browsing/` verwendet werden.
Hinweis: Verwenden Sie als Grundlage den Code aus Quelltext 3.6.

Optional: Verändern Sie die Anzahl der Threads und beobachten Sie die Ausführungszeiten.

Ü

Übung 3.3

Erstellen Sie eine Funktion, mit welcher es möglich ist, sich auf einer Wordpress-Webseite anzumelden. Die Funktion soll testen, ob die Anmeldung erfolgreich war oder nicht. Als Eingabe dient eine Textdatei welche eine Liste von Passwörtern enthält. Die Passwörter sind durch Zeilenumbrüche getrennt.
Achtung:
Dieser Angriff darf nicht auf einen fremden Server angewendet werden. Dies gilt auch, wenn es sich um eine eigene Wordpress-Instanz auf einem fremden Server handelt.
Gehen Sie in Ihrem Skript nach folgendem Ablauf vor

1. Absetzen einer GET-Anfrage auf die Anmeldeseite (Bespiel: http://localhost/wordpress/wp-admin)
2. Absetzen einer POST-Anfrage (Anmeldung) auf die im Location-Header der vorigen Antwort (Response) befindliche URL
3. Absetzen einer weiteren GET-Anfrage auf die im Location-Header der vorigen Antwort befindliche URL
4. Durchsuchen des HTML-Inhalts der vorigen Antwort nach Anhaltspunkten, ob die Anmeldung erfolgreich war. (Bei erfolgreicher Anmeldung muss beispielsweise das Wort “Howdy“ im HTML-Inhalt vorkommen)

Verzeichnisse

I. Abbildungen

II. Beispiele

III. Definitionen

IV. Exkurse

V. Kontrollaufgaben

VI. Literatur

BSI. Studie - Durchführungskonzept für Penetrationstests. Website, November 2003. `https://www.bsi.bund.de/DE/Publikationen/Studien/pentest/index_htm.html`.

Johannes Ernesti and Peter Kaiser. *Python 3 - Das umfassende Handbuch*. Galileo Press, 3. Auflage, Bonn, 2012. ISBN 978-3-836-21925-9.

ITWissen. 3-Wege-Handshake - 3-way handshake. Website, Juni 2013. `http://www.itwissen.info/definition/lexikon/3-Wege-Handshake-3-way-handshake.html`.

Mark Lutz. *Learning python*. O'Reilly Media, Inc., 2013.

Alex Martelli, Anna Ravenscroft, and David Ascher. *Python cookbook*. O'Reilly Media, Inc., 2005.

Adam Maxwell. The very unofficial dummies guide to scapy. Website, Juni 2012. `https://scapy-guide.googlecode.com/files/ScapyGuide.pdf`.

Jon Oberheide. dpkt Tutorial #2: Parsing a PCAP File. Website, Oktober 2008. `https://jon.oberheide.org/blog/2008/10/15/dpkt-tutorial-2-parsing-a-pcap-file/`.

TJ O'Connor. *Violent Python - A Cookbook for Hackers, Forensic Analysts, Penetration Testers, and Security Engineers*. Newnes, London, 1st edition, 2012. ISBN 978-1-597-49957-6.

Larry L Peterson and Bruce S Davie. *Computer networks: a systems approach*. Elsevier, 2007.

prontosystems.org. Passives und aktives ftp. Website, April 2016. http://www.prontosystems.org/it/ftp.

Vivek Ramachandran. Simulating an SSH Worm using Python. Website, Juni 2013. http://hackoftheday.securitytube.net/2013/04/simulating-ssh-worm-using-python.html.

Jörg Riether. BackTracks Erben: Kali Linux 1.0. Reinkarnation. Website, Mai 2013. http://heise.de/-1843614.

J. Seitz. *Black Hat Python: Python Programming for Hackers and Pentesters*. No Starch Press, Incorporated, 2014. ISBN 9781593275907.

slacksite.com. Active ftp vs. passive ftp, a definitive explanation. Website, April 2016. http://www.slacksite.com/other/ftp.html.

Andrew S Tanenbaum. *Computer networks, 4-th edition*. Wetherall, 2003.

Michael Weigend. *Objektorientierte Programmierung mit Python 3 - Einstieg, Praxis, professionelle Anwendung*. Hüthig Jehle Rehm, 4. Auflage, München, 2010. ISBN 978-3-826-61750-8.

Anhang

A. Schlüsselwörter

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

Open C³S
Open Competence Center for Cyber Security

Fort- und Weiterbildung

Neue Bedrohungszenarien stellen Sicherheitsexperten und IT-Verantwortliche in Unternehmen und einschlägigen Behörden vor immer größere Herausforderungen. Neue Technologien und Anwendungen erfordern zusätzliches Know-how und personelle Ressourcen.

Zur Erhöhung des Fachkräftepools und um neues Forschungswissen schnell in die Praxis zu integrieren, haben sich die im Bereich lehrenden und forschenden Verbundpartner zum Ziel gesetzt, ein hochschuloffenes transdisziplinäres Weiterbildungsprogramm im Sektor Cyber Security zu entwickeln. Auf der Grundlage kooperativer Strukturen werden wissenschaftliche Weiterbildungsmodule im Verbund zu hochschulübergreifenden Modulpaketen und abschlussorientierten Ausbildungslinien konzipiert und im laufenden Studienbetrieb empirisch getestet.

Die Initiative soll High Potentials mit und ohne formale Hochschulzugangsberechtigung über innovative Weiterbildungsangebote (vom Zertifikat bis zum Masterprogramm) zu Sicherheitsexperten aus- und fortbilden. Hierzu werden innovative sektorale Lösungen zur Optimierung der Durchlässigkeit von berulicher und hoch- schulischer Bildung entwickelt und für eine erfolgreiche Implementierung vorbereitet. Unter prominenter Beteiligung einschlägiger Verbände, der Industrie sowie Sicherheits und Ermittlungsbehörden verfolgt die Initiative das Ziel, im deutschsprachigen Raum eine Generation von Fachkräften wissenschaftlich aus- und weiterzubilden, die unser Internet schützen kann.

Open Competence Center for Cyber Security

Open C³S ist aus dem Verbundvorhabens Open Competence Center for Cyber Security entstanden. Das Gesamtziel des Programms war die Entwicklung eines hochschuloffenen transdisziplinären Programms wissenschaftlicher Weiterbildung im Sektor Cyber Security. Das Bundesministerium für Bildung und Forschung (BMBF) fördert das Großprojekt im Rahmen des Wettbewerbs „Aufstieg durch Bildung: offene Hochschulen“, der aus BMBF-Mitteln und dem Europäischen Sozialfonds finanziert wird.

Neun in Forschung und Lehre renommierte Hochschulen und Universitäten aus dem gesamten Bundesgebiet haben sich zum Ziel gesetzt, Online-Studiengänge auf dem Gebiet der Cybersicherheit zu entwickeln. Dieses Konzept soll den Studierenden ermöglichen, sich berufsbegleitend auf hohem Niveau wissenschaftliche Qualifikationen anzueignen und akademische Abschlüsse zu erlangen. Beruflich erworbene Kompetenzen können eingebracht werden. Die Bezeichnung „Open“ steht auch für die Öffnung des Zugangs zu akademischer Bildung ohne klassischen Hochschulzugang.

Mission der Initiative ist es, dringend benötigte Sicherheitsexperten aus- und fortzubilden, um mit einer sicheren IT-Infrastruktur die Informationsgesellschaft in Deutschland und darüber hinaus zu stärken.

Umsetzungsnahes Wissen ist ein wesentlicher Schlüssel um der wachsenden digitalen Bedrohung zu begegnen. Solange wir nicht in der Lage sind, Systeme hinreichend zu härten, Netzwerke sicher zu designen und Software sicher zu entwickeln, bleiben wir anfällig für kriminelle Aktivitäten. Unser Ziel ist es, die Mitarbeiter von heute zu Sicherheitsexperten und Führungskräften von morgen auszubilden und dafür zu sorgen, dass sich die Zahl und die Fertigkeiten dieser Experten nachhaltig erhöht.

Weitere Informationen zu Open C³S: https://open-c3s.de

Z203 Python 2 - Penetration Testing

Penetrationstests werden durchgeführt, um Schwachstellen in IT-Systemen (logisch oder physikalisch) oder bezüglich des Faktors Mensch aufzudecken und sollen somit unmittelbar dazu führen, die Sicherheit eines Systems zu erhöhen. Dies kann durch Schließen der Sicherheitslücken bzw. Schulung der Mitarbeiter geschehen.

Zur Durchführung von Penetrationstests hat sich eine spezielle Methodik bewährt, die ein strukturiertes Vorgehen ermöglicht. Für Penetrationstest werden Angriffe ausgeführt, die mittels von Python-Programmen flexibel und kofigurierbar gestaltet werden können.

Ziel dieses Moduls ist es, aus abstrakten Aufgabenstellungen zu Penetrationstests ablauffähige Programme zu entwickeln. Als Programmiersprache wird Python verwendet. Python ist eine leistungsfähige Skriptsprache, die im Forensik- und Pentest-Umfeld häufig verwendet wird.

Nach Bearbeitung dieses Moduls soll der Student Netzwerkprotokolle analysieren und deren Inhalt aufschlüsseln können. Weiter soll die Implementierung von Penetrationstests das Verständnis über Angriffe auf IT-Strukturen erweitern. Die Implementierung und Anwendung von Proxy-Diensten und die Fertigkeit des Python-gestützten Mailen und Browsens runden das Wissensspektrum ab.

Der Studienbrief 1 behandelt das Thema Netzwerkanalyse.

Der Studienbrief 2 befasst sich mit dem Penetrationstesten von Systemen.

Der Studienbrief 3 befasst sich mit allgemeinen Pythonhacks.