Antivirus Analyse Mechanismen

Thema 20

im Modul 106: IT-Sicherheit und IT-Angriffe

vorgelegt von: Michael Koll

Matrikelnummer:

Erstgutachter:

Inhaltsverzeichnis

Abkürzungsverzeichnis

AV-Scanner Anti-Viren-Scanner.

DDoS Distributed Denial of Service.

IDS Intrusion Detection System.

MAVM Multi-Antivirus-Maschine.

PE Portable Executable Format.

SSDT System Service Dispatch Table.

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

1. Einleitung

Malware ist heutzutage eine der größten Bedrohungen der Integrität und Vertraulichkeit von IT-Systemen. Im Jahre 2017 registrierten Antivirus-Hersteller bis zu 4 Millionen neue Samples pro Tag. Gleichzeitig sind eine Vielzahl von Gegenmaßnahmen zur Erkennung und Analyse von Schadsoftware einfach zu beschaffen und erschweren den Umgang mit neuer oder veränderter Schadsoftware [5].

In dieser Hausarbeit wird eine Malware-Analyse-Sandbox `Cuckoo` vorgestellt, welche eine automatisierte Analyse von potentiell schädlichen Dateien ermöglicht. Die Samples werden in der Sandbox einer statischen und dynamischen Analyse unterzogen und die Ergebnisse im Anschluss mit diversen Auswertungsmechanismen bewertet. Die automatische Analyse bietet eine schnelle und komfortable Möglichkeit einen Überblick über das Verhalten einer Schadsoftware zu bekommen und eine erste Bewertung des Verhaltens vorzunehmen. Sie ersetzt allerdings nicht, gerade bei komplexerer Schadsoftware, die manuelle Auswertung, sondern ist als Einstieg in die Analyse eines Samples zu verstehen.

Nach einer formalen Definition folgt in Kapitel 2.2 eine Darstellung der verschiedenen Malwaretypen und eine kurze Charakterisierung. Der theoretische Teil in Kapitel 3 beschreibt gängige Erkennungsmethoden von Anti-Virus-Scannern und Analysemethoden für Schadsoftware. Es wird eine Kategorisierung der Analysemethoden vorgenommen. Kapitel 3.3 schließt den theoretischen Teil mit der Vorstellung von verschiedenen Anti-Analysemethoden ab.

Der praktische Teil beginnt mit der Installation und Vorbereitung der Cuckoo-Sandbox in Kapitel 4.1. Die Installation wurde mit Hilfe von Ansible vollständig automatisiert. In Kapitel 4.2 folgen abschließend zwei durch die Sandbox erstellte Analysen, für die eine Bewertung hinsichtlich der Gefährlichkeit und Funktionsweise der Samples vorgenommen wird.

2. Hintergrund

In diesem Kapitel wird ein Überblick über Schadsoftware gegeben, dazu wird Schadsoftware definiert und die wichtigsten Arten vorgestellt.

2.1. Definition von Schadsoftware

Als Schadsoftware (oder Malware) wird eine bösartige, schädliche Software bezeichnet, die ein Computersystem ohne Einverständnis des Besitzers infiltriert oder beschädigt. Schadsoftware wird zu diesem Zweck erzeugt und steht damit im Gegensatz zu Software, die z. B. durch einen Fehler ungewollt Schaden anrichtet([18, S.4]). Schadsoftware ist häufig in der Lage sich selbständig weiter zu verbreiten bzwzu reproduzieren, um weitere IT-Systeme zu infizieren.
Der Schaden kann sowohl aus dem Verlust der Vertraulichkeit bei schutzbedürftigen Daten, aus dem Verlust der Integrität bei Informationen und IT-Systemen oder aus dem Verlust der Informationen oder der Beschädigung des IT-Systems bestehen. Eine Kombination dieser Schadensbilder ist ebenfalls möglich ([16]). Schadsoftware nutzt im allgemeinen Architektur-Schwächen von Software aus, um IT-Systeme zu infizieren. Mögliche Schwächen sind die gemeinschaftliche Nutzung von Programmen und Daten, die universelle Interpretierbarkeit von Daten und die Transitivität des Informationsflusses (Vgl. [12, S. 105]).

2.2. Typen von Schadsoftware

Parallel zur rasanten Entwicklung von IT-Systemen hat sich auch die Vielfalt von Schadsoftware entwickelt,angefangen bei Viren und Würmern bis zu heutzutage weit verbreiteter Ransomware. Schadsoftware unterscheidet sich in ihrem technischen Aufbau, ihrer Verschleierungsmöglichkeiten,ihrer Verbreitungsstrategie und dem vorgesehenen Anwendungzweck. Die in diesem Kapitel vorgestellten Typen sind nicht vollzählig, stellen allerdings eine Liste der häufigsten anzutreffenden Schadsoftware dar.

Virus und Wurm Viren und Würmer sind die älteste Variante von Schadsoftware. Viren sind Befehlsketten, die ein Wirtsprogramm zur Ausführung benötigen.

Das Wirtsprogramm muss auf einem System durch den Nutzer ausgeführt werden, um seine Schadenswirkung zu entfalten. Würmer hingegen sind eigenständige, ablauffähige Programme, die sich selbständig, das heißt ohne Interaktion ein Nutzers, über verschiedene Netzwerktechnologien verbreiten (z. B. E-Mail, Peer-to-Peer-Programme, LAN oder WAN). Beide Typen kompromittieren die Vertraulichkeit und Integrität von IT-Systemen.

Trojanisches Pferd Trojanische Pferde (bzw. kurz Trojaner genannt) enthalten verdeckte Eigenschaften um ein System hinsichtlich der Vertraulichkeit, Integrität und Verfügbarkeit zu kompromittieren. Zur Tarnung täuscht ein Trojaner den Nutzern in der Regel sichtbare, korrekte Ergebnisse vor. Es existiert also eine Diskrepanz zwischen Ist- und Soll-Funktionalität. Ein Beispiel ist der sogenannte **Staatstrojaner**[1], der die Möglichkeit bietet Informationen eines IT-Systems aufzuzeichnen und an einen Command-and-Control-Server zu übermitteln. Weiterhin kann über diesen Server zusätzliche Funktionalität nachgeladen werden.

Spyware und Adware Spyware verfolgt das Ziel Informationen eines IT-System auszulesen und diese Informationen zu übermitteln. Häufig wird Spyware durch einen Trojaner, z. B. integriert in herunterladbaren Inhalten, verbreitet. ([18, S. 10]).

Bot Durch Bots infizierte Systeme können durch den Angreifer bzw. einen Command-and-Control-Server ferngesteuert werden. Dies ermöglicht es einem Angreifer die infizierten Systeme für andere Zwecke einzusetzen, z. B. zur Verschleierung der eigenen Identität oder als Teilnehmer eines Distributed Denial of Service (DDoS)-Angriffes. Normalerweise ist ein Bot ein Wurm oder Trojaner, ergänzt durch die Möglichkeit zur Fernsteuerung [8, S. 62].

Ransomware Ransomware verhindert, dass der Nutzer weiterhin Zugriff auf seine Daten bzw. das System hat, entweder durch Sperren möglicher Zugänge zum Betriebssystem, Anwendungen und zum Netzwerk oder durch Verschlüsselung von Daten. Üblicherweise wird einem Opfer die Möglichkeit zur Wiederherstellung des Zugriffs nach einer Lösegeldzahlung versprochen und teilweise auch ermöglicht.

[1] `https://www.ccc.de/de/updates/2011/staatstrojaner`

3. Theoretischer Teil

3.1. Erkennungsverfahren von AV-Scannern

Anti-Viren-Scanner (AV-Scanner) sind die am häufigsten genutzte Anti-Virus-Software gegen Malware. Die ersten Anti-Viren-Scanner wurden in den 1980er Jahren entwickelt. Seit dieser Zeit haben sich die (Verschleierungs-)Methoden von Malware-Autoren stark verändert, weshalb auch die Hersteller von Anti-Viren-Scannern ihre Software an die Gegebenheiten anpassen mussten.
AV-Scanner untersuchen eine Datei vor dem Öffnen bzw. Ausführen auf eine mögliche Infizierung, entweder nach manueller Aufforderung durch den Nutzer (On-Demand-Scaner) oder automatisch (On-Access-Scanner).Sollte eine mögliche Gefährdung festgestellt werden, können AV-Scanner üblicherweise die schädliche Datei isolieren und einen weiteren Zugriff, und damit eine Ausbreitung der Schadsoftware, verhindern. Zur Erkennung von Schadsoftware untersuchen die meisten AV-Scanner eine Datei nach gewissen Mustern ab und vergleichen diese mit einer Datenbank (Signaturbasierte Verfahren,siehe Kapitel 3.1.1). Zusätzlich werden heuristische Verfahren angewandt, um neue oder leicht veränderte Schadsoftware identifizieren zu können (siehe Kapitel 3.1.2). Aufgrund einer Vielzahl möglicher Verschleierungsmaßnahmen (siehe Kapitel 3.3) können die Sicherheitsmaßnahmen um weitere Verfahren wie die verhaltensbasierte Analyse (Sandboxing) oder Erkennung auf Basis von maschinellem Lernen erweitert werden (siehe 3.1.3)
Um die Möglichkeit einer Nicht-Entdeckung zu verringern sollte eine Kombination mehrerer Verfahren angewandt werden,basierend auf den verfügbaren Ressourcen und dem Grad der Sensitivität der zu schützenden Daten. Eine gute Möglichkeit der Kombination unterschiedlicher Scan-Engines ist die Nutzung von Multi-Antivirus-Maschinen (MAVMs), welche eine Vielzahl unterschiedlicher AV-Scanner einbindet[2] [3]. Zu beachten ist, dass auch wenn alle genutzten AV-Scanner keine Schadsoftware auffinden konnten, dies nicht bedeutet, dass keine Schadsoftware auf dem überprüften System vorhanden ist ([9, S. 161]).

3.1.1. Signaturbasierte Verfahren

Signaturbasierte Verfahren sind heutzutage immer noch die Basis von AV-Scannern. Der Grundgedanke von signaturbasierten Verfahren ist es, bestimmte Eigenschaften

[2] https://www.virustotal.com/de/about/credits/
[3] https://www.opswat.com/metadefender-core

einer potentiell schädlichen Datei zu extrahieren und diese mit einer Signaturdatenbank abzugleichen. Signaturen waren anfänglich eine bestimmte Abfolge von Bytes, die eine spezifische Schadsoftware enthielt. Heutzutage enthält die Signatur einer Schadsoftware deutlich mehr Informationen zum Aufbau und Inhalt derselben, um möglichen Verschleierungsmaßnahmen und der steigenden Komplexität von Schadsoftware zu begegnen. Wenn eine neue Schadsoftware bekannt wird, wird diese üblicherweise von Anti-Virus-Herstellern analysiert und eine geeignete Signatur in der Datenbank abgelegt. Damit anschließend ein AV-Scanner nicht jede Datei vollständig abgleichen muss, wird z. B. nur ein Hashwert einer bestimmten Stelle (z. B. die ersten zwei Bytes) mit der Signatur verglichen. Zusammengefasst wird bei signaturbasierten Verfahren der disassemblierte Code analysiert, bestimmte Merkmale extrahiert und diese Merkmale in einer Signaturdatenbank zum späteren Abgleich durch den AV-Scanner gespeichert. Signaturbasierte Verfahren haben den großen Nachteil, dass sie nur bekannte Schadsoftware, also solche, deren Signatur bekannt und in der Datenbank gespeichert ist, erkennen können. Man spricht von einem reaktiven Erkennungsverfahren ([19]).

3.1.2. Heuristische Verfahren

Heuristische Verfahren untersuchen Programme mit Regeln und Algorithmen nach bestimmten Merkmalen und sind daher theoretisch in der Lage neue unbekannte Schadsoftware oder Abwandlungen einer bekannten Schadsoftware anhand ihres Verhaltens zu entdecken. AV-Scanner, die mit Heuristiken arbeiten, werden proaktive AV-Scanner genannt. Heuristische Scan-Engines versuchen die Wahrscheinlichkeit zu beurteilen, ob eine Datei mit einer Schadsoftware infiziert ist oder nicht. Dazu wird das Verhalten eines Programms analysiert und nach Hinweisen auf eine Schadsoftware gesucht. Solche Hinweise können z. B. das Prüfen auf ein bestimmtes Datum, eine unüblich große Datei oder der versuchte Zugriff auf Adressbücher sein. Wenn mehrere solcher Merkmale vorhanden sind, werden diese gewichtet und durch den AV-Scanner eine Wahrscheinlichkeit für eine vorhandene Schadsoftware berechnet. Heuristische Analysen können auch umgekehrt angewandt werden, indem bestimmte Merkmale für eine Schadsoftware(-familie) ausgeschlossen werden können. Dies kann z. B. eine Mindestgröße einer bekannten Schadsoftware sein. Folgende Liste stellt mögliche statische Methoden dar, die in heuristischen Verfahren angewandt werden ([1, S. 125]):

String Scanning-Methode Sucht nach bestimmten Abfolgen von Bytes (oder Strings), die für eine bestimmte Schadsoftware typisch sind

Wildcards-Methode Sucht anhand regulärer Ausdrücke nach Vorkommnissen bestimmter Bytefolgen

Bookmarks-Methode Vergleicht den Abstand zwischen dem Anfang der Datei und einem bestimmten Ausdruck

Skeleton-Methode Anhand der essentiellen Befehle innerhalb einer Datei wird ein Skelett erstellt, welches mit der Datenbank abgeglichen werden kann

Um potentiell schädliches Verhalten zu erkennen, können zwei unterschiedliche Ansätze verfolgt werden: Anomalie- oder Verhaltensbasierte Erkennung analysiert das Systemverhalten im nicht infizierten Zustand und vergleicht dieses mit dem Verhalten einer möglichen Schadsoftware. Anhand dieser Differenz kann der Algorithmus angepasst werden. Spezifikationsbasierte Erkennung basiert nicht auf maschinellem Lernen sondern baut auf einer exakten Beschreibung des Soll-Verhaltens eines Systems auf. Im Gegensatz zur Anomaliebasierten Erkennung weist die Spezifikationsbasierte Erkennung eine deutlich geringere Fehlalarmrate auf([8, S. 64]). Alle heuristischen Verfahren haben gegenüber signaturbasierten Verfahren den Vorteil neue oder abgewandelte Schadsoftware erkennen zu können. Ein weiterer Vorteil ist, dass die Algorithmen durch ein Update des AV-Scanners verändert werden können, ohne dass der Autor einer Schadsoftware Kenntnis über den genauen Ablauf hat. Ein Nachteil heuristischer Verfahren ist der erhöhte Bedarf an Ressourcen, wie z. B. CPU-Rechenzeit und Speicherauslastung.

3.1.3. Weitere Verfahren

Zusätzlich zu den signaturbasierten und heuristischen Scannern ist es möglich eine Schadsoftware in einer Sandbox auszuführen und automatisiert zu analysieren. Durch die Ausführung der Schadsoftware ist es möglich, im Gegensatz zu signaturbasierten und heuristischen Methoden, dynamische Analysemethoden anzuwenden, wie z. B. Debugging,Prozessbeobachtung oder Netzwerk-Analyse.Sandboxen sind nicht Teil klassischer AV-Scanner, können aber zur Erkennung von Schadsoftware in komplexen Sicherheitssystemen wie Intrusion Detection Systeme (IDSs) integriert werden. In Kapitel 4 wird näher auf die Sandbox **Cuckoo** eingegangen und anhand einer praktischen Analyse demonstriert.

MAVMs stellen kein eigenes Verfahren zur Erkennung von Malware dar. Mit MAVMs ist es möglich (automatisiert) eine Vielzahl von AV-Scannern zu benutzen, um durch die Kombination der unterschiedlichen heuristischen Algorithmen die Erkennungsrate zu erhöhen([10]). Beispiele für MAVMs sind VirusTotal[4], welcher mit über 70

[4] `https://www.virustotal.com/`

unterschiedlichen Scan-Engines und AV-Scannern arbeitet oder Metadefender[5] mit über 30 unterschiedlichen Scan-Engines.

3.2. Methoden zur Analyse von Malware

Dieses Kapitel beschreibt unterschiedliche Analysemethoden zur Untersuchung einer Schadsoftware. Ziel einer Analyse ist es das Verhalten einer Schadsoftware zu beschreiben, um

- Signaturen für die signaturbasierte Erkennung zu gewinnen
- Heuristische Erkennungsalgorithmen zu verbessern
- die Auswirkungen einer erfolgreichen Infizierung zu beurteilen
- Maßnahmen zur Entfernung der Schadsoftware zu entwickeln

In den folgenden Kapiteln wird auf den Ablauf einer Malwareanalyse eingegangen. Dabei werden hauptsächlich Methoden dargestellt, die zur Analyse einer Schadsoftware für Windows-Betriebssysteme geeignet sind.

3.2.1. Vorbereitung und Ablauf

Malin, Casey und Aquilina teilen den Erkennungs- und Analyseprozess von Malware in fünf Phasen auf ([9, S. XXV]) und beschreiben diese im Detail:

1. Sichern und Untersuchen von volatilen Daten
2. Untersuchung des Arbeitsspeichers
3. Analyse von persistenten Daten
4. Identifizierung und Profiling einer Schadsoftware
5. Statische und dynamische Analyse der Schadsoftware

Es wird davon ausgegangen, dass mit den Phasen 1-3 bereits eine Schadsoftware erkannt und extrahiert wurde. Ein weiteres methodisches Vorgehen zur Erkennung von Schadsoftware ist in [14]dargestellt. Die in den folgenden Kapiteln vorgestellten Methoden eignen sich zur Durchführung der Phasen 4 und 5, um eine erkannte Schadsoftware zu untersuchen und Aussagen über ihr Verhalten zu ermöglichen.

[5] `https://metadefender.opswat.com/`

Zur Untersuchung einer erkannten Schadsoftware ist ein methodisches Vorgehen zu empfehlen, um alle relevanten Gesichtspunkte zu erfassen und die Ergebnisse reproduzierbar dokumentieren zu können.Auch wenn jede Schadsoftware ein unterschiedliches Verhalten aufweist und damit unterschiedliche Methoden oder ein anderer Ablauf zur Analyse notwendig ist, kann ein Vorgehen nach folgendem Schema empfohlen werden ([21],[12, S. 94ff]):

1. Automatische Analyse der Malware in einer Sandbox
2. Laborumgebung zur Ausführung der Malware bereitstellen
3. Statische Eigenschaften untersuchen
4. Verhaltensanalyse durchführen
5. Statische Codeanalyse
6. Dynamische Codeanalyse
7. Falls notwendig entpacken
8. Arbeitsspeicheranalyse durchführen
9. Wiederholen der Schritte 4-8 mit angepassten Konfigurationen
10. Dokumentieren und archivieren der Analysergebnisse

Zur Vorbereitung einer Analyse in einer Sandbox oder Laborumgebung sollten einige Konfigurationen zur Absicherung vorgenommen werden. Dies kann z. B. das Isolieren der Laborumgebung oder das Anonymisieren des Netzwerkverkehrs sein[6]. Weiterhin sollte für eine Laborumgebung sichergestellt sein, dass es jederzeit möglich ist einen bestimmten Zustand zu sichern oder wiederherzustellen (Snapshots).

Die einzelnen Schritte sind voneinander abhängig und gefundene Ergebnisse beeinflussen die Reihenfolge der Methoden und das detaillierte Vorgehen. Die Analysemethoden lassen sich anhand ihrer Eigenschaften in drei Kategorien aufteilen. Schritte 3 und 5 benötigen statische Analysemethoden, die Malware wird also nicht ausgeführt (siehe Kapitel 3.2.2). Die Untersuchung erfolgt durch eine Analyse der verdächtigen Datei hinsichtlich ihrer Eigenschaften, verlinkter Bibliotheken und Metadaten. Für Schritte 4 und 6 werden dynamische Analysemethoden verwendet (siehe Kapitel 3.2.3). Dabei wird die Schadsoftware ausgeführt und währenddessen die Interaktionen mit dem System beobachtet. Die Ergebnisse der statischen Analyseschritte fließen direkt in die dynamische Analyse ein und können Hinweise auf besonders

[6] `https://zeltser.com/build-malware-analysis-toolkit/`

zu beobachtende Ereignisse geben.Schritt 1, die Ausführung der Malware in einer Sandbox, nutzt sowohl statische, als auch dynamische Analysemethoden und dient der ersten Erkenntnisgewinnung. Eine nähere Beschreibung zu den angewandten Methoden in einer Sandbox ist in Kapitel 4 zu finden.
Eine besondere Stellung nimmt die Arbeitsspeicheranalyse ein, da zur Aufzeichnung der Arbeitsspeicherzustände die Ausführung der Malware notwendig ist, die Analyse aber statisch anhand der aufgezeichneten Daten erfolgt. Die Arbeitsspeicheranalyse wird als eigenständige dritte Kategorie eingeordnet.

3.2.2. Statische Analysemethoden

In diesem Kapitel werden verschiedene statische Analysemethoden kurz vorgestellt.

AV-Scanner und MAVM Eine potentielle Schadsoftware mit AV-Scanners zu untersuchen kann erste Erkenntnisse zur Identifizierung und/oder Klassifizierung der Malware liefern.

Hashing Beim Hashing wird ein eindeutiger Fingerabdruck der Datei erstellt, welcher zur Identifizierung und Suche nach vorhandenen Analysen dieser Datei genutzt werden kann ([17, S. 43])

Suche nach Zeichenfolgen (String detection) Es wird innerhalb der Datei nach bestimmten Zeichenfolgen gesucht, welche z. B. Dateinamen, URLs, IP-Adressen, Registry-Schlüssel oder Funktionsnamen der Windows-API sein können ([9, S. 255ff.]). Je nach gefundenen Ergebnissen kann ein Rückschluss auf die Funktionalität der Malware geführt werden.

Dynamische oder statische Bibliotheken Die Analyse der verknüpften Bibliotheken kann einen guten Hinweis auf die Funktionalität der Malware geben. Weiterhin können dadurch wertvolle Hinweise für die dynamische Analyse gewonnen werden ([9, S. 259ff.])

PE-Header analysieren Durch eine Analyse der **PE**-Header können Informationen über die Art der Anwendung, Segmentangaben zum Speicherbild und Arbeitsspeicherbild sowie Import- und Exportverzeichnisse gefunden werden. Die Ergebnisse können Schadsoftware-Indikatoren offenlegen ([12, S. 92-93])

Untersuchung auf gepackte Schadsoftware Als Verschleierungsmaßnahme kann Schadsoftware gepackt oder verschlüsselt werden. Durch Untersuchung der Entropie einer Datei können diese Maßnahmen aufgedeckt werden. Näheres dazu in Kapitel 3.3.

Statische Codeanalyse Das Programm wird disassembliert und kann anschließend in seinem Ablauf nachvollzogen werden. Üblicherweise wird nach verdächtigen Befehlen oder Zeichenfolgen gesucht. Die statische Codeanalyse benötigt einen erheblichen Zeitaufwand und eine hohe Expertise. Andererseits lässt sich, sofern der Angreifer keine komplexen Gegenmaßnahmen getroffen hat, die Funktionalität der Schadsoftware dadurch detailliert beschreiben.

3.2.3. Dynamische Analysemethoden

In diesem Kapitel werden verschiedene dynamische Analysemethoden kurz vorgestellt. Die ersten drei Schritte sind Teil der Verhaltensanalyse.

System- und Prozessanalyse Mit geeigneten Tools werden Prozessaktivitäten, Windows-API-Aufrufe und Veränderungen am Dateisystem aufgezeichnet und analysiert. Dadurch kann eine Aussage über die Interaktion der Malware mit dem System und anderen Artefakten getroffen werden.

Registryanalyse Untersuchung von Zugriffen und Änderungen von Registryeinträgen

Analyse des Netzwerkverkehrs Analyse des Netzwerkverkehrs und angeforderter Netzwerkressourcen. In einem weiteren Schritt kann der Zugriff auf die geforderten Ressourcen gewährt oder simuliert werden, um eventuell ein verändertes Verhalten der Schadsoftware zu erzwingen.

Dynamische Codeanalyse (Debugging) Die dynamische Codeanalyse dient der Untersuchung komplizierter Befehlsstrukturen und -abläufe, die in der statischen Codeanalyse nicht nachvollzogen werden können bzwnur während der Ausführung vorhanden sind (z. B. dynamisch generierte Daten). Mittels geeigneter Haltepunkte und eventueller Manipulation von Daten während der Laufzeit kann die Funktionsweise einer Schadsoftware vollständig nachvollzogen werden. Diese Methode benötigt ein hohes Maß an Expertise und einen erheblichen Zeitaufwand.

3.3. Verschleierungsmaßnahmen

Schadsoftware wird üblicherweise mit einem bestimmten Ziel erstellt, also z. B. privilegierten Zugriff ermöglichen, Nutzung eines infizierten Hosts für Flooding-Attacken, sensible Daten abgreifen und vieles mehr. Die Grundfunktionalität von Schadsoftware wird aber häufig durch eine Vielzahl an Funktionen und Maßnahmen erweitert, um

die Erkennung und Analyse durch AV-Scanner oder Malware-Analysten zu erschweren. Dabei können grundsätzlich zwei Kategorien von Verschleierungsmaßnahmen unterschieden werden:Anti-Erkennungsmaßnahmen und Anti-Analysemaßnahmen. Die Abgrenzung der beiden Kategorien ist allerdings nicht zwingend, da viele Maßnahmen für beide Kategorien angewandt werden können.

Anti-Erkennungsmaßnahmen zielen darauf ab die Erkennung durch Endnutzer und AV-Scanner zu erschweren. Dabei wird sowohl die Ausführung der Malware verschleiert, als auch das äußere Erscheinungsbild, also die Signatur, verändert. Gängige Methoden sind

Verborgene Ausführung (Covert Launching Techniques) Launcher enthalten, häufig in komprimierter oder verschlüsselter Form, die eigentliche Schadsoftware im Ressourcebereich. Der Launcher an sich sieht für AV-Scanner unverdächtig aus, da der eigentliche Schadcode hierdurch verschleiert wird. Um den Schadcode auszuführen werden Techniken wie Process Injection (Injizierung des Schadcodes in einen anderen, undschädlichen Prozess), Process Replacement (Überschreiben eines unschädlichen Prozesses im Arbeitsspeicher) oder API Hooks (Umleiten der systeminternen Nachrichten auf die schädliche DLL) verwendet (vgl. [17, S. 253ff.]). Ein Beispiel für einen Launcher wird in Kapitel 4.2.2 gegeben.

Codieren Das Codieren des Codes führt dazu, dass statische Analysen wie z. B. die Suche nach Zeichenketten ohne Kenntnis des verwendeten Codierungsalgorithmus erschwert werden. Häufig angewandt wird die XOR-Codierung, wodurch jedes Zeichen mit einem festen Byte-Wert verknüpft wird. Das Ergebnis sind häufig nicht druckbare (und lesbare) ASCII-Werte, welche die Analyse des betroffenen Codes erschweren.

Packen und Verschlüsseln Durch Packen einer Anwendung werden die verwendeten Importe verschleiert und sind von außen nicht mehr nachvollziehbar. Ein AV-Scanner oder Malwareanalyst sieht nur die gepackte Anwendung. Es gibt eine Vielzahl von Packern (z. B. UPX, PECompact, ASPack oder Armadillo). Jeder Packer nutzt unterschiedliche Strategien zum Packen der Anwendung, daher ist es für eine Analysesoftware oder Analysten notwendig herauszufinden, welcher Packer genutzt wurde.Zusätzlich zum Packen kann die Schadsoftware verschlüsselt werden, was die Rekonstruktion einer Anwendung weiter erschwert. Ob ein Packer oder eine Verschlüsselung genutzt wurde kann häufig, aber nicht immer, mit Hilfe der Entropie der Schadsoftware ermittelt werden (siehe [17, S. 387])

Olygomorphie, Polymorphe und Metamorphose Malware Aufgrund der Tatsache, dass statische Pack- und Verschlüsselungstechniken eindeutige Signaturen hinterlassen (in Form der Entschlüsselungsanwendung)wurden verschiedenste Techniken entwickelt,mit denen Schadsoftware die eigene Signatur mehrfach oder sogar bei jeder Infektion verändern kann (siehe [20]). Eines der ersten Beispiele für einen metamorphen Wurm ist der Storm-Wurm aus dem Jahre 2007, der ständig sowohl den Payload, als auch den Verbreitungsweg ändert (siehe [15]).

Weitere Maßnahmen dienen explizit dazu die Analyse einer Schadsoftware durch Analysten zu erschweren:

Anti-Disassembly Anti-Disassembly-Maßnahmen erschweren oder verlangsamen sowohl die manuelle Analyse durch einen Analysten, als auch die automatische Erkennung durch heuristische Scanner.Gängige Maßnahmen sind Sprungbefehle mit demselben Ziel, Sprungbefehle mit einer konstanten Bedingung oder die Nutzung von Funktionszeigern. Alle Maßnahmen haben zum Ziel die Analyse durch IDAs zu erschweren (siehe [11] und [17, S. 328ff.]). Ein Beispiel für Spaghetti-Code, also das exzessive Verwenden von (unnötigen) Sprüngen wurde bei der Analyse eines FinFisher-Samples durch Microsoft entdeckt (siehe [2]).

Anti-Debugging Um das Debuggen einer Malware zu erschweren existieren eine Vielzahl an Möglichkeiten (siehe [6]), die je nach Betriebssystem unterschiedlich sind. Die Erkennung von lokalen oder Remote-Debuggern ist durch das Vorhandensein bestimmter Prozesse, Dateien oder injizierten Modulen in den eigenen Prozess erkennbar. Im Beispiel von FinFisher wird als erstes versucht jegliche Debugger zu beenden,anschließend wird geprüft ob ein weiterer Debugger aktiv ist. Als dritte Maßnahme versucht FinFisher die Möglichkeit zum Anlegen von Haltepunkten zu unterbinden. Eine weitere von FinFisher verwendete Anti-Debugging-Maßnahme ist die Nutzung mehrerer, eigens für diesen Zweck angepasster virtueller Maschinen. Jede dieser virtuellen Maschinen nutzt einen eigenen Befehlssatz,was sowohl die Analyse durch eine IDA, als auch die dynamische Analyse durch einen Debugger nahezu unmöglich macht ([2]).

Anti-VM Eine weitere Kategorie von Verschleierungsmaßnahmen ist die Erkennung von virtuellen Maschinen. Falls die Malware erfolgreich erkennt, dass sie in einer virtuellen Maschine ausgeführt wird, kann das Verhalten verändert oder die Ausführung komplett verhindert werden. Virtuelle Maschinen können anhand von Artefakten (z. B. Existenz bestimmter Registry-Schlüssel,Dateien

oder Arbeitsspeicherinhalte) oder ihrem Verhalten (Stichwort No Pill und Red Pill, siehe [17, S. 374f.]) erkannt werden.

Um einen Eindruck zu bekommen, welche Möglichkeiten Malware nutzen kann um zu erkennen, dass sie in einer Analyseumgebung ausgeführt wird, wurden die Cuckoo-Analysemaschinen mittels Pafish[7] getestet. Pafish versucht eine Vielzahlmöglicher Anti-VM-Maßnahmen zu prüfen und gibt gefundene Artefakte in einem Report aus. Ein Auszug des Reports ist in Abbildung 3.1 zu sehen. Listing 3.1 zeigt das vollständige Protokoll der gefundenen Artefakte. Auch wenn einige Artefakte erkannt wurden, soll diese Konfiguration für die Analysen im Rahmen dieser Hausarbeit ausreichen. Mit weiteren manuellen Anpassungen wäre es möglich die Erkennungswahrscheinlichkeit weiter zu verringen (siehe [4]).

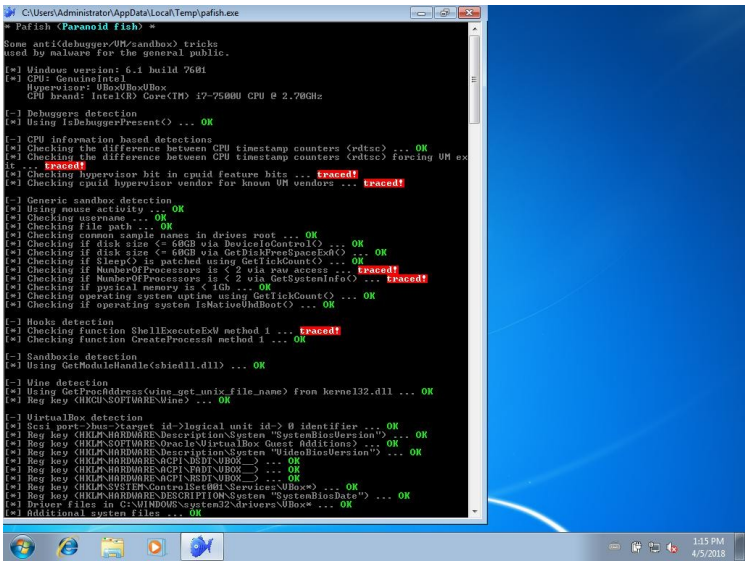

Abbildung 3.1.: Pafish win7x64-01

Listing 3.1: pafish.log

```
1 [pafish] Start
2 [pafish] Windows version: 6.1 build 7601
3 [pafish] CPU: GenuineIntel (HV: VBoxVBoxVBox) Intel(R) Core(TM) i7-7500U CPU @ ←-
,→ 2.70GHz
4 [pafish] CPU VM traced by checking the difference between CPU timestamp counters ←-
,→ (rdtsc) forcing VM exit
5 [pafish] CPU VM traced by checking hypervisor bit in cpuid feature bits
6 [pafish] CPU VM traced by checking cpuid hypervisor vendor for known VM vendors
7 [pafish] Sandbox traced by checking if NumberOfProcessors is less than 2 via raw ←-
,→ access
8 [pafish] Sandbox traced by checking if NumberOfProcessors is less than 2 via ←-
,→ GetSystemInfo()
9 [pafish] Hooks traced using ShellExecuteExW method 1
```

[7] https://github.com/a0rtega/pafish

```
[pafish] VirtualBox device identifiers traced using WMI
[pafish] End
```

4. Praktischer Teil

4.1. Malwareanalyse mit Cuckoo

Cuckoo[8] ist eine Sandbox, die es erlaubt schädliche Dateien in einer isolierten Umgebung zu analysieren. Cuckoo benötigt eine oder mehrere virtuelle Maschinen,in denen statische und dynamische Analysen durchgeführt werden können. Die Dateien werden über einen pythonbasierten Agenten an die virtuelle Maschine übermittelt. Mittels Cuckoo ist es möglich

- ausführbare Dateien, Office-Dokumente, PDF-Dateien und E-mails in Windows-, Linux-, Mac OS- oder Android-Umgebungen zu analysieren
- eine statische Analyse (PE-Header, Strings, Signaturen) der Schadsoftware durchzuführen
- eine Verhaltensanalyse (API-Aufrufe, Registryzugriffe) der Schadsoftware durchzuführen
- mit verschiedenen Strategien (Drop, INetSim, TOR) Netzwerkverkehr zu analysieren
- eine Arbeitsspeicheranalyse mit Volatility durchzuführen

Zur Unterstützung bietet Cuckoo mehrere Apps, die das Handling der Sandbox erleichtern sollen. Das Cuckoo-Webinterface bietet eine komfortable Möglichkeit Schadsoftware mit den gewünschten Parametern an die Analysemaschinen zu übermitteln. Der Cuckoo-Rooter vereinfacht das Handling der iptables-Regeln und konfiguriert das Routing angepasst für jede Analyse. Nach dem Abschluss einer Analyse können die Ergebnisse in dem Webinterface eingesehen werden.

Für diese Hausarbeit wurde die Installation und Konfiguration einer Cuckoo-Umgebung mit Hilfe von Ansible[9] automatisiert. In den folgenden Kapiteln werden die zur Installation notwendigen Schritte beschrieben.

[8] `https://cuckoosandbox.org/`
[9] `https://www.ansible.com/`

4.1.1. Vorbereitung und Installation

Die Installation wurde nach den Vorgaben der Entwickler automatisiert und in ein Ansible-Playbook implementiert (siehe [7] oder Anhang C). Die Automatisierung ermöglicht die einfache Installation und Einrichtung von Cuckoo für zukünftige Verwendungen. Nach einem Durchlauf des Ansible-Playbooks ist der Host für die Verwendung von Cuckoo vorbereitet, inklusive der Installation von VirtualBox, Anpassung aller Konfigurationen für Routingoptionen und der Installation optionaler Analysefeatures. Alternativ kann die Installation manuell nach den Vorgaben der Cuckoo-Entwickler durchgeführt werden[10].

Vorbereitung des Hosts

Für die automatische Installation wird ein Host mit Ubuntu 16.04 LTS benötigt. Die Installation von Ansible erfolgt mit

Listing 4.1: Ansible installieren

```
$ sudo apt-get update
$ sudo apt-get install software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get update
$ sudo apt-get install ansible
```

Anschließend wird das Ansible-Playbook mit folgendem Befehl aus dem Root-Ordner des Playbooks gestartet und der Host für die Verwendung von Cuckoo vorbereitet. Standardmäßig wird Cuckoo mit den Konfigurationswerten aus der Datei `inventory/host_vars/cuckoo_host` konfiguriert. Falls veränderte Konfigurationen benötigt werden, können diese hier angepasst werden.

Listing 4.2: Cuckoo installieren

```
$ ansible-playbook -i inventory/hosts site.yml -kK
```

Sofern keine Änderungen an der Konfiguration vorgenommen wurden,ist Cuckoo anschließend einsatzbereit. Als Analysemaschinen wurden eine Windows-7-VM und eine Windows-XP-VM in der Konfiguration hinterlegt.Die Nutzung von tcpdump (Netzwerkverkehr aufzeichnen), Volatility (Arbeitsspeicheranalyse) und Yara (Signaturerkennung) ist vorbereitet. Als Routingoptionen sind standardmäßig Drop (jeglichen Netzwerkverkehr unterbinden),INetSim (Simulation von Internetdiensten) und TOR (Routing übe das TOR-Netzwerk) konfiguriert und können jeweils für eine Analyse aktiviert werden. Als Standardinterface wird für die virtuellen Maschinen `vboxnet0` im IP-Bereich `192.168.56.1/24` konfiguriert.

[10] `https://cuckoo.sh/docs/installation/host/index.html`

Um den INetSim-Dienst zu nutzen wurde eine virtuelle Maschine auf Basis von Remnux[11] installiert. Remnux bietet neben INetSim weitere Dienste, wie z. B. FakeDns an, welche für die Malwareanalyse genutzt werden können. Die Dienste der Remnux-VM wurden unter der IP `192.168.56.2` konfiguriert. Die Remnux-VM muss manuell installiert werden, dies wird nicht durch das Ansible-Playbook erledigt.

Vorbereitung der Gastmaschinen

Für die in dieser Hausarbeit durchgeführten Analysen wurden zwei virtuelle Maschinen mit dem Tool `VMCloak`[12] verwendet. VMCloak wurde von den Cuckoo-Entwicklern zur Automatisierung der Erstellung von Cuckoo-Analysemaschinen erstellt. VMCloak automatisiert den Prozess der Installation von Windows-VMs und deren Vorbereitung für die Nutzung von Cuckoo. Weiterhin werden diverse Anti-VM-Detection-Maßnahmen implementiert.

Im folgenden wird die Installation der virtuellen Maschinen beschrieben, wie sie für diese Hausarbeit verwendet wurden.Sofern unverändert wurde VMCloak während der Ausführung des Cuckoo-Ansible-Playbooks in einer `virtuelenv`-Umgebung im Cuckoo-Working-Directory installiert. Die Befehle müssen bei aktivierter `virtuelnv`-Umgebung ausgeführt werden. Die durch VMCloak erstellten Dateien werden im Homeverzeichnis des aktiven Benutzers unter `/.vmcloak/` angelegt.

Zur Vorbereitung der Installation muss ein Hardwareprofil des Hosts angelegt und die Windows-ISO-Dateien gemountet werden.

Listing 4.3: Hardwareprofil anlegen und ISOs mounten

```
$ su cuckoo
$ source ~/cuckoo_cwd/venv/bin/activate
(venv)$ vmcloak
(venv)$ sudo vmcloak-gethwconf tuxedo
(venv)$ sudo mkdir -p /mnt/win7
(venv)$ sudo mkdir -p /mnt/winxp
(venv)$ sudo mount -o loop,ro /datadisk/ISOs/win7x64pro/win7x64proEnglish.ISO ←-
,→ /mnt/win7/
(venv)$ sudo mount -o loop,ro /datadisk/ISOs/winxpx32pro/winxpProSP3En.iso ←-
,→ /mnt/winxp/
```

Im Anschluss werden mittels VMCloak die virtuellen Maschinen erstellt. Dazu wird von VMCloak eine automatische Installation von Windows vorgenommen und grundlegende Konfigurationen für Cuckoo durchgeführt (Deaktivieren der UAC, Installation des Cuckoo-Agents, Deaktivieren von Sicherheitseinstellungen). Das in diesem

[11] `https://remnux.org/`
[12] `https://github.com/jbremer/vmcloak`

Schritt angelegte Image dient als Basis für weitere Analysemaschinen, dazu wird im Anschluss an die Initialisierung ein Klon-Image angelegt und die weiteren Schritte an diesem durchgeführt.

Listing 4.4: Analysemaschinen erstellen

```
(venv)$ vmcloak init win7x64proClean --win7x64 --vm virtualbox \
    --iso-mount /mnt/win7 --adapter vboxnet0 --ip 192.168.56.101  \
    --netmask 255.255.255.0 --gateway 192.168.56.1 \
(venv)$ vmcloak init winXpSP3Clean --winxp --vm virtualbox \
    --iso-mount /mnt/winxp --adapter vboxnet0 --ip 192.168.56.101 \
    --netmask 255.255.255.0 --gateway 192.168.56.1 --serial-key XXX-XXX
(venv)$ vmcloak clone win7x64proClean win7x64pro
(venv)$ vmcloak clone winXpSP3Clean winXpSP3
```

Auf den nun erstellten virtuellen Maschinen werden einige Programme installiert, die für die Ausführung und Analyse von Malware benötigt werden (können). Die im folgenden Listing durchgeführte Konfiguration wurde in den Analysen dieser Hausarbeit verwendet.

Listing 4.5: Software installieren

```
(venv)$ vmcloak install win7x64Pro adobepdf dotnet:4.0 dotnet:4.5 \
    flash ie9 java pillow vcredist removetooltips
(venv)$ vmcloak install vmName extract extract.iso=office2013pro.ISO \
      extract.dir=iso
(venv)$ vmcloak install winXpSP3 adobepdf flash java pillow \
    removetooltips
```

Im Anschluss wurde in der Windows 7-Maschine manuell Office installiert (Setupdateien durch `vmcloak install <vmName> extract` in der virtuellen Maschine hinterlegt). Manuelle Anpassungen können in den VMs durch`vmcloak modify <vmName>` durchgeführt werden. Dabei wurde als zweiter DNS-Server in beiden Maschinen die IP der INetSim-VM hinterlegt.

Mit dem Befehl `vmcloak snapshot <vmName>` können nun Snapshots der konfigurierten Images angelegt werden. Dadurch werden die Maschinen in VirtualBox registriert, eine feste IP zugewiesen und ein Snapshot `vmcloak` angelegt. Die Maschinen sind nun für den Einsatz von Cuckoo vorbereitet.

Listing 4.6: Snapshots erstellen

```
(venv)$ vmcloak snapshot win7x64Pro win7x64-01 192.168.56.101
(venv)$ vmcloak snapshot winXpSP3 winXpSP3-01 192.168.56.110
```

4.1.2. Aufbau

Der endgültige Aufbau der Netzwerkstruktur und der virtuellen Maschinen ist in Abbildung 4.1 dargestellt. Der Host dient in Kombination mit der Cuckoo-App `rooter` (analysebasierte Konfiguration der iptables-Routen) als Router für den Netzwerkverkehr. Eine gesamte Übersicht der verwendeten Software und Betriebssysteme und deren Versionen ist in Anhang B aufgelistet.

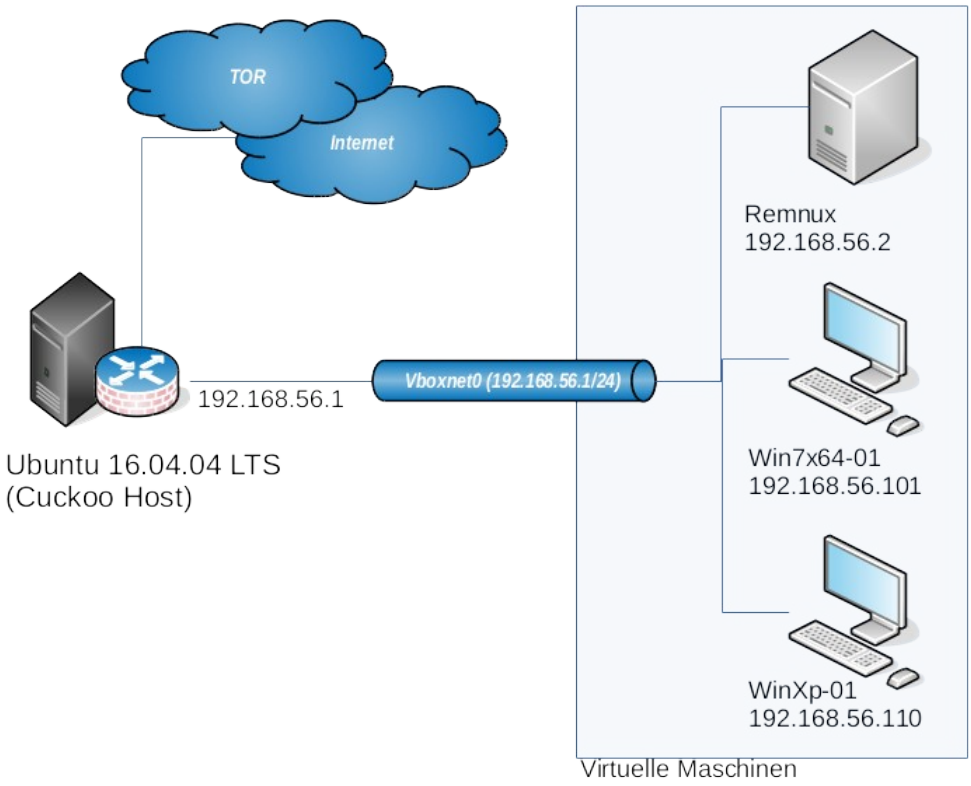

Abbildung 4.1.: Aufbau der Infrastruktur

Die Cuckoo-Dienste können über die folgenden Befehle gestartet werden:

Listing 4.7: Cuckoo starten

```
1 # Cuckoo Rooter starten
2 ~/cuckoo_cwd$ cuckoo rooter --sudo
3 # Cuckoo starten
```

```
~/cuckoo_cwd$ cuckoo
# Cuckoo Webinterface starten
~/cuckoo_cwd$ cuckoo web
```

Das Cuckoo-Webinterface ist in einem beliebigen Browser unter `http://127.0.0.1:8000` erreichbar.

4.2. Analysen

In den folgenden Kapiteln werden einige Analysen potentieller Schadsoftware mit der Cuckoo-Sandbox durchgeführt und ausgewertet.Nach einer kurzen Zusammenfassung werden auffällige Ergebnisse der statischen Analyse, der Verhaltensanalyse und der Netzwerkanalyse beschrieben. Weiterhin werden aus der analysierten Datei extrahierte Dateien aufgeführt und sonstige Auffälligkeiten genannt. Zum Schluss folgt eine Zusammenfassung der ermittelten Daten und anhand dessen ein Rückschluss auf die Funktion und die Auswirkungen des Samples. Die vollständigen Ergebnisse sind im Anhang dieser Hausarbeit beigelegt. Alle Samples wurden passwortgeschützt verpackt, das Passwort lautet`infected`.

4.2.1. Analyse 1

Dieses Sample stammt aus dem Lehrmaterial zum Buch "Practical Malware Analysis"von Michael Sikorski (siehe [17]). Das Sample wurde gewählt um das bekannte Verhalten mit Cuckoo zu analysieren und mit der Auswertung zu experimentieren. Das Programm setzt einen Timer auf den 01.01.2100. Sobald dieser Zeitpunkt erreicht ist, werden HTTP-Anfragen an eine definierte Adresse geschickt.

Übersicht

Dateiname	Lab07_01.exe
Typ	PE32 executable (console)
MD5 Hash	c04fd8d9198095192e7d55345966da2e
Quelle	`https://github.com/mikesiko/PracticalMalwareAnalysis-Labs` (Lab-07-01)
Analyseergebnisse	Anhang D und `../Analysen/1/`
Cuckoo-Node	win7x64-01
Routing	TOR
Zusätzliche Parameter	- keine -

Cuckoo-Score	3.2/10

Tabelle 4.1.: Übersicht Analyse 1

Signaturen Abbildung 4.2 zeigt die durch Cuckoo gefunden Signaturen in dem untersuchten Sample.Hierbei ist zu betonen, dass die Signatur 3 ignoriert werden kann, da diese durch ein fehlendes Whitelisting der Cuckoo-Sandbox entsteht[13].

Als relevante Signaturen wurden die Installation eines Programms im Autostart von Windows und die Registrierung als Dienst ermittelt.Weiterhin wurde das Sample von 43 Anti-Viren-Scannern als schadhaft erkannt. Die Anwendung wurde mit Hilfe des Packers `Armadillo` gepackt.

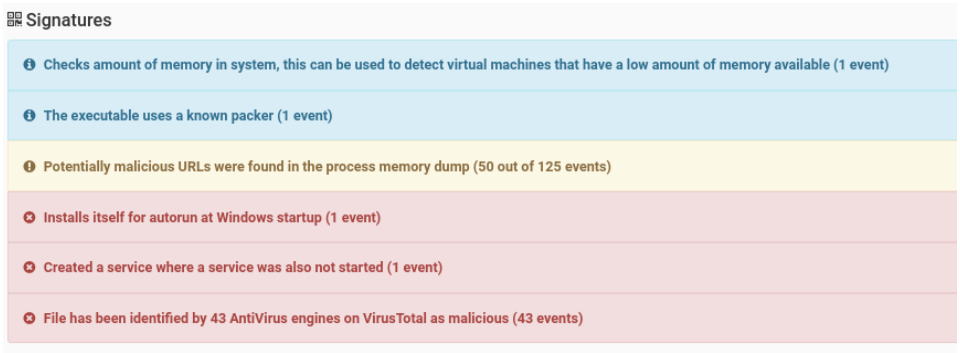

Abbildung 4.2.: Erkannte Signaturen Analyse 1

Statische Analyse

Imports und Exports

KERNEL32.dll Kernfunktionalitäten der Win32 API

ADVAPI32.dll Funktionen zur Verwaltung der Registry, Services und Benutzeraccounts

WININET.dll Stellt Internetfunktionen bereit

Mit Hilfe der von Sikorski zusammengestellten Liste mit Windows-Funktionen, die häufig von Schadsoftware benutzt werden (siehe [17, S. 453–463]) konnten folgende auffällige Funktionen durch die statische Analyse ermittelt werden (vollständige Liste siehe Abbildung D.1):

[13] https://github.com/cuckoosandbox/cuckoo/issues/1987

CreateWaitableTimer/SetWaitableTimer Das Programm erstellt einen Timer und aktiviert diesen[14] [15].

CreateMutex/OpenMutex Erstellt ein Mutex-Objekt, welches sicherstellt, dass die Schadsoftware genau einmal zur gleichen Zeit ausgeführt wird. Anhand des Mutex-Namens kann es möglich sein die Schadsoftware zu erkennen (z. B. in der String-Analyse)[16] [17].

CreateService Erstellt einen Service und fügt diesen der Service-Kontrollmanager-Datenbank hinzu[18].

InternetOpen/InternetOpenUrl Diese Funktionen ermöglichen den Zugriff auf eine FTP- oder HTTP-Ressource[19] [20].

Strings Aus der vollständigen Liste der extrahierten Zeichenketten (siehe Abbildung D.2) wurden folgende als relevant identifiziert:

String	Beschreibung
!This program cannot be run in DOS mode.	Elementarer Bestandteil des PE-Headers einer Windows-Anwendung.
MalService	Möglicherweise der Name des erstellten Services
HGL345	Keine bekannte Zuordnung, eventuell der Name des erstellten Mutex-Objekts
`http://www.malwareanalysisbook.com`	URL die eventuell durch die Schadsoftware aufgerufen wird.

Tabelle 4.2.: Relevante Strings Analyse 1

Verhaltensanalyse

Die extrahierte Daten aus der Verhaltensanalyse wurden anschließend gezielt nach weiteren Informationen zu den bereits gefundenen Hinweisen untersucht. Das Pro-

[14] https://msdn.microsoft.com/en-us/library/windows/desktop/ms682492(v=vs.85).aspx
[15] https://msdn.microsoft.com/en-us/library/windows/desktop/ms686289(v=vs.85).aspx
[16] https://msdn.microsoft.com/en-us/library/windows/desktop/ms682411(v=vs.85).aspx
[17] https://msdn.microsoft.com/en-us/library/windows/desktop/ms684315(v=vs.85).aspx
[18] https://msdn.microsoft.com/en-us/library/windows/desktop/ms682450(v=vs.85).aspx
[19] https://msdn.microsoft.com/en-us/library/windows/desktop/aa385096(v=vs.85).aspx
[20] https://msdn.microsoft.com/en-us/library/windows/desktop/aa385098(v=vs.85).aspx

gramm erstellt einen Service `Malservice` mit Starttyp 2, welches einen automatischen Start während des Systemstarts zur Folge hat (siehe Abbildung 4.3).

Abbildung 4.3.: Services Analyse 1

Weiterhin kann nachgewiesen werden, dass ein Mutex-Objekt mit dem Namen `HGL345` angelegt wird (siehe Abbildung 4.4).

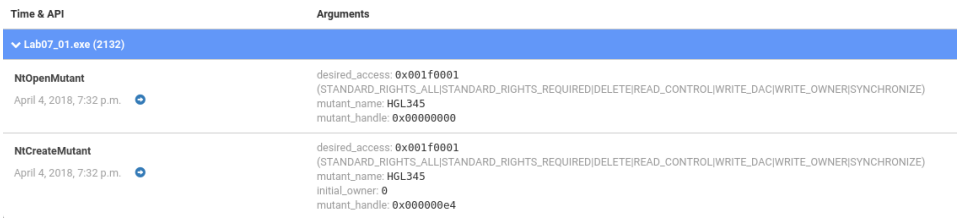

Abbildung 4.4.: Mutex Analyse 1

Hinweise auf den Aufruf der Domain `http://www.malwareanalysisbook.com` konnten hier nicht nachgewiesen werden. Ebenso werden keine relevanten Registryeinträge gelesen oder manipuliert.

Netzwerkanalyse

Die Netzwerkanalyse erbrachte keine relevanten Erkenntnisse.

Extrahierte Dateien

Es konnten keine weiteren Dateien extrahiert werden.

Sonstiges

Keine weiteren Auffälligkeiten.

Zusammenfassung

Mit den in der Sandbox extrahierten Daten können folgende Aussagen getroffen werden:

- Die Schadsoftware registiert sich selbst als Service `Malservice`.
- Der Malservice wird während des Systemstarts gestartet.
- Die Schadsoftware erstellt ein Mutex-Object mit dem Namen `HGL345`.
- Die Schadsoftware erstellt ein Timerobjekt, über das keine weiteren Informationen verfügbar sind.
- Die Schadsoftware enthält eine Domain, welche aber während der Ausführung nicht aufgerufen wird.

Weitere Durchläufe mit verschiedenen Parametern (erzwungene Laufzeit 10min) und Volatility brachten keine weiteren Erkenntnisse.

4.2.2. Analyse 2

Dieses Sample wurde aufgrund des relativ hohen Alters gewählt, um möglichst Anti-Analyseverfahren auszuschließen. Weiterhin besteht die Möglichkeit Netzwerktraffic bei einem Bot zu beobachten.

Übersicht

Dateiname	BOTBINARY.EXE (IllusionBot_May2007.zip)
Typ	PE32 executable (GUI)
MD5 Hash	9b9e083a9cf6a1db6251e189e5966a4d
Quelle	`https://github.com/ytisf/theZoo/tree/master/malwares/Binaries/IllusionBot_May2007`
Analyseergebnisse	Anhang E und `../Analysen/2/`
Cuckoo-Node	winXp-01
Routing	TOR
Zusätzliche Parameter	Volatility aktiviert, Erzwungener Timeout nach 5 Minuten
Cuckoo-Score	5.4/10

Tabelle 4.3.: Übersicht Analyse 2

Signaturen Bei der Analyse dieses Samples wurden 9 Signaturen erkannt,die in Abbildung 4.5 dargestellt sind. Das Programm erstellt einen Service mit Namen `ntndis` und registriert diesen für den Autostart. Weiterhin wird versucht eine Verbindung zu einem IRC-Server aufzubauen.Die Signaturerkennung durch Volatility hat außerdem einen injizierten Prozess gefunden.

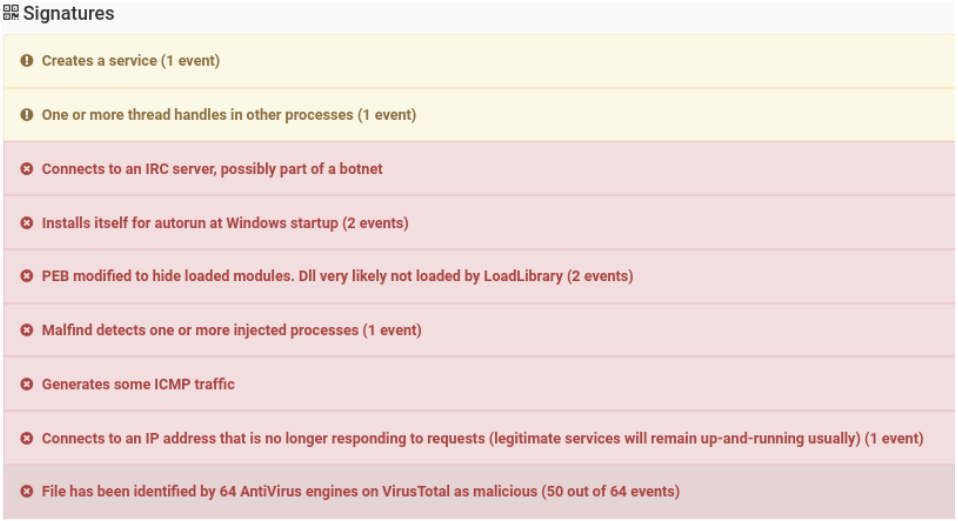

Abbildung 4.5.: Erkannte Signaturen Analyse 2

Statische Analyse

Imports/Exports Cuckoo konnte nur den Import der `KERNEL32.dll` extrahieren (siehe Abbildung 4.6). Wie bereits durch Volatility erkannt ist es wahrscheinlich, dass das Laden weiterer DLLs durch die Manipulation des PE-Headers verschleiert wird.

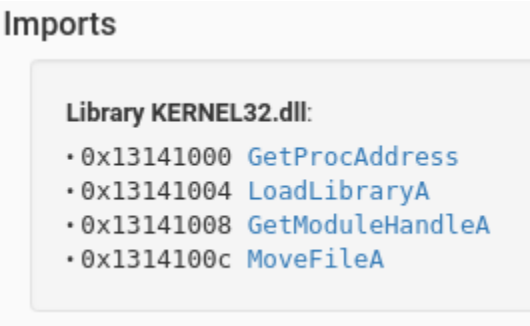

Abbildung 4.6.: Imports Analyse 2

Strings Durch die Betrachtung der extrahierten Strings ist es möglich Hinweise auf verwendete Windows-Funktionen zu erhaltenDa über 600 Strings extrahiert wur-

den, wird hier nur ein Auszug dargestellt. Die vollständige Liste ist im Dateianhang in dem JSON-Protokoll nachzulesen.

Interessant ist das zweifache Vorkommen der Zeichenkette `This program cannot be run in DOS mode.`, was daraufhin deutet, dass eine weitere ausführbare Datei eingebettet ist. Diese wurde durch Cuckoo ebenfalls extrahiert und auf Seite 33 weiter analysiert.

Weitere Einträge lassen auf den Import der Bibliotheken `wininet.dll, ntdll.dll, gdi32.dll, advapi32.dll, user32.dll, kernel32.dll und ws2_32.dll` schließen. Ebenfalls wurden eine Vielzahl von Windows-Funktionen gefunden, die Funktionalitäten zum Lesen und Ändern von Registrykeys, Erzeugen von Prozessen, Herstellen von Netzwerkverbindungen und Auslesen von Systeminformationen bieten.

Die Datei enthält eine Vielzahl an Strings, die auf eine Command-und-Control-Funktionalität hinweisen, mit denen sich das Verhalten des Hosts beeinflussen lässt, ebenso mögliche IRC-Befehle. Folgendes Listing zeigt einen Auszug:

Listing 4.8: CnC-Zeichenketten

```
1 Usage: !httpflood <host> [port] [path_to_script]
2 HTTP Flooder started
3 HTTP Flooder terminated
4 Usage: udpflood <host> [port]
5 Usage: !email <server> <port> <from> <to> <attach>
6 Usage: !get <url> <local> [noexec]
7 USER %s 0 * :%s
8 NICK %s
9 PASS %s
```

Verhaltensanalyse

Wie durch die Analyse der Strings vermutet erstellt die Schadsoftware eine weitere Datei `ntndis.sys`, welche sie als Service `ntndis` anlegt und für den Autostart registriert (siehe Abbildung E.1 und E.2). Die Schadsoftware verschiebt sich selber in den Pfad C:\WINDOWS\system32\drivers\ntndis.exe

Die Schadsoftware erstellt eine Firewall-Regel für sich selbst, um den Zugriff auf Netzwerkressourcen zu ermöglichen (siehe Abbildung E.3).

Die in der String-Analyse gefunden DLLs werden über die undokumentierte NTAPI-Funktion LdrLoadDll geladen (siehe Abbildung E.4).

Netzwerkanalyse

Die Schadsoftware versucht eine Verbindung zu einem IRC-Server mit der IP `217.16.29.198` aufzubauen (siehe Abbildung E.5). Dazu wird mittels PASS das Passwort gesetzt. Anschließend erfolgt das Festlegen des Benutzernamens mittels NICK, wofür der Computername verwendet wird. Es wird versucht dem Channel `skaters_1990` beizutreten. Zum Schluss wird die Verbindung beendet. Der IRC-Server ist nicht mehr aktiv, daher kann keine Verbindung aufgebaut bzw. keine Antwort empfangen werden.

Extrahierte Dateien

Die extrahierte Datei `ntndis.sys` wird durch Cuckoo bzw. VirusTotal als Rootkit identifiziert[21], welches vermutlich über System Service Dispatch Table (SSDT)-Hooking (siehe [3]) die in der String-Analyse gefundenen DLLs lädt und so den privilegierten Zugriff ermöglicht.Die Datei wurde ebenfalls mit Cuckoo analysiert, die Ergebnisse liegen dem Dateianhang bei.

Anzeichen für ein SSDT-Hooking könnten die abweichenden Einsprungadressen sein (siehe Abbildung 4.7). Eine tiefergehende Analyse wird im Rahmen dieser Hausarbeit aufgrund des Umfangs nicht durchgeführt.

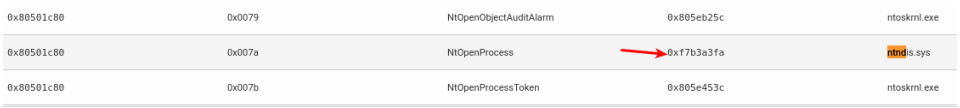

Abbildung 4.7.: SSDT Analyse 2

Sonstiges

Zusammenfassung

Aufgrund der durch Cuckoo gewonnenen Analysergebnisse können folgende Aussagen getroffen werden:

- Die Applikation erstellt einen Autostart-Service `ntndis`
- Die Applikation ermöglicht sich selbst den Netzwerkzugriff, indem sie eine Firewall-Regel erstellt

[21] `https://www.sophos.com/en-us/threat-center/threat-analyses/viruses-and-spyware/Troj~RKProc-F/detailed-analysis.aspx`

- Die Bibliotheks-Importe sind verschleiert und werden nicht durch LoadLibrary ausgeführt
- Die extrahierten Strings weisen auf eine Command-and-Control-Funktion hin, die verschiedene Verwendungszwecke für einen infizierten Host ermöglicht
- Die Applikation versucht eine Verbindung zu einem inaktiven IRC-Server aufzubauen
- Die Datei `ntndis.sys` wird aus der Datei `BOTBINARY.exe` extrahiert, die aus dem Rootkit extrahierten Strings bilden eine Teilmenge der Hauptdatei
- Das eingebettete Rootkit nutzt SSDT-Hooking, um die DLLs zu laden und privilegierten Zugriff zu ermöglichen

Da der IRC-Server nicht mehr aktiv ist konnte die weitere Funktionalität der Schadsoftware nicht getestet werden.

Literatur

[1] Essam Al Daoud, Iqbal Jebril und Belal Zaqaibeh. *Computer Virus Strategies and Detection Methods*. 2.09.2008. URL: https://www.researchgate.net/publication/228937695_Computer_virus_strategies_and_detection_methods (besucht am 02. 03. 2018).

[2] Andrea Allievi und Elia Floria. *FinFisher exposed: A researcher's tale of defeating traps, tricks, and complex virtual machines*. Microsoft. 1.03.2018. URL: https://cloudblogs.microsoft.com/microsoftsecure/2018/03/01/finfisher-exposed-a-researchers-tale-of-defeating-traps-tricks-and-complex-virtual-machines/ (besucht am 07. 03. 2018).

[3] Frank Block. *Investigating Memory Analysis Tools. SSDT Hooking via Pointer Replacement*. 4.04.2018. URL: https://insinuator.net/2015/12/investigating-memory-analysis-tools-ssdt-hooking-via-pointer-replacement/.

[4] Byte-Atlas. *Hardening Win7 x64 on VirtualBox for Malware Analysis*. 5.02.2017. URL: http://byte-atlas.blogspot.de/2017/02/hardening-vbox-win7x64.html (besucht am 05. 04. 2018).

[5] European Union Agency for Network and Information Security, Hrsg. *ENISA Thread Landscape Report2017*. 2018. URL: https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2017 (besucht am 27. 02. 2018).

[6] Peter Ferrie. *The Ultimate Anti-Debugging Reference*4.05.2011. URL: http://pferrie.host22.com/papers/antidebug.pdf (besucht am 05. 04. 2018).

[7] Michael Koll. *Ansible-Cuckoo*. 6.04.2018. URL: https://github.com/michkoll/ansible-cuckoo (besucht am 06. 04. 2018).

[8] Jyoti Landage und M. P. Wankhade. „Malware and Malware Detection Techniques: A Survey“. In: *International Journal of Engineering Research & Technology*. 2(12). 2013, S. 61–68. URL: https://www.ijert.org/download/6744/malware-and-malware-detection-techniques--a-survey (besucht am 02. 03. 2018).

[9] Cameron H. Malin, Eoghan Casey und James M. Aquilina. *Malware forensics field guide for Windows systems. Digital forensics field guides*. eng. Digital forensics field guides. Waltham, MA: Syngress, 2012. 518 S. ISBN: 978-1-59749-472-4.

[10] *Multi-scanning: Scan with Multiple Leading Anti-malware Engines.*Opswat. 2018. URL: https://www.opswat.com/products/metadefender/core/multi-scanning (besucht am 07. 03. 2018).

[11] Rahul Nair. *ANTI-DISASSEMBLY TECHNIQUES USED BY MALWARE (A PRIMER).* URL: https://www.malwinator.com/2015/11/22/anti-disassembly-used-in-malware-a-primer/ (besucht am 05. 04. 2018).

[12] Martin Rieger, Tobias Scheible und David Schlichtenberger. *IT-Sicherheit und IT-Angriffe.* Version 4. Auflage. Balingen: Hochschule Albstadt-Sigmaringen, 2018. (Besucht am 27. 02. 2018).

[13] Martin Rieger und David Schlichtenberger. *Themen Hausarbeit M106 WS 17/18.* 2018. URL: https://elearning.hs-albsig.de/goto.php?target=file_194992_download&client_id=HS-ALBSIG (besucht am 27. 02. 2018).

[14] SANS DFIR. *Windows Forensic Analysis Poster.* 2017. URL: https://digital-forensics.sans.org/media/Poster_Windows_Forensics_2017_WEB.pdf (besucht am 28. 02. 2018).

[15] Bruce Schneier. *The Storm Worm.* 2007URL: https://www.schneier.com/blog/archives/2007/10/the_storm_worm.html (besucht am 02. 03. 2018).

[16] Oliver Schonschek. *Malware verstehen, erkennen und abwehren.* Hrsg. von Security-Insider. 31.01.2017. URL: https://www.security-insider.de/malware-verstehen-erkennen-und-abwehren-a-578417/ (besucht am 01. 03. 2018).

[17] Michael Sikorski und Andrew Honig. *Practical Malware Analysis. A Hands-On Guide to Dissecting Malicious Software.* eng. San Francisco: No Starch Press, 2012. 802 S. ISBN: 978-1-59327-290-6.

[18] Ronghua Tian. „An Integrated Malware Detection and Classification System“. Thesis. Deakin University, 2011.URL: https://dro.deakin.edu.au/eserv/DU:30043244/Tian-thesis-2011.pdf (besucht am 04. 03. 2018).

[19] Tutanch und Peter Schmitz. *Was sind Virenscanner.*2017. URL: https://www.security-insider.de/was-sind-virenscanner-a-570942/ (besucht am 01. 03. 2018).

[20] Ilsun You und Kangbin Yim. „Malware Obfuscation Techniques: A Brief Survey“. In: *2010 International Conference on Broadband,Wireless Computing, Communication and Applications.*2010 International Conference on Broadband, Wireless Computing, Communication and Applications (BWCCA). (Fukuoka, TBD, Japan). IEEE, 4.11.2010 - 06.11.2010, S. 297–300. ISBN: 978-1-4244-8448-5. DOI: 10.1109/BWCCA.2010.85 . URL: DOI%2010.1109/BWCCA.2010.85.

[21] Lenny Zeltser. *Cheat Sheet for Analyzing Malicious Software*. 2017. URL: https://zeltser.com/malware-analysis-cheat-sheet/ (besucht am 28. 02. 2018).

A. Aufgabenstellung

Die Aufgabenstellung lautet wie folgt (siehe [13, S. 21]):

- THEORETISCHER TEIL:
  - Welche grundsätzlichen Verfahren nutzen AV-Scanner, um Malware zu erkennen?
  - Welche Analyseverfahren für Malware gibt es? Kategorisieren Sie diese.
  - Welche Verfahren und Mechanismen nutzen Angreifer, um die Erkennung ihrer Malware zu erschweren oder zu verhindern? Erläutern Sie die Verfahren anhand von Beispielen.
- PRAKTISCHER TEIL
  - Im praktischen Teil sollen Sie ihre eigene Schadsoftware-Analyse-Sandbox (auf Basis von Cuckoo) erstellen und damit experimentieren.
  - Als Basis können Sie wie in `https://www.heise.de/security/artikel/Malware-Analyse-Do-ItYourself-3910855.html?seite=all` beschrieben vorgehen.
  - Dokumentieren Sie ihre Vorgehensweise der Installation und Konfiguration.
  - Prüfen Sie exemplarisch Malwaresamples, aber auch unschädliche Dateien und interpretieren Sie die Ergebnisse detailliert.

B. Verwendete Software

Folgende Softwareversionen wurden auf dem Host verwendet:

Software	**Version**
Ubuntu	16.04.04 LTS
Python	2.7.12
Ansible	2.5.0
PIP	8.1.1
tcpdump	4.9.2
Volatitlity	2.5
Cuckoo	2.0.5.3
setuptools	39.0.1
weasyprint	0.36
yara	3.3.0
vmcloak	0.4.5
MongoDB	2.6.10
VirtualBox	5.1.34

Tabelle B.1.: Software Host

Folgende Softwareversionen wurden auf dem Cuckoo-Node win7x64-01 verwendet:

Windows 7	SP 1
Adobe Reader	9.0.0
Adobe Flash Player	11.7.700.169
Microsoft Office	2013
Java RE	7.0.0
.NET Framework	4.5
Internet Explorer	8.0.7601
Microsoft Visual C++ Redist	8.0.56336

Tabelle B.2.: Software win7x64-01

Folgende Softwareversionen wurden auf dem Cuckoo-Node winXp-01 verwendet:

Windows XP	SP3
Adobe Reader	9.0.0

Java RE	7.0.0
Internet Explorer	8.0.6001

Tabelle B.3.: Software winXp-01

C. Dateien

- `Hausarbeit.pdf` Hauptdokument
- `Analysen` Enthält alle Dateien zu den durchgeführten Analysen
  - `1` Analysedaten der Datei Lab07_01.exe
    - `Report` zugehöriger Cuckoo-Report
    - `Sample` zugehöriges Sample (Passwort`infected`)
  - `2` Analysedaten der Datei BOTBINARY.EXE und der extrahierten Datei ntndis.sys
    - `Report` zugehörige Cuckoo-Reports
    - `Sample` zugehörige Samples (Passwort`infected`)
- `Misc` zusätzliche Dateien
  - `Ansible-Cuckoo` Ansible-Playbook zur Cuckoo-Installation
    - `inventory`
      - `hosts` Ansible-Inventory
      - `host_vars` Variablendefinition für Playbook
    - `roles` Ansible-Rollen; Kernfunktionalität des Playbooks
    - `tests` Travis-Tests
    - `site.yml` Ansible-Playbook; Hauptdatei
    - `README.md` Beschreibung der Funktionalität und Bedienung

D. Analyse 1

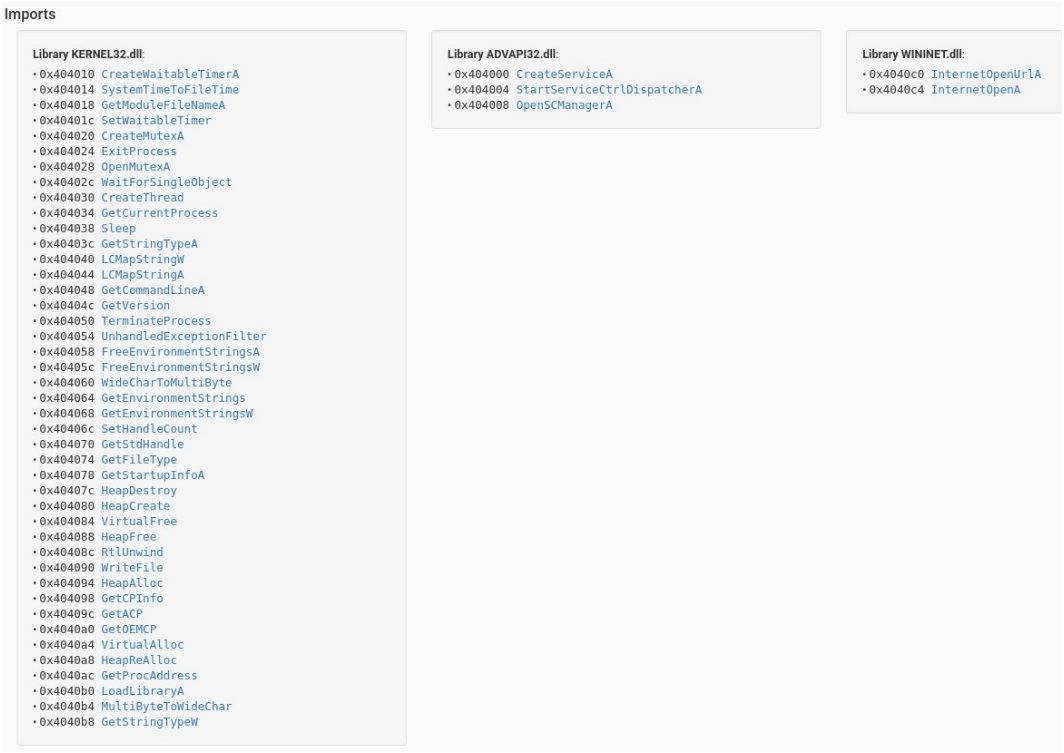

Abbildung D.1.: Imports Analyse 1

Abbildung D.2.: Strings Analyse 1

E. Analyse 2

Abbildung E.1.: ntndis.sys Analyse 2

Abbildung E.2.: Service Analyse 2

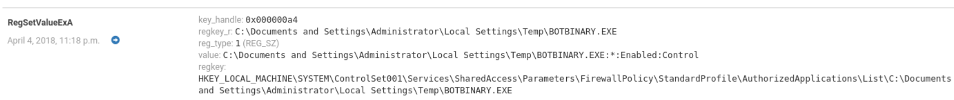

Abbildung E.3.: Firewall Analyse 2

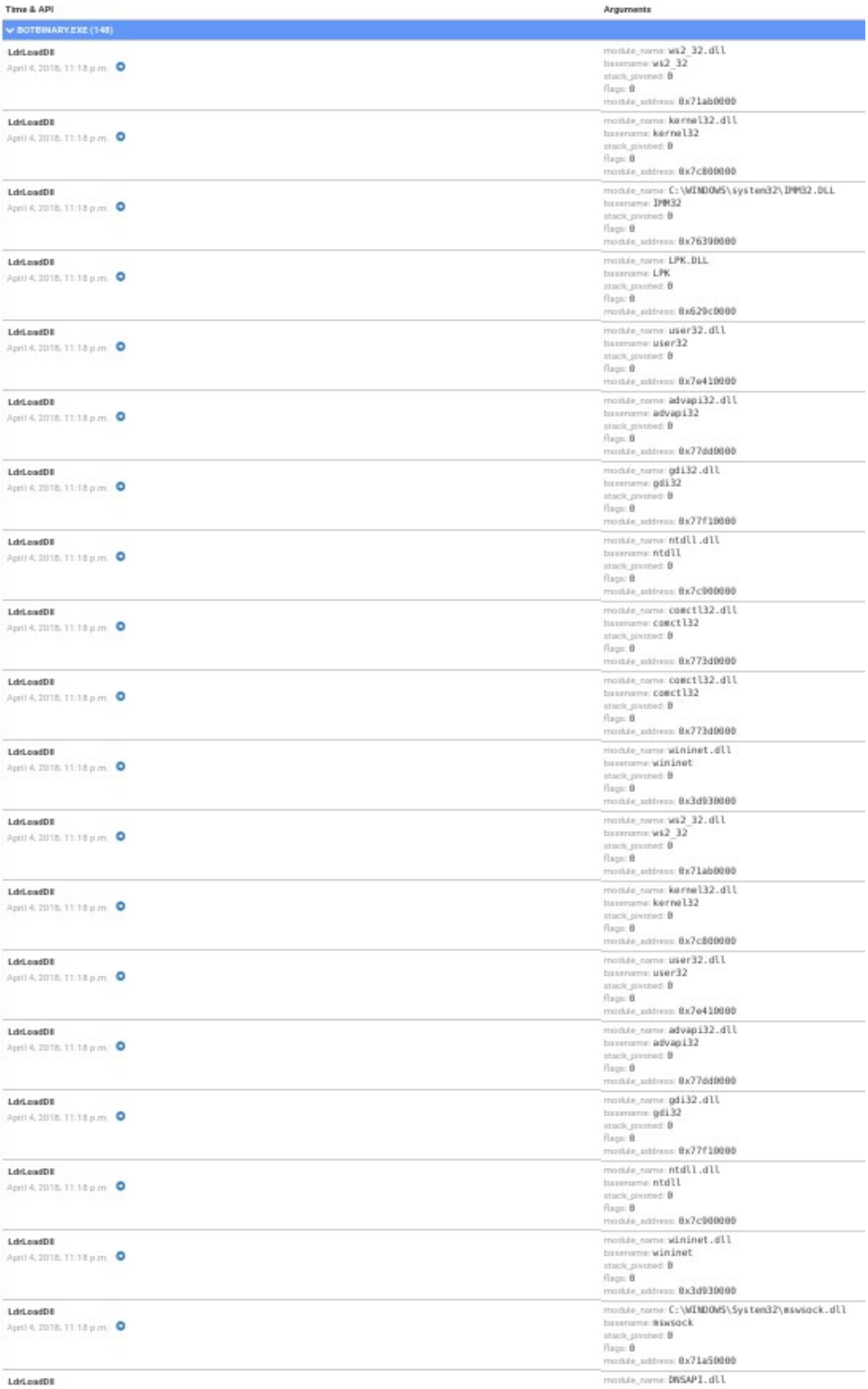

Abbildung E.4.: LdrLoadDll Analyse 2

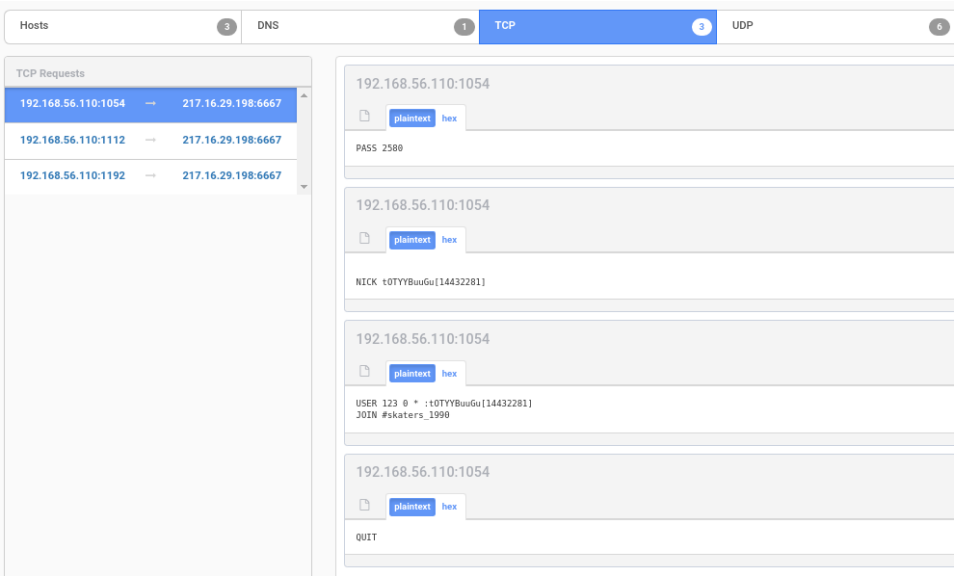

Abbildung E.5.: Netzwerk Analyse 2

Glossar

C

Command-and-Control-Server Server,der zur Steuerung eines Botnetzes verwendet wird.

D

Distributed Denial of Service Nichtverfügbarkeit eines Dienstes, die durch Vielanfragen an diesen Dienst hervorgerufen wird..

I

Integrität Schutzziel der Informationssicherheit; Integrität beschreibt die Gewährleistung der Qualität von Informationen..

M

Malware Zusammengestetzt aus *malicious* und *software*, siehe Schadsoftware.

O

On-Access-Scanner AV-Scanner,der ständig im Hintergrund das System nach Malware scannt ohne eine manuelle Aktivierung zu benötigen.

On-Demand-Scaner AV-Scanner,der nur auf Aktivierung durch einen Benutzer einen Scan durchführt..

P

persistent Gegenteil von volatil, dauerhaft verfügbare Daten,die auch nach dem Ausschalten eines Computersystems zur Verfügung stehen.

S

Sandbox Abgeschottete, meist durch eine virtuelle Maschine realisierte, Umgebung zur Analyse von potentiell schädlichen Dateien.

Scan-Engine Die Scanengine ist der elementare Bestandteil eines AV-Scanner und ist für die eigentliche Untersuchung der Daten eines Computer zuständig. Die Scanengine ist maßgeblich für die Effizienz eines AV-Scanner verantwortlich.

Schadsoftware Bösartige, schädliche Software, die ungewollte Aktionen auf einem Computersystem ausführt.

V

Vertraulichkeit Schutzzielder Informationssicherheit;Vertraulichkeit gewährleistet den Zugriff auf Informationen nur für berechtigte Beteiligte..

volatil flüchtig; hier: Flüchte Daten innerhalb eines Computersystems, wie z. B. Informationen im Arbeitsspeicher, die bei Ausschalten des Systems nicht mehr zur Verfügung stehen.