Städtisches Gymnasium Rheinbach

Entwicklung und Erklärung eines Antiviren-Programms (mit Signaturen) in der Sprache „Python“

FACHARBEIT

Leistungskurs Informatik

Herr Faßbender

Jahrgangsstufe 11

Daniel Zavelberg

Schuljahr 2017/2018

Inhaltsverzeichnis

1 Einleitung

Seit nun ungefähr zwei Jahren beschäftige ich mich mit der Programmiersprache „Python“. Ich habe bereits einen Gaming-Laptop, auf dem ich schon mehrere Skripte erstellt habe und zwei Einplatinenrechner „Raspberry Pi 3 Model B“ für eine eigene Cloud und einen iterativen Web-Crawler. Jedoch ist der Gaming-Laptop eher nicht geeignet für das Schreiben von neuen „Python“-Skripten und die zwei „Raspberry Pi 3 Model B“ nicht sehr benutzerfreundlich durch ihre mangelnde Leistungsstärke. Daher habe ich mir ein zweites Laptop gekauft, das sich eher zum Programmieren eignet. Wie schützt man sein neues Laptop vor Viren? Mit einem Virenscanner, der am besten kostenlos ist und von dem man weiß, dass er zuverlässig den Anwender schützt. Wichtig ist, dass fortlaufend Softwareupdates installiert werden und der Besuch nicht vertrauenswürdiger Webseiten immer unter äußerster Vorsicht erfolgt. Zu diesem Zweck habe ich mir als erstes die Software „Malwarebytes“ heruntergeladen, ein Antiviren-Programm. Auch wenn ich von „Malwarebytes“ überzeugt war, wollte ich noch die Konkurrenz ausprobieren. Somit habe ich mir „Avira Free Antivirus“ gedownloadet, das ähnlich zu „Malwarebytes“ ein kostenloses Antiviren-Programm ist.

Dann hatte ich die Idee, statt kostenloser Virenscanner aus dem Internet zu vertrauen, mit meinen Programmierkenntnissen in „Python“ ein eigenes Antiviren-Programm zu entwickeln. Dieses kann zwar nur einen einfachen und vollständigen Scan mit Signaturen durchführen, jedoch erfüllt es seinen grundsätzlichen Sinn und Zweck.

In meiner Facharbeit werde ich auf die Funktionsweise meines Antiviren-Programms eingehen (Kapitel 2). Im Anschluss stelle ich die zwei Methoden „scan“ und „full_scan“ (2.1) vor. Im Kapitel 2.2 beschreibe ich zwei eigens erstellte Module, die einen PC nach Dateien durchsuchen (2.2.1) und infizierte Programme sicherstellen (2.2.2). In Kapitel 3 werde ich auf zwei Scan-Techniken eingehen, die unter anderem auch von kostenpflichtigen Antiviren-Programmen, wie zum Beispiel „Kaspersky“, verwendet werden. Diese sind das Scannen mit Signaturen (3.1) und Heuristik (3.2) sowie die zwei Arten der Heuristik (3.2.1, 3.2.2). Danach gehe ich auf das Zustandekommen zwei besonderer Fälle bei Antiviren-Programmen ein, false-positive (3.3) und false-negative (3.4). Im Anschluss folgt die Analyse von Schwachstellen und Problemen meines entwickelten Antiviren-Programms (Kapitel 4). Mit einem Fazit und Ausblick (Kapitel 5) zu den behandelten Themen schließt meine Facharbeit.

2 Funktionsweise meines Antiviren-Programms

Die Funktionsweise meines Antiviren-Programms ist relativ einfach zu erklären. Die Software erstellt aus dem Inhalt der zu untersuchenden Datei „D“ einen MD5-Hash, beziehungsweise eine Signatur „S“. „S“ wird mit einer Datenbank von über 30 Millionen Signaturen abgeglichen. Die hierbei genutzten Signaturen stammen von der Webseite www.virusshare.com.[1]

2.1 Erklärung der Methoden „scan“ und „full_scan“

Die Methoden „scan“ und „full_scan“ unterscheiden sich beim genaueren Betrachten nur in einer Funktionsweise. Wie der Name der Methode „full_scan“ bereits andeutet, untersucht diese Methode alle vorhandenen Dateien auf dem PC mit den Rechten, die der Benutzer hat, bei welchem das Skript ausgeführt wird. Den Dateipfad speichert das Programm neben vielen anderen Dateipfaden in einer Liste ab, vom Modul „SystemFileScanner“ (2.2.2) werden die Pfade in eine separate Textdatei übertragen. Diese Textdatei wird in der Methode „full_scan“ aufgerufen und in einer weiteren Liste „L“ verwaltet. Die Aufgabe ist, alle Dateien von einem Programm oder mit einer einzelnen Instanz des Virenscanners zu überprüfen. Aus diesem Grund wurde die Methode auf acht Threads aufgeteilt. Jeder einzelne Thread befasst sich so mit 12,5 % der vorhandenen Dateien, beziehungsweise von „L“. Thread-1 behandelt die Datei(L*0) bis Datei(L*0,125), Thread-2 die Datei(L*0,125+1) bis Datei(L*0,25), Thread-3 die Datei(L*0,25 + 1) bis Datei(L*0,375). Die restlichen fünf Threads verfahren in der gleichen Weise mit den Scans (vgl. Abbildung 1).

Die Methode „scan“ ist die Basis für die Methode „full_scan“ und bietet eine einfache Untersuchung an. „Einfach“ bedeutet die Untersuchung einer einzigen vom Benutzer ausgesuchten Datei. Zunächst wird ein Boolean „M“ mit dem Wahrheitswert „false“ initialisiert und dann nach der Datei „D“ gefragt, die durch die Methode „scan“ untersucht werden soll. Nun wird von „D“ ein MD5-Hash, eine Signatur „S“, erstellt. Anschließend werden die Textdateien, welche die Signaturen von www.virusshare.com beherbergen, einzeln vom Skript gelesen und in eine Liste „L“ übertragen. Wenn „S“ nun in „L“ vorkommt, wird „M“ auf den Wahrheitswert „true“ gesetzt und diese einfache Untersuchung abgeschlossen, sowie eine Bedrohung für den PC signalisiert. Wurde beim Prüflauf eine Bedrohung ausgeschlossen, signalisiert das Antiviren-Programm keine Bedrohung. Die Methoden unterscheiden sich darin, dass bei „full_scan“ nach einem

[1] https://virusshare.com/hashes.4n6

Treffer die vollständige Untersuchung nicht abgebrochen wird, sondern die betroffene Datei mit der Base64-Kodierung verschlüsselt und in einen Quarantäneordner verschoben wird, und die Untersuchung weiterläuft.

2.2 Eigens erstellte Module

Um nicht alles in ein „Python“-Skript zu schreiben, erstellte ich zwei weitere Skripte erstellt, die Module „quarantaene“ und „SystemFileScanner“. Diese Vorgehensweise bietet mehr Übersicht. Module sind nichts anderes als Bibliotheken oder packages, die den Umgang in „Python“ erheblich erleichtern. Beide Module sollten, sowohl Windows-Betriebssysteme als auch Linux-Distributionen, wie zum Beispiel „Ubuntu“, unterstützen.

2.2.1 Das Modul „quarantaene“

Das Modul „quarantaene“ besteht aus der Methode „encode_base64“ und „decode_base64“. Beide Methoden greifen auf einen extra erstellten Quarantäneordner zurück, sowie auf andere Verzeichnisse auf dem PC.

Die Methode „encode_base64“ nimmt zunächst einen String entgegen, der dem Pfad einer in Quarantäne zu behandelnden Datei entspricht. Der Dateiinhalt wird mit dem Base64-Verfahren kodiert. Anschließend wird eine neue Datei im Quarantäneordner mit gleichem Namen der soeben kodierten Datei erstellt. Das alte Verzeichnis wird in der Datei gespeichert und danach der kodierte Inhalt der Datei. Hiernach wird die in der Methode übergebene Datei gelöscht. Für den Fall, dass die Dateien auch wieder benutzbar gemacht werden sollen, erledigt die Methode „decode_base64“ die gleiche Aufgabe wie „encode_base64“, nur spiegelverkehrt. Das heißt, dass eine bestimmte, sich in Quarantäne befindende Datei zunächst aufgerufen und ihr Inhalt eingelesen wird. Man nimmt nun das alte Verzeichnis der in Base64 kodierten Datei, danach den restlichen kodierten Inhalt. Nun wird im alten Verzeichnis eine neue Datei erstellt, in dieser der dekodierte Inhalt gespeichert wird. Zuletzt entfernt man die kodierte Datei aus dem Quarantäneordner, sodass die dekodierte Datei nutzbar ist.

2.2.2 Das Modul „SystemFileScanner“

Das Modul „SystemFileScanner“ besteht, wie das Modul „quarantaene“, aus den zwei Methoden „partitions“ und „indeces“.

Die Methode „partitions“ sucht auf einem Windows-System nach den gängigen Partitionen „A:\“ bis „Z:\“ und ruft dann die Methode „indeces“ auf. Auf einem Linux-

System ruft „partitions“ direkt die Methode „indeces“ auf, da Linux-Distributionen normalerweise keine Festplattenpartitionen besitzen. Die Methode „indeces“ sucht auf einem Windows-System die einzelnen Partitionen auf ihre ganzen Verzeichnisse ab und speichert diese in einer Liste ab. Auf einem Linux-System wird dieser Schritt aufgrund fehlender Partitionen ausgelassen. Anschließend selektiert „indeces“ alle Dateien und speichert diese in einer Textdatei ab. Diese Textdatei ruft mein Antiviren-Programm ab, wenn es einen vollständigen Scan ausführt, und speichert die Datei erneut in einer Liste ab (vgl. Kapitel 2.1).

3 Scantechniken von Antiviren-Programmen

Antiviren-Programme führen heutzutage nicht nur Untersuchungen mit Signaturen durch. Während früher nur mit Signaturen nach bösartigen Dateien gesucht wurde, gibt es mittlerweile neben dieser Technik auch andere effektivere und effizientere Verfahren. Dazu zählt unter anderem Heuristik, welche in zwei Kategorien aufgeteilt werden kann. Im Folgenden werde ich das von mir selbst verwendete Verfahren, das Vergleichen von Signaturen, und die zwei Arten der heuristischen Analyse erläutern.

3.1 Scannen mit Signaturen

Beim Scannen mit Signaturen wird eine Signatur, beziehungsweise ein MD5/SHA1-Hash (in meinem Fall ein MD5-Hash), von einer Datei erstellt und mit einer Datenbank durch weitere Hashs verglichen. Befindet sich in der Datenbank ein Ebenbild vom zu untersuchenden Hash, wurde eine Bedrohung gefunden. Dieses Verfahren zeigt die primäre Vorgehensweise, um Viren aufzuspüren. Moderne Firewalls verwenden zum Beispiel solche Verfahren. Ein Vorteil ist die einfache und weit verbreitete Technik, mit der sehr schnell überprüft werden kann, ob es sich bei einem Programm um eine Bedrohung handelt oder nicht. Das Scannen mit Signaturen bildet deshalb die Basis für jedes Antiviren-Programm und ist die effektivste Methode gegen alte, bereits bekannte Bedrohungen. Der Nachteil dieser älteren Technik ist, dass das Verfahren sehr einfach zu umgehen ist. Der Angreifer braucht seine Schadsoftware nur um einen einzigen Bit zu verändern, um später einen anderen Hash zu erzeugen und den Scan nach Viren zu umgehen. Das Verfahren ist dadurch anfällig für polymorphe[2] und metamorphe[3] Viren.

[2] https://medical-dictionary.thefreedictionary.com/Polymorphic
[3] https://www.cknow.com/cms/vtutor/metamorphic-viruses.html

Polymorph beschreibt einen Zustand in mehreren Formen. Für Computerviren bedeutet dies, dass sie meist von einer Mutation Engine (MtE) verschlüsselt werden, um am Ende schwerer von Antiviren-Programmen entdeckt zu werden. Ein Beispiel dafür ist die Dark Avenger Mutation Engine (DAME). Diese verlinkt einen Virus zu sich. Wenn ein Programm mit diesem jetzt polymorphen Virus infiziert wird, sieht dieser Teil für ein Antiviren-Programm unbrauchbar aus. Der Decryptor (Entschlüssler/MtE) selbst liegt unverschlüsselt im infizierten Programm vor und entschlüsselt den Virus beim Ausführen der Software.

Metamorphe Viren schreiben sich bei jeder Infektion selbst neu, um von Antiviren-Programmen nicht identifiziert zu werden. Polymorphe Viren dürfen nicht mit metamorphen Viren verwechselt werden, da erstere sich selbst verschlüsseln und letztere sich nur neu schreiben. Der einfachste Weg für einen metamorphen Virus ist es, sich durch einen sogenannten NOP-Befehl neu zu schreiben. Dies ist ein Befehl, genauer gesagt eine Prozessoranweisung, die keine andere Funktion hat, als den Befehlszähler zu inkrementieren. Im Grunde ist es die perfekte Anweisung, um die Funktion des Virus nicht zu verändern und dennoch seine Absicht(en) vor einem Antiviren-Programm zu verstecken. Bei metamorphen und polymorphen Viren wird deutlich, dass es sehr viele Möglichkeiten gibt, einen Virus in verschiedenen Formen darzustellen und somit einen spezifischen Hash praktisch unbrauchbar zu machen.

Das US-amerikanische Softwarehaus „Symantec“, Hersteller des Antiviren-Programms „Norton“, behauptete 1992 in einem Bericht, knapp 900.000 verschiedene Mutationen von nur einem Virus mit der Mutation Engine „DAME“ herstellen zu können.[4] Dies verdeutlicht, dass ein Verfahren, wie das Scannen mit Signaturen, nach einer gewissen Zeit sehr leicht zu umgehen ist.

3.2 Heuristik

Heuristik ist laut der Website www.informatik-verstehen.de „die Lehre des Gewinnens neuer Kenntnisse auf methodischem Weg, […] auf der Basis der zugrunde liegenden, komplexen Problemstruktur.“[5]

Für ein Antiviren-Programm bedeutet dies die Suche nach neuen und alten Viren, basierend auf einem menschlich erstellten Regelwerk. Das Antiviren-Programm ist ein virtueller und lernfähiger Malware-Analyst, der sein Regelwerk ausbauen kann,

[4] http://www.textfiles.com/magazines/CRYPT/crptlettr6.vir
[5] http://www.informatik-verstehen.de/lexikon/heuristik

beziehungsweise die Leistungsstärke des Antiviren-Scanners verbessern soll. Dies führt unter anderem zu der Annahme, dass eine solche Scan-Technik langsam ist. Heutzutage verwenden die meisten Antiviren-Scanner statische und dynamische Analyseverfahren zusammen, um neue Schadsoftware aufzuspüren.[6]

3.2.1 Statische Analyse

Bei der statischen, heuristischen Analyse wird eine Datei auf verdächtige Attribute (Prozessoranweisungen) untersucht. Dies geschieht, indem ein Disassembler die binär kodierte Maschinensprache eines ausführbaren Programms in eine für Menschen lesbarere Assemblersprache umwandelt.[7] Wenn das Antiviren-Programm bei seiner Analyse nun auf verdächtige Attribute stößt, wird ein Zähler bei jedem Fund inkrementiert. Ist der Zähler am Ende gleich einer bestimmten Anzahl, so wird das untersuchte Objekt als potentielle Gefahr eingestuft. Nach einem Artikel des Softwareherstellers „Kaspersky Lab“ vom 01.03.2013 stellt sich diese Methode als höchst ineffektiv heraus, da die Erkennungsrate von Schadsoftware im Vergleich zu der false-positive Rate sehr gering ausfällt.[8] Der Grund dafür ist, dass die statische Analyse eine Bandbreite von Dateien mit gleichen Charakteristiken aufweist und demnach eine harmlose Datei als Schadsoftware behandelt werden kann (vgl. Kapitel 3.3).

Ein Beispiel für die statische Analyse ist die Untersuchung des „Bundestrojaners“ durch die Hackervereinigung „Chaos Computer Club“ am 08.10.2011. Die statische Analyse wurde anschließend in einem spezifischen Scanner für den „Bundestrojaner“ übernommen.[9]

3.2.2 Dynamische Analyse

Anstelle eines Disassemblers, der den Assembly-Code analysiert und daraufhin die Entscheidung fällt, ob eine Datei schädlich für den PC ist, wird bei der dynamischen, heuristischen Analyse eine Datei in einem Emulator[10], ein Programm, das ein Betriebssystem nachstellt, getestet. Im Grunde genommen ist der Emulator eine virtuelle Maschine. Anhand der Aktivität nach der Ausführung des Programms in der virtuellen Maschine wird entschieden, ob es sich bei der Software um Schadsoftware handelt oder

---

[6] https://support.kaspersky.com/6324
[7] https://hydrasky.com/malware-analysis/malware-analysis-bypass-static-heuristic-antivirus
[8] Vgl. 6
[9] https://de.scribd.com/document/68043486/CCC-Analyse-einer-Regierungs-Malware
[10] https://de.wikipedia.org/wiki/Emulator

nicht. Ein Vorteil dieser Technik besteht darin, dass selbst polymorphe Viren, die durch eine signaturbasierte Untersuchung nicht entdeckt wurden, leicht aufzuspüren sind, und eine statisch, heuristische Analyse bei einem verschleierten Code nichts bewirkt. Diese Art der Untersuchung ist jedoch sehr ressourcenaufwändig, da man ein Betriebssystem emuliert und mindestens ein weiteres Programm in der virtuellen Umgebung ausführt. Außerdem kann ein dynamischer Scanner, der eine Schadsoftware zum Beispiel um 12 Uhr entdecken soll, so getäuscht werden, dass diese nur um 14 Uhr ausgeführt wird. Ein dynamischer, heuristischer Scanner ist nicht unbedingt in der Lage, eine Schadsoftware, die erst zu einer bestimmten Uhrzeit auf dem Zielrechner ausgeführt wird, zu entdecken. Aus diesen Gründen verwenden Antiviren-Programme diese beiden heuristischen Verfahren meistens gemeinsam.

3.3 False-positive

Wenn das Antiviren-Programm eine Datei als Schadsoftware identifiziert, diese jedoch keine Bedrohung darstellt, handelt es sich um einen false-positive, ein falscher Verdächtiger. Die weit verbreitete Spieleplattform „Steam“ bietet zu diesem Problem eine Informationsseite an. Sie beschreibt, dass einige Spiele von Antiviren-Programmen fälschlicherweise als Virus oder Trojaner eingestuft werden. Die Problembehebung ist in diesem Fall einfach, da nur der Pfad des Spiels anzugeben ist, um dem Antiviren-Programm zu signalisieren, dass es sich bei dem angeblichen Virus um keinen handelt.[11]

3.4 False-negative

Wenn das Antiviren-Programm eine Datei nicht als Bedrohung identifiziert, obwohl sie das System schädigt, spricht man von einem false-negative, ein Verdächtiger, der aufgrund mangelnder Beweise mit seinen Taten davonkommt. In einem Beitrag des YouTube-Kanals „SemperVideo“[12] wird eine Backdoor mit dem Programm „Metasploit“ erstellt. „Metasploit“ ist ein kostenloses Pentest-Werkzeug, das zum Beispiel ermöglicht, ein Netzwerk auf Schwachstellen zu prüfen. In diesem Beitrag wird das Antiviren-Programm „Norton Internet Security“ verwendet, welches diese Backdoor nicht erkennen kann. Eine Backdoor ist zum Beispiel fähig, ein Foto über eine vorhandene Webcam zu machen, sowie ein Video aufzunehmen/zu streamen. Man kann auch einen Keylogger auf dem betroffenen PC starten, der jeden Tastenschlag aufschreibt und einem Angreifer Passwörter übermittelt.

[11] https://support.steampowered.com/kb_article.php?ref=4361-MVDP-3638&l=german
[12] https://www.youtube.com/watch?v=-DyufLiA0yU

4 Schwachstellen meines Antiviren-Programms

Die erste Schwachstelle meines entwickelten Antiviren-Programms (vgl. Kapitel 2) ist die effektive und effiziente Verwaltung der einzelnen Signaturen, welche die Software zum Untersuchen der Dateien verwendet. Bei einem Scan müssen zunächst 30.329.199 Signaturen durchsucht werden (Stand: 09.04.2018).[13] Das momentane Speichern der Signaturen erfolgt in mehreren Textdateien. Zwar kann man so schnell prüfen, welche MD5-Hashdatei von VirusShare nicht vorhanden ist. Das Aufrufen jeder einzelnen Datei ist im Gegensatz zu einer einzigen größeren Datei, die man nur einmal aufruft, um mit ihr weiterzuarbeiten, nicht sehr effizient. Das Abspeichern der Hashdateien ist dennoch notwendig, da die Programmiersprache „Python“ bei meinen Tests nur 1 GB im Cache verwalten konnte. Das Aufrufen der Dateien, die zusammen knapp 950 MB groß sind, würde ansonsten einen MemoryError mit Programmabsturz hervorrufen und somit das Antiviren-Programm unbrauchbar machen. Eine Möglichkeit, das Problem mit einer großen Datei zu umgehen, ist ihr Öffnen mit dem Modus „rb“ in der eingebauten Funktion „open“. „rb“ ruft den Lesemodus einer Datei in Byte-Darstellung auf. Der Vorteil ist, dass anders als beim normalen Lesemodus „r“ nur Zahlen abgespeichert werden, also Bytes. Der Computer kann diese viel schneller berechnen und einfacher in einer Datenstruktur, wie einer Liste oder in einem String aus Bytes, verwalten. Eine Untersuchung eines Programms, die vorher 14 Sekunden gedauert hat, liegt nun meistens bei unter einer Sekunde. Die zweite Schwachstelle ist, dass „Python“ im Allgemeinen langsamer agiert als andere Programmiersprachen, wie zum Beispiel „Java“. Dies hat vor allem den Grund, dass „Python“ eine interpretierte Programmiersprache ist.[14] Das bedeutet, dass Variablen keinen bestimmten Datentypen haben, sondern ihnen direkt ein Datentyp zugewiesen wird, der beim Ausführen eines Skriptes interpretiert wird und dann als solcher behandelt wird. „Python“-Skripte liegen also nicht direkt in Maschinensprache vor. Projekte können jedoch schneller als in anderen Programmiersprachen umgesetzt werden, da „Python“ der gesprochenen Sprache ähnelt. Ein weiteres Problem kann zukünftig durch leichtes Umgehen und Ausnutzen meines Antiviren-Programms entstehen. Dies liegt daran, dass ein „Python“-Skript, wie bereits geschildert, keine direkte ausführbare Datei ist, sondern bei der Ausführung interpretiert wird. Zwar gibt es Open-Source-Programme, wie „py2exe“ oder „PyInstaller“, die eine ausführbare Datei aus einem „Python“-Skript erstellen, diese jedoch nur konsolenbasierte Anwendungen unterstützen.

---

[13] https://virusshare.com
[14] https://www.“Python“.org/doc/essays/blurb

5 Fazit und Ausblick

Mein Antiviren-Programm verfügt über die Funktion, einen MD5-Hash zu erstellen und diesen mit einer Liste von über 30 Millionen weiteren MD5-Hashes zu vergleichen. Diese Funktion wird in der Methode „scan“ für eine einfache Untersuchung und in der Methode „full_scan“ für eine vollständige Untersuchung verwendet. Ziel ist es, dass mein Antiviren-Programm auch über die erläuterte statische, heuristische Analyse verfügt und eine Firewall ergänzt wird. Zudem soll eine Echtzeit-Schutzfunktion implementiert werden, die den Benutzer besser schützt. Bisher können infizierte Dateien nur gelöscht oder wiederhergestellt werden, dabei jedoch nicht desinfiziert werden, wie es einige Antiviren-Programme mittlerweile anbieten. Des Weiteren ist eine Umsetzung in den Programmiersprachen „Java“ und „C++“ angedacht. Mit beiden Programmiersprachen kann dann mein entwickeltes Antiviren-Programm auch auf Windows-Betriebssystemen ausgeführt werden, ohne einen „Python“-Interpreter zu benötigen.

6 Anhang

6.1 Glossar

Assembler-Sprache: (kurz: Assembler) hardwarenahe und für den Menschen verständliche Programmiersprache

Backdoor: (deutsch: „Hintertür“) ermöglicht einen alternativen Zugriff auf den PC

Boolean: Wahrheitswert, der nur den Zustand „true“ oder „false“ annehmen kann

Bundestrojaner: staatliches Überwachungsprogramm, das den Ermittlungsbehörden unbemerkten Zugriff auf einen PC ermöglichen soll

Disassembler: übersetzt Binärcode in Assembler-Sprache

Hash: (deutsch: „Prüfsumme“) kann bei einem Bit ein anderes Ergebnis liefern; MD5-Hash (Message-Digest Algorithm 5): dient zur leichten Überprüfung, ob ein Programm schädlich ist

Mutation Engine: (kurz: MtE) erstellt polymorphe Viren

Skript: Quellcodedatei einer interpretierten Programmiersprache, z.B. „Python“

String: Zeichenkette oder Zeichenfolge

Thread: Prozess, der neben dem Hauptprozess des Programms ausgeführt wird

Trojaner: Programm, das sich als nützliche Anwendung tarnt, jedoch im Hintergrund ohne Wissen des Anwenders andere Funktionen ausführt

Web-Crawler: Programm, das genutzt wird, um Informationen von Webseiten zu sammeln

6.2 Quellenverzeichnis

1 https://virusshare.com

2 https://de.scribd.com/document/68043486/CCC-Analyse-einer-Regierungs-Malware

3 https://support.steampowered.com/kb_article.php?ref=4361-MVDP-3638&l=german

4 https://www.youtube.com/watch?v=-DyufLiA0yU

5 https://www.infosecurity-magazine.com/opinions/malware-detection-signatures

6 https://www.cknow.com/cms/vtutor/metamorphic-viruses.html

7 https://medical-dictionary.thefreedictionary.com/Polymorphic

8 http://virus.wikidot.com/dark-avenger-mutation-engine

9 https://de.wikipedia.org/wiki/Computervirus

10 https://de.wikipedia.org/wiki/Mutations_Engine

11 https://de.wikipedia.org/wiki/Nulloperation

12 http://www.textfiles.com/magazines/CRYPT/crptlettr6.vir

13 http://lexikon.stangl.eu/1963/heuristik/

14 https://support.kaspersky.com/6324

15 https://www.youtube.com/watch?v=d7boMCLmnaA

16 https://de.wikipedia.org/wiki/Emulator

17 http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201701-201712-bar

18 https://pdfs.semanticscholar.org/b421/0671dc58a5cd38f658e14cb13f1fb567f2a5.pdf

19 https://pdfs.semanticscholar.org/459e/9ecd2093f8e971b192155b1c27429da554e8.pdf

20 https://www.“Python“.org/doc/essays/blurb

Letzter Zugriff auf die Internetseiten (1-20) erfolgte am 09.04.2018.

6.3 Abbildungsverzeichnis

Abbildung 1:

```
if part == 1:
    i = int(len(files)*0.125)
    tmp = 0
if part == 2:
    i = int(len(files)*0.25)
    tmp = int(len(files)*0.125)
if part == 3:
    i = int(len(files)*0.375)
    tmp = int(len(files)*0.25)
if part == 4:
    i = int(len(files)*0.5)
    tmp = int(len(files)*0.375)
if part == 5:
    i = int(len(files)*0.625)
    tmp = int(len(files)*0.5)
if part == 6:
    i = int(len(files)*0.75)
    tmp = int(len(files)*0.625)
if part == 7:
    i = int(len(files)*0.875)
    tmp = int(len(files)*0.75)
if part == 8:
    i = int(len(files))
    tmp = int(len(files)*0.875)
```

Abbildung 1 stammt von mir und zeigt einen Ausschnitt meines Antiviren-Programms.

7 Versicherung der selbstständigen Erarbeitung

Ich versichere, dass ich die vorliegende Arbeit einschließlich evtl. beigefügter Zeichnungen, Kartenskizzen, Darstellungen u. ä. m. selbstständig angefertigt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Alle Stellen, die dem Wortlaut oder dem Sinn nach anderen Werken entnommen sind, habe ich in jedem Fall unter genauer Angabe der Quelle deutlich als Entlehnung kenntlich gemacht.

Swisttal-Morenhoven, den 12.04.2018

(Ort) (Datum)

_____

(Unterschrift)