Automatisierung von Vulnerabiltiy Scans

Autor: Martin Schagerl

Letzte Änderung am 10.07.2014

Version 1.3

Lexon e.U., Inh. Martin Schagerl, Holzing 52, A 3252 Bergland

T: +43 / 676 / 7119622, E: office@lexon.at, I: http://www.lexon.at

Kurzfassung

In dieser Arbeit werden Konzepte für die Automatisierung von Vulnerability Scans beschrieben. Dadurch wird ermöglicht, immer wiederkehrende Schritte ohne manuellen Aufwand maschinell abzuarbeiten. Dabei werden Techniken und Werkzeuge sowohl für aktive als auch für proaktive Überprüfungen erklärt. Durch die Automatisierung können Zielsysteme in regelmäßigen Abständen geprüft und die gewonnen Ergebnisse beliebig weiterverarbeitet werden.

Für die proaktiven Scans werden mehrere im Internet zu Verfügung stehende Dienste vorgestellt und deren Anwendung erläutert. Bei den aktiven Überprüfungen werden die Werkzeuge Nmap und Nessus detailliert beschrieben sowie praktische Beispiele gegeben. Es wird für jedes Werkzeug bzw. Dienst ein Weg zur Automatisierung vorgestellt.

Als Entwicklungsumgebung wird das Betriebssystem Backtrack 5R2 und der Entwicklungseditor Eclipse verwendet. Zudem werden die verschiedenen Scans mit der Skriptsprache Python automatisiert, wobei die Ergebnisse in einer SQLite Datenbank abgespeichert werden. Für all diese Technologien wird ein kurzer Einblick gegeben.

Administratoren von Unternehmen können die beschriebenen Ansätze dazu verwenden, um Schwachstellen im eigenen Unternehmen aufzudecken. Penetrationstester können viele ihrer bisherigen manuellen Schritte durch die hier erwähnten Konzepte ersetzen und somit den Einsatz ihrer Ressourcen optimieren.

Abstract

This work describes concepts to automate vulnerability scans. Therefor it is possible to process repetitive steps without manual actions. There will be explanations about the techniques and tools of active and proactive inspecting. By automating the scans, the target systems can be tested in regular intervals and the results are processed as desired.

The work shows and explains the application of different services in the internet to execute proactive scans. For the active scans the tools nmap and nessus will be described with the help of practical examples. Code snippets will show a way for automating the scans.

The operating system backtrack 5R2 and the IDE Eclipse are used as development environment. The scans are automated with the script language python and the results are stored in a SQLite database. For all of these technologies there will be a brief introduction.

Administrators can use these aspects to uncover weaknesses in their own company. Penetration testers may replace many manual steps through these concepts to optimize the use of their resources.

Inhaltsverzeichnis

1. EINLEITUNG 5
1.1. MOTIVATION UND RELEVANZ 5
1.2. ZIELSETZUNG 5
1.3. ANWENDUNGSSZENARIO 5
1.4. AUFBAU DER ARBEIT 6
2. FACHLICHES UMFELD 7
2.1. ENTWICKLUNGSUMGEBUNG: ECLIPSE & BACKTRACK 7
2.2. DATENÜBERTRAGUNG: TCP, UDP, IP 7
2.3. PYTHON 9
2.4. SQLITE 12
2.5. EINTEILUNG VON SCANS ZUM AUFDECKEN VON SCHWACHSTELLEN 13
3. PROAKTIVE ÜBERPRÜFUNG 15
3.1. EINFÜHRUNG 15
3.2. ERFORDERLICHE SCHRITTE 17
3.3. DOMAINS UND IP ADRESSEN ERMITTELN 18
3.3.1. DNS Abfragen 18
3.3.2. RIPE NCC 21
3.3.3. Bing 22
3.4. SOFTWARE- UND PROTOKOLLINFORMATIONEN ERMITTELN 23
3.5. ZIELE AUF SCHADCODE ÜBERPRÜFEN 25
3.6. SENSIBLE INFORMATIONEN AUFDECKEN 26
4. AKTIVE ÜBERPRÜFUNG 30
4.1. NMAP SCAN 30
4.1.1. TCP/UDP Portscan 30
4.1.2. Dienst und Betriebssystem Erkennung 33
4.1.3. Automatisierung 33
4.2. NESSUS SCAN 36
4.2.1. Aufbau von Nessus 36
4.2.2. Automatisierung 37
5. ZUSAMMENFASSUNG UND AUSBLICK 40

Abbildungsverzeichnis

Abbildung 1: Einteilung von TCP, UDP und IP 8
Abbildung 2: Abarbeitung eines Python-Codes vgl. [5, S. 21] 10
Abbildung 3: Geschwindigkeit von Programmiersprachen [6, S. 3] 10
Abbildung 4: Speicherauslastung von Programmiersprachen [6, S. 3] 11
Abbildung 5: Umfang des Quelltextes von Programmiersprachen [6, S. 5] 11
Abbildung 6: Werkzeuge zum Aufdecken interner Schwachstellen [10, S. 15] 15
Abbildung 7: Arten von proaktiven Überprüfungen [10, S. 9] 16
Abbildung 8: RIPE Abfrage (https://apps.db.ripe.net/search/query.html) 21
Abbildung 9: Bing Abfrage (http://www.bing.com) 22
Abbildung 10: Suchergebnis von Shodan 24
Abbildung 11: Neue CSE erstellen 28
Abbildung 12: Ablauf eines SYN Scan/"half open" Scan, vgl. [20, S. 89] 31
Abbildung 13: Ablauf eines Connect Scan, vgl. [20, S. 88] 32

Listings

Listing 1: Verwendung von nslookup 19
Listing 2: DNS Abfragen in Python 20
Listing 3: Bing Abfragen in Python 23
Listing 4: Shodan Abfragen in Python 25
Listing 5: VirusTotal Abfragen in Python 26
Listing 6: Google Abfragen in Python 29
Listing 7: NMAP Prozess mit Python starten 34
Listing 8: NMAP Ergebnis in Python verarbeiten 35
Listing 9: Nessus Scans mit Python durchführen 38

1. Einleitung

1.1. Motivation und Relevanz

Heutzutage werden beinahe alle Arten von Daten in digitaler Form abgespeichert – von privaten Bildern bis hin zu hochsensiblen Entwicklungsdaten. Durch den Einsatz von modernen Computersystemen, sowie Kommunikationsmedien wie das Internet, ergeben sich durch die Digitalisierung eine Menge von Vorteilen. Zu den wichtigsten zählt die automatisierte Verarbeitung, sowie die effiziente und kostengünstige Speicherung.

Da Unternehmen, Institute, Organisationen und öffentliche Einrichtungen ihre Geschäftsprozesse kaum ohne Internet abwickeln können, entsteht durch die globale Vernetzung ein hohes Risiko für deren Daten und IT Infrastrukturen. Mittels verschiedenen Attacken ist der Zugriff auf fremde Einrichtungen über das Internet möglich. Dadurch kann ein enormer Schaden jeglicher Art entstehen: Finanzieller-, Wirtschaftlicher- bzw. Imageschaden.

Wie in der letzten Zeit durch Anonymous & Co gezeigt wurde, sind viele große Unternehmen bzw. Öffentliche Einrichtungen durch diverse Attacken angreifbar. Um sich vor solche Angriffe besser zu schützen, gibt es mehrere Möglichkeiten um IT Infrastrukturen abzusichern.

Da durch solche Sicherheitstests viele manuelle Arbeit entsteht, wäre es wünschenswert den gesamten Prozess zu automatisieren. Daher ist die Motivation dieser Arbeit, Sicherheitslücken in IT Infrastrukturen automatisiert aufzudecken.

1.2. Zielsetzung

Ziel der Arbeit ist es, dass sowohl aktive als auch proaktive Überprüfungen automatisiert durchgeführt werden können. Es werden zwei völlig unabhängige, modular aufgebaute Skripts erstellt. Durch den modularen Aufbau besteht die Möglichkeit, dass einzelne Module für andere Aufgaben wiederverwendet werden können.

Die Ergebnisse werden in einer Datenbank abgespeichert. Durch die strukturierte Datenablage ergibt sich der Vorteil, dass ein Export in unterschiedliche Formate vereinfacht wird.

1.3. Anwendungsszenario

IT Dienstleister, welche Sicherheitsüberprüfungen für ihre Kunden anbieten – sogenannte Penetrationtests - kennen die Tatsache, dass ein Großteil der gesamten Überprüfung häufig nach demselben Schema abläuft. Diese Abläufe nehmen eine Menge Zeit in Anspruch und werden daher für den Kunden sehr kostenintensiv. Durch die Automatisierung könnten Penetrationtests einerseits für Kunden attraktiver

angeboten werden und andererseits kann ein wirtschaftlicher Vorteil für den IT Dienstleister gewonnen werden.

Häufig dürfen Penetrationstests nicht während der Geschäftszeit durchgeführt werden, da hiermit ein Risiko für den produktiven Betrieb entstehen könnte. Daher sind arbeiten am Wochenende oder in der Nacht notwendig, was zu einen finanziellen und organisatorischen Problem führen kann. Auch hier kann die Automatisierung Abhilfe schaffen: Die Skripts können jederzeit gestartet werden und können über eine längere Zeit laufen. Die Ergebnisse können dann in der üblichen Geschäftszeit ausgewertet werden.

Auch für Unternehmen, Institute oder öffentlichen Einrichtungen welche ihre eigene Infrastruktur und Applikationen in regelmäßigen Abständen selbst überprüfen wollen, ist das hier beschriebene Konzept ein guter Anhaltspunkt. So können die Tests in einen akzeptablen Zeit- und Kostenaufwand durchgeführt werden.

1.4. Aufbau der Arbeit

Im Kapitel 2 wird ein fachlicher Einstieg geschaffen, damit die späteren Ansätze besser verstanden werden können. Dabei werden verwendete Techniken und Konzepte kurz angesprochen und erklärt.

Der Hauptteil gliedert sich in zwei große Bereiche: proaktive und aktive Scans. Zu beiden Bereiche gibt es einerseits theoretische Beschreibungen von Methoden und Werkzeugen und zum anderen wird ein Einblick in die Implementierung der Automatisierung gegeben.
Die wichtigsten Methoden bzw. Werkzeuge werden detailliert beschrieben sowie mit praktischen Beispielen erklärt. Da es jedoch den Rahmen sprengen würde, alle zu Verfügung stehenden Tools und Methoden zu beschreiben, werden manche nur am Rande gestreift sowie auf weiterführende Literatur verwiesen.

Im letzten Kapitel werden die wichtigsten Punkte zusammengefasst und ein Ausblick gegeben.

2. Fachliches Umfeld

2.1. Entwicklungsumgebung: Eclipse & Backtrack

Die gesamte Entwicklung dieser Arbeit wird auf dem Betriebssystem Backtrack durchgeführt. Bei Backtrack handelt es sich um eine Linux Distribution mit zusätzlichen Werkzeugen für Sicherheitsüberprüfungen. Es wurde die derzeit aktuellste Version 5 R2 verwendet, welche seit 1. März 2012 verfügbar ist. Es basiert auf Ubuntu mit dem Kernel 3.2.6, wodurch die Ubuntu Paketverwaltung genutzt werden kann, was ein großer Vorteil gegenüber den vorherigen Versionen ist. [1] Das Betriebssystem kann kostenlos unter www.backtrack-linux.org heruntergeladen werden. Des Weiteren werden auf dieser Webseite Anleitungen zur Installation bereitgestellt.

Um die späteren Skripts komfortabel zu schreiben und zu testen wird Eclipse als Entwicklungseditor verwendet. Eclipse ist eine universelle Entwicklungsplattform, welche durch ein Plugin-System beliebig angepasst und erweitert werden kann. [2, S. 6]
Wie unter 2.3 beschrieben, wird Python als Skriptsprache verwendet. Da Eclipse standardmäßig ohne Python-Unterstützung bereitgestellt wird, muss das Plugin „PyDev“ in die Eclipse Umgebung hinzugefügt werden.
Zusätzlich kann die Bibliothek „pep8“ verwendet werden, um sicherzustelle, dass der Quelltext entsprechend der Python Konvention dargestellt wird. Dazu muss legendlich die Pep8 Funktionalität aktiviert werden, welche bereits mit der dem „PyDev“ Plugin mitinstalliert wurde.

Weitere mögliche Python-Entwicklungseditoren mit Debug-Funktionalität wären:

- **Netbeans:** Mit diesem Editor kann auch ein Python Plugin installiert werden. Gegenüber Eclipse ist die Bedienung jedoch weniger komfortabler und des Weiteren war die Debugfunktion in den Tests sehr unstabil. Netbeans kann kostenlos unter http://netbeans.org/ heruntergeladen werden.

- **Wing IDE Professional:** Dieser Editor hat standardmäßig eine Python Funktionalität integriert, allerdings ist die Professional Version nicht frei erhältlich. Es werden auch freie Versionen angeboten, die jedoch durch eine eingeschränkte Funktionalität nicht attraktiv für die Entwicklung größerer Projekte sind. Alle Wing IDE Version können unter http://wingware.com/ heruntergeladen werden.

2.2. Datenübertragung: TCP, UDP, IP

Da viele der Sicherheitsüberprüfungen auf Basis von Netzwerkprotokollen stattfinden, werden diese im Vorfeld erklärt, um die nachfolgenden Konzepte zu verstehen.

Damit Daten zwischen heterogenen Geräten im Netzwerk ausgetauscht werden können, müssen bestimmte Regeln eingehalten werden. Dazu wurden sogenannte Netzwerkprotokolle entworfen, welche die Adressierung der Kommunikationsteilnehmer sowie die Steuerung und Kontrolle des Datenflusses regeln. [3, S. 7-9] Im Lauf der Zeit haben sich dabei die zwei Transportprotokolle TCP (Transmission

Control Protocol) und UDP (User Datagram Protocol) wie auch das Adressierungsprotokoll IP (Internet Protocol) durchgesetzt. Das Transportprotokoll ist für die korrekte Zuordnung der Datenpakete an die richtigen Dienste zwischen zwei Geräten im Netzwerk zuständig. Um dies zu bewerkstelligen horcht jeder Dienst auf einen bestimmten Port und benutzt eines der Transportprotokolle. Die Geräte selbst werden mit einer eindeutigen IP Adresse adressiert.

Die Transportprotokolle lassen sich folgendermaßen beschreiben:

- **TCP:** Hierbei handelt es sich um ein zuverlässiges, verbindungsorientiertes Übertragungsprotokoll. Es ist zuverlässig, weil alle gesendeten Datenpakete von der Gegenseite bestätigt werden müssen. Geht ein Paket verloren, so kümmert sich TCP um das erneute senden der Daten. Ein Entwickler einer Anwendung muss sich somit nicht um die Fehlerbehandlung auf der Netzwerkebene kümmern. Ein wesentlicher Aspekt ist, dass das Protokoll verbindungsorientiert arbeitet. Das bedeutet, es wird vor dem senden der eigentlichen Daten eine Session zwischen Sender und Empfänger aufgebaut. [4, S. 217-224] Dazu wird das sogenannte Drei-Wege-Handshake durchgeführt, wodurch Sender und Empfänger durch senden von SYN und ACK Segmenten eine Verbindung etablieren. Durch diese Tatsache können Portscan rasch durchgeführt werden. [4, S. 242]

- **UDP:** Dieses Übertragungsprotokoll arbeitet genau gegengleich als TCP: Es ist unzuverlässig und verbindungslos. Somit muss sich der Programmierer einer Anwendung selbst um Fehlerbehandlung und Flusskontrolle kümmern. Da das Protokoll verbindungslos ist, wird keine Verbindung aufgebaut. [4, S. 207, 208] Dadurch gestaltet sich ein Portscan zeitaufwändiger, da es keine Bestätigung für den Empfang eines Datenpakets gibt. UDP wird dann eingesetzt, wenn Daten rasch übertragen werden müssen und wenn einzelne verlorene Datenpakte nicht die Funktionalität beeinflussen. Ein Beispiel dafür wäre das Übertragen eines Videostreams: Wenn hierbei jeder übertragene Frame bestätigt werden müsste, dann würde das Video womöglich nicht flüssig laufen. Zudem würde das menschliche Auge nicht merken, wenn hin und wieder ein einzelner Frame fehlt.

- **IP:** Um weltweit mit anderen Systemen zu kommunizieren, muss eine global anerkannte Adressierung angewendet werden. Genau für diesen Zweck wird IP eingesetzt. Die Adressierung besteht aus 32 Bit und bietet die Möglichkeit Pakete zwischen unterschiedliche Netze zu routen. [4, S. 87, 88]

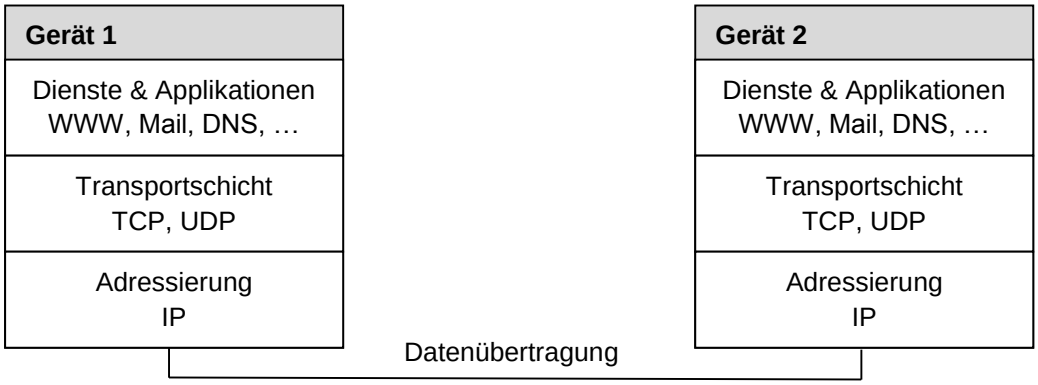

Abbildung 1: Einteilung von TCP, UDP und IP

In der Abbildung 1 werden die oben beschriebenen Protokolle in der richtigen Kommunikationsreihenfolge dargestellt. Es ist zu beachten, dass hier nicht alle Techniken und Protokolle für eine

Netzwerkkommunikation erwähnt werden. Um einen genaueren Einblick in diese Thematik zu bekommen, empfehle ich das Buch „TCP/IP – Konzepte, Protokolle und Architekturen“ von Douglas E. Comer.

2.3. Python

Die späteren Skripte sind Großteils in der Skriptsprache Python geschrieben. Die Auswahl der Sprache basierte auf folgenden Kriterien:

- **Programm-Portabilität:** Die Python Skripte können sowohl unter einem Windows als auch unter einen Unix Betriebssystem ausgeführt werden. Da keine grafische Benutzeroberfläche implementiert wird, sollten zwischen den unterschiedlichen Betriebssystemen keine Änderungen im Quelltext notwendig sein. [5, S. 4] Unter Windows muss allerdings der Python Interpreter nachinstalliert werden, welcher bei Unix/Linux Distributionen bereits in der Standardinstallation enthalten ist.

- **Bibliotheken Unterstützung:** In Python gibt es eine große Sammlung von vorgefertigten Methoden in den Standardbibliotheken. Durch die weite Verbreitung gibt es außerdem eine hohe Anzahl an Bibliotheken von Dritten, welche in die eigenen Skripte eingebunden werden können. [5, S. 4] Dadurch wird Entwicklungszeit gespart, da bestimmte Lösungen nicht neu erfunden werden müssen. Auch für die nachfolgende Automatisierung werden viele Bibliotheken von Dritten verwendet, welche bereits gut ausgereift sind und somit stabil laufen.

- **Komponenten-Integration:** Ein Python Code kann sowohl in andere Python Skripts eingebunden werden, wie auch von einer Vielzahl anderer Programmiersprachen verwendet werden. So können C/C++, Java, .NET mit Python interagieren. Auch für die Kommunikation über das Netzwerk werden Schnittstellen wie SOAP und XML-RPC angeboten. [5, S. 4]

- **Einfachheit, Lesbarkeit und Produktivität:** Die Syntax von Python ist relativ einfach zu erlernen. Man braucht sich nicht um komplizierte Speicherverwaltungen wie z.b in C zu kümmern.
  der Regel 1/3 bis 1/5 so lange wie äquivalenter Code in C++ oder Java“ [5, S. 4] Dadurch wird einerseits Zeit gespart und andererseits bleibt der Code übersichtlich und lesbar.

Wie bereits angesprochen, muss ein Python-Interpreter auf jenem System verfügbar sein, auf welchem ein Python Skript ausgeführt werden soll. Sollte der Interpreter nicht installiert sein, so kann dieser für das entsprechende Betriebssystem auf der Webseite http://python.org heruntergeladen und installiert werden.

Um performante und stabile Programme zu schreiben, sollte ein Entwickler nicht nur die Syntax einer Sprache kennen, sondern es ist auch wichtig zu wissen, wie der Code im Hintergrund verarbeitet wird. In Python ist keine Kompilierung wie in C/C++ notwendig, sondern der Quellcode muss lediglich dem Interpreter übergeben werden. Dieser generiert dann einen sogenannten Bytecode (.pyc Dateien) und gibt diesen zur Ausführung an die Python Virtual Maschine (PVM) weiter. Die PVM führt dann die Anweisungen in einer Schleife Schritt für Schritt aus. Um die Performance zu erhöhen besitzt die PVM einige Optimierungen, wie z.b Cachefunktionen damit nicht jede Anweisung im Quellcode wiederholt analysiert und geparst werden muss. [5, S. 21]

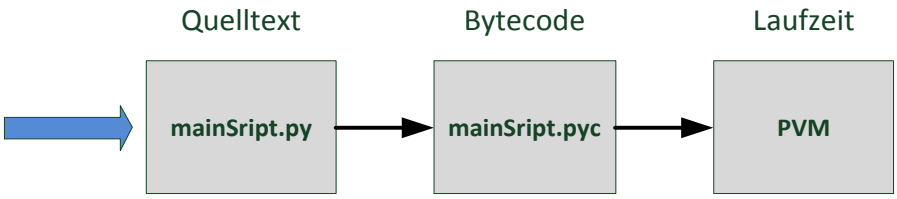

Abbildung 2: Abarbeitung eines Python-Codes vgl. [5, S. 21]

In der Abbildung 2 wird die Abarbeitung eines Python-Codes gezeigt. Dabei wird der Quelltext vom Interpreter in einen Bytecode übersetzt, welcher dann von der PVM ausgeführt wird.

Python ist etwas langsamer als die Sprachen C oder C++, jedoch ist dieser Aspekt für diese Arbeit nicht von großer Bedeutung, da die Geschwindigkeit der Abarbeitung von den externen Programmen sowie der Übertragungsgeschwindigkeit des Netzwerkes abhängig ist. [6, S. 3]

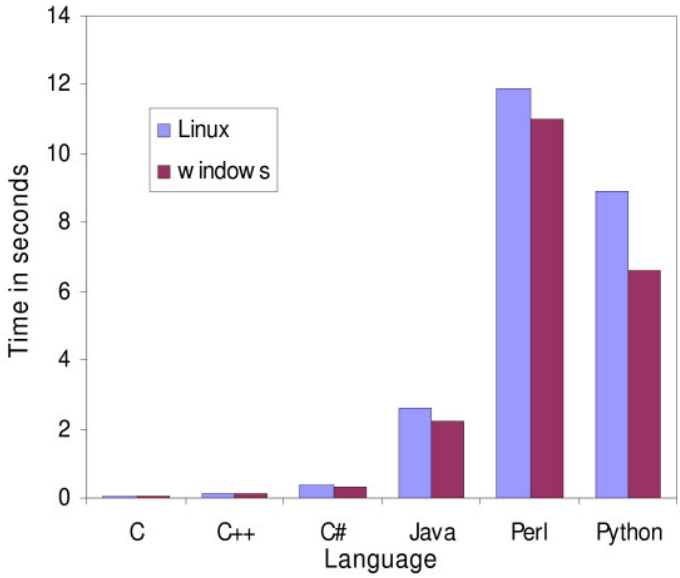

Abbildung 3: Geschwindigkeit von Programmiersprachen [6, S. 3]

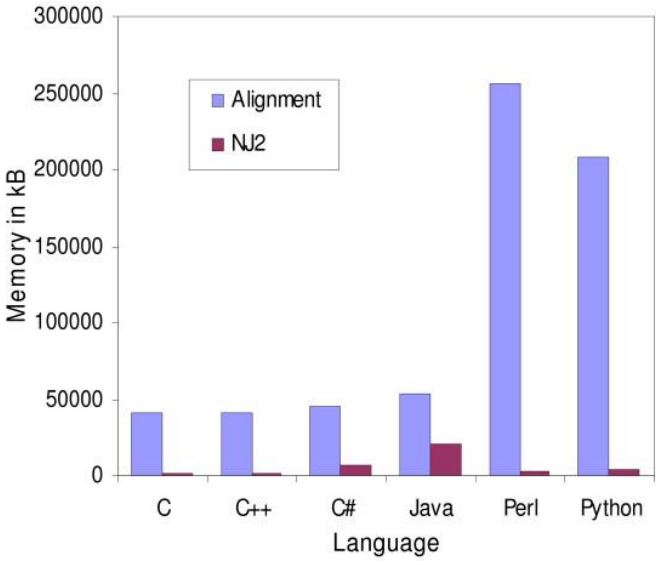

Abbildung 4: Speicherauslastung von Programmiersprachen [6, S. 3]

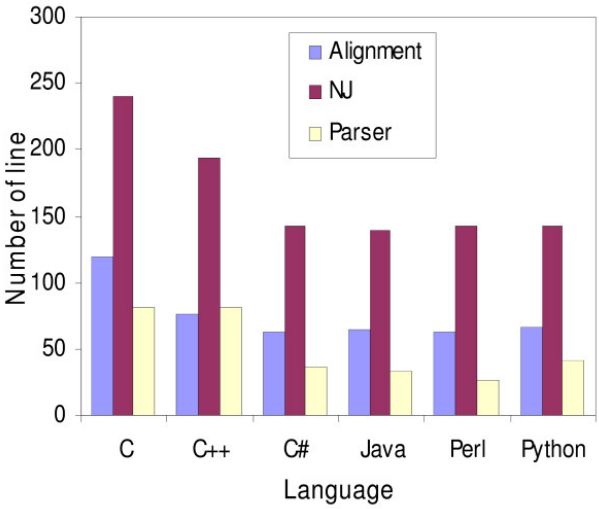

Abbildung 5: Umfang des Quelltextes von Programmiersprachen [6, S. 5]

In den Abbildungen 3 - 5 werden die Programmiersprachen C, C++, C#, Java sowie die Skriptsprachen Perl und Python mit unterschiedlichen Algorithmen aus der Bioinformatik getestet.
In der Abbildung 3 wird die Ausführungsgeschwindigkeit des Neighbor-Joining Algorithmus zwischen allen sechs Programmiersprachen verglichen. Zwischen den beiden Skriptsprachen Perl und Python erkennt man, dass Python sowohl auf Linux und Windows schneller arbeitet. Es ist zu beachten, dass C# eine Windows orientierte Sprache ist und somit das Ergebnis unter Linux nicht ernst genommen werden darf.
Die Abbildung 4 vergleicht die Speicherauslastung zwischen den Sprachen beim Ausführen des Neighbor-Joining Algorithmus sowie ein allgemeines Testprogramms (Alignment).

In der Abbildung 5 sieht man, dass bei den Sprachen C und C++ bei jedem Algorithmus ein höherer Tippaufwand gegenüber den anderen Sprachen besteht. Dafür sind diese beiden Sprachen bei der Geschwindigkeit und Speicherauslastung deutlich effizienter. [6, S. 3-5]

2.4. SQLite

Im Zuge von aktiven oder proaktiven Scans werden ein Menge Daten produziert, welche auf einen Speichermedium abgelegt werden müssen, damit diese Informationen später ausgewertet beziehungsweise analysiert werden können. Um diese Prozesse automatisiert durchführen zu können (z.b das Erstellen eines PDF Berichts), müssen die Daten in einer strukturierten Form abgespeichert werden. Dazu stehen Formate wie XML (Extensible Markup Language), JSON (JavaScript Object Notation), CSV (Comma-Separated Values) oder Datenbanken zu Verfügung.

Der Nachteil von XML, JSON oder CSV gegenüber einer Datenbank ist, dass keine Sprache zum Abfragen und Bearbeiten (Löschen, Verändern, Einfügen) der Daten zu Verfügung steht.
Hierzu kann bei relationalen Datenbank die Datenbanksprache SQL (Structured Query Language) verwendet werden.

Herkömmliche Datenbanksysteme wie Oracle oder Microsoft SQL Server arbeiten nach einem Client-Server Prinzip. Dieses Konzept ist jedoch für diese Arbeit nicht vorteilhaft, da die Sicherheitsüberprüfungen unabhängig von einer vorgegebenen Infrastruktur sowie performante Hardware durchführbar sein sollen. Daher wurde für diese Arbeit SQLite als Datenablage ausgewählt.

SQLite ist eine Open Source, eingebettete relationale Datenbank. Die Datenbank stellt einen Weg für Anwendungen bereit, um Daten ohne Overhead zu verarbeiten. Da der SQLite Prozess als Teil der jeweiligen Anwendung läuft, benötigen diese weder einen zentralen Datenbankserver noch einen dezentralen eigenständigen Prozess bzw. Dienst. Damit dies möglich ist, muss eine API (Application programming interface) für die entsprechenden Sprache geladen werden. [7, S. 1]

Ein Erfolgsfaktor von SQLite ist, dass die in C programmierte API performant und kompakt ist und daher auf vielen Embedded Systemen eingesetzt wird. Die Datenbank wird beispielsweise in den gängigsten Smarthphones und Tabletes wie Android eingesetzt und hat dadurch eine weite Verbreitung. [8, S. 83]

Ein weiteres Entscheidungskriterium für SQLite waren unter anderem die mitgelieferten Exportfunktionen. Beispielsweise können Tabellen mit nur wenigen Befehlen in eine CSV oder HTML Datei exportiert werden. Um solche Exportfunktionen anzuwenden, müssen die folgenden Befehle angewendet werden, um eine Tabelle beispielsweise in eine CSV Datei zu exportieren: [9]

- Die Datenbank mit dem sqlite3 Prozess öffnen

- Ausgabeoptionen konfigurieren (Format, Ausgabepfad & Dateiname)

- Daten abfragen, welche exportiert werden sollen

```

```

- SQLite Prozess beenden

```

```

In Linux Distributionen wird SQLite in einer Standardinstallation mitinstalliert. Sollte SQLite nicht installiert sein, so kann unter http://www.sqlite.org/download.html ein Installationspaket für das gewünschte Betriebssystem heruntergeladen werden.

2.5. Einteilung von Scans zum Aufdecken von Schwachstellen

In den nachfolgenden zwei Hauptkapiteln werden unterschiedliche Methoden zum Aufdecken von Schwachstellen beschrieben. Um den Zusammenhang besser zu verstehen, wird hier ein Überblick gegeben, wann welche Methoden eingesetzt bzw. wie diese kombiniert werden können.

Die Überprüfungen können folgendermaßen eingeteilt werden:

	Proaktive Überprüfung	**Aktive Überprüfung**
Angewendete Technik	Es werden Dienste im Internet abgefragt.	Es werden Datenpakete direkt an die Zielsysteme geschickt.
Einsatzzweck	- Ermitteln von IP Adressen und Domains - Abfragen, ob Zielsysteme auf Blacklists eingetragen sind - Abfragen, ob Schwachstellen von den Zielsystemen bereits bekannt sind - Ermitteln, ob sensible Informationen wie Passwörter über das Internet aufgedeckte werden können.	- Aufdecken von offenen Ports - Eingesetzte Services und Betriebssysteme erkennen - Schwachstellen wie unsichere Verschlüsselungen, Einsatz von veralteten Produktversionen, usw. ermitteln.
zeitlicher Aufwand	Gering, da an die Dienste nur eine Anfrage pro Zielsystem gestellt werden muss.	Hoch, da sehr viele Datenpakete pro Zielsystem gesendet und ausgewertet werden müssen.
Aktualität der gewonnen Information	Das ist für jeden Dienst unterschiedlich zu bewerten. Es muss jedoch beachtet werden, dass	Die Ergebnisse sind aktuell, da die Zielsysteme direkt geprüft werden.

	Ergebnisse veraltet und somit ungültig sein können.	

Um ein gutes Ergebnis zu bekommen, müssen die beiden Methoden kombiniert werden. Über die proaktiven Überprüfungen werden Adressinformationen (Domains und IP Adressen) ermittelt, welche danach in den aktiven Scan mit einbezogen werden. Das bedeutet, man führt in der Regel zuerst einen proaktiven Scan durch und verwendet die gewonnen Ergebnisse als Input für eine aktive Überprüfung.

3. Proaktive Überprüfung

Bei der proaktiven Überprüfung handelt es sich um das Sammeln von digitalen Informationen über ein ausgewähltes Ziel. Häufig sind Daten bzw. Bereiche eines Unternehmens ungewollt öffentlich zugänglich. Deshalb sollte ein sicherheitsbewusstes Unternehmen in regelmäßigen Abständen prüfen, ob unerwünschte Informationen sowie Systeme über das Internet erreichbar sind und diese gegebenenfalls in einen sicheren Bereich verlagern.

In den nachfolgenden Unterkapiteln wird ein tieferer Einblick in das Thema gegeben. Dabei werden namhafte Dienste beschrieben und ein Weg zur Automatisierung aufgezeigt.

3.1. Einführung

Das Aufdecken von sicherheitskritischen Vorfällen bzw. Informationen kann in zwei Kategorien unterteilt werden:

- **Internes Monitoring:** Hierbei werden Ressourcen verwendet, die nur im internen Netzwerk zu Verfügung stehen. Dies können Firewalls, Antiviren Systeme, alle möglichen Logdateien, usw. sein. [10, S. 4, 5]

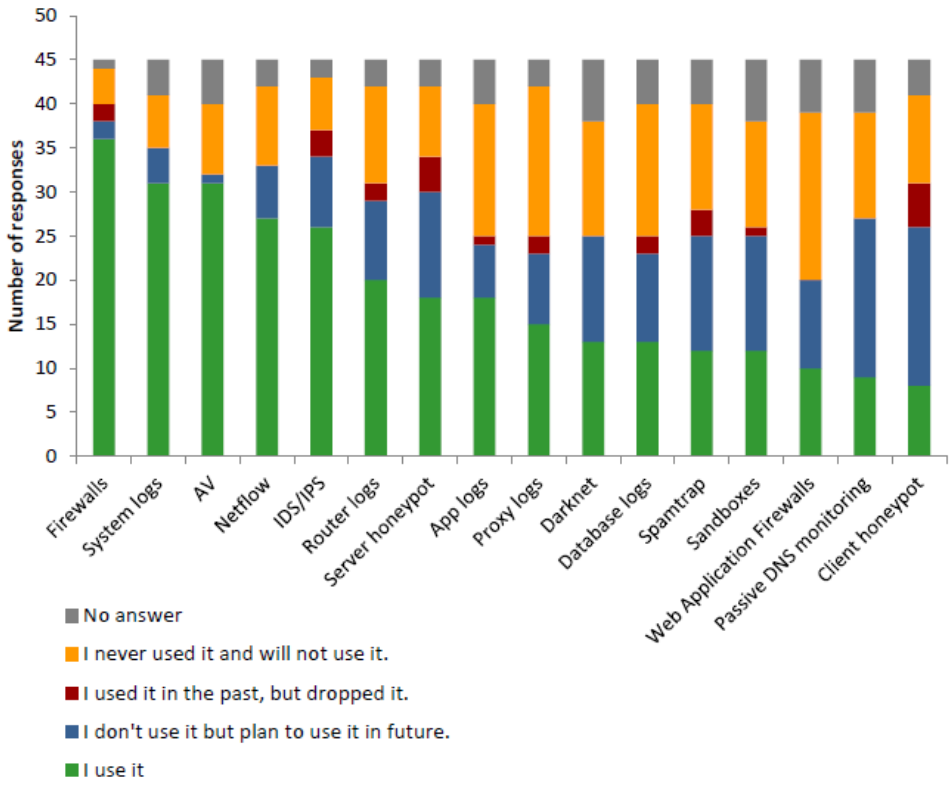

Abbildung 6: Werkzeuge zum Aufdecken interner Schwachstellen [10, S. 15]

In der Abbildung 6 werden die unterschiedlichen Möglichkeiten für internes Monitoring gezeigt. Für diese Statistik wurden Administratoren von öffentlichen Bereichen, Unternehmen, Provider sowie Hochschulen befragt. Aus der Grafik ist zu erkennen, dass komplizierte Werkzeuge für die mehr Wissen erforderlich ist, weniger oft eingesetzt werden. [10, S. 15]
Dieser Teil wird jedoch in dieser Arbeit nicht näher behandelt und wurde nur der Vollständigkeit halber erwähnt.

- **Externe Dienste/Quellen:** Wenn man sich in die Lage eines Angreifers versetzt, hat dieser meist nur wenige Informationen über sein Ziel. Bevor er zum Hacken eines Unternehmens beginnen kann, muss dieser wissen, welche Systeme sowie Dienste überhaupt existieren und welche durch Schwachstellen ausnutzbar sind. Um derartige Informationen zu ermitteln, gibt es Dienste bzw. Quellen im Internet, welche bekannte Schwachstellen, Zugangsdaten, usw. zu Verfügung stellen. Diese Dienste können öffentlich, nicht öffentlich oder kommerziell sein.

  In den nachfolgenden Kapiteln werden die Dienste „Shodan“, „VirusTotal“, „Google Custom Search Engine“, „Bing“ und „Ripe NCC“ näher beschrieben.

  In eine etwas andere Art der externen Quellen fällt das Incoming Incident Reporting. Hierbei handelt es sich um Dienstleister wie Secunia, SecurityFoucus oder SecurityTracker, welche über aktuelle Schwachstellen in diversen Softwareprodukten informieren. Dabei erhält man sogenannte Advisories, die eine Beschreibung (Ausnutzung, Verbreitung, betroffene Systeme, Gegenmaßnahmen) der Schwachstelle enthalten. Wird nun ein betroffenes Produkt im Unternehmen eingesetzt, dann sollte die Schwachstelle mithilfe der im Advisory beschriebenen Gegenmaßnahmen behoben werden.

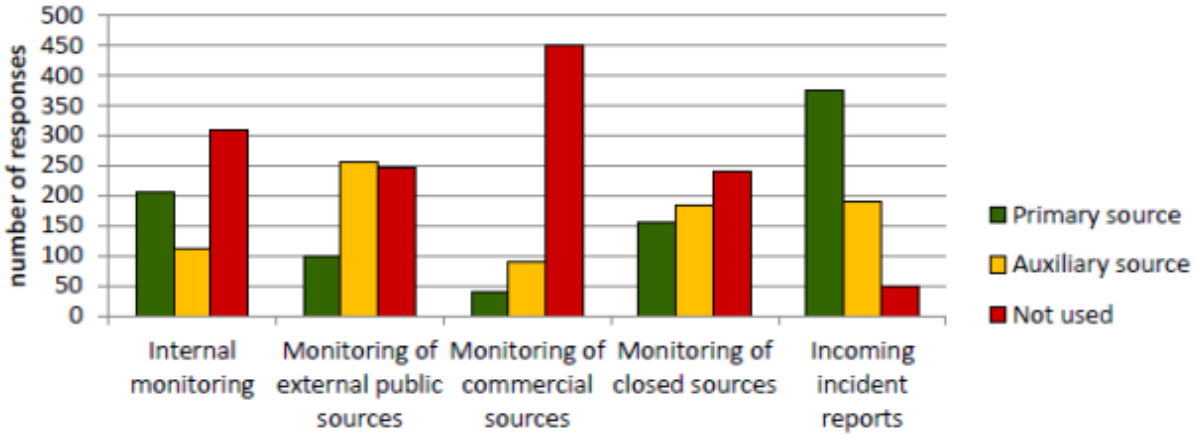

Abbildung 7: Arten von proaktiven Überprüfungen [10, S. 9]

Die Statistik in der Abbildung 7 zeigt, dass „Incoming Incident Repots“ und „Inernal monitoring“ die beliebtesten Methoden zum Ermitteln von Sicherheitskritischen Informationen sind. [10, S. 9] Der Grund für die geringe Nutzung von kommerziellen Quellen dürfte unteranderem der Aufwand durch Formalismus (Verträge, Geldüberweisung, ...) sowie die Möglichkeit von freien Quellen sein. [10, S. 9]

Sollten IT Mitarbeiter eines Unternehmen nicht das Wissen für diese Aufgabe besitzen, so kann diese auch von unterschiedlichen CERTs (Computer Emergency Response Team) durchgeführt werden. CERTs beobachten täglich die aktuelle Entwicklung von Sicherheitslücken, Viren, Trojaner, usw. und sind daher eine empfehlenswerte Anlaufstelle.

3.2. Erforderliche Schritte

Um ein Ziel anzugreifen, sind viele einzelne Schnitte notwendig. In der Regel werden am Begin alle öffentlichen IP Adressen und Domainnamen des Ziels ermittelt. Auf Basis dieser Information können dann diverse Dienste im Internet auf Sicherheitslücken abgefragt werden. Eine proaktive Überprüfung sollte daher in den folgenden Schritten durchgeführt werden:

- **Domains und IP Adressen ermitteln:** Um Angriffe erfolgreich durchführen zu können, müssen so viele öffentlich erreichbare Systeme und Services wie möglich aufgedeckt werden. Dazu können DNS (Domain Name System) Server, Suchmaschinen oder zentrale Registrierungsstellen abfragen werden. Umso mehr Domains oder IP Adressen man besitzt, desto höher wird die Wahrscheinlichkeit, dass auch eine Schwachstelle gefunden wird.

- **Auswahl der Dienste:** Im Internet werden viele Dienste zum Abfragen von Schwachstellen angeboten, jedoch werden hier nur jene beschrieben, welche durch Skripts automatisierbar sind. Einige Dienste können nur durch Eingabe eines CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) verwendet werden und sind daher für die Automatisierung unbrauchbar. Weitere Entscheidungskriterien der Dienste sind [10, S. 22-26]:

  Aktualität: Damit ist die Zeit gemeint, die zwischen Aufdecken einer Schwachstelle und dem zu Verfügung stellen der Informationen über diese Schwachstelle vergeht. Im Idealfall werden die Ergebnisse sofort bereitgestellt.

  Qualität: Hier ist zum einem die Genauigkeit der Ergebnisse und zum anderem die Seriosität des Anbieters gemeint. Ist ein Dienst dafür bekannt, dass viele False Positives Meldungen bereitgestellt werden, so sollte dieser eher gemieden werden.

  Ergebnisformat: Damit die Ergebnisse maschinell weiterverarbeitet werden können, sind Quellen mit einen standardisierten Ergebnisformat (z.b XML oder JSON) gut geeignet. So können die Resultate komfortabel ausgewertet und in der eigene Datenbank abspeichert werden.

- **Zielsysteme auf Schadcode, Blacklists und auffällige Muster prüfen:** Nachdem nun alle Domains und IP Adressen bekannt sind, können diese auf Schadcode getestet werden. Dazu werden die oben ausgewählten Internetquellen abgefragt. Ebenfalls kann geprüft werden, ob Adressen auf Blacklists stehen. Des Weiteren ist es mit Hilfe von Suchmaschinen möglich, auffällige Muster von Webapplikationen zu erkennen. Dies könnte zum Beispiel eine Dateiauflistung („Index of“) von einem Verzeichnis am Webserver sein, wodurch ungewollte Daten bereitgestellt werden.

3.3. Domains und IP Adressen ermitteln

Es wird davon ausgegangen, dass mindestens eine öffentliche IP Adresse bzw. Domain vom Ziel bekannt ist. Das ist in der Regel die Adresse der öffentlichen Webseite des ausgewählten Unternehmens. Da größere Firmen jedoch meist mehrere öffentliche Adressen besitzen (VPN Zugänge, Mail Dienste, …), ist es notwendig diese aufzudecken, um alle erreichbaren Systeme bzw. Dienste auf Schwachstellen überprüfen zu können.

In den nächsten Unterkapiteln wird gezeigt, wie möglichst alle öffentlichen IP Adressen sowie Domains eines Unternehmens ermittelt werden können.

3.3.1. DNS Abfragen

Besitzt man nun eine IP Adresse oder Domainnamen, so kann dies genutzt werden um weitere DNS Einträge zu bekommen. DNS teilt die unterschiedliche Informationstypen in sogenannte Resource Records (RR) ein, welche mithilfe von Tools wie „nslookup“ abgefragt werden können. [11, S. 16] In dieser Arbeit werden die folgenden RR verwendet: [11, S. 16-17]

- **A bzw. AAAA**: Mit diesem RR wird die Zuordnung zwischen Hostnamen und einer IPv4 bzw. Ipv6 Adresse hergestellt.
- **PTR:** Ein PTR Eintrag ermöglich ein Reverse Lookup, d.h damit ist es möglich einer IP Adresse den Hostnamen zuzuordnen.
- **MX:** Damit werden die zuständigen Mailserver einer Domain angegeben. Diese Einträge sind nur vorhanden, wenn ein Maildienst für die Domain genutzt wird.
- **SOA:** Dieser Record enthält wichtige Informationen zur Verwaltung einer Domain. Es wird unteranderem der Primäre Namensserver angegeben.
- **NS:** Mit diesen Record werden entweder die zuständigen Namenserver definiert oder es wird eine Delegatierung durchgeführt.

Um diese theoretischen Aspekte verständlicher zu machen, werden in der Listing 1 DNS Abfragen für die Domain fhtstp.ac.at durchgeführt.

Listing 1: Verwendung von nslookup

Im Listing 1 werden die folgenden Punkte veranschaulicht:

- Zeile 5 -12: Es wird die IPv4 Adresse für den Host www.fhstp.ac.at aufgelöst.

- Zeile 14 – 25: Es werden die zuständigen Mailserver für die Domain fhstp.ac.at ermittelt. Dabei werden die dazugehörigen IP Adressen auch ausgegeben.
- Zeile 27 – 33: Durch ein Reverse Lookup werden die Hostnamen für die IP Adresse 195.202.144.6 aufgedeckt.
- Zeile 35 – 48: Der SOA Eintrag für die Domain fhstp.ac.at wird ausgelesen. Dabei erfährt man den primären Namenserver (n1.fh-stpoelten.ac.at)

Somit ist die IP Adresse des Webservers, die Adressen der Mailserver sowie der primäre Namenserver bekannt. Es wurde also mithilfe von einem Domainnamen (Webseite der FH St. Pölten) drei weitere Adressen (195.202.144.4, 195.202.144.2, n1.fh-stpoelten.ac.at), welche als Grundlage für die für die späteren Angriffe dienen aufgedeckt.

Um diesen Vorgang in Python zu automatisieren, muss das Modul „adns“ in das Skript eingebunden werden. Dieses besitzt ein Schnittstelle zu „GNU adns“, wobei es sich um eine C/C++ Bibliothek zum Auflösen von DNS Anfragen handelt. Das Modul kann unter http://adns-python.googlecode.com/files/adns-python-1.2.1.tar.gz herunter geladen werden. [12]

Danach kann der folgende Code implementiert werden:

```
'          '

'                    '

'          '

'          '
```

Listing 2: DNS Abfragen in Python

In dem Code Ausschnitt von Listing 2 werden die folgenden Python Anweisungen ausgeführt:

- Zeile 1: Das Modul wird in das Skript eingebunden
- Zeile 3: Eine neue Instanz wird erstellt
- Zeile 5: Mit der erstellten Instanz wird eine A RR Abfrage durchgeführt
- Zeile 7: Mit der erstellten Instanz wird eine PTR RR Abfrage durchgeführt
- Zeile 9: Mit der erstellten Instanz wird eine MX RR Abfrage durchgeführt
- Zeile 11: Mit der erstellten Instanz wird eine SOA RR Abfrage durchgeführt

Da mit den hier beschriebenen DNS Abfragen nicht alle öffentlichen Adressen eines Unternehmens ermittelt werden können, wird im nachfolgendem Kapitel die Datenbank von jener Organisation abgefragt, welche für die Vergabe von IP Adressen verantwortlich ist.

3.3.2. RIPE NCC

Die RIPE NCC (Réseaux IP Européens Network Coordination Centre) ist eine gemeinnützige, unabhängige Organisation, welche für die Vergabe von IP Adressen und AS (Autonome System) Nummern in Europa, dem Nahen Osten und Zentralasien zuständig ist. [13]

Die RIPE stellt eine online Datenbank mit alle registrierten IP Adressen zu Verfügung. Diese Adressen können unter https://apps.db.ripe.net/search/query.html abgefragt werden. Dazu werden Informationen wie Besitzer, IP Range, Land, usw einer IP Adresse preisgegeben.

Um die Abfrage zu demonstrieren, suchen wir hier alle Information zu der zuvor ermittelten IP Adresse der FH St. Pölten.

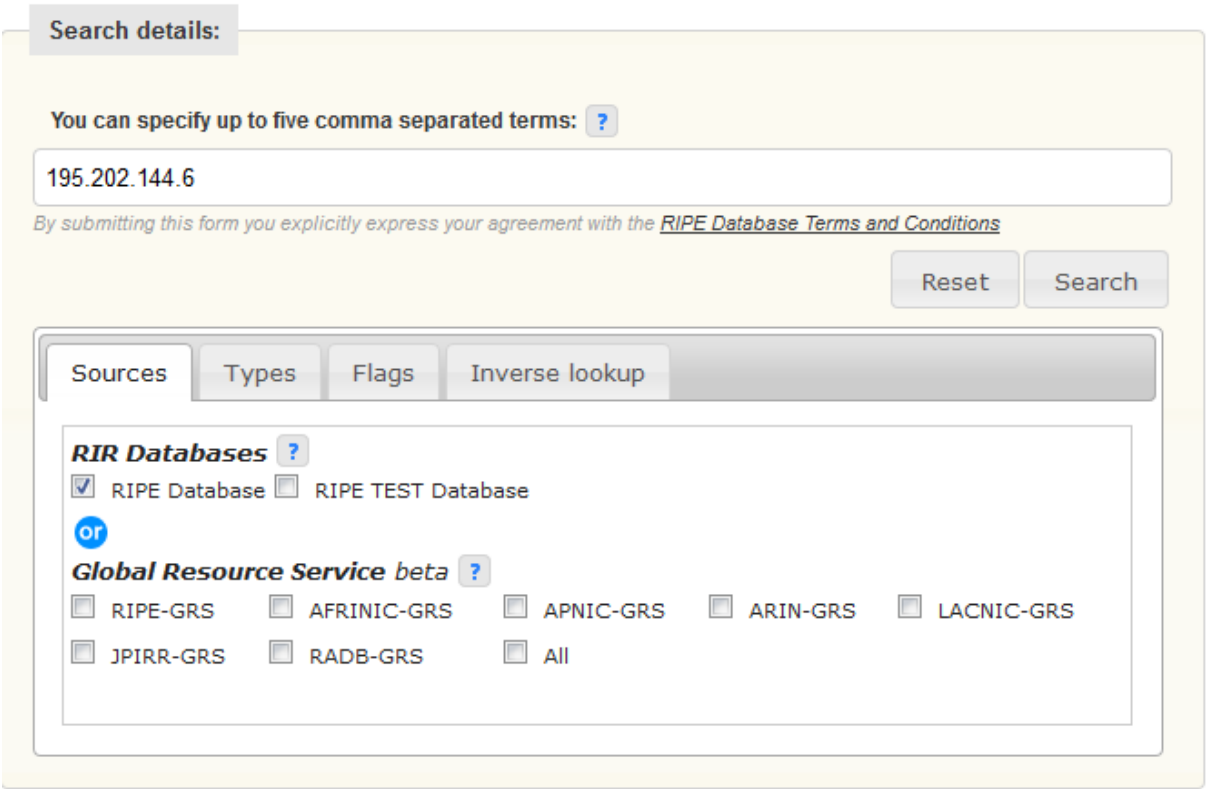

Abbildung 8: RIPE Abfrage (https://apps.db.ripe.net/search/query.html)

In der Abbildung 8 wird eine RIPE NCC Abfrage für die IP Adresse der FH St. Pölten durchgeführt. Das relevanteste Attribut in dem Suchergebnis ist der IP Bereich indem die Adresse fällt. Dadurch lässt sich sagen, dass die FH St. Pölten noch weitere 62 öffentliche Adressen besitzt.

Somit wurden 62 Adressen auf Basis der URL von der öffentlichen Webseite aufgedeckt. Da jedoch auf eine IP Adresse mehrere Domains verweisen können, müssen im nächsten Schritt alle indizierten Domainnamen der aufgedeckten IP Adressen ermittelt werden. Diese Technik wird im nächsten Kapitel näher beschrieben.

3.3.3. Bing

Bei Bing handelt es sich um eine Suchmaschine von Microsoft. Mithilfe einer speziellen Funktionalität dieser Suchmaschine kann herausgefunden werden, welche Domains von einer IP Adresse gehostet werden. Damit kann ein Angreifer Webseiten bzw. Webapplikationen aufdecken, welche eventuell Schwachstellen enthalten und somit ausgenutzt werden können.

Um diese Funktionalität zu nutzen, muss unter http://www.bing.com/ im Suchfeld der Suchoperator „IP:“ und die gewünschte Adresse eingegeben werden. In der Abbildung 9 wird dies mit der IP Adresse der FH St. Pölten demonstriert. Im Ergebnis werden alle Domains aufgelistet, welche zu dieser IP Adresse verweisen.

Dieser Vorgang kann nun mit den restlichen 62 öffentlichen Adressen durchgeführt werden.

Abbildung 9: Bing Abfrage (http://www.bing.com)

Einige Domainnamen aus dem Suchergebnis von Abbildung 9 sind:

- english.fhstp.ac.at
- sport.fhstp.ac.at
- ball.fhstp.ac.at
- wwwtest.fhstp.ac.at
- …

Da es ein hoher manueller Aufwand ist, alle 62 öffentlichen IP Adressen einzugeben und die Ergebnisse auszuwerten, wird dieser Schritt wieder mit einem Python Skript automatisiert. Dazu wird das Modul

„pyping“ in das Skript eingebunden. Dieses kann mit folgenden Befehl unter Linux nachinstalliert werden: [14]

Um die Bing API zu verwenden, wird ein API Key benötigt. Dieser kann unter http://www.bing.com/developers/createapp.aspx angefordert werden. Dazu wird ein Micorsoft Zugang vorausgesetzt (MNS Messenger, Hotmail, ...), welcher bei nichtvorhandensein kostenlos erstellt werden kann. [14]

Danach kann der folgende Code implementiert werden:

Listing 3: Bing Abfragen in Python

Bei den Python Anweisungen im Listing 3 wurden folgende Aktionen durchgeführt:

- Zeile 1: Das Modul “pybing” wird eingebunden
- Zeile 3: Eine neue Instanz wird erstellt
- Zeile 5: Eine Abfrage wird an Bing geschickt. Bei mehreren Abfragen kann dieser Schritt in einer Schleife wiederholt werden.
- Zeile 7 – 8: Die Ergebnisse der Abfrage werden in einer Schleife durchlaufen und zur Demonstration wird die gefundene URL ausgegeben

Durch diesen Vorgang wurden nun weitere öffentliche Domains ermittelt. Diese dienen als Grundlage für die weiteren Überprüfungen.

3.4. Software- und Protokollinformationen ermitteln

Da nun die öffentlichen Adressen über das Ziel gesammelt wurden, kann dies genutzt werden, um Informationen über die Systeme und Dienste hinter diesen Adressen zu ermitteln. Für diesen Zweck stehen spezielle Suchmaschinen wie Shodan zu Verfügung. Der Dienst wertet die Banner von Web-, FTP-, SNMP-, SSH- und Telnet-Servern weltweit aus, um Angaben über das Produkt, die Version und diverse weiteren Daten zu enthalten. [15, S. 5-7]

Die Suchmaschine von John Matherly stellt verschiedene Filtermöglichkeiten zu Verfügung. Es kann nach Ländern, Services, Hostnamen und Betriebssystemen gefiltert werden. Um beispielsweise alle indizierten Apache Server der FH St. Pölten zu finden, genügt es, auf Shodan als Suchbegriff “apache hostname:.fh-stpoelten.ac.at“ einzugeben um eine Liste mit den gewünschten Informationen zu bekommen. Mehrere Suchattribute können mit “+“ oder “–“ kombiniert werden. [15, S. 14-23] Alle zu Verfügung stehenden Suchattribute werden unter http://www.shodanhq.com/help/filters beschrieben.

Shodan steht frei und ohne Registrierung zu Verfügung. Mit einem Login können jedoch zusätzliche Export- und Filterfunktionen benutzt werden. Wenn man sich nicht neu registrieren möchte, können auch Konten von Google, Twitter, Yahoo, AOL, Facebook oder OpenID als Zugang verwendet werden. [15, S. 12, 13]

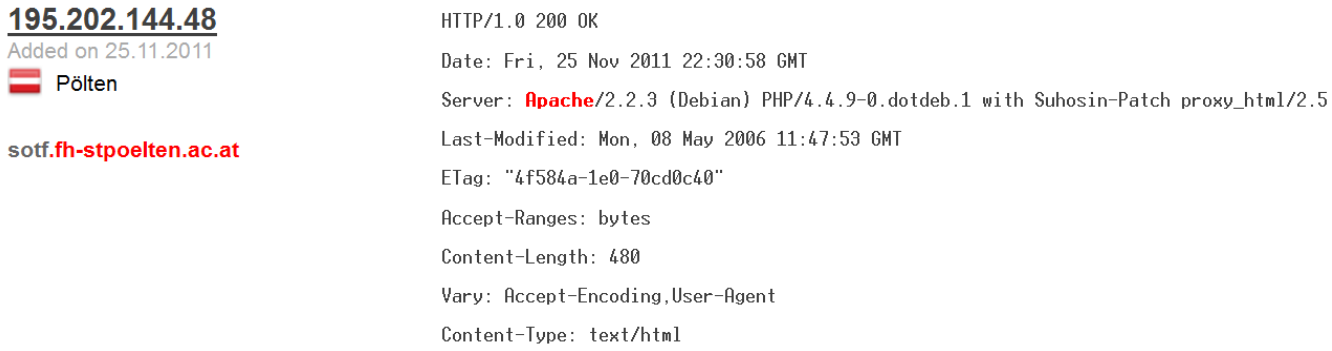

Abbildung 10: Suchergebnis von Shodan

In der Abbildung 10 wird ein Suchergebnis von Shodan gezeigt. Bei den Ergebnissen sollte immer das Datum der Indizierung (Added on) beachtet werden, da der Eintrag bei einem älteren Datum eventuell nicht mehr gültig ist. In diesem Fall wurde der Host am 25.11.2011 indiziert. Zusätzlich zum Datum werden Metadaten aus dem HTTP Response sowie das Land und die Stadt indem sich der Host befindet aufgelistet.

Um diese Abfragen zu automatisieren, muss die Python Bibliothek für Shodan nachinstalliert werden. Dazu unter Linux den folgenden Befehl ausführen:

Auch hier wird zum Benutzen der Shodan API ein Key gefordert. Dieser kann nach einem Login kostenlos angefordert werden. Dazu die beschriebenen Schritte auf der Seite http://www.shodanhq.com/api_doc durchführen.

Danach kann der folgende Code implementiert werden:

Listing 4: Shodan Abfragen in Python

Bei den Python Anweisungen im Listing 4 wurden folgende Aktionen durchgeführt:

- Zeile 1: Das Modul “shodan” wird eingebunden
- Zeile 3: Es wird die Funktion „WebAPI“ vom Modul „shodan“ aufgerufen. Als Parameter wird der zuvor angeforderte API Key übergeben. Als Rückgabewert bekommt man ein neues Objekt.
- Zeile 5: Eine Suchanfrage wird an Shodan geschickt. Bei mehreren Abfragen kann dieser Schritt in einer Schleife wiederholt werden.
- Zeile 7 – 8: Die Ergebnisse der Abfrage werden in einer Schleife durchlaufen und zur Demonstration wird das Datum der Indizierung sowie der Hostname ausgegeben.

Alle weiteren Details zur Shodan API können unter http://docs.shodanhq.com/python/tutorial.html nachgelesen werden.

Durch diese Suchabfragen wurden die eingesetzten Produkte sowie deren Versionen auf den Zielsystemen aufgedeckt. Nun kann recherchiert werden, ob gefundene Produktversionen veraltet und somit durch bekannte Schwachstellen ausnutzbar sind.

3.5. Ziele auf Schadcode überprüfen

Da es sowohl für einen Angreifer als auch für ein Unternehmen interessant ist, ob Webseiten bereits Schadcode enthalten, kann dies über Webdienste abgefragt werden. Der Anbieter VirusTotal bietet dabei sowohl die Überprüfung von Dateien als auch von Webseiten an.

Bei VirusTotal werden über 60 Viren-Engines abgefragt. Der Einsatz der Vielzahl von Antivirenprogrammen ermöglicht es, verlässlichere Ergebnisse als mit nur einem Scanner zu erhalten. [16] Die manuelle Überprüfung einer URL kann entweder direkt über die Webseite https://www.virustotal.com sowie über eine Mozilla Firefox Extension durchgeführt werden. Um automatisierte Abfragen zu tätigen, kann die öffentliche API verwendet werden. [16]

Um die API zu nutzen, muss lediglich ein POST Request mit den nachfolgenden Parameter an https://www.virustotal.com/vtapi/v2/url/scan übermittelt werden: [17]

- **url:** Mit diesem Parameter werden die zu überprüfenden Adressen angegeben. Um mit einem Request mehrere Urls abzuarbeiten, müssen diese mit Beistrich getrennt angegeben werden.
- **apikey:** Um Abfragen über die API zu tätigen, muss ein Konto erstellt werden um einen persönlichen API Key zu bekommen. Dieser muss dann mit diesem Parameter übertragen werden.

Es können maximal vier Request pro Minute durchgeführt werden. Als Ergebnis erhält man ein JSON Objekt. [17]

Die Automatisierung kann mit folgenden Python Code implementiert:

Listing 5: VirusTotal Abfragen in Python

Im Listing 5 werden die folgenden Python Anweisungen durchgeführt:

- Zeile1: Die Bibliothek “simplejson” wird eingebunden. Diese wird zur Verarbeitung von JSON Objekte verwendet.
- Zeile 2 - 3: Um HTTP Requests zu erzeugen, werden die Bibliotheken „urllib“ und „urllib2“ eingebunden.
- Zeile 5 – 6: Alle benötigten Daten für den HTTP Post Request werden zusammengestellt. Wie zuvor erwähnt, können hier mehrere URLs angegeben werden.
- Zeile 8: Mithilfe dem „urlilib“ - Modul werden die Parameter in ein geeignetes Format konvertiert.
- Zeile 9 – 10: Der Request wird gesendet. Die Antwort wird in einem neuen Objekt abgespeichert.
- Zeile 12 – 13: Das JSON Objekt aus dem HTTP Response wird nun gelesen und mittels „simplejson“ verarbeitet.
- Zeile 15: Das neu entstandene Objekt wird nun in einer Schleife durchlaufen und dabei wird auf das Feld „result“ des Arrays zugegriffen und ausgegeben.

Sollte ein Unternehmen merken, dass eine Ihrer Webseiten mit Schadecode infiziert ist, so sollte diese schnellstmöglich bereinigt werden, da das bekanntwerden einer verseuchten Unternehmenswebseite einen fatalen Imageverlust bedeuten kann.

3.6. Sensible Informationen aufdecken

Suchmaschinen durchlaufen mit Hilfe von Webcrawlern das Internet und fügen gefundene Dokumente zu ihrem Suchindex hinzu. Nach eigenen Angaben ermittelte Google im Jahr 2008 das Vorhandensein von

mehr als einer Billion URLs im Internet. [18] Bei der Indizierung wird eine Kopie des Dokuments sowie Metadaten wie Dateigröße, Dokumententitel und URL in der Datenbank abgespeichert. [19, S. 492]

Wird eine einfache Sucheingabe mit mehreren Begriffen erstellt, so werden diese mit einem booleschen UND verknüpft um nach allen Begriffen zu suchen. Um das Sucherergebnis zu verfeinern, stellt Google unter http://www.google.com/help/operators.html öffentliche Suchoperatoren zu Verfügung. So könnte man in der Suche mit einen Minus Zeichen (-) Begriffe angeben, welche im Suchergebnis nicht vorkommen sollen.

In der Hackszene sind über die Jahre nicht veröffentliche Suchoperatoren von Google aufgedeckt worden. Zu den bekanntesten und wichtigsten zählen: [19, S. 493]

- **intitle:** Mit der Abfrage „intitle“ ist es möglich, nur nach Dokumenten mit einen bestimmten Titel zu suchen. Um zum Beispiel Webserver zu suchen, welche durch eine unsichere Konfiguration eine Verzeichnisauflistung erlauben, kann der Suchbegriff „intitle:“Index of““ verwendet werden.

- **inurl:** Um nach bestimmten Vorkommnissen in der URL zu suchen, kann der Operator “inurl” angewendet werden. So könnten beispielsweise mit der Suchabfrage “inurl:indexFrame.shtml Axis“ öffentliche Webcams von dem Hersteller Axis gesucht werden. Das Ergebnis ist erstaunlich: Man bekommt Zugriff auf die Steuerung unterschiedlichster Webcams und sogar Einblick in die Straßenkameras einer Regierung.

- **filetype:** Damit nur gewisse Dokumente im Suchergebnis angezeigt werden, kann der “filetype” Operator eingesetzt werden. Dabei kann Google den Inhalt von 13 Datentypen auslesen und somit indizieren. Zu diesen zählen Adobe Portable Document Format (pdf), Postscript (ps), Microsoft’s Excel, Word, PowerPoint, Works, Write (xls(x), doc(x), ppt(x), wks, wri), Rich Text Format (rtf), Shockwave Flash (swf) und Text (ans, txt) Dateien.

- **site:** Dieser Operator wird angewendet, wenn nur in bestimmte Domains gesucht werden soll. Um zum Beispiel alle Österreichischen Regierungsseiten zu durchsuchen, kann der Suchbegriff “site:gv.at“ in die Abfrage miteinbezogen werden.

- **link:** Damit können Seiten aufgedeckt werden, welche eine Verbindung zu einer bestimmten URL haben. Die FH St.Pölten könnte mit dem Suchbegriff link:fhstp.ac.at überprüfen, welche Seiten auf die eigene Webseite verweisen.

Durch diese Operatoren und deren Kombination können sensible Information aus der Google Datenbank abgefragt werden. Diese Methode der Informationsbeschaffung wird in der Community als Google Hacking bezeichnet. [19, S. 491]

Über die Jahre haben Sicherheitsexperten, Hacker, usw. Suchabfragen entwickelt, um gezielt Schwachstellen oder sensible Informationen über die Google Suchmaschine aufzudecken. Diese Sammlung ist in der Google Hacking Database (GHDB) von Johnny Long abgelegt und steht frei zu Verfügung. Ein Eintrag in dieser Datenbank wird als Dork bezeichnet. [19, S. 494] Derzeit besitzt die GHDB 4459 Dorks und kann unter http://www.stachliu.com/resources/tools/google-hacking-diggity-project/attack-tools/ kostenlos herunter geladen werden.

Um Webseiten mit der GBHD automatisiert zu überprüfen, muss unter http://www.google.com/cse/ eine neue Google Custom Search (CSE) angelegt werden. In der neuen CSE werden dann alle zu durchsuchenden Domains angegeben. In der Abbildung 11 wird dies mit einen Beispiel veranschaulicht. Mit dieser Suchmaschine würden die vier angegebenen Domains mit der gesamten Dorkliste durchsucht Nachdem die Suchmaschine erstellt wurde, bekommt man in der URL die ID dieser Suchmaschine angezeigt.

Beispiel:

URL: http://www.google.com/cse/home?cx=016070887149123456789:z0lignygxyz

➔ Die ID wäre in diesem Fall 016070887149123456789:z0lignygxyz

Des Weiteren muss zum Verwenden der Google API ein API Key unter https://code.google.com/apis/console generiert werden.

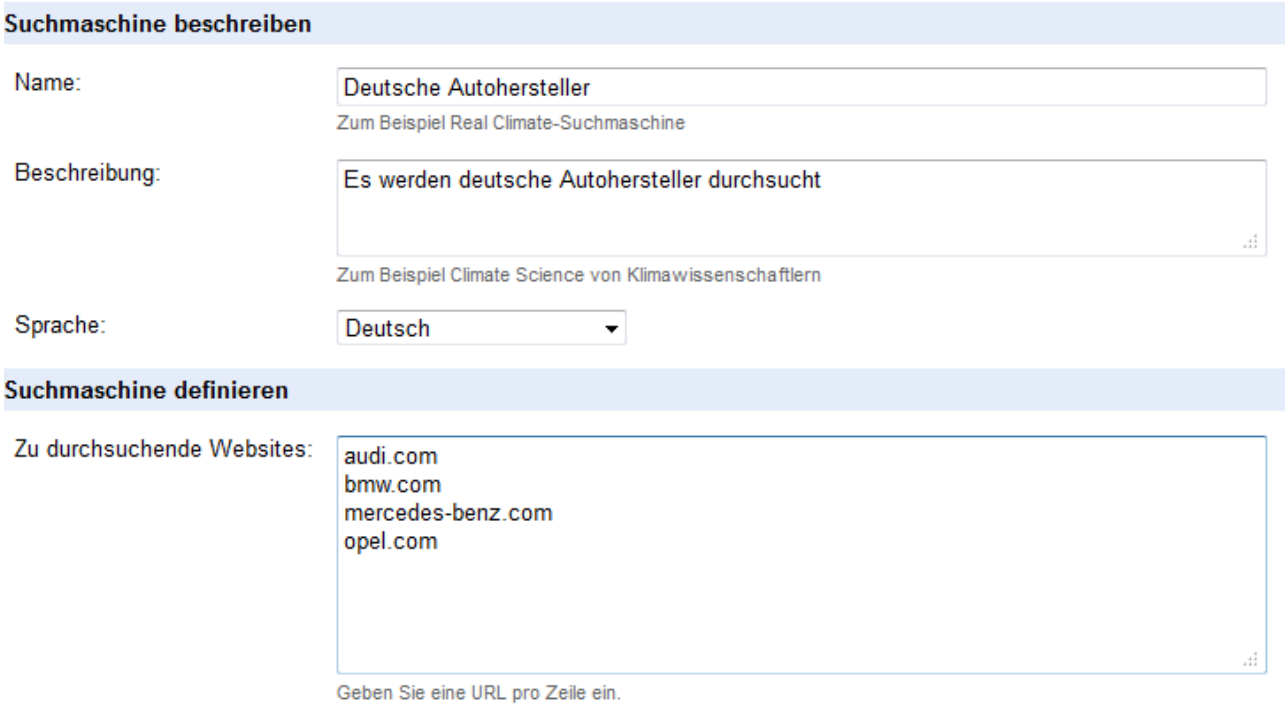

Abbildung 11: Neue CSE erstellen

Nachdem nun die CSE sowie der API Key vorhanden sind, kann der nachfolgende Python Code implementiert werden:

Listing 6: Google Abfragen in Python

Im Listing 6 werden die folgenden Anweisungen durchgeführt:

- Zeile 1: Die Bibliothek zum Ansprechen der Google API wird eingebunden.
- Zeile 3: Es wird die Funktion „build“ von der zuvor eingebunden Bibliothek aufgerufen. Die Funktion erwartet als dritten Parameter einen API Key. Die ersten beiden Parameter besagen, dass die Custom Search v1 verwendet werden soll. Als Rückgabewert bekommt man ein neues Objekt.
- Zeile 5 – 8: Da die exportierte GHDB Datei zusätzlich zu dem eigentlichen Dorks noch weitere Informationen wie zum Beispiel eine Beschreibung besitzt, muss jede Zeile bei dem Zeichen ;; gesplittet werden um den benötigte Dork von der dritten Spalte herauszufiltern.
- Zeile 10: Alle eingelesenen Dorks werden nun in einer Schleife durchlaufen.
- Zeile 11: Es wird eine Suchanfrage an die zuvor erstellte Suchmaschine gestellt. Als Suchbegriff wird der gerade durchlaufene Dork verwendet. Um eine Verbindung zur Suchmaschine herzustellen, muss die ID der Custom Search Engine angegeben werden (cseID).
- Zeile 13 – 14: Die Ergebnisse der Suchabfrage werden in einer Schleife durchlaufen und dabei wird die URL vom Ergebnis ausgegeben.

Sollten Dorkabfragen Ergebnisse liefern, so müssen diese auf Relevanz geprüft werden. Eventuell sind PDF Dateien gefunden worden, welche jedoch bewusst in einen öffentlichen Bereich liegen und daher keine Gefahr darstellt. Wurden jedoch Zugangsdaten oder ähnlich kritische Informationen aufgedeckt, so wäre es ratsam, diese schnellstmöglich in einen sicheren Bereich zu verlagern.

4. Aktive Überprüfung

Bei den aktiven Überprüfungen werden ausgewählte Systeme bzw. Dienste direkt geprüft, d.h. es sind keine externen Quellen notwendig. Um mit dieser Methode offene Zugänge bzw. Schwachstellen aufzudecken, werden Datenpakete an das Ziel gesendet und deren Antworten ausgewertet. Die gewonnen Informationen sind somit aktuell. Da im Gegensatz zu den proaktiven Überprüfungen viele Datenpakete übertragen werden müssen, nehmen aktive Scans einen deutlich längeren Zeitraum in Anspruch.

Es ist zu beachten, dass eine aktive Überprüfung keine Alternative zu einer proaktiven Überprüfung ist, da viele Ergebnisse wie IP Adresse und Domains aus den proaktiven Scans als Basis für die aktiven Scans dienen.

In den nachfolgenden Unterkapiteln werden zwei bekannte Tools zur Realisierung der aktiven Überprüfung vorgestellt. Dabei werden die eingesetzten Methoden beschrieben, sowie Automatisierungskonzepte vorgeschlagen.

4.1. NMAP Scan

Bei NMAP handelt es sich um ein verbreitetes Programm für Portscans, welches alle gängigen Scan-Techniken beherrscht. [20, S. 91] Dabei beinhaltet NMAP nützliche Funktionen um Dienste und Betriebssysteme sowie deren Versionsstände aufzudecken. Es kann auch festgestellt werden, ob Firewalls stateful arbeiten oder ob ein Spoofen von TCP-Verbindungen möglich ist. [21, S. 290]

Während NMAP bei den meisten Linux-Distributionen bereits installiert ist, muss das Programm bei Windows mit einer separaten Installation nachgerüstet werden. [21, S. 290] Aktuelle Installationspakete können kostenlos unter http://nmap.org/download.html heruntergeladen werden.

4.1.1. TCP/UDP Portscan

Das Ziel eines Portscan ist es festzustellen, ob Ports auf bestimmten Zielsystemen geöffnet sind und ein Dienst aktiv ist. Um dies zu erreichen, werden entsprechend präparierte TCP- und UDP Datenpakete an die Zielsysteme gesendet und deren Antwort ausgewertet. [20, S. 87] Dabei können diese Scans in folgende Kategorien eingeteilt werden: [20, S. 87]

- **Vertikaler Portscan:** Hierbei werden auf einen bestimmten Ziel eine Reihe von Ports geprüft. Dadurch dass ein konkreter Host gescannt wird, kann dies relativ einfach von einem Intrusion-Detection-System (IDS) erkannt werden.

- **Horizontaler Portscan:** Bei dieser Methoden werden an vielen Zielsystemen ein bestimmter Port gescannt. Dadurch kann systematisch nach einen verwundbaren offenen Dienst gesucht werden. Da hiermit nur ein einziger Zugriff pro Host notwendig ist, ist die Erkennung von einem IDS wesentlich schwieriger.

- **Blockscan:** Diese Methode ist eine Kombination aus dem Vertikalen und Horizontaler Portscan. Es werden sowohl viele Zielsysteme als auch Ports geprüft.

Um nun offene TCP Ports zu finden, können unterschiedliche Techniken angewendet werden. Jede der nachfolgenden Methode nützt die TCP Flags, um eine Aussage über den Portstatus des Zielsystems zu treffen. Je nach Szenario muss entschieden werden, welche Methode am sinnvollsten einzusetzen ist.

Folgende Methoden stehen zu Verfügung: [20, S. 89-91]

- **SYN/"half open" Scan:** Bei dieser Scan-Technik wird keine vollständige TCP Session aufgebaut. Sollte vom Zielsystem durch antworten eines TCP SYN|ACK-Flag ein offener Port signalisieren werden, so wird ein TCP RST-Flag gesendet um den Verbindungsaufbau abzubrechen. Wird ein TCP RST|ACK-Flag empfangen, so werden keine weiteren Pakete gesendet und der Port wird als geschlossen behandelt. (siehe Abbildung 12)

  Da hier kein üblicher Verbindungsaufbau stattfindet, werden Administrator bzw. Root Rechte für das Scannertool benötigt.

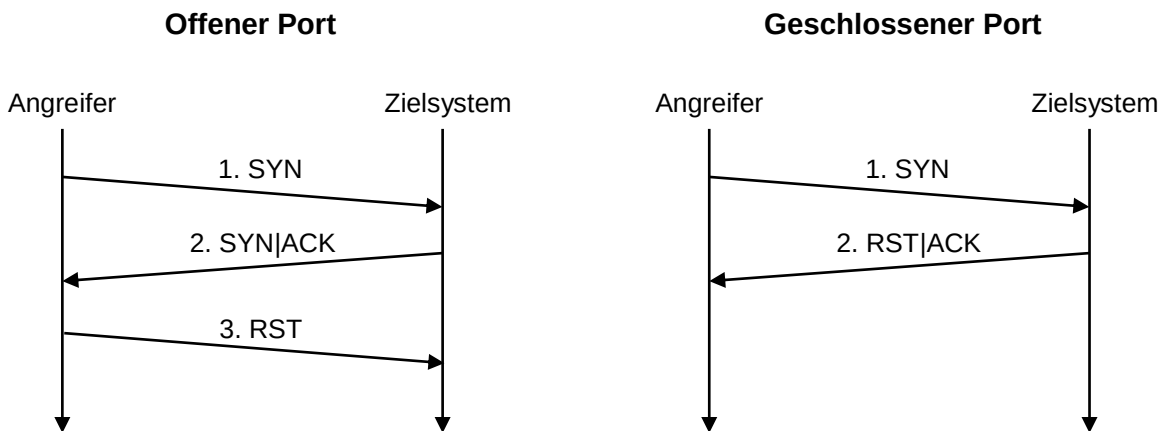

Abbildung 12: Ablauf eines SYN Scan/"half open" Scan, vgl. [20, S. 89]

Dieser Scan kann von NMAP mit dem Parameter „-sS“ durchgeführt werden:

- **Connect Scan:** Hier wird versucht eine vollständige TCP Session aufzubauen. Diese Überprüfung ist im Vergleich mit dem SYN/"half open" Scan zeitaufwändiger, da mehrere Pakete übertragen werden. Da hier ein üblicher Verbindungsaufbau stattfindet, können Standard APIs bzw. System Calls verwendet werden und es sind somit keine Administrator bzw. Root Rechte für das Scannertool notwendig.

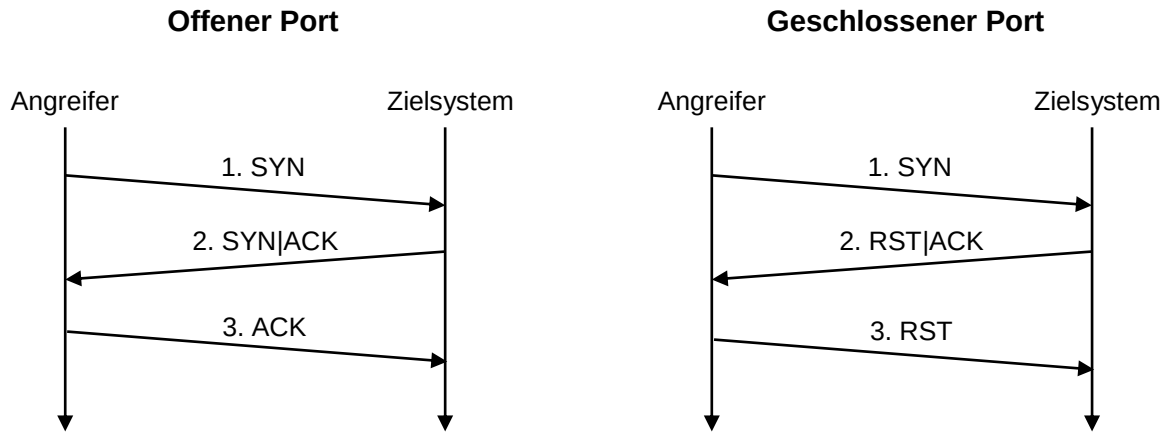

Abbildung 13: Ablauf eines Connect Scan, vgl. [20, S. 88]

Wie man in der Abbildung 13 sieht, wird nach dem Erhalt eines TCP Paket mit gesetzten RST|ACK-Flag mit einem RST-Flag geantwortet, um die Verbindung ordnungsgemäß abzubauen.

Dieser Scan kann von NMAP mit dem Parameter „-sT“ durchgeführt werden:

- **ACK/NULL/FIN/XMAS Scan:** Diese Überprüfungen werden eingesetzt um Hosts hinter einer Firewall zu erkennen. Dabei werden TCP Pakete mit unterschiedlichen Flag-Kombinationen an einen Port des Zielsystems gesendet. Es kann nicht garantiert werden, ob ein Port offen oder geschlossen ist, jedoch ob der Port durch eine Firewall gefiltert oder nicht gefiltert wird.

Nachdem verbreitete Dienste wie zum Beispiel DNS das Transportprotokoll UDP verwenden, müssen auch diese Ports aufgedeckt werden. Da bei UDP kein Verbindungsaufbau durchgeführt wird, muss eine andere Scan-Technik angewendet werden: Es wird ein leeres UDP Pakete an das Zielsystem gesendet. Antwortet dieses System mit einem „ICMP port unreachable“ Nachricht, so ist der Port geschlossen. Wenn keine Antwort oder ein UDP Paket als Antwort empfangen wird, so ist dies ein Anzeichen für einen offenen Port. [20, S. 91] Der Nachteil dieser Technik ist, dass beim Blocken eines Ports durch ein Firewall falscherweise dieser als offener gekennzeichnet wird. Um diesem Problem entgegen zu wirken, können zusätzlich Anwendungsspezifische UDP Pakete gesendet und ausgewertet werden. So kann beispielsweise eine DNS Anfrage auf Port 53 geschickt und auf eine korrekte Antwort gewartet werden. Bei einen NMAP-UDP Scan wird dieses Konzept angewendet. [22]

Ein UDP Scan kann mittels NMAP mit dem Parameter „-sU“ durchgeführt werden:

4.1.2. Dienst und Betriebssystem Erkennung

Wie Eingangs bereits erwähnt, kann zusätzlich zu den Portstatus auch der eingesetzte Dienst und das Betriebssysteme sowie deren Versionsstände ermittelt werden. Diese Informationen können für einen Angreifer hilfreich sein, da hiermit ältere und somit verwundbare Systeme aufgedeckt werden können. Diese Funktion kann durch die folgenden Techniken bewerkstelligt werden:

- **Fingerprinting:** Hier werden präparierte Datenpakete an offene und geschlossene Ports des Zielsystems gesendet. Die Antwort wird mit einer Datenbank abgeglichen, in der unterschiedlichste Betriebssystem-Signaturen abgespeichert sind. Gibt es eine Übereinstimmung, so kann eine Aussage über das entfernte Betriebssystem getroffen werden. Hierbei wird also die unterschiedliche Implementierung des Netzwerk-Stacks von Betriebssystemen ausgenutzt. [23, S. 3]

  Damit man auch aktuelle Patchstände von Betriebssystemen erkennt, muss die Fingerprinting-Datenbank aktuell gehalten werden. Dies kann durch updaten der NMAP Version realisiert werden.

- **Banner-Grabbing:** Hierbei wird ein vollständiger Verbindungsaufbau zu offene Ports durchgeführt. In Abhängigkeit von der Konfiguration der Dienste senden diese nach einer aufgebauten Verbindung einen sogenannten Banner, welcher Informationen über den laufenden Dienst und dessen Versionsstand enthält. [20, S. 94] Durch bestimmte Parameter in der Konfiguration können diese Bannerinformationen eingeschränkt bzw. abgeschaltet werden.

  Anhand der Daten des Banners können auch Rückschlüsse auf das Betriebssystem gezogen werden, da der Dienst eventuell nur für ein bestimmtes Betriebssystem verfügbar ist. [20, S. 94]

Um von NMAP eine Betriebssystemerkennung durchzuführen, muss der Parameter -O angegeben werden:

Bei der Versionserkennung kann spezifiziert werden, wie aggressiv die Überprüfung durchgeführt werden soll. Es kann ein Level zwischen 0 und 9 konfiguriert werden, wobei 7 der Standardwert ist. Je höher das Level, umso mehr Informationen werden ermittelt. Die Diensterkennung wird mit dem Parameter „-sV“ angegeben, die Idensität mit „--version-intensity <intensity>“:

4.1.3. Automatisierung

Um eine Reihe von Systemen zu überprüfen und die Ergebnisse maschinell zu verarbeiten, wird eine Python Script erstellt. Dazu ist es mit NMAP möglich, die gesamte Ausgabe in eine XML Datei zu schreiben, welche danach analysiert und verarbeitet werden kann. Um dies zu erreichen, muss der Parameter „-oX <dateiname>“ an dem NMAP Prozess übergeben werden.

In Python kann man Prozesse mit der Bibliothek „subprocess“ starten und verwalten. Um einen NMAP Prozess zu starten, muss folgender Quelltext implementiert werden:

Listing 7: NMAP Prozess mit Python starten

Im Listing 6 werden die folgenden Anweisungen durchgeführt:

- Zeile 1: Die Bibliothek zum Starten und Verwalten von Prozessen wird eingebunden.
- Zeile 3: Ein Prozess wird erstellt und gestartet. Im ersten Parameter wird der Befehl für den Prozessaufruf inklusiv den Prozessparametern angegeben. Der zweite Parameter besagt, dass keine Shell-Features benutzt werden sollen. Dies wird aus Sicherheitsgründen in der Dokumentation empfohlen. Mit dem dritten und vierten Parameter wird signalisiert, dass STDOUT und SDERR Nachrichten nicht abgefangen werden. [24]

  Um Beispielsweise einen NMAP Prozess zu starten, kann der folgende Befehl „nmap –sS 192.168.1.1 –O –sV –oX nmapResult“ übergeben werden. Damit würde man einen TCP Scan mit Betriebssystem- und Diensterkennung durchführen. Die Ergebnisse werden in die Datei nmapResult.xml geschrieben.
- Zeile 4: Es wird gewartet, bis der Prozess beendet ist.

Im nächsten Schritt muss die entstandene Ergebnisdatei eingelesen und analysiert werden. Dazu wird die Bibliothek „minidom“ verwendet. Es kann somit der folgende Code implementiert werden:

Listing 8: NMAP Ergebnis in Python verarbeiten

Im Listing 8 werden die folgenden Anweisungen durchgeführt:

- Zeile 1: Die Bibliothek zum Parsen von XML Dateien wird eingebunden. Da diese im Unterverzeichnis /xm/dom vom Pythonpfad liegt, muss der relative Pfad mit einem Punkt getrennt angegeben werden.
- Zeile 3: Das Ergebnisfile von NMAP wird eingelesen und analysiert. Es entsteht ein neues Objekt, mit dem in den nachfolgenden Schritten auf die XML Daten zugegriffen wird.
- Zeile 5: Alle gescannten Hosts werden in der Schleife durchlaufen.
- Zeile 7: Die IP Adresse des gescannten Zielsystems wird ermittelt.
- Zeile 9: Um eine übersichtliche Ausgabe zu produzieren, wird hier die Adresse des Hosts ausgegeben. Nachfolgend werden dann die weiteren Informationen ermittelt.
- Zeile 11: Die gescannten Ports des aktuellen Hosts werden in einer Schleife durchlaufen.
- Zeile 12 – 15: Es werden Informationen zu den Ports eingelesen. Dabei erhält man die Portnummer, das Protokoll (TCP oder UDP), den Status des Ports (open, closed, filtered) und der Grund für den festgestellten Status (z.b syn-ack).
- Zeile 17 – 20: Es werden Informationen zu den Diensten ermittelt. Dabei kann der Name, das verwendete Produkt sowie die Version eingelesen werden. Teilweise werden auch zusätzliche Informationen wie beispielweise eingesetzte SSL Verschlüsslungen angegeben.
- Zeile 22 – 24: Die Port- und Dienstergebnisse werden in einer formulierten Form ausgegeben.
- Zeile 26: Alle möglichen Betriebssystem Varianten des entfernte Systems werden durchlaufen.

- Zeile 27, 28: Der Name des Betriebssystems sowie die Genauigkeit der Übereinstimmung wird ermittelt.
- Zeile 30: Die Informationen über das Betriebssystem werden in einer formulierten Form ausgegeben.
- Zeile 32, 33: Zwei Leerzeilen werden ausgegeben, damit erkannt wird, dass mit einem anderen Host fortgefahren wird.

Um NMAP Scans nur in der Nacht oder am Wochenende durchzuführen, damit der produktive Geschäftsablauf nicht gestört wird, kann das hier gezeigt Code-Snippet verwendet werden. Dabei können die Zielsysteme in einer Schleife durchlaufen werden um eines nach dem anderen zu überprüfen. Die generierten Ergebnisse können dann von Mitarbeitern in der üblichen Geschäftszeit ausgewertet werden.

4.2. Nessus Scan

Der Vulnerability Scanner Nessus ist eines der beliebtesten Tools für Security-Audits. [25] Er wurde 1998 zum ersten Mal als Open Source Projekt veröffentlicht. Der Scanner konnte bis 2005 unter der GPL (Gerneral Public License) verwendet werden. Danach wurde Nessus als kostenpflichte Version von der Organisation Tenable weiterentwickelt, welche auch Support für das Produkt anbietet. [26, S. 26,27] Eine alternative zu Nessus ist OpenVAS, wobei es sich um ein Open Source Programm handelt, welches auf Nessus aufbaut.

Derzeit kann sowohl eine private als auch eine kommerzielle Version unter http://www.tenable.com erworben werden. Bei der kostenlosen Edition wird kein Support angeboten und die Überprüfung ist auf 16 Adressen beschränkt. Nessus befindet sich derzeit in der Version 5.0, welche am 15. Februar 2012 veröffentlicht wurde. [27]

4.2.1. Aufbau von Nessus

Einer der Erfolgsfaktoren von Nessus ist der gut durchdachte Produktaufbau. Die flexible Architektur von Nessus kann in die folgenden Kategorien unterteilt werden:

- **Client-Server-Prinzip:** Um die Nachteile von einem clientbasierten Scanner entgegen zu wirken, wurde Nessus auf einem Client-Server Prinzip entwickelt. Das bedeutet, es muss auf einen zentralen Computer (Server) der Neuss Dienst (nessusd) installiert werden. Danach kann Nessus von berechtigten Benutzern (Clients) über das Netzwerk mittels einer Webapplikation bedient werden. [26, S. 31] Dadurch sind keine Installationen auf den Clients notwendig und deren Notebooks bzw. PCs können in der Zeit der Überprüfung für andere Aufgaben verwendet werden.

  Die Kommunikation zwischen den Benutzer und den Server Dienst ist standardmäßig mittels TLS Verschlüsslung abgesichert. Für die Bedienung von Nessus muss eine Authentifizierung mittels Passwort oder Zertifikat durchgeführt werden. Dadurch kann man Nessus in eine vorhandene Public Key Infrastructure (PKI) integrieren. [26, S. 33]

  Betreibt man im Netzwerk einen MySQL Server und möchte diesen zum Speichern der Ergebnisse verwenden, so kann dies per Nessus Konfiguration realisiert werden. Dadurch können andere

Anwendungen die von Nessus genierten Daten weiterverarbeiten. [26, S. 33, 34] Standardmäßig müssen die generierten Berichte einer Überprüfung jedoch über die Webapplikation heruntergeladen werden.

- **Plugin:** Nessus stellt Funktionen zum Einbinden von Erweiterungen zu Verfügung. So können Sicherheitsleute eigene Überprüfungen erstellen und diese in der Nessus Umgebung laufen lassen. Dazu müssen sogenannte NASL (Nessus Attack Scripting Language) Skripts erstellt werden. Die Syntax von NASL orientiert sich stark an der Sprache C und ist speziell für Sicherheitsüberprüfungen designed. [26, S. 34] So kann man unternehmensspezifische Gegebenheiten wie Konfigurationen, Anwendungen, usw. durch eigene Skripte überprüfen. Durch die weite Verbreitung von Nessus gibt es bereits viele NASL Skripte zum Herunterladen.

  Um ein Plugin in die Nessus Umgebung hinzuzufügen, muss die entsprechende NASL Datei in das Unterverzeichnis „plugins“, welches im Installationsverzeichnis von Nessus existiert, abgelegt werden. Das Skript wird dann während eines Scanvorgangs ausgeführt. Es besteht auch Möglichkeit die Skripte manuell über die Konsole mittels NSAL Interpreter (nasl) auszuführen.

- **Knowledge Base:** In der Knowledge Base werden alle Ergebnisse von durgeführten Sicherheitstests gesammelt. [28, S. 19] Da Plugins auf diese Datenquelle zugreifen können, müssen bestimmte Informationen nicht wiederholt ermittelt werden. So kann beispielsweise das Wissen über offene Ports eines Hosts, welche von vorherigen Plugin bereits aufgedeckt wurden, von anderen Plugins verwendet werden. So sind die Überprüfungen schneller und mit weniger Rechenaufwand durchführbar.

4.2.2. Automatisierung

Um Nessus Scans über ein eigenes Programm bzw. Skript zu starten und die generierten Reports automatisiert weiterverarbeiteten zu können, kann die XML-RPC (Extensible Markup Language Remote Procedure Call) Schnittstelle von Nessus verwendet werden.

Bei XML-RPC handelt es sich um eine genormte Spezifikationen, um Methoden auf entfernten Systemen aufzurufen. Dabei werden die auszuführenden Methodennamen und Parameter in XML Form mithilfe dem HTTP (Hypertext Transfer Protocol) an den Server übertragen. Durch den Einsatz von standardisierten Komponenten, kann XML-RPC unabhängig von Plattformen und Programmiersprachen implementiert werden. [29, S. 1]

Um die Automatisierung in Python durchzuführen, kann unter http://code.google.com/p/nessusxmlrpc/downloads/list eine vollständige XML-RPC Implementierung für die Nessus Schnittstelle heruntergeladen werden. Diese Bibliothek kümmert sich um die korrekte Zusammenstellung des XML formatierten Inhalts und dessen Übertragung.

Danach kann der folgende Python Code implementiert werden:

Listing 9: Nessus Scans mit Python durchführen

Im Listing 9 werden die folgenden Anweisungen durchgeführt:

- Zeile 1: Die zuvor heruntergeladene Bibliothek von der Goolge Developer Seite wird eingebunden. Diese ermöglicht den Zugriff auf die Nessus Schnittstelle.
- Zeile 2: Die Bibliothek „time“ wird eingebunden. Diese wird verwendet, um den Programmablauf für eine bestimmte Zeit zu pausieren.
- Zeile 4: Es wird eine Verbindung mit dem Nessus Server hergestellt. Dazu muss die Adresse und der Port auf dem der Nessus Dienst horcht angegebene werden. Des Weiteren erwartet die Funktion als dritten und vierten Parameter einen gültigen Benutzernamen sowie dessen Passwort.
- Zeile 5: Es wird eine neue Überprüfung angestoßen. Dabei muss ein noch nicht existierender Scanname angegeben werden. Um nicht einen bereits vorhanden Namen zu verwenden, empfiehlt es sich, einen Timestamp an den Namen anzuhängen. Im zweiten Parameter werden die Adressen der Zielsysteme angegeben, wobei mehrere Adressen durch einen Beistrich getrennt werden. Der letzte Parameter gibt den Namen der zu verwendenden Policy an. Diese muss bereits am Nessus Server existieren.
- Zeile 7: In einer Schleife wird überprüft, ob der Scan abgeschlossen ist.
- Zeile 8: Alle fertigen Reports am Server werden ermittelt.
- Zeile 9: Die Reports werden durchlaufen.
- Zeile 10: Es wird überprüft, ob der Report abgeschlossen ist und ob es sich um den zuvor gestarteten Scan handelt. Wenn nicht, wird eine Sekunde gewartet und mit dem nächsten Bericht fortgesetzt. Das Warten hat den Sinn, dass es zu keiner übermäßig hohen Prozessorauslastung kommt.
- Zeile 11: Der vollständige Report von dem zuvor gestarteten Scan wird herunter geladen und der Inhalt in eine Variable gespeichert.
- Zeile 12 – 14: Der Report wird auf das Filesystem geschrieben. Der XML formatierte Inhalt könnte auch analysiert und beliebig weiterverarbeitet werden, dazu siehe Listing 8.
- Zeile 15: Die Endlosschleife wird beendet

Es ist zu beachten, dass im Listing 9 nur ein Einblick in die Verwendung der XML-RPC Schnittstelle gegeben wurden. Um sich über alle zu Verfügung stehenden Methoden zu informieren, stellt Tenable unter

http://www.nessus.org/documentation/nessus_XMLRPC_protocol_guide.pdf eine detaillierte Dokumentation zum Download bereit.

5. Zusammenfassung und Ausblick

Ziel dieser Arbeit war es, aktive und proaktive Überprüfungen automatisiert durchzuführen. Dabei wurden mehrere Dienste für die proaktiven Scans untersucht und die geeignetsten mit einen Python Script automatisiert. Für die aktiven Überprüfungen wurde Nmap und Nessus als Basiswerkzeug verwendet. Auch hier wurden die Scans automatisiert gestartet und deren Ergebnis verarbeitet. Alle eingesetzten Technologien sind am Markt frei erhältlich und können somit beliebig eingesetzt werden.

Da die Ergebnisse nach den Scans strukturiert in einer SQLite Datenbank vorliegen, wäre es möglich beliebige Auswertungen zu erstellen. Bei einer Weiterentwicklung dieser Arbeit könnten somit automatisierte Berichte aus dem vorhandenen Datenbestand erstellt werden. So wäre am Ende ein einziges gut gegliedertes und formatiertes Dokument vorhanden, in dem alle Schwachstellen aufgelistet sind. Zusätzlich wäre man im Stande grafische Statistiken zu erstellen, um das Ergebnis auch für Personen verständlich zu machen, welche wenig oder keine technischen Kenntnisse besitzen. Dadurch könnte zum Beispiel eine Geschäftsleitung bzw. ein Abteilungsleiter davon überzeugt werden, mehr in die technische Sicherheit des Unternehmens zu investieren.

Mit den Scripts der proaktiven Überprüfungen wäre es Denkbar, den gesamten österreichischen Adressraum abzuscannen. Dieser Vorgang würde zwar einiges an Zeit in Anspruch nehmen, jedoch würden man dadurch sicherlich viele Schwachstellen von Unternehmen, Institute bzw. Organisation aufdecken. Ein solches Service könnte beispielsweise von der österreichischen CERT angeboten werden, um die IT Sicherheit in Österreich zu fördern. Alle öffentlichen IP Bereiche aus Österreich können aus der Datenbank von http://www.maxmind.com/ abgefragt werden.

Literaturverzeichnis

[1] BackTrack Linux. BackTrack 5 R2 Released. [Online]. http://www.backtrack-linux.org/backtrack/backtrack-5-r2-released/(Letzter Zugriff: 1 März 2012)

[2] Steve Holzner, *Eclipse*, 1st ed. Köln, Deutschland: O'Reilly Verlag GmbH & Co. KG, 2004.

[3] Holtkamp Heiko , "Einführung in TCP/IP," Universität Bielefeld, Standards und Protokolle April 2011.

[4] Commer E. Douglas, *TCP/IP-Konzepte, Protokolle und Architekturen*.: Pearson Education, Inc., 2003.

[5] Mark Lutz and David Ascher, *Einführung in Python*, 2nd ed.: O'Reilly Verlag GmbH & Co.KG, 2007.

[6] Mathieu Fourment and Michael Gillings, "A comparison of common programming languages used in bioinformatics," BioMed Central Ltd., Sydney, 2008.

[7] Grant Allen and Mike Owens, *The Definitive Guide to SQLite*, 1st ed. New York: Springer Verlag, Inc., 2010.

[8] Bi Chunyue, "Research and Application of SQLite Embedded Database Technology," *WSEAS TRANSACTIONS on COMPUTERS*, vol. 1, no. 8, pp. 83 - 92, Januar 2009.

[9] SQLite-Team. SQLite. [Online]. http://www.sqlite.org/sqlite.html(Letzter Zugriff: 14 April 2012)

[10] Tomasz Grudziecki, Paweł Jacewicz, Przemysław Jaroszewski, Łukasz Juszczyk, Piotr Kijewski Katarzyna Gorzelak, "Proactive Detection of Network Security Incidents," ENISA, 2011.

[11] Harald Derksen, "Kommunikationsprotokolle - Domain Name System," RWTH Aachen - Lehrstuhl für Informatik IV, Aachen, 2002.

[12] Python Software Foundation. Package Index. [Online]. http://pypi.python.org/pypi/adns-python/1.2.1(Letzter Zugriff: 04 Mai 2012)

[13] RIPE NCC. RIPE Network Coordination Centre. [Online]. https://www.ripe.net/lir-services/ncc/functions(Letzter Zugriff: 23 April 2012)

[14] Google Inc. Google Project Hosting. [Online]. http://code.google.com/p/pybing/(Letzter Zugriff: 29 April 2012)

[15] Shearer Michael, "Shodan for Penetration Testers," DEF CON® Hacking Conference, Mai 2012. [Online]. https://www.defcon.org/images/defcon-18/dc-18-presentations/Schearer/DEFCON-18-Schearer-SHODAN.pdf

[16] VirusTotal. About Virus Total. [Online]. https://www.virustotal.com/about/(Letzter Zugriff: 28 April 2012)

[17] VirusTotal. VirusTotal Public API. [Online]. https://www.virustotal.com/documentation/public-api(Letzter Zugriff: 30 April 2012)

[18] Google Inc. We knew the web was big. [Online]. http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html(Letzter Zugriff: 03 Mai 2012)

[19] Lisa Lancor and Robert Workman, "Using google hacking to enhance defense strategies," *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, vol. 1, no. 39, pp. 491-495, März 2007.

[20] Hans-Joachim Hof, Stephan Krause, Lars Völker, Uwe Walter, and Martina Zitterbart, "Hacking und Hackerabwehr," Universität Karlsruhe (TH), Karsruhe, ISSN 1613-849X, 2007.

[21] Ralf Spenneberg, *Linux-Firewalls - Sicherheit für Linux-Server und -Netzwerke mit IPv4 und IPv6*. München: Addison-Wesley Verlag, 2011.

[22] nmap.org. Port Scanning Techniques. [Online]. http://nmap.org/book/man-port-scanning-techniques.html(Letzter Zugriff: 09 Mai 2012)

[23] Barroso Berrueta David, "A practical approach for defeating Nmap OS−Fingerprinting," 2003.

[24] Python Software Foundation. Subprocess management. [Online]. http://docs.python.org/library/subprocess.html(Letzter Zugriff: 09 Mai 2012)

[25] PC-Welt. Schwachstellen im Netzwerk entdecken mit Nessus. [Online]. http://www.pcwelt.de/news/Netzwerksicherheit-Schwachstellen-im-Netzwerk-entdecken-mit-Nessus-67657.html(Letzter Zugriff: 26 Mai 2012)

[26] Mark Carey, Paul Criscuolo, and Mike Petruzzi, *Nessus Network Auditing*, 2nd ed. Burlington: Syngress Publishing, Inc., 2008.

[27] Tenable Network Security. Nessus for Home. [Online]. http://www.tenable.com/products/nessus/nessus-homefeed(Letzter Zugriff: 15 Mai 2012)

[28] Tasnuva Syeda Ahsan and Rabiul Samina Alam, "Detecting known host security flaws over a network connection," BRAC University, Bachelor of Science in Electronics & Communication Engineering 2009.

[29] Mark Allman, "An Evaluation of XML-RPC," *SIGMETRICS Perform. Eval. Rev.*, vol. 30, no. 4, pp. 2-11, März 2003.

[30] HD Moore, Noam Rathaus Jay Beale, *Nessus - Network Auditing*, 1st ed. Rockland, USA: Syngress Publishing, Inc., 2004.