Technische
Hochschule Ulm
University of
Applied Sciences

Machine Learning im Kontext von Cyber Security

Masterarbeit
zur Erlangung des Grades eines Master of Science (M.Sc.) im Studiengang Informationssysteme

vorgelegt von
Kathrin Rodi

Matrikelnummer: 3129378

21. November 2019

Erstgutachter: Prof. Dr. Reinhold von Schwerin

Zweitgutachter: Prof. Dr. Markus Schäffter

Betreuer: Hans-Martin Münch

Eigenständigkeitserklärung

Diese Abschlussarbeit wurde von mir selbständig verfasst. Es wurden nur die angegebenen Quellen und Hilfsmittel verwendet. Alle wörtlichen und sinngemäßen Zitate sind in dieser Arbeit als solche kenntlich gemacht.

Kathi Rodi, 21. November 2019

Abstract

Machine Learning Ansätze sind im Kontext von Cyber Security essenziell, da es durch immer anspruchsvoller werdende Sicherheitsbedrohungen nicht mehr möglich ist deren Indikatoren manuell zu ermitteln und zu klassifizieren. Diese Aufgabe von Menschen bearbeiten zu lassen wäre deutlich zu kostenintensiv und zu ineffizient.
Anhand einer Literaturrecherche nach Webster&Watson wird überprüft, welchen Mehrwert Machine Learning in Bezug auf Informationssicherheit bieten kann. Dazu werden bestehende Ansätze aus der Industrie sowohl als auch aus der Wissenschaft klassifiziert, wobei die jeweils verwendeten Algorithmen, Features, Evaluationskriterien sowie die durchgeführte Evaluation der jeweiligen Ergebnisse untersucht werden.
Des Weiteren wird der Begriff Indicator of Compromise (IoC) geklärt, und besonders auf dessen Bedeutung, in Bezug auf Malware Erkennung eingegangen.
Zusätzlich wird recherchiert welche Datensätze in Bezug auf Cyber Security bestehen und welche Qualität diese aufweisen.
Ergänzend werden Diskrepanzen zwischen Ansätzen aus der Industrie und der Wissenschaft überprüft, um etwaige Aussagen aus der Industrie herauszufiltern, welche noch nicht wissenschaftlich belegt wurden, respektive wissenschaftliche Ansätze zu finden, welche bereits von der Industrie genutzt werden. Ziel der Arbeit ist es eine umfassende Übersicht über bestehende Machine Learning Ansätze, welche dabei helfen Indikatoren von Cyberangriffen zu klassifizieren, zu gewinnen und einer dieser gefunden Ansätze beziehungsweise einen gefundenen qualifizierten Datensatz wissenschaftlich zu validieren.

Inhaltsverzeichnis

1. Einleitung

Bereits im 19. Jahrhundert träumte der Polymath Charles Babbage vom mechanisierten Rechnen. Dieser Wunsch basierte hauptsächlich auf dem Zorn über die Unzulänglichkeit der damaligen analogen mathematischen Anwendungen. Babbage entwickelte ein Konzept für analytische Maschinen, also einen programmierbaren Allzweckrechner. Seine Kollegin, die britische Mathematikerin, Ada Lovelace lieferte die entsprechenden Ideen zur Programmierung seiner Maschine. Allerdings konnte das Konzept der *Analytical Engine* niemals umgesetzt werden und besteht seither, rein als Entwurf. Dennoch macht diese Forschung die beiden bis heute zu Pionieren des modernen Computers und dessen Programmierung.
Babbages Wunschtraum von damals ist nicht nur längst Wirklichkeit geworden, er hat sich in rasendem Tempo weiterentwickelt. Heute können Computer nicht nur fehlerfrei Logarithmen berechnen, sie sind selbst in der Lage einen Großteil unseres Lebens zu digitalisieren. Bankgeschäfte, Einkäufe, die Steuererklärung und bald auch Arztbesuche sind nur ein kleiner Teil dessen, was wir online erledigen. Dabei produzieren wir eine enorme Masse an persönlichen Daten, welche in falschen Händen, zu einem physischen Schaden für uns führen können. Gerade deshalb gilt es diesen Teil unseres Lebens zu schützen. Wie wir unsere physischen Habseligkeiten schützen in dem wir beispielsweise Schlösser verwenden, gilt es ebenso unsere digitalen Artefakte zu schützen, um finanziellen, reputativen sowie physischen Schaden zu verhindern. Studien zeigen allerdings, dass wir der nötigen Sicherheit weit hinterherhinken.

1.1. Motivation

> *Cybercrime umfasst die Straftaten, die sich gegen Datennetze, informationstechnische Systeme oder deren Daten richten [...] oder die mittels Informationstechnik begangen werden. (Bundeskriminalamt 2018)*

Das Bundeskriminalamt (BKA) verzeichnete allein im Jahr 2017 knapp 86.000 Fälle von Cybercrime. Davon waren über 1.400 Phishing Angriffe im Onlinebanking bei

denen ein durchschnittlicher Schaden von 4000 € pro Fall entstand (Bundeskriminalamt 2018). Laut einer Studie des Bundesverband Informationswirtschaft,Telekommunikation und neue Medien e.V. (BITKOM) ist bereits, jeder zweite Deutsche Opfer eines Cyberangriffs geworden, lediglich 18% hätten diesbezüglich angegeben, Anzeige bei der Polizei erstattet zu haben (e.V. 2017). Dies lässt vermuten, dass die Dunkelziffer der tatsächlichen Cybercrime-Straftaten weit über den 86.000 gemeldeten Fällen liegt. AVTest registriert täglich bis zu 350.000 neue Schadhafte Programme (AV-TEST 2019). Das Problem hierbei ist nicht allein die Quantität der Software sondern auch die Qualität. Immer bessere Verschleierungstaktiken sorgen dafür, dass Malware schwerer identifizierbar wird. Sicherheitsüberprüfungen die auf Signaturabgleichen beruhen, funktionieren beispielsweise nur bei bereits bekannten Signaturen, neuartige Malware kann von ihnen nicht erkannt werden. Diese Komplexität und Fülle an Malware überfordert nicht nur Intrusion Detection Systeme, sondern auch Sicherheitsexperten. Wie schon im Jahre 2010 von Evans und Reeder (2010) vorhergesagt, fehlt es an Expertise für diese Flut an Angriffen. Da der Mangel an Fachkräften, wenn überhaupt, erst in Jahren ausgeglichen werden kann, bedarf es alternativer Lösungen für die Sicherheit von Heute.

Machine Learning kann der Schlüssel hierfür sein. Diese Technologie kann für die automatische Verarbeitung von Sicherheitsereignissen genutzt werden. Gängige Warnungen können leicht von Machine Learning Verfahren überprüft werden, dadurch haben Sicherheitsexperten mehr Kapazität sich um besondere Warnungen zu kümmern. Des Weiteren ist es schwierig Warnsignale zusehends zu priorisieren und zu kategorisieren. Auch hierbei können Algorithmen helfen. Beispielsweise lässt sich ein System implementieren, welches eine Klassifizierung in gutartig oder bösartig durchführt. Dabei spricht man von einer *binären* Klassifikation. Gleichzeitig ist es möglich, die als bösartig gelabelten Daten in diverse Kategorien einzustufen. Beispielsweise kann Malware, durch *Multi-Klassen Klassifikation*, in Subklassen wie Viren, Würmer, Trojaner und Ransomware aufgeteilt werden, wodurch die spezifische Untersuchung und Bekämpfung effizienter gestaltet werden kann. Eine weitere Fähigkeit von Machine Learning Algorithmen (MLAs) ist das *Clustering*. Diese Technik fasst grundsätzlich ähnliche Inhalte zusammen. Dabei entstehen Gruppen mit Daten die eine hohe interne Homogenität, verglichen mit anderen Gruppen jedoch eine hohe Heterogenität aufweisen. Clustering kann unter anderem dazu genutzt werden, Hypertext Transfer Protocol (HTTP) Verkehr zu analysieren und herauszufinden, um welche Art von Anfragen es sich handelt. Die Requests können beispielsweise zu Botnet-, Mobiltelefon- oder gängige Benutzeranfragen geclustert werden. Dies stellt eine immense Erleichterung für Sicherheitsexperten dar, da sich diese unmittelbar dem potenziell gefährlichen Cluster widmen können. Machine Learning Algorithmen besitzen die Fähigkeit des eigenständigen Lernens. Da sie so zu neuen Erkenntnissen gelangen und nicht auf bereits bekannte Warnun-

gen, wie beispielsweise bösartige Signaturen, angewiesen sind, können mit ihrer Hilfe sowohl Advanced Persistent Threats (APTs) als auch Zero-days erkannt werden. Anhand der dadurch verringerten Antwortzeit auf Attacken, kann nicht nur ein Verlust von Daten, sondern auch ein finanzieller Schaden stark abgemildert werden.

Forschungen belegen die Wirksamkeit von Machine Learning Ansätzen im Bereich Cyber Security und somit die hier aufgeführten Thesen. Um dies zu verdeutlichen, werden adäquate Untersuchungen in Kapitel 4 *Bestehende Analyseverfahren* beschrieben.

1.2. Ziel der Arbeit

Die Ziele dieser Arbeit belaufen sich auf die folgenden vier Punkte:

1. Zunächst soll der Begriff *Indicator of Compromise* geklärt und in Bezug auf Malware untersucht werden.
2. Des Weiteren soll der momentane Stand der Forschung im Bereich Malware Analyse, mit Hilfe von Machine Learning Verfahren, erörtert werden und etwaige Diskrepanzen mit dem Stand der Industrie aufgedeckt werden.
3. Um eine aussagekräftige Malware Analyse zu tätigen, Bedarf es qualitativ hochwertiger Datensätze. Um einen solchen Datensatz zu ermitteln gilt es, zu evaluieren welche Datensätze einer Analyse dienlich sind.
4. Ferner soll eine prototypische Umsetzung einer Analyse mit einem der evaluierten Datensätze durchgeführt werden, um dessen Tauglichkeit für zukünftige Analysen im Bereich Cyber Security mit Hilfe des maschinellen Lernens zu prüfen.

Die Arbeit soll somit sowohl Sicherheitsexperten als auch Data Scientists dienen, um einen Überblick über den momentanen Stand der Forschung zu liefern. Zu dem soll es diesem Publikum durch die Evaluierung der Datensätze erleichtert werden, eigenständige neue Analysen durchzuführen oder bestehende Analyseverfahren zu optimieren. Die prototypische Implementierung soll diesbezüglich als Beispiel dienen.

1.3. Aufbau der Arbeit

Das erste Kapitel dient der Einführung in das Thema, wobei zusätzlich die Relevanz der Forschung erläutert wird. Ferner werden die Ziele der Arbeit abgesteckt. Das folgende Kapitel *Forschungsmethode* erläutert das fundierte Vorgehen, durch welches Informationen generiert und Erkenntnisse erlangt wurden. Der Hauptteil besteht aus drei Teilen: der Untersuchung der *Bestehenden Analyseverfahren*, der Evaluierung der *Datensätze* sowie der *prototypischen Implementierung* eines Analyseverfahrens anhand eines der evaluierten Datensätze. Im Anschluss wird zu den Ergebnissen kritisch Stellung bezogen, sowie ein Fazit gezogen. Abschließend wird ein Ausblick für potenzielle zukünftige Forschungen gegeben.

2. Forschungsmethoden

In diesem Kapitel werden die Forschungsmethoden erläutert, auf welcher der Informationsgewinn basiert und wodurch neue Erkenntnisse gewonnen werden konnten. Das Literaturreview wurde zu Beginn der Arbeit durchgeführt, um Informationen bezüglich des Themas zu sammeln und um den momentanen Stand der Forschung zu identifizieren.
Der Cross-Industry Standard Process for Data Mining (CRISP-DM) wird als Vorgehensmodell ausgewählt, da hierdurch eine strukturierte Vorgehensweise sichergestellt werden kann. Der genaue Aufbau dieses Models wir in Kapitel 2.2 beschrieben.

2.1. Literaturreview

Im Rahmen einer Literaturrecherche nach Webster und Watson (2002) wurden die wissenschaftlichen Datenbanken ACM Digital Library, ScienceDirect und IEEE sowie die akademische Suchmaschine Google Schoolar nach relevanten Inhalten durchsucht. Hierbei wurde darauf geachtet, dass es sich bei den Ergebnissen, um Peer-Reviewed Journals sowie Peer-Reviewed Konferenzen handelt, um eine bestmögliche Qualität der zu verwendenden Quellen zu garantieren. Ferner wurde lediglich nach Publikationen seit 2015 gesucht, um die Aktualität der Ansätze zu gewährleisten. Da sich besonders im Bereich Cyber Security binnen eines Jahres enorme Entwicklungen zeigen, wäre durch das Hinzuziehen älterer Publikationen kein Mehrwert entstanden. Als Suchstring wurde die logische Kombination aus den Begriffen „Machine Learning“OR „Deep Learning“AND „Cyber Security“OR „Information Security“NOT „Android“NOT „IoT“NOT „Mobile“verwendet. Dies beruht darauf, dass Machine Learning und Deep Learnning, sowie Information Security und Cyber Security oftmals synonym verwendet werden. Da sich die Arbeit nicht mit dem Thema mobil oder Internet of Things (IoT) basierter Applikationen beschäftigt, wurden diese Keywords bei der Suche ausgegrenzt. Eine Forschung in diesem Bereich ist gleichermaßen umfangreich und bedarf einer eigenständigen Arbeit. Diese Suche ergab insgesamt 308 Treffer. Zusätzlich wurde sowohl eine Vorwärts - als auch eine Rückwärtssuche durchgeführt, welche zu weiteren 24 Treffern führte. Durch die Rückwärtssuche konnten weitere relevante Ansätze von Machine Learning im Bereich Cyber Security, sowie hilfreiche Informationen zu

bestehenden Datensets ausfindig gemacht werden. Auch die Vorwärtssuche, welche mit Google Schoolar umgesetzt wurde, führte zu hochaktuellen Beiträgen zum Thema. Von den dadurch 332 ausfindig gemachten Quellen wurden 266 anhand Titel, Abstract, Einleitung und Schluss, in Ermangelung von Relevanz oder wegen Überschneidungen bereits gefundener Ansätze, aussortiert. Hingegen wurden 66 Quellen für die hier vorliegende Arbeit verwendet. Eine Übersicht über den Prozess der Literaturrecherche kann im Anhang Anlage 1 eingesehen werden. Wie von Webster und Watson (2002) empfohlen, wurden die gefundenen Quellen anschließend akribisch in einer Liste nach Inhalt und Relevanz gefiltert. Zunächst wurden die ausgewählten Quellen in vier Themenblöcke aufgeteilt:

- Ansatz inklusive Datenset
- Ansatz ohne Datenset
- Indicators of Compromise (IoCs)
- Datensatz

Die Quellen wurden anschließend den einzelnen Blocks zugewiesen. Zudem wurden weitere Blocks erstellt die jedoch keinen Einfluss auf die Relevanz der Quelle hatten und somit hier nicht gelistet sind. Zu jeder Quelle wurde die Kernaussage notiert, sowie gegebenenfalls bereits inhaltsrelevante Punkte, wie verwendete Machine Learning Verfahren, Namen von Datensets oder interessante Ergebnisse. Anschließend wurde die Relevanz der Quellen untersucht. Um diesbezüglich ein systematisches Vorgehen zu garantieren wurden folgende Relevanzkriterien erstellt:

1. Hoher Themenbezug zu mindestens einem der Themen: IoCs oder Datensets
2. Ausführung eines Ansatzes
3. Ausführung eines Ansatzes inklusive verfügbarem Datenset

Die Quellen wurden anhand dieser Skala bewertet, wobei 1 für eine geringe Relevanz und 3 für eine hohe Relevanz steht. Zusätzlich wurde die Anzahl der Zitationen festgehalten, um die wissenschaftliche Relevanz innerhalb der Forschungsgemeinde zu evaluieren. Einen Ausschnitt der daraus resultierenden Literaturliste kann im Anhang Anlage 2 eingesehen werden.

2.2. CRISP-DM

Bereits im 18. Jahrhundert legte Thomas Bayes mit seinem *Satz von Bayes*, der die Berechnung bedingter Wahrscheinlichkeiten beschreibt, den Grundstein dafür was wir heute *Data Mining*, also den Erkenntnisgewinn aus Daten, nennen. Als in den 1950er Jahren die Produktion kommerzieller Seriencomputer startete konnte die Datenanalyse automatisiert werden. Daraus entwickelten sich die ersten Neuronalen Netze und Cluster Analysen wie wir sie heute kennen. Einen weiteren Aufschwung erlebte Data Mining in den 1990ern, wo auch der CRISP-DM von DaimlerChrysler, SPSS und NCR entwickelt wurde (SmartVisionEurope 2015). Dieser Prozess beschreibt eine Methodik für Data Scientists, um eine effiziente, robuste und universelle Vorgehensweise zu garantieren (Chapman u. a. 1999). Wie in Abbildung 2.2.1 dargestellt,besteht dieses Vorgehensmodell aus sechs Phasen.

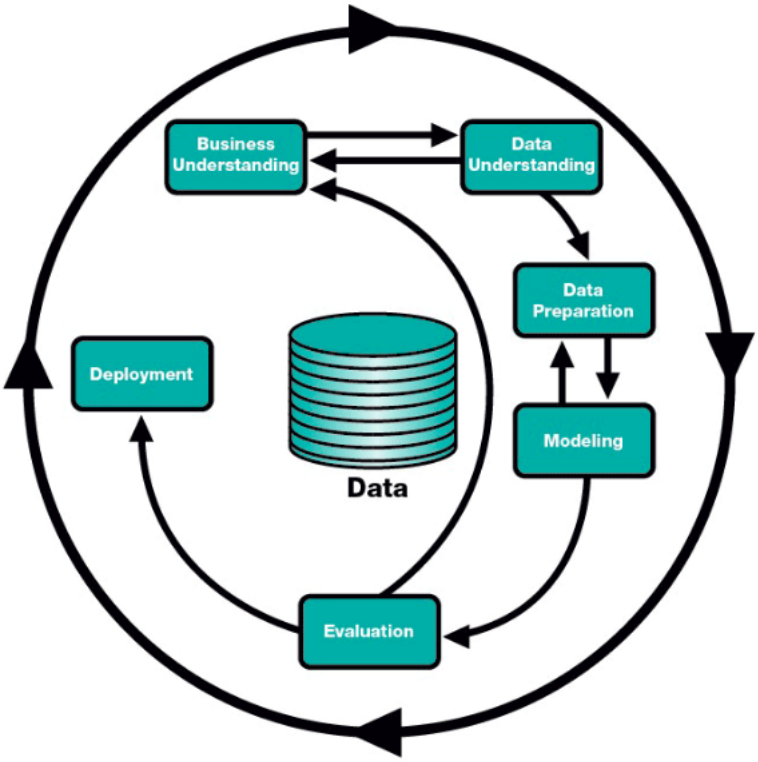


Abbildung 2.2.1.: CRISP-DM Phasen (SmartVisionEurope 2015)

Business Understanding: diese Phase beschäftigt sich mit der Frage nach dem Ziel der Analyse. Dementsprechend, werden die Aufgaben erstellt und ein Plan festgelegt.
Data Understanding: die zweite Phase zielt darauf ab Daten zu sammeln und

durch ein erstes Screening, deren Qualität festzustellen. Wie die Grafik 2.2.1 zeigt, kann dies dazu führen, die Ergebnisse aus der ersten Phase noch einmal anzupassen.
Data Preparation: nachdem Daten gesammelt wurden, gilt es anschließend diese für Analysen auf zubereiten. Hierbei liegt der Fokus darauf, die bestmögliche Konstruktion des finalen Datensatzes für die anschließende Modellierung zu gewinnen. Dazu ist es nötig relevante Daten auszuwählen und die Daten zu bereinigen. Dazu gehört sowohl das Entfernen und Korrigieren von Datenfehlern, als auch das Schätzen fehlender Daten durch Interpolation beispielsweise.
Modeling: diese Phase beschäftigt sich zunächst mit der Erstellung verschiedener Modelle wie zum Beispiel eines Decision Trees oder eines Neuronalen Netzes und der anschließenden Auswahl der adäquatesten Modellierungstechnik. Dazu gehört das kreieren eines Test- und eines Trainingsdatensets, womit verschiedene Modelle getestet werden können. Gegebenenfalls bedarf dies dem Wiederholen der Datenvorbereitung, um ein Datenset nochmals zu justieren.
Evaluation: während dieser Phase wird das Modell welches die in Phase eins definierten Ziele am besten erfüllt, ausgewählt.
Deployment: in der letzten Phase werden die Ergebnisse aufbereitet und präsentiert und zusätzlich in einem Dokument festgehalten (SmartVisionEurope 2015).

Dieses Vorgehensmodell wurde für diese Arbeit ausgewählt, da es ein strukturiertes Vorgehen ermöglicht und dadurch die Qualität der Ergebnisse gesteigert werden kann. Das *Business Understanding* besteht in dieser Arbeit darin, herauszufinden, was der momentane Stand der Forschung bezüglich Machine Learnning im Bereich Cyber Security ist. Anschließend werden bestehende Datensets untersucht, die der späteren Analyse dienen. Um diese Daten anwenden zu können werden diese zunächst in der *Data Preparation* Phase entsprechend aufbereitet. In der nächsten Phase, dem *Modelling* werden diverse Algorithmen auf deren Passgenauigkeit überprüft. Anschließend wird der Algorithmus, welcher die Anforderungen am besten erfüllt, implementiert. Darüber hinaus wird das ganze Vorgehen ausführlich dokumentiert.

3. IOC - Bösartiges Verhalten

Die permanente Steigerung in Größe und Komplexität von Computersystemen, bietet nicht nur einen höheren Nutzen für Kunden, sondern auch mehr Angriffsfläche für Hacker. Dies erschwert die Arbeit von Sicherheitsexperten. Da es darum geht die Kompromittierung eines Systems so früh wie möglich zu erkennen, um potenziellen Schaden zu verhindern, beziehungsweise diesen so gering wie möglich zu halten, arbeiten Experten gegen die Zeit.
Wurde ein System Opfer eines Angriffs, gilt es dieses forensisch zu untersuchen. Normalerweise hinterlässt ein Angreifer Spuren seines Einbruchs. Die Aufgabe der IT-Security ist es, diese zu finden. Diese Hinterlassenschaften werden als *Indicator of Compromise (IoC)* bezeichnet, also Indikatoren, welche darauf hindeuten, dass ein System kompromittiert wurde.
IoCs müssen jedoch differenziert betrachtet werden. Es gibt eindeutige Indikatoren, welche kaum einen Zweifel daran lassen, dass ein System kompromittiert wurde. Angenommen ein Sicherheitsexperte findet Schadsoftware auf einem System und stellt gleichzeitig fest, dass es zu einem Datenupload auf einen nicht identifizierbaren Server kam, welcher von der Malware initiiert wurde. In diesem Fall kann davon ausgegangen werden, dass das System tatsächlich kompromittiert wurde. Der Indikator bildet sich hierbei aus den beiden Indizien: Malware und unautorisierter Upload.
Des Weiteren gibt es Indikatoren welche nicht eindeutig sind. Dieses Phänomen verdeutlicht das folgende Beispiel: angenommen, auf einer Maschine werden Prozesse erkannt, welche nicht von dieser selbst gestartet wurden, sondern durch remote gesendete Befehle. Diese können durch das Windows Tool `PsExec` übermittelt worden sein. Mit Hilfe dessen, lassen sich administrative Tätigkeiten, wie beispielsweise Systemupdates oder Passwort Änderungen, anhand von Remote-Befehlen durchführen. So vorteilhaft dieses Tool in den richtigen Händen erscheint, so gefährlich ist es in den falschen. Angreifer können `PsExec` für bösartige Zwecke missbrauchen. Zwar verlangt der Remote-Zugriff eine IP-Adresse mit korrespondierenden Benutzerinformationen, diese können jedoch durch andere Arten von Angriffen beschaffen werden. Da es sich bei `PsExec` um ein legitimiertes Tool zur Systemkoordination handelt, wird es von Anti-Viren Programmen nicht erkannt. Dadurch wird die Entdeckung eines Missbrauchs deutlich erschwert. Da die Benutzerinformationen allerdings unverschlüsselt übertragen werden, können immerhin diese über Tools

wie `Wireshark` oder `Tcpdump` abgefangen werden. Der Nachweis über die Nutzung dieses Tools allein reicht also nicht aus, um eine Kompromittierung annehmen zu können.

Bei der Malware spezifischen Analyse gilt es zunächst herauszufinden, was genau passiert ist und welches Schadprogramm für den Angriff verantwortlich ist. Traditionelle Anti-Virus Programme arbeiten basierend auf Datenbanken, in welchen sie bereits bekannte Signaturen und Heuristiken anwenden, um Malware zu identifizieren. Das Problem hierbei ist, dass es für Angreifer ein leichtes ist, ihren Code zu modifizieren, um die Signatur zu verändern, wodurch das Schadprogramm nicht mehr als solches erkannt wird. Verschleierungstaktiken wie diese, lassen sich in drei Gruppen einteilen (P. He u. a. 2017):

- **Packing** Dies Bezeichnet die Technik exekutierbare Dateien zu komprimieren. Um die komprimierte Malware zu erkennen muss diese zunächst entpackt werden. Gleichzeitig ist dies aber auch ein guter IoC, da ausführbare Dateien im Regelfall nicht komprimiert vorliegen.
- **Metamorphismus** Hierbei wird die Erkennung erschwert in dem der Binärcode mutiert wird. Das bedeutet, die Sequenz der Opcodes wird bei jeder Ausführung geändert.
- **Polymorphismus** Eine polymorphe Schadsoftware generiert bei jeder Ausführung eine weitere Version der Malware, sodass eine große Anzahl an divergierender Signaturen für dasselbe Programm entstehen.

Diese Techniken erschweren das Erkennen von Malware anhand gängiger Anti-Virus Programme deutlich. Zukünftig kann Machine Learning hierbei eine große Rolle spielen. Denn wie Han u. a. (2019) bereits erfolgreich untersuchten, ist auf MLAs basierende Erkennungssoftware in der Lage, Malware trotz dieser Verschleierungstechniken zu konstatieren.

Das Erkennen von Malware basiert im Regelfall auf der Untersuchung von Portable Executable Dateien (PE-Dateien). Diese beinhalten ausführbare Daten im Binärformat. Dazu gehören Windows `.exe` Dateien, Objektcode und Dynamic Link Libraries (DLLs). Eine PE-Datei ist folgendermaßen aufgebaut:

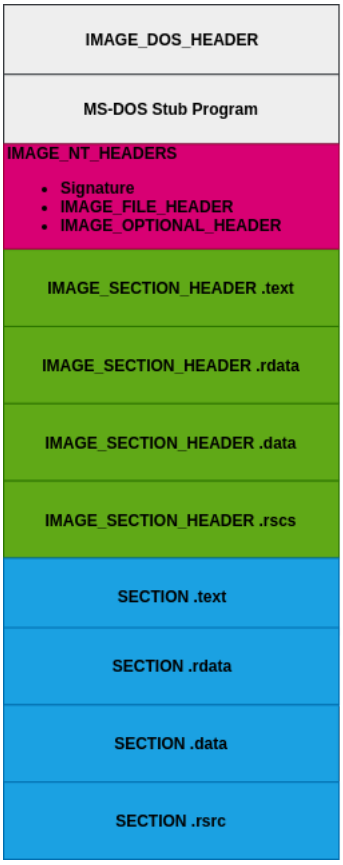


Abbildung 3.0.1.: Aufbau einer PE-Datei (eigene Darstellung)

Die in Abbildung 4.3.1 grau hinterlegten Bereiche sind für die Analyse von PE-Dateien irrelevant, sie dienen unter anderem lediglich dazu, eine Fehlermeldung auszugeben, falls eine `.exe` Datei in einem Betriebssystem ausgeführt werden soll, mit welchem diese nicht kompatibel ist.
Der Bereich `IMAGE_NT_HEADERS` bietet bereits Informationen für eine Analyse. `IMAGE_FILE_HEADER` beinhaltet grundlegende Informationen bezüglich der Datei. Beispielsweise wann diese ausgeführt wurde, was einer Analyse sehr nützlich sein kann. Der Sektor `IMAGE_OPTIONAL_HEADER` ist entgegen dem was der Name vermuten lässt nicht *optional*. Hier werden wichtige Informationen wie der Programmeinstiegspunkt, die Stackgröße zu Beginn sowie die Verwendung eines Graphical User Interface (GUI) (dt. *Grafische Benutzeroberfläche*) oder einer Konsole definiert.
Die grün hinterlegten `IMAGE_SECTION_HEADER` bieten die interessantesten Informationen für eine Analyse. Diese *Header* werden vom Compiler generiert und benannt, sodass der Benutzer wenig Kontrolle über die Namen hat. Dementsprechend konsistent ist die Benennung im Regelfall. Im PE-Header finden sich also relevante Informationen wie Imports, Exports, die Namen der verschiedenen Bereiche (blau

hinterlegt), sowie deren Speichergröße auf der Festplatte und im Random-Access Memory (RAM), sowie die Ressourcen welche von einem Programm benötigt werden.

Grundsätzlich gibt es zwei Methoden um eine Malware Analyse durchzuführen: eine statische und eine dynamische.
Die Dynamische Analyse beinhaltet das Ausführen Schadhafter Programme. Dabei wird Malware in einer sicheren Umgebung ausgeführt und so dessen Verhalten analysiert. Dadurch kann im Gegensatz zur statischen Analyse die tatsächliche Verhaltensweise einer Datei untersucht werden, denn nicht jeder String der in einer Binärdatei gefunden wird, muss zwangsläufig ausgeführt werden. Des Weiteren können Logdateien analysiert werden, welche erst durch das Ausführen eines Programms entstehen. Dynamische Analysen werden im Regelfall in einer *Sandbox*, also in einem isolierten Bereich durchgeführt, wodurch kein Schaden an der Umgebung genommen wird. Der Nachteil dieser Analyse besteht darin, dass die Malware die virtuelle Umgebung erkennen kann und sich somit stoppt. Des Weiteren können von der Schadsoftware benötigte Registry Keys oder Dateien in der virtuellen Umgebung fehlen, sodass deren Verhalten nicht korrekt aufgezeichnet werden kann.

Eine weitere Methode zur Malware Untersuchung bietet eine statische Analyse. Hierbei werden PE-Dateien erforscht. Zunächst durchläuft potenzielle Malware diverse Virenscanner, um die Entdeckung einer bösartigen Signatur zu erhöhen.
Des Weiteren kann *Hashing* zum Einsatz kommen. Dabei wird ein eindeutiger *hash* generiert, welcher verwendet werden kann, um zu recherchieren, ob dieser bereits von anderen Antivirus-Dienstleistern analysiert wurde. Hashing bietet zudem den Vorteil, dass die Datei selbst noch nicht geteilt werden muss. Des Weiteren ist der Austausch eines Hashes auch um einiges schneller als der Upload einer Datei.
Programme verwenden Strings beispielsweise um sich mit einer URL verbinden zu können oder um Textausgaben zu drucken. Diese können einen guten Einblick über das Verhalten einzelner Programme liefern. Beispielsweise können dadurch IP Adressen für Command and Control (C2)-Systeme identifiziert werden, welche von Angreifern zum Verwalten von Remotesitzungen von infizierten Hosts verwendet werden.
Handelt es sich bei Dateien um sogenannte *Packer*, also komprimierte Programme, müssen diese zunächst entpackt werden, um eine erfolgreiche Analyse durchführen zu können. Bei Dateien die relativ wenige Strings enthalten, handelt es sich meist um Packer. Wie die nachfolgende Tabelle zeigt, bietet die statische Analyse eine Vielzahl an Indikatoren, welche darauf hinweisen können, dass es sich bei der untersuchten Datei um Schadsoftware handelt.

Ort	Indikator	Verhalten
System	neue/modifizierte Dateien	Veränderung des Dateisystems durch Malware
System	Registry Einträge	Veränderung/Erstellung von Registry Keys
PE Header	wenige Imports	durch Packer komprimierte Dateien, um die Erkennung und Analyse zu erschweren
PE Header - Imports	`SetWindowsHookEx`	empfängt Tastatureingaben (Keylogger)
PE Header - Imports	`RegisterHotKey`	bestimmte Tastenkombination startet Anwendung (Keylogger)
IMAGE_FILE_HEADER	Kompilierungszeit	unsinnige Kompilierungszeit ist verdächtig
Sections	Abweichende Namen	z.B. .srtsa anstatt .data
SECTION .text	divergierende Speichergröße von Virtual Size und Raw Size	Packer extrahiert Code nach .text
SECTION .rsrc	eingebettetes Programm, Treiber	weitere durch Malware gestartete Aktionen

Tabelle 3.1.: Auszug an, durch statische Analysen identifizierte Indikatoren für einen Malware Angriff (In Anlehnung an Sikorski (2012))

Diese Indikatoren aus Tabelle 3.1, sind nur ein kleiner Teil dessen, was bei einer statischen Analyse entdeckt werden kann. Dennoch wird dadurch deutlich, wie hilfreich PE-Dateien im Erkennen von Malware sein können.
Welche Features aus diesen Dateien generiert werden können, um eine aussagekräftige Untersuchung mit Hilfe von MLAs zu generieren, wird in diversen Ansätzen beschrieben, mit welchen sich das folgende Kapitel beschäftigt.

4. Analyseverfahren

Das folgende Kapitel beschäftigt sich mit Machine Learning Verfahren, welche für die Erkennung von Cyber Security Angriffen verwendet werden. Diese Verfahren wurden anhand einer umfangreichen Literaturrecherche ermittelt. Jedes Vorgehen wird auf dessen verwendete Algorithmen sowie der ausgewählten Features untersucht. Des Weiteren wird analysiert welche Evaluationskriterien verwendet wurden um die Effektivität zu messen und zu welchem Ergebnis der jeweilige Ansatz führte. Das Vorgehen hierbei, orientiert sich an einem problembezogenen Ansatz. Dazu wird im Folgenden eine Übersicht darüber generiert, welche Probleme aus der Cyber Security bereits von Machine Learning Algorithmen in Angriff genommen wurden. Dies dient einer übersichtlichen Darstellung der Möglichkeiten und zeigt die Diversität auf, in welcher MLAs die Arbeit von Cyber Security Spezialisten unterstützen und verbessern können.

Zunächst werden Verfahren erläutert, welche bereits wissenschaftlich belegt wurden. Wurden, dem Ansatz entsprechende Adaptionen aus der Industrie identifiziert, wurde der jeweilige Ansatz um den industriellen erweitert.

4.1. Grundbegriffe aus der Cyber Security

Angriffsvektor, Attacken erklären

4.2. Grundbegriffe des maschinellen Lernens

Leistungsindikatoren, Algorithmen erklären

4.3. Abgewandte Ansätze

Für die Recherche wurden alle Verfahren in einem Zeitraum von 2015 bis heute berücksichtigt. Diese Periode wurde gewählt, da sich die Zahl der Cyberangriffe,

sowie die zur Verfügung stehende Schadsoftware bereits innerhalb weniger Jahre deutlich vermehrt beziehungsweise verändert. Somit soll verhindert werden veraltete Angriffsvektoren zu analysieren, welche bereits von moderneren überholt wurden. Zusätzlich gibt es bereits vergleichbare Arbeiten aus dem Jahr 2016 (s. Buczak und Guven 2016), in welchem Machine Learning Ansätze vor dieser Zeit analysiert werden.
In der folgenden Auflistung wird die Bezeichnung „Erkennung“für binäre Klassifikation verwendet. Beispielsweise, wenn sich ein Ansatz darauf beschränkt Daten entweder in die Rubrik A „bösartig“oder in die Rubrik B „gutartig“zu klassifizieren. Erfolgt in einem Analyseverfahren eine Einteilung in mehrere Klassen (mehr als zwei), wird nachfolgend der Begriff „Klassifizierung“verwendet.

4.3.1. Erkennung von Malware - Hybride Analyse (2015)

Im Jahr 2015 haben Shijo und Salim (2015) einen, auf zwei Analysen basierenden, Ansatz gewählt, um Malware zu erkennen. Dabei vermischten sie die statische Analyse mit der dynamischen, um so einen hybriden Ansatz zu erreichen. Zum einen verwendeten sie ein statisches Analyseverfahren bei dem sie *Printable Strings*, also nicht kodierte Zeichenfolgen wie z. B. *FindFirstFile* aus Binärdateien extrahierten. Zum anderen konfigurierten sie eine Cuckoo Sandbox, in der sie Schadsoftware ausführten und deren API Aufrufe in einer Logdatei speicherten.
Sie untersuchten die Ähnlichkeit in API-Aufrufsequenzen anhand von n-Grammbasierter Ähnlichkeitsmessung. Als Features dienten Tri- und Tetragramme ab einer gewissen Häufigkeit, sowie PrintableStrings ab einer Häufigkeit von zwei. Für die Klassifizierung wurden die Algorithmen Random Forest (RF) und Support Vector Machine (SVM) verwendet. Es wurden jeweils beide Ansätze separat, sowie in Kombination getestet. Analysen mit SVM erzielten eine Genauigkeit von 95.88 % für die statische Analyse und 97.16 % für die dynamische Analyse und waren somit erfolgreicher, als Untersuchungen mit Random Forest. Die besten Ergebnisse erzielte der hybride Ansatz mit SVM mit einer Genauigkeit von 98.71 % und der geringsten False Positive Rate (FPR) von 0.026.
Die Forschung von Shijo und Salim (2015) zeigt also, dass mit den von ihnen gewählte Features, mit einem hybriden Ansatz, deutlich genauere Aussagen, als mit rein statischen oder rein dynamischen Analysen, getroffen werden können.

4.3.2. Erkennung von Malware - Statische Analyse (2016)

More und Gaikwad (2016) untersuchten EXE-Dateien auf Schadsoftware. Dazu konvertierten sie die Dateien zunächst in Operation Code (Opcode), also in den Teil der Maschinensprachanweisung der die auszuführenden Operationen angibt, z.B. `55 8B EC 83 EC 5C 83 7D 0C 0F 74 2B 83 7D 0C 46`. Das ausgewählte Feature Datenset wurde anschließend nochmals zu einer Attribute-Relation File Format (ARFF) Datei konvertiert, um die Datei nachfolgend mit der Machine Learning Software Weka bearbeiten zu können. In Weka wurden die Algorithmen JRip, C4.5 und IBk verwendet. Wobei es sich bei JRip und C4.5 um Decision Tree (DT) und bei IBk um k-nearest-neighbor (k-NN) Implementierungen handelt. Um die Erkennungsgenauigkeit zu erhöhen, wurden nicht nur die einzelnen Algorithmen, sondern ein Klassifikatorensemble angewandt, um Methoden wie Mehrheitsvoting, Veto-Voting und vertrauensbasiertes Veto-Voting verwenden zu können. Ersteres folgt demokratischen Regeln, das heißt, die Klasse mit den meisten Stimmen ist das Ergebnis. Veto-Voting hingehen basiert auf Annahmen über die Wahl der anderen Algorithmen. Vertrauensbasiertes Veto-Voting ergänzt voriges Voting um eine Vertrauensberechnung, wodurch jedem Algorithmus ein bestimmtes Vertrauensniveau zugeteilt wird. Weiterführende Informationen bezüglich dieser Methoden können Shahzad und Lavesson (2013) entnommen werden.
More und Gaikwad (2016) konnten zeigen, dass durch die Verwendung von Veto-Voting eine Genauigkeit von 80.7 % erzielt werden kann. Im Vergleich dazu, lag das beste Ergebnis, welches durch singulären Algorithmeneinsatz von IBk erzielt wurde, bei einer Genauigkeit von nur 73.5 %.

Dieses Ergebnis stützt die These von Shijo und Salim (2015) aus dem Jahr zuvor, welche zeigten, dass ein nicht-hybrider Ansatz weniger genau ist, als einer, der die statische und die dynamische miteinander verknüpft.

4.3.3. Erkennung bösartiger XML-basierter Office Dokumente(2016)

Durch das neue Dateiformat, welches Microsoft 2007 auf den Markt gebracht hat, sollten Sicherheitslücken geschlossen werden. Das Binärformat wurde durch ein XML basiertes Dateiformat ersetzt. Dadurch werden neue digitale Funktionen unterstützt, sowie vertrauensbasierte Bereiche geschaffen, welche das Format weniger riskikoreich gestalten sollen. Dennoch können Attacken gegen XML-basierte Office Dokumente gestartet werden. Zu den möglichen Angriffsvektoren zählen beispielsweise macrobasierte Attacken. Durch den Missbrauch von Visual Basic for

Applications (VBA) kann die zugehörige shell gestartet werden, um willkürliche Kommandos zu senden. Außerdem können externe Bibliotheken sowie Programme aufgerufen werden, welche Schaden verursachen können. Eine weitere Bedrohung durch den Gebrauch von Macros bildet die Fähigkeit dieser, bösartige Dateien aus dem Internet herunterzuladen.
Cohen u. a. (2016) haben in ihrer Arbeit diese Art von Angriffsvektoren untersucht, um eine Infizierung durch bösartige Macros frühzeitig zu erkennen. Dazu haben sie zunächst Office Dokumente in eine Liste von Pfaden konvertiert. Diese dienen der Analyse als Features.

Feature	Structural path	Category
f1	*word\vbaProject.bin*	*Macro*
f2	*word_rels\vbaProject.bin.rels*	*Macro*
f3	*word_rels\vbaProject.bin.rels\Relationships*	*Macro*
f4	*word_rels\vbaProject.bin.rels\Relationships\Relationship*	*Macro*
f5	*word\vbaData.xml*	*Macro*
f6	*word\vbaData.xml\wne:vbaSuppData*	*Macro*
f7	*word\vbaData.xml\wne:vbaSuppData\wne:mcds*	*Macro*
f8	*word\vbaData.xml\wne:vbaSuppData\wne:mcds\wne:mcd*	*Macro*
f9	*word\embeddings*	*Embedded*
f10	*word\embeddings\oleObject1.bin*	*OLE*
f11	*word\vbaData.xml\wne:vbaSuppData\wne:docEvents*	*Macro*
f12	*word\vbaData.xml\wne:vbaSuppData\wne:docEvents\wne:eventDocOpen*	*Macro*
f13	*word\media\image1.emf*	*EMF*

Abbildung 4.3.1.: Features als Pfade dargestellt (Cohen u. a. 2016)

Dadurch wurde jedoch eine so hohe Anzahl an Features generiert, dass die Untersuchung mit verschiedenen Datensets durchgeführt wurde, welche Top Features von 10 bis 2000 beinhalteten. Um die Feature-Repräsentation, also das Vorhandensein bzw. die Wichtigkeit von Features zu bestimmen, wurden zwei Verfahren angewandt. Zum einen ein binäres Verfahren, welches lediglich die Ab-, respektive Anwesenheit eines Features misst und zum anderen wurde das statistische Verfahren Term Frequency - Inverse Document Frequency (TF-IDF) verwendet, um die Wichtigkeit eines Terms in Bezug auf ein Dokument zu bestimmen. Anschließend wurden die Daten mit folgenden Algorithmen untersucht: J48, RF, Logistic Regression (LR), Naïve Bayes (NB), Bayesian Network (BN), LogitBoost (LB), Sequential Minimal Optimization (SMO), Bagging und AdaBoost (AB).

Wie die Ergebnisse zeigen, erzielt das Datenset mit den Top 200 Features, welches mit Random Forest analysiert wurde die besten Werte mit einem F-measure von 0.66. Wie sich demonstrieren ließ, ist es möglich bösartige Office Dokumente durch eine Analyse deren Pfade zu erkennen. Die Untersuchung beschränkt sich jedoch auf direkte Gefahren innerhalb von Dokumenten. Indirekte Gefahren, wie

etwa die durch weiterführende Links, wurden in dieser Forschungsarbeit nicht berücksichtigt.

4.3.4. Erkennung bösartiger HTTP-Anfragen (2016)

Pham, Hoang und Vu (2016) haben in ihrer Arbeit ein Intrusion Detection System (IDS) für HTTP - Anfragen entwickelt. Dazu nutzen sie ein von ihnen entwickeltes Modul, um Netzwerkpakete zu erfassen. Dieses Modul basiert auf derselben Erfassungstechnik die `Wireshark` verwendet. Um ein geeignetes Model zu zu evaluieren, welches es ermöglicht die Pakete in Echtzeit zu klassifizieren, wurden diverse MLAs anhand des CSIC 2010 HTTP Datensets von Carmen Torrano Giménez, Alejandro Pérez Villegas (2010) getestet. Dieses besteht aus 223585 Daten die entweder als *normal* oder *anomal* gelabelt sind. Es enthält Attacken wie SQL-Injections, Buffer Overflow und Cross-Site Scripting (XSS). Pham, Hoang und Vu (2016) trainierten und testeten die Daten mit den Algorithmen Decision Tree, Random Forest, AdaBoost, Logistic Regression und dem SGDClassifier. Als Evaluationsmethode wurden Precision, Recall und F-measure verwendet. Den höchsten dieser Werte erzielte die Logistic Regression Methode mit einem F1-measure von 0.96 for abnormen und 95.82 für normalen Verkehr. Zukünftig sollen diese Ergebnisse an Paketen, welche durch das eigens entwickelte Modul der Forscher erfasst wurden, evaluiert werden.

4.3.5. Klassifizierung von Netzwerkattacken (2017)

Yin u. a. (2017) implementierten ein IDS, für welches sie Recurrent Neural Networks (RNNs) verwendeten. Zusätzlich wurde die Leistung des Models bei binärer als auch bei Multi-Klassen Klassifikation untersucht. Um die Effizienz zu prüfen, wurde ferner ein Vergleich mit diversen MLAs gezogen. Die Analyse basiert auf dem NSL-KDD Datenset aus dem Jahr 2009. Dieses beinhaltet neben dem normalen Netzwerkverkehr, Daten zu vier verschiedenen Angriffstypen die da wären: Denial of Service (DoS), Remote to Local (R2L), User to Root (U2R) und Probe Attacken. Das Datenset besteht aus 41 Features. Um Vergleiche mit anderen MLAs herzustellen, wurden parallel Experimente mit Artificial Neural Network (ANN), RF, NB, Multi-Layer Perceptron (MLP), SVM, J48, Random Tree und Naïve Bayesian Tree (NBTree) für die binäre Klassifikation durchgeführt. Auf dieselbe Weise wurde die Multi-Klassen Klassifikation überprüft. Als Qualitätsmerkmal der Ergebnisse wurde der Wert *Accuracy* gewählt, welcher die Genauigkeit der Analyse wider spiegelt. Dieser Wert basiert auf dem Prozentsatz der korrekt klassifizierten Daten im Vergleich zur Gesamtheit der Daten. Das Ergebnis zeigt, dass RNNs eine qualitativ hochwertigere

Analyse produzieren als die zu verglichenen MLAs. Bei der binären Klassifikation des Testsets erzielte auf RNNs basierende Untersuchungen eine Genauigkeit von 83.28% gefolgt von NBTree mit 82.02%. Bei der Klassifizierung in fünf Klassen erreichte das Recurrent Neural Network (RNN) mit 81.29% ebenfalls ein besseres Ergebnis als die restlichen Algorithmen. Jedoch fanden Yin u. a. (2017) heraus, dass RNNs deutlich mehr Zeit für das Training beanspruchen, dieses Problem soll zukünftig durch Nutzung der Graphics Processing Unit (GPU) beschleunigt werden.

4.3.6. Erkennung von Ransomware - Dynamische Analyse (2017)

Maniath u. a. (2017) haben eine dynamische Analyse entwickelt um *Ransomware* anhand von API Aufrufen zu klassifizieren. Ransomware beschreibt eine Art Schadprogramm, welche dem Benutzer Ressourcen entzieht und eine Lösegeldsumme verlangt um diese wieder verfügbar zu machen. Um die Aufrufe zu extrahieren verwendeten die Forscher die dafür entwickelte Umgebung von Cuckoo Sandbox. Dadurch konnten die Schadprogramme in einer Umgebung ausgeführt werden ohne Schaden zu erzeugen. Die Anwendung erfasst alle API Aufrufe und speichert diese in einer `.json` Datei. Für die Analyse wurden 157 schadhafte Dateien aus nicht näher beschriebenen Onlinequellen verwendet. Dabei konnten 239 Aufrufe extrahiert werden, welche der Untersuchung als Features dienten. Um eine einheitliche Länge dieser zu gewährleisten, wurden die einzelnen Aufrufe entsprechend der längsten Sequenz mit Nullen aufgefüllt. Anschließend wurden die Daten entweder mit 0 für *gutartig* oder mit 1 für *ransomware* gelabelt. Anhand dieser Daten wurde ein Long-Short Term Memory (LSTM) Netzwerk trainiert. Nach einer Trainingszeit von zwei Stunden erreichte das Model eine Genauigkeit von 96.67% bei der Analyse der Testdaten. Die komplette Prozess einschließlich der Gewinnung der API Sequenzen dauerte allerdings 56 Stunden, was in Anbetracht der geringen Menge an Datensätzen doch sehr nachteilig ist.

4.3.7. Erkennung von Malware - Imageanalyse (2017)

Eine potenziell schnellere Methode um Malware zu erkennen, entwickelten Choi u. a. (2017) anhand einer Image Analyse. Dazu generieren sie Bilder von ausführbaren Dateien. Dabei hat jedes Pixel einen Wert zwischen 0 und 255. Um aus einer Datei ein Bild zu erhalten, wird jedes Byte eingelesen und zu einer Ganzzahl konvertiert die einem Pixel entspricht. Dadurch entstehen 256x256 Bilder. Da dadurch während einer Analyse der Speicher zur Neige geht wurden die Daten auf 32x32 reduziert.

Die dadurch generierten Bilder dienen der Analyse mit einem Convolutional neural network (CNN) als Features. Die 2000 dafür verwendeten Schadprogramme, stammen aus einem koreanischen Cyber Security Forschungszentrum. Als Metrik zur Überprüfung der Genauigkeit wurde hier ebenfalls die Genauigkeit gewählt, welche sich auf 95.66% beläuft.
Bedauerlicherweise haben Choi u. a. (2017) nicht erwähnt in welcher Geschwindigkeit sie ihre Analyse durchführen konnten. In Anbetracht der von ihnen aufgestellten These - Imageanalysen seien viel schneller als statische und dynamische Analysen - wäre dies ein interessantes Detail gewesen.

4.3.8. Erkennung böswilliger MS Office Dateien (2017)

Bearden und Lo (2017) konzentrierten sich in ihrer Forschung darauf eine Methode zu entwickeln, mit welcher sich Microsoft Office Dokumente anhand ihrer Macros nach *gutartig* und *bösartig* klassifizieren lassen. Dazu untersuchten sie 158 Dateien, von welchen 40 schadhaft waren. Dokumente die Macros enthalten, beinhalten nicht nur VBA Code sondern auch *p-code*. Dabei handelt es sich um Assembler-Code, welcher vom VBA-Interpreter generiert wird, nachdem der Code einmal ausgeführt wurde (Bearden und Lo 2017). Diese Codes dienten der Analyse mit k-NN als Features. Die Effizienz wurde, wie die beiden zuletzt erläuterten Ansätze, anhand der Genauigkeit gemessen. Es wurden verschiedene Experimente mit unterschiedlichen K und L durchgeführt, wobei K die Anzahl an Clustern und L die Anzahl der besten Features impliziert. Den besten Wert erzielte eine Kombination mit K=3 und den Top 75 Features mit einer Genauigkeit von 96.3%.
Es wurde ausschließlich ein Algorithmus getestet, da die Intention der Forschung darin bestand , einen *Proof of concept* für die Erkennung bösartiger Macros anhand von p-codes bereitzustellen. Der Vorteil den Bearden und Lo (2017) mit ihrer Forschung geschaffen haben, besteht darin, dass potenziell bösartige Dateien nicht geöffnet werden müssen, bevor sie analysiert werden können. Jedoch weisen auch sie auf den Mangel an adäquaten Trainingsdaten hin, welchen es zukünftig zu beheben gilt.

4.3.9. Klassifizierung von DDoS Attacken (2017)

Bereits seit den 80er Jahren gibt es DoS Attacken, welche Netzwerk Ressourcen erschöpfen und dadurch die Verfügbarkeit von Services blockieren. Es gibt zwei Arten diese Attacken zu erkennen: Zielseitige Verteidigung und Quellenseitige Verteidigung(Z. He, Zhang und Lee 2017). Die Zielseitige Erkennung hat den Nachteil, dass Attacken erst entdeckt werden nachdem sie bereits beim Opfersystem

ankamen. Z. He, Zhang und Lee (2017) forschen an einem pro aktiven Ansatz, bei dem die Erkennung auf der Quellenseite erfolgen soll, wodurch Angriffe auf mehrere Systeme verhindert werden können. Dabei konzentrieren sie sich auf vier gängige DDoS Angriffe aus der Cloud: Secure Shell (SSH) brute-force Attacken, Internet Control Message Protocol (ICMP) flooding Attacken, Domain Name System (DNS) reflection Attacken und Transmission Control Protocol (TCP) synchronize (SYN) Attacken. Als Feature für SSH brute-force Attacken dient die Anzahl an Diffie-Hellman Schlüsselaustausche, da dieser Wert während einer solchen Attacke steigt (Z. He, Zhang und Lee 2017). Als Feature für DNS reflection Attacken wurde das Verhältnis von eingehenden und ausgehenden DNS-Paketen gewählt, da bei einer Attacke mehr Anfragen als Antworten gesendet werden. Die ICMP Paketrate wurde als Feature für ICMP flooding Attacken gewählt, da bei normalem Verkehr eine geringere Anzahl dieser Pakete vorhanden ist. Um actcp SYN Attacken zu identifizieren, wurde das SYN/acknowledge (ACK) Verhältnis als Feature gewählt, da während einer solchen Attacke mehr SYN Tags als ACK Tags in den Paketen zu finden sind. Für das Experiment wurde der Netzwerkverkehr 9 Stunden lang verfolgt und anschließend mit den Algorithmen LR, SVM, DT, NB, RF, k-means und Gaussian-Mixture Model for Expectation-Maximization (GMM-EM) klassifiziert. Zur Evaluation dienten die Metriken: Precision, Genauigkeit, Recall und F1-measure. Das beste Ergebnis mit einem F1-measure von 0.9975 und einer Genauigkeit von 99.73% konnte mit SVM erzielt werden. Durch diesen vielversprechenden Ansatz sollen zukünftig weitere DDoS Attacken pro aktiv erkannt werden.

4.3.10. Schwachstellen Scanner für Web Applikationen (2017)

Um Schwachstellen in Web Applikationen zu finden, wurde bei diesem Ansatz ein System namens Bug Terminating Bot (BTB) entwickelt (Robin Tommy, Gullapudi Sundeep 2017). Dieser Schwachstellenscanner überprüft Websites auf potenzielle Angriffsvektoren und liefert gleichzeitig Lösungen, um diese zu beheben. Zunächst überträgt der Bot alle Seiten einer Webapplikation und sucht innerhalb dieser nach ausnutzbaren Schwachstellen. Das Überprüfen basiert auf dem Ausführen von Payloads für gefundene Konflikte. Beispielsweise können Seiten auf SQL-Injections und XSS Attacken überprüft werden, in dem adäquate Payloads ausgeführt werden. Anschließend werden Code Vorschläge geliefert, welche die gefundenen Schwachstellen schließen sollen. Das Ergebnis des Scans, sowie die Verbesserungsvorschläge basieren auf Machine Learning. Nach dem Scan werden die Daten an einen zentralisierten Server geschickt, auf welchem eine SVM Informationen zu Schwachstellen analysiert und entsprechende Vorschläge für Payloads und gleichzeitig verfügbare Patches liefert. Um die Effizienz dieses Systems zu messen, wurde ein Leistungsfaktor E berechnet. Dieser setzt sich aus der benötigten Zeit für den ersten Scan sowie

dem des letzten Scans zusammen. Wie Experimente zeigen, nimmt die Dauer der Scans mit BTB ab, während Überprüfungen mit Scannern ohne Machine Learning Komponenten in der Dauer konstant bleiben.
Mit dieser Forschung wurde gezeigt, dass durch Machine Learning schnellere Ergebnisse bereitgestellt werden können.

4.3.11. Erkennung von Malware anhand von PE-Header (2017)

Raff, Sylvester und Nicholas (2017) verzichten in ihrem Ansatz auf explizite Feature Konstruktionen und zeigen, dass Malware anhand von Bytes von Neuronalen Netzen erkannt werden kann. Für diese Analyse untersuchten sie zwei verschiedene Ansätze, ein Fully Connected Neural Network ((FC) Neural Network) und ein RNN, bei welchem sie sich für das LSTM Model entschieden. Gerade deshalb war es nötig sich bei der Analyse auf den Header der PE-Dateien zu konzentrieren, da LSTM Modelle für die Berechnung aller Daten enorm viel Zeit und Ressourcen in Anspruch nehmen würden (Raff, Sylvester und Nicholas 2017). Die dabei verwendeten Features bestehen aus 328 geordneten Bytes. Zusätzlich wurden Modelle entwickelt um Vergleiche mit den Ergebnissen der Neuronalen Netze anzustellen. Dazu gehören Extra Random Trees (ET), RF und LR. Um die Vorhersagen zu evaluieren wurden die Metriken Genauigkeit sowie Area Under the Curve (AUC) verwendet. Entsprechende Daten sammelten die Forscher sowohl bei *VirusShare* als auch bei *OpenMalware*. Wie die Ergebnisse zeigen liegt der vielversprechendste Ansatz in (FC) Neural Network mit einer Genauigkeit von 89.9% gefolgt von dem RNN LSTM mit 79.7%.
Durch diese Forschung zeigen die Autoren das Potenzial Neuronaler Netze im Erkennen von Schadsoftware lediglich anhand von rohen Bytes. Dies impliziert nicht nur eine enorme Zeit sondern auch eine bemerkenswerte Ressourcen Einsparung.

4.3.12. Erkennung von Malware anhand von PE-Header mit erweitertem Feature-Set (2017)

Im Gegensatz zu dem zuvor erläuterten Ansatz von Raff, Sylvester und Nicholas (2017), verwenden Kumar, Kuppusamy und Aghila (2017) ein erweitertes Feature Set für ihre Analyse. Dieses setzt sich aus *rohen* Features, wie beispielsweise der Anzahl der `FILE_HEADER`-Abschnitte, und *abgeleiteten* Features zusammen. Bei Letzterem handelt es sich um Werte, die durch den Abgleich von Feature-Werten mit vorab definierten Regeln entstehen. Beispielsweise könnte der Wert der Kompilierungszeit, bei welchem es sich um eine Ganzzahl, die die vergangene Zeit seit 1969 in Sekunden angibt, handelt, allein wenig Aussagekraft besitzen (Kumar,

Kuppusamy und Aghila 2017). Deswegen wird die Zahl in ein Datumsformat konvertiert und mit einem bestimmten Gültigkeitsbereich verglichen. Dadurch ergibt sich ein boolescher Wert, welcher in die Analyse mit einfließt. Kumar, Kuppusamy und Aghila (2017) verwenden insgesamt 11 abgeleitete und 55 rohe Features. Für ihre Analyse sammelten die Autoren bösartige Dateien bei *VirusShare* und *download.cnet*. Es wurden Experimente anhand rohen Feature Sets als auch anhand eines integrierten Feature Sets (welches aus rohen und abgeleiteten Features besteht) mit LR, Linear Discriminant Analysis (LDA), RF, DT, NB und k-NN durchgeführt. Als Evaluationsmetriken wurden Genauigkeit, Precision, Recall und der F1-measure verwendet. Bis auf k-NN erzielten alle Algorithmen ein besseres Ergebnis, wenn das integrierte Feature Set verwendet wurde. Den besten Wert erreicht RF mit einer Genauigkeit von 89.23% und einem F1-measure von 0.90.
Diese Ergebnisse erzielen somit eine höhere Genauigkeit als die der Untersuchungen von Raff, Sylvester und Nicholas (2017), jedoch ist eine weitaus intensivere Vorarbeit zu leisten.

4.3.13. Erkennung von Exfiltration und C&CTunnels (2017)

Das u. a. (2017) entwickelten ein auf Machine Learning basierendes System um die Exfiltration von Daten eines kompromittierten Systems, sowie den Aufbau eines C&C-Servers zu erkennen. Um Daten zu exfiltrieren kann Schadsoftware DNS Abfragen nutzen. Dazu kodiert und/oder komprimiert sie die zu versendenden Informationen zunächst. Anschließend kann die daraus entstandene Zeichenkette als Subdomain einer DNS Abfrage angehängt werden.
Über DNS TXT-Records können Texte anstatt einer IP Adresse an einen Benutzer gesendet werden. Angreifer können diese Funktion nutzen, um einen Tunnel zum Senden von Anweisungen oder zum Öffnen einer Sitzung einzurichten. Bei der Exfiltration gehen die Autoren davon aus, dass eine kodierte Subdomain ein Indiz hierfür ist, weshalb diese Zeichenkette analysiert wird. Zunächst generieren sie aus dieser acht Features wie beispielsweise dem Verhältnis der Anzahl an Ziffern zum Rest der Elemente. Das von Das u. a. (2017) entworfene Model basiert auf Logistic Regression und erreicht einen F1-measure von 0.96.
Für die Klassifizierung von TXT-Records wählten sie einen *unsupervised learning* Ansatz, d.h. das Modell arbeitete mit einem ungelabelten Datenset. Für die Analyse verwendeten sie zehn Features, unter anderem die Anzahl der Groß- sowie der Kleinbuchstaben, Punkte oder Unterstriche sowie die Anzahl von Ziffern. Dadurch konnten von 2356 bösartigen TXT-Records 2160 von ihrem Modell identifiziert werden.

4.3.14. Erkennung bösartiger PowerShell-Befehle (2018)

Hendler, Kels und Rubin (2018) untersuchten in ihrer Forschung von 2018 bösartige PowerShell Befehle. Dazu analysierten sie ein Datenset welches aus über 66.000 Befehlen bestand. Die hierbei verwendeten bösartigen Befehle stammen von Microsoft-Sicherheitsexperten. Das Datenset musste zunächst vorverarbeitet werden. Dazu wurden beispielsweise kodierte Befehle zunächst dekodiert, Leerzeichen wurden entfernt und Nummern durch ein Sternchen (*) ersetzt. Anschließend wurden die Daten sowohl mit einem CNN als auch mit einem auf Natural Language Processing (NLP) basierenden Detektoren untersucht. Das beste Ergebnis erzielte allerdings ein Ensemble-Detektor, der einen NLP-basierten Klassifikator mit einem CNN-basierten kombiniert. Dieser erzielte eine True Positive Rate (TPR) von 0.92. Da das verwendete Datenset nicht einsehbar ist, bleibt hier unklar, was einen PowerShell Befehl bösartig macht. Da viele Befehle von Administratoren als auch von Angreifern gleichermaßen genutzt werden, wäre dies ein interessantes Detail gewesen.

4.3.15. Klassifizierung von Netzwerkverkehr in 5 Klassen (2018)

Ding und Zhai (2018) wählten den Ansatz eines CNN um ein IDS aufzubauen, da laut ihnen Systeme, welche mit traditionellen MLAs arbeiten nicht explizit und nicht zuverlässig genug sind. Um diese These zu verifizieren verglichen sie die Ergebnisse des CNN mit traditionellen Algorithmen wie RF und SVM und Deep Learning Methoden wie Deep Belief Network (DBN) und LSTM. Wie Yin u. a. (2017), verwendeten sie das NSL-KDD Datenset, welche dieses bereits mit RNN analysierten. Auch Ding und Zhai (2018) wählten als Leistungsindikator die Genauigkeit, sowie die TPR und die FPR. Tatsächlich lag die Genauigkeit des CNN mit 80.13% deutlich über der, der traditionellen als auch der Deep Learning Methoden. Jedoch erzielten Yin u. a. (2017) mit ihrem RNN eine um knapp 2% bessere Analyse bei der Klassifikation des Netzwerkverkehrs in 5 Klassen. Da laut Yin u. a. (2017) RNNs deutlich mehr Trainingszeit benötigen als traditionelle MLAs, wäre es interessant zu vergleichen gewesen, wie sich die Trainingszeit bei dem von Ding und Zhai (2018) entwickelten CNN verhielt. Leider ist dieses Detail in der Arbeit nicht dokumentiert.

4.3.16. Anomalieerkennung anhand von Systemprotokollen (2018)

Computersysteme produzieren täglich Terabytes an Daten, welche unmöglich manuell inspiziert werden können. Dadurch scheint dies ein für MLAs prädestinierter Use Case zu sein. Brown u. a. (2018) untersuchten in ihrer Arbeit die Analysefähigkeit von RNNs in Systemprotokollen. Da die Autoren auf unbeaufsichtigte Lernmethoden setzen, kann auf die zeitaufwändige Beschaffung von gelabelten Daten verzichtet werden. Um das von ihnen entwickelte Modell zu evaluieren, verwendeten sie das Los Alamos National Laboratory (LANL) Datenset, welches Host Ereignisse protokolliert. Dieses besteht aus über einer Milliarde Protokollzeilen, welche jeweils 21 Features beinhalten. Von diesen verwendeten die Autoren lediglich acht für ihre Analyse und reduzierten den Datensatz auf 14 Millionen Protokollzeilen, da sie nur mehr zwei Tage der Aufzeichnungen berücksichtigten. Um die Ergebnisse zu überprüfen verwendeten sie die area under the receiver operating characteristic curve (AUC ROC). Dieser ergab einen AUC Wert von 0.976 und zeigt deutlich das die Analyse mit unbeaufsichtigten RNNs ein hohes Potenziell für die Untersuchung von Logdateien besitzt.

4.3.17. Erkennung von bösartigem Netzwerkverkehr (2018)

Aldwairi, Perera und Novotny (2018) verwendeten in ihrer Arbeit eine Restricted Boltzmann Machine (RBM) um Netzwerkverkehr binär nach *normal* oder *anormal* zu klassifizieren. Im Vergleich zu Neuronalen Netzen die aus *Input*, *hidden* und *output* Schicht bestehen, enthält eine RBM lediglich die erste beiden Schichten. Im Vergleich zu Yin u. a. (2017) und Ding und Zhai (2018), verwenden Aldwairi, Perera und Novotny (2018) nicht das veraltete NSL-KDD Datenset, sondern das ISCX Datenset (Shiravi u. a. 2012), in welchem Netzwerkverkehr aus dem Jahr 2012 aufgezeichnet wurde. Für ihre Analyse verwendeten die Forscher 16 Features die verschiedene Merkmale des Netzwerkverkehrs beschreiben. Um die Leistung der RBM zu evaluieren verwendeten auch sie den Leistungsindikator der Genauigkeit und erzielten dabei einen Wert von 89%. Außerdem vermerkten sie eine TPR von 88.4% und eine False Negative Rate (FNR) von 88.8%. Da Aldwairi, Perera und Novotny (2018) lediglich eine zweischichtige RBM verwendeten, wäre es interessant weitere Experimente mit so genannten Deep Restricted Boltzmann Machines (DRBMs), welche aus mehreren Schichten bestehen, durchzuführen.

4.3.18. Erkennung von Botnetzen (2018)

Botnetze sind eine Sammlung an kompromittierten, miteinander verbundenen Maschinen, die durch einen Master gesteuert werden. Diese Netze können Bedrohungen wie DoS Attacken, Spamming oder Datendiebstahl auslösen. Laut Leonard, Xu und Sandhu (2009) besteht der Lebenszyklus eines Botnetzes aus vier Phasen: Formation, C&C, Attacke, post Attack-Phase. Um diese Art von bösartigem Verhalten zu Analysieren, entwickelten Mathur, Raheja und Ahlawat (2018) ein Modell, welches Botnetze in den ersten beiden dieser Phasen erkennt. Um Attacken so früh wie möglich zu erkennen, setzen die Forscher auf ein zeitsparendes Analyseverfahren bei welchem lediglich der Header von TCP/User Datagram Protocol (UDP)-Paketen untersucht wird. Um entsprechende Daten zu generieren wurde Netzwerkverkehr von Linux als auch von Windows Systemen erfasst. Zusätzlich verwendeten die Forscher das CTU-13 (Garcia u. a. 2014), sowie das ISOT (Victoria 2010) Datenset. Insgesamt wurden 11 Features wie beispielsweise Fließdauer, Ziel und Quell IP Adresse sowie das zu verwendete Protokoll verwendet. Bei der Analyse kamen die Algorithmen LR, Random SubSpace, Randomizable Filtered Classifier, MultiClass Classifier und Random Committee zum Einsatz. Dabei erzielte der MultiClass Classifier eine Genauigkeit von 98.4% sowie eine FPR von lediglich 0.004 in einer Zeit von 0.03s. Somit beweisen Mathur, Raheja und Ahlawat (2018) einen praktikablen Ansatz im Erkennen von Botnetzen, allerdings bemängeln sie die Genauigkeit bei einem exponentiellen Anstieg von Netzwerkverkehrsdaten, sie empfehlen daher für eine größere Menge an Daten neuronale Netze zur Erkennung zu verwenden.

4.3.19. Klassifizierung von Microsoft Malware (2018)

Sabar, Yi und Song (2018) setzen bei der Erkennung von Malware auf SVM. Jedoch optimieren sie deren Konfiguration durch Verwendung von hyper Heuristiken. Dabei handelt es sich um eine Suchmethode welche unter diversen Heuristiken auswählt und diese gegebenenfalls miteinander kombiniert, um eine bestmögliche problemspezifische Konfiguration zu gewährleisten. Die Daten für die Analyse entnahmen sie der Microsoft Kaggle Challenge von 2015 (Kaggle 2015), bei welcher es 500 GB Schadsoftware Dateien in neun verschiedene Klassen zu klassifizieren galt. Zusätzlich verwendeten sie das NSL-KDD Datenset. Um den von ihnen entwickelten Ansatz zu evaluieren, analysierten sie die Datensätze zusätzlich mit den Algorithmen Gaussian Naive Bayes Tree (GNBT), Fuzzy Classifier (FC) und DT. Als Leistungsindikatoren wählten sie die Genauigkeit sowie den *logloss* (dt. Logarithmischer Verlust), welcher die Leistung eines Modells misst, dessen Ausgabe einen Wahrscheinlichkeitswert zwischen 0 und 1 annehmen kann. Dabei steigt der Wert wenn die Annahme von dem tatsächlichen Wert abweicht und sinkt wenn er

mit diesem übereinstimmt. Dementsprechend ist ein Wert welcher gegen 0 strebt wünschenswert. Tatsächlich erzielte der Ansatz von Sabar, Yi und Song (2018) den niedrigsten logloss mit einem Wert von 0.0031. Des Weiteren konnte ebenfalls damit die höchste Genauigkeit von 85.69% erreicht werden. Dadurch konnten die Forscher die Effektivität ihres Ansatzes beweisen.

4.3.20. Klassifizierung von Malware anhand von Datenpaketen (2018)

Mit ihrem Ansatz möchten Yeo u. a. (2018) die Schwächen von portbasierten Ansätzen und deep packet inspection (DPI) kompensieren. Laut ihnen ist der portbasierte Ansatz nicht verlässlich bei unbekannten Ports und DPI (Dharmapurikar u. a. 2003) führt zu einer zu zeit intensiven Analyse, welche für große Datenmengen nicht praktikabel ist. Für ihre Arbeit verwendeten sie das Datenset des Stratosphere IPS Projekts (Garcia u. a. 2014), welches Malware Datenpakete vor, sowie nach einer Infektion eines Windows XP Systems beinhaltet. Die Daten wurden den sechs Klassen *Neris, rbot, Virut, Murlo, NSIS* und *normal* zugeordnet. Insgesamt wurden 35 Features aus den Netzwerkpaketen extrahiert. Dabei wurde beispielsweise die Größe der jeweiligen Pakete sowie die Dauer der Übertragung untersucht. Für die Analyse verwendeten sie ein CNN sowie die Klassifikatoren MLP, SVM und RF. Um die Leistung der jeweiligen Algorithmen zu evaluieren, wurde die Genauigkeit, Precision und Recall verwendet. Dabei wurde deutlich, dass die besten Ergebnisse sowohl durch RF mit einer Genauigkeit von 93% als auch durch Analysen mit einem CNN mit 85%iger Genauigkeit erzielt werden können.
Leider gingen die Forscher nicht auf ihre anfangs aufgestellte These ein und erläuterten nicht, ob ihr Ansatz nun effektiver und zeitsparender ist als die portbasierte bzw die DPI Analyse.

4.3.21. Erkennung von Port-Scans (2018)

Aksu und Ali Aydin (2018) konzentrierten sich in ihrer Arbeit auf die Erkennung von Port-Scans. Dafür verwendeten sie das CICIDS2017 Datenset, welches vom *Canadian Institute for Cyber Security* entwickelt wurde. Dieses besteht aus insgesamt knapp 290.000 Aufnahmen von Netzwerkverkehr, wobei jede über 85 Features wie Quell-IP, Quell Port, Ziel Port, Dauer und weitergeleitete Pakete verfügt. Um Port-Scans zu identifizieren wurden Deep Learning Algorithmen, sowie eine SVM verwendet. Die Leistung wurde anhand von Precision, Recall, Genauigkeit und dem F1-measure evaluiert. Für ihre Analyse verwendeten Aksu und Ali Aydin

(2018) alle Features des Datensets und verzichteten somit auf zusätzliche Feature Auswahl Algorithmen. Dadurch erzielten sie eine Genauigkeit von 69.8% und einem F1-masure von 0.65 mit der SVM. Allerdings konnten sie diese Ergebnisse mit Hilfe von Deep Learning Algorithmen deutlich verbessern, dadurch wurde eine Genauigkeit von 97.80% und ein F1-measure von 0.99 erreicht.

4.3.22. Erkennung von Netzwerkverkehr (2018)

Teoh u. a. (2018) wenden Deep Learning an, um bösartigen Netzwerkverkehr zu identifizieren. Bei ihrer Analyse setzen sie auf die Klassifikation mit MLP. Durch ihre Forschung soll gezeigt werden, dass die Zukunft von Malware Erkennung in Deep Learning liegt. Als Grundlage für das Experiment diente das Advanced Security Network Metrics & Non-Payload-Based Obfuscations (ASNM-NPBO) Datenset (Homoliak u. a. 2013), welches aus legitimer und bösartiger TCP Kommunikation besteht. Das Datenset führt zwei Arten von Kennzeichnung auf. Zum einen wird der Verkehr binär in *gutartig* oder *bösartig* klassifiziert, zum anderen werden die Daten drei Klassen zu geordnet: *gutartig*, *direkte Attacke* oder *verschleierte Attacke*. Teoh u. a. (2018) beschränkten sich bei ihrer Untersuchung auf das binär klassifizierte Datenset, welches aus circa 9000 Aufzeichnungen besteht. Von den knapp 900 Attributen verwendeten die Forscher lediglich 15 für ihre Analyse. Anschließend wurden zwei Modelle entwickelt: MLP und J48 *decision tree*. Letzterer erzielte eine Genauigkeit von 99.35%. Mit Hilfe des MLP wurden lediglich 15(!) Daten analysiert, welche eine Genauigkeit von 100% aufweisen. Da dies aber keine repräsentative Anzahl an Daten ist kann über die Effizienz einer Analyse mit MLP sowie über die zu Beginn aufgestellte These - die Zukunft von Malware Erkennung liegt in Deep Learning - keine relevante Aussage getroffen werden.

4.3.23. Erkennung bösartiger SQL-Abfragen (2018)

Jayaprakash und Kandasamy (2018) stellen in ihrer Arbeit einen Anomalie basierten Ansatz vor, um ein IDS für SQL Datenbanken zu entwickeln. Laut Jayaprakash und Kandasamy (2018) reichen bisherige signaturbasierte Lösungen hierfür nicht aus, da diese lediglich auf bekannte Signaturen reagieren und dadurch neuartige Angriffe nicht erkennen. Bei ihrer Analyse möchten sie Angriffe von innerhalb einer Organisation als auch von außerhalb erfassen. Dabei setzen sie auf eine Datenstruktur welche aus einer Beziehung von acht Arrays besteht. In diesen Arrays werden SQL Abfragen entsprechend repräsentiert. Für die Analyse verwendeten sie den Naïve Bayes Klassifikator, welcher Daten in Klassen, entsprechend der vorhandenen Benutzerrollen zuteilt und die Protokolldatei als Eingabe verwendet.

Dabei wird ein Profil erstellt, welches mit bisherigen Profilen verglichen, und mit einem Punktesystem von 0 bis 10 bewertet wird. Befindet sich der dabei ermittelte Schweregrad über 8.0 wird der Administrator alarmiert. Liegt der Wert zwischen 6.0 und 8.0 wird die Abfrage geblockt. Andererseits wird die Abfrage ausgeführt. Das Modell wurde mit 239 Abfragen getestet. Dabei konnten 59.92% korrekt klassifiziert werden.

4.3.24. Erkennung von LDDoS Attacken (2018)

In ihrer Arbeit stellen Siracusano, Shiaeles und Ghita (2018) eine Methode vor um LDDoS Angriffe zu erkennen. Bei dieser Art von Angriff handelt es sich im Gegensatz zu DoS Attacken, bei welchen ein Server mit Anfragen geflutet wird, um Anfragen welche sehr langsam und nacheinander an einen Host gesendet werden. Dadurch hält der Server Ressourcen bereit die auf den Rest der Nachricht warten, wodurch andere Benutzeranfragen nicht abgearbeitet werden können. Um LDDoS Attacken zu erkennen, verwenden sie TCP-Verbindungsparameter. Diesbezügliche Daten extrahierten sie aus einem eigens zu diesem Zweck simulierten Netzwerk aus Clients, Angreifern und einem Webserver. Zusätzlich verwendeten sie das CIC Datenset (Jazi u. a. 2017), welches DoS Attacken auf der Anwendungsebene beinhaltet. Diverse Analysen wurden mit LR, k-NN, SVM, DT, RF und Deep Neural Network (DNN) durchgeführt. Die Ergebnisse zeigen das besonders Analysen mit k-NN und DT sehr effizient sind. Alle Modelle erreichten eine Genauigkeit von 95%. Die Analyse anhand eines Decision Tree erreichte nicht nur eine FPR und einer FNR von 0, sondern konnte zudem mit einer Evaluierungszeit von 0.019s am schnellsten durchgeführt werden. Somit wurde nicht nur gezeigt, dass eine Analyse von TCP Daten mit MLAs erfolgreich sein kann, sondern auch welcher Algorithmus sich am besten dafür eignet.

4.3.25. Klassifizierung von Wi-Fi Netzwerkdaten (2018)

Qin u. a. (2018) verwendeten für ihre Analyse das Aegean WiFi Intrusion Dataset (AWID) welches Angriffe auf kabellose Netzwerke beinhaltet. In ihrer Analyse klassifizierten die Forscher die Daten nach Flooding oder Injection Attacken sowie nach normalem Netzwerkverkehr. Für ihre Analyse verwendeten sie 18 von 154 möglichen Features wie zum Beispiel der Dauer der Verbindung, das Zeitdelta vom letzten erfassten Frame, die Datenrate sowie die Quelladresse. Sie wählten eine SVM um die Daten zu analysieren. Da Support Vector Machines (SVMs) grundsätzlich nur binär klassifizieren wurde eine Methode verwendet die eine zusätzliche Bibliothek benötigt um eine Multi-Klassen Klassifikation durchführen zu können. Hierbei

werden mehrere SVMs verwendet um Daten zu klassifizieren. Gibt es k Klassen, werden $k(k-\frac{1}{2})$ SVMs für die Analyse verwendet (Qin u. a. 2018). Diese Methode erzielte eine Genauigkeit für Flooding Attacken, Injection Attacken und normale Daten von 89.18%, 87.34%, und 99.88%.

4.3.26. Klassifizierung von verschleierter Malware (2019)

Han u. a. (2019) möchten mit ihrem Ansatz das Problem der Schadsoftwareverschleierung lösen. Diese Technik wird von Malware mehr und mehr verwendet um einer Erkennung zu entgehen (Li, Peng u. a. 2016). Zu diesen Techniken zählen Packing, Metamorphismus und Polymorphismus. Bei ersterem handelt es sich um komprimierten Schadcode, welcher zunächst entpackt werden muss um diesen analysieren zu können. Bei Polymorphismus wird eine Technik angewandt bei welcher der Binärcode verschlüsselt und mutiert wird. Dabei wird sobald der Code in den Speicher geladen wird, eine neue Version desselben generiert, wodurch divergierende Signaturen für denselben Code entstehen können. Metamorphismus ist ein weiterer Verschlüsselungsansatz bei welchem die Opcode Sequenz bei jedem Programmstart geändert wird. Dies erschwert dem Detektor ein stabiles Featureset für die Malware zu erstellen. Weitere Informationen zu diesen Techniken können P. He u. a. (2017) entnommen werden.
Um Daten zu generieren setzen die Forscher auf eine dynamische Analyse der Malware. Das bedeutet die Schadsoftware wird in einer gesicherten Umgebung ausgeführt. Aus den dadurch erfassten API und Dynamic Link Library (DLL) Informationen, der Interaktion mit Dateien und dem Netzwerk, sowie Informationen aus den PE-Dateien, werden Features abgeleitet, mit welchen der Klassifikator trainiert wird. Dadurch entwickelten sie folgende Modelle: DT, RF, k-NN und Extreme Gradient Boosting (XGBoost). Als Leistungsindikatoren wählten sie Accuracy, Precision, Recall und F1-measure. Durch diverse Experimente konnten sie nachweisen, dass durch die Analyse der Informationen bezüglich der Interaktion mit Dateien, dem Netzwerk und der Registry durch RF eine Genauigkeit von 97.21% erzielt werden kann. Dadurch konnte eine verlässliche Analyse selbst bei verschleierter Malware nachgewiesen werden.

4.3.27. Klassifizierung von Malware - Imageanalyse (2019)

Um Polymorphismus, Metamorphismus und Packing mit traditionellen MLAs zu erkennen ist ein umfangreiches Feature Engineering, sowie beträchtliche Kenntnisse auf Domain-Ebene nötig (Rhode, Burnap und Jones 2018). Zudem können Angreifer der automatischen Malware Erkennung entgehen sobald sie die verwendeten Features

kennen (Anderson, Kharkar u. a. 2017). Diesen Problemen wollen Vinayakumar u. a. (2019) mit ihrem Deep Learning Ansatz begegnen.
Dazu verglichen sie klassische Algorithmen für maschinelles Lernen mit Deep Learning Architekturen. Die Vergleiche basieren auf statischen und dynamischen Analysen, sowie auf Bildverarbeitungstechnologien.
Für die statische Analyse verwendeten sie privat gesammelte Proben sowie das Ember Datenset (Anderson und Roth 2018), welches aus je rund 70.000 bösartigen und gutartigen Dateien besteht. Mit Hilfe von LR, NB, k-NN,DT, AB, RF, SVM, Light Gradient Boosting Machine (LightGBM) sowie DNN und CNN entwickelten sie einen hybriden Ansatz um Malware statisch zu klassifizieren. Die Ergebnisse zeigen, dass Deep Neural Networks (DNNs) eine höhere Genauigkeit (98.9%) als traditionelle MLAs (LightGBM: 97.5%) erreichen.
Auch bei der dynamischen Analyse konnten Deep Learning Architekturen klassische MLAs übertreffen. Für diese Art Analyse wurden Daten in einer Cuckoo Sandbox generiert. Das beste Ergebnis erzielte hierbei ein CNN mit einem AUC von 0.9978.
Als drittes Experiment wurde eine Imageanalyse durchgeführt, wobei Malware Dateien als grau skalierte Bilder dargestellt werden. Für diese Analyse verwendeten sie das Malimg Datenset (Nataraj u. a. 2011), welches knapp 10.000 Malware Bilder beinhaltet, sowie privat gesammelte Proben. Bei Experimenten wurde deutlich, dass Analysen mit LSTM, mit einer Genauigkeit von 96.3% die höchste Effizient bieten.
Wie sich zeigte ist die Imageanalyse schneller als die statische und die dynamische Analyse, da diese auf Rohdaten basiert, unabhängig von Packing ist und komplett auf Zerlegung oder Ausführung von Code verzichtet.

4.3.28. Erkennung von FF Netzwerken (2019)

Um böswillige Netzwerkangriffe wie DDoS, Phishing und Spaming zu verschleiern, setzen Angreifer vermehrt auf FF. Durch diese Technologie entsteht ein sich ständig änderndes Netzwerk kompromittierter Hosts die als Proxy dienen. Durch die schnelle Änderung der IP Adresse des Kontrollterminals werden solche Angriffe häufig nicht erkannt, da IP Blacklists hierbei nicht funktionieren. Zudem erschwert die Analogie zu Content Distribution Networks (CDNs) einer Differenzierung dieser beiden Arten von Netzwerken. Diesem Differenzierungsproblem haben sich Chen u. a. (2019) in ihrer Arbeit angenommen. Als eines der Features verwenden sie den Domain Namen. Dieser ist in FF Netzen schlecht leserlich, da die Reihenfolge von Konsonanten und Vokalen unregelmäßig und zudem mit Nummern vermischt ist. Außerdem sind hierbei die Domain Namen oft länger als in CDNs. Als weiteres Feature wird der CNAME verwendet, welcher einer Domäne als zusätzlichem Namen bzw Alias dient. Zusätzlich wird der A-Record, welcher einer Domain eine

feste IP Adresse zuteilt, als Feature verwendet. Im Gegensatz zu CDNs sind FF Domain Namen kurzlebiger und weisen einen geringeren Netzwerkverkehr auf, daher wurde auch dieses Merkmal als Feature aufgenommen. Als zusätzliches Feature werden geographische Unterschiede der Ergebnisse von Domänen verwendet. In CDNs werden Knoten in verschiedenen geographischen Gebieten bereit gestellt. Deswegen bekommen Nutzer die weit voneinander entfernt sind unterschiedliche Auflösungen als Ergebnis, wenn sie Domain Namen abfragen. In FF Netzen werden jedoch immer dieselben Ergebnisse geliefert, da diese über eine viel kleinere Anzahl von IP Adressen verfügen. Ihre Erkennungsmethode basiert auf einem LSTM dessen Effizienz sie anhand der Genauigkeit, dem Recall und dem F1-measure gemessen haben. Die Analyse erzielte bei einer Trainingszeit zwischen 35 und 75 Sekunden eine Genauigkeit und einen F1-measure von über 0.95.

4.3.29. Erkennung von drive-by Download-Attacken bei Twitter (2019)

Das Kürzen von Links innerhalb von Tweets hat den Vorteil, dass Nutzer auch in anbetracht der vorgegebenen Länge von 140 Zeichen pro Tweet, lange URLs tweeten können. Dieses Feature birgt allerdigs gleichzeitig die Gefahr Opfer eines drive-by Download Angriffs zu werden. Hierbei nutzen Angreifer die Kürzung der Links, um Benutzer über Bilder oder Text auf bösartige Seiten zu locken. Diese Links können dazuführen, dass der Angreifer Fernzugriff zum System des Opfers bekommt, von welchem er Daten extrahieren oder dessen Computer in ein Botnetz integrieren kann (Provos u. a. 2007). Javed, Burnap und Rana (2019) haben in ihrer Arbeit diese Art von Links untersucht und nach *bösartig* und *gutartig* klassifiziert. Für ihre Analyse sammelten sie Tweets über die Twitter Streaming API, zur Fussball Europameisterschaft von 2016 (#Euro2016) und zu den Olympischen Spielen (#Rio2016) desselben Jahres. Da diese zu den Top Themen dieses Jahres gehörten beinhalteten diese die meisten Tweets. Für ihre Experimente nutzten sie ein Tool, welches jede URL in einer sicheren Umgebung besucht und etwaige Systemlevel Operationen, wie besispielsweise Datei-, Prozess- oder Registry-Änderungen aufzeichnet. Ergeben sich daraus clientbasierte Änderungen wie die Freigabe von Speicher, Startzeit eines Prozesses oder gesendete Bytes, fließen diese als Features in die Analyse ein. Insgesamt werden dabei 54 Metriken aufgezeichnet. Der zweite Teil des Feature-Sets, setzt sich aus 24 Tweet Attributen wie Beutzernamen, Art des Benutzers, Anzahl an Followern und der Anzahl der Retweets zusammen. Diese insgesamt 78 Features analysierten sie anhand von vier Modellen: Naïve Bayes, Bayesian Network, J84 (Decision Tree) und MLP. Als Leistungsindikatoren verwendeten sie Precision, Recall, F1-measure und FPR. Die Modelle wurden mit Daten der

Fussball Europameisterschaft trainiert und anschließend mit Daten der Olympiade getestet. Dabei kam ein Votum Meta-Klassifikator zum Einsatz, welcher es erlaubt Ergebnisse mehrerer Klassifikatoren zu verbinden und somit die beste Klassifiezierungswahrscheinlichkeit zu generieren. Wie sich zeigte führen Kombinationen aus J48 und NB sowie NB und MLP zu den besten Ergebnissen. Erstere erzielte einen F1-measure von 0.862, letztere erreichte einen Wert von 0.75.

4.3.30. Erkennung von DGA Domains (2019)

Der Domain Generation Algorithm (DGA) ist ein Algorithmus, mit dem Domainnnamen generiert werden, die häufig von Malware verwendet werden, um domainnbasierten Firewall-Steuerelementen auszuweichen. Dadurch können C2 Server verschleiert werden.
Li, Xiong u. a. (2019) fokussieren sich in ihrer Arbeit darauf DGA Domains zu identifizieren. Für ihre Analyse verwendeten sie Feed-Listen (Bambenek 2019) welche DGA-generierte Domänen beinhalten, die von Malware genutzt werden. Diese Listen sammelten sie über einen Zeitraum von einem Jahr. Für die Analyse der DGA Domains betrachteten sie jede Domain als Zeichenkette, aus welcher sie zwei Arten von Features extrahierten: sprachliche Features und DNS-Features. Zu den sechs sprachlichen Features gehören beispielsweise die Länge, das Verhältnis bedeutungsvoller Wörter, sowie der Prozentsatz numerischer Zeichen. Unter den 27 DNS-Features befinden sich beispielsweise Erzeugungszeit und Ablaufdatum, da DGA-Domains zeitnah erstellt werden und nur eine sehr kurze Zeit gültig sind. Ihre Analyse besteht aus zwei Teilen: zunächst werden die Domains nach *normal* oder *DGA* klassifiziert. Dafür wurden die folgenden Modelle entwickelt: J48, ANN, SVM, LR, NB, Gradient Boosting Tree (GBT) und RF. Anhand der Ergebnisse wurde deutlich, dass durch die Decision Tree Implementierung J48 die besten Werte mit einer Genauigkeit von 95.89% erzielt werden können, weshalb dieser Algorithmus als endgültiger Klassifikator gewählt wurde. Im Anschluss an diese erste Analyse wurden Domains, welche der Klasse DGA angehörten anhand des DBSCAN Algorithmus entweder in eines von drei Malware Cluster, oder dem Cluster für *normale* Domains zugewiesen. Hierbei beträgt die Durchschnittsgenauigkeit 87.64%. Zusätzlich entwickelten sie Deep Learning Modelle und verglichen diese mit dem erfolgreichsten Machine Learning Algorithmus der vorherigen Experimente, J48. Dabei fanden sie heraus, dass gerade bei einr hohen Anzahl an Daten ein LSTM Modell die höchste Genauigkeit mit 98.77% liefert. Dadurch konnten sie die Effizienz von DNNs auch bei der Erkennung von DGA-Domains nachweisen.

4.3.31. Erkennung von Phishing Websites (2019)

Phishing zielt im Vergleich zu anderen Attacken nicht darauf ab Schwachstellen im System auszunutzen, sondern durch Sicherheitslücken beim Menschen an dessen sensitive Informationen wie Benutzernamen und Passwörter zu gelangen. In der Forschung gibt es momentan vier Verfahren, um Phishing Websites zu erkennen: Blacklists, Heuristiken, Inhaltsanalysen und Machinelles Lernen. Blacklists gleichen URLs mit bekannten Phishing Websites ab, Heuristiken verwenden Signaturdatenbanken bekannter Angriffe um sie mit der Signatur eines heuristischen Musters abzugleichen. Inhaltsanalysen versuchen Phishing Websites mit Hilfe bekannter Algorithmen wie TF-IDF zu identifizieren.
Der im Folgende beschriebene Ansatz von Alswailem u. a. (2019) verwendet Machine Learning Verfahren, um Phishing Websites zu erkennen.

4.3.32. Erkennung von Insider Bedrohungen (2019)

Le und Nur Zincir-Heywood (2019)...

4.3.33. Erkennung von bösartigen PDFs (2019)

Jeong, Woo und Kang (2019)...

5. Datensätze

6. Prototypische Implementierung

7. Diskussion

8. Fazit

9. Zukünftige Forschung

A. Literatur

Bücher und Journals

Aksu, D. und M. Ali Aydin (2018). "Detecting Port Scan Attempts with Comparative Analysis of Deep Learning and Support Vector Machine Algorithms". In: *2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)*. IEEE, S. 77–80 (siehe S. 28).

Aldwairi, T., D. Perera und M. A. Novotny (2018). "An evaluation of the performance of Restricted Boltzmann Machines as a model for anomaly network intrusion detection". In: *Computer Networks* 144, S. 111–119 (siehe S. 26).

Alswailem, A. u. a. (Mai 2019). "Detecting Phishing Websites Using Machine Learning". In: *2019 2nd International Conference on Computer Applications & Information Security (ICCAIS)*. IEEE, S. 1–6 (siehe S. 35).

Anderson, H. S., A. Kharkar u. a. (2017). "Evading machine learning malware detection". In: *Black Hat* (siehe S. 32).

Anderson, H. S. und P. Roth (2018). "Ember: an open dataset for training static PE malware machine learning models". In: *arXiv preprint arXiv:1804.04637* (siehe S. 32).

Bearden, R. und D. C.-T. Lo (2017). "Automated microsoft office macro malware detection using machine learning". In: *2017 IEEE International Conference on Big Data (Big Data)*. Bd. 2018-Janua. IEEE, S. 4448–4452 (siehe S. 21).

Brown, A. u. a. (2018). "Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection". In: *Proceedings of the First Workshop on Machine Learning for Computing Systems*, S. 1 (siehe S. 26).

Buczak, A. L. und E. Guven (2016). "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection". In: *IEEE Communications Surveys & Tutorials* 18.2, S. 1153–1176 (siehe S. 16).

Bundeskriminalamt (2018). "Cybercrime, Bundeslagebild 2017". In: (Siehe S. 2, 3).

Chapman, P. u. a. (1999). "The CRISP-DM user guide". In: *4th CRISP-DM SIG Workshop in Brussels in March*. Bd. 1999 (siehe S. 8).

Chen, X. u. a. (2019). "A Deep Learning Based Fast-Flux and CDN Domain Names Recognition Method". In: *Proceedings of the 2019 2nd International Conference on Information Science and Systems - ICISS 2019*. Bd. Part F1483. New York, New York, USA: ACM Press, S. 54–59 (siehe S. 32).

Choi, S. u. a. (Okt. 2017). "Malware detection using malware image and deep learning". In: *2017 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, S. 1193–1195 (siehe S. 20, 21).

Cohen, A. u. a. (2016). "SFEM: Structural feature extraction methodology for the detection of malicious office documents using machine learning methods". In: *Expert Systems with Applications* 63, S. 324–343 (siehe S. 18).

Das, A. u. a. (Dez. 2017). "Detection of Exfiltration and Tunneling over DNS". In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. Bd. 2018-Janua. IEEE, S. 737–742 (siehe S. 24).

Dharmapurikar, S. u. a. (2003). "Deep packet inspection using parallel bloom filters". In: *11th Symposium on High Performance Interconnects, 2003. Proceedings.* IEEE, S. 44–51 (siehe S. 28).

Ding, Y. und Y. Zhai (2018). "Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks". In: *Proceedings of the 2018 2nd International Conference on Computer Science and Artificial Intelligence - CSAI '18*. New York, New York, USA: ACM Press, S. 81–85 (siehe S. 25, 26).

e.V., B. (2017). *Cybercrime: Jeder zweite Internetnutzer wurde Opfer*. (Besucht am 01. 10. 2019) (siehe S. 3).

Evans, K. und F. S. Reeder (2010). *A human capital crisis in cybersecurity : technical proficiency matters : a report of the CSIS Commission on Cybersecurity for the 44th Presidency*. November, S. 35 (siehe S. 3).

Garcia, S. u. a. (2014). "An empirical comparison of botnet detection methods". In: *computers & security* 45, S. 100–123 (siehe S. 27, 28).

Han, W. u. a. (Jan. 2019). "MalInsight: A systematic profiling based malware detection framework". In: *Journal of Network and Computer Applications* 125.June 2018, S. 236–250 (siehe S. 11, 31).

He, P. u. a. (2017). "Model approach to grammatical evolution: deep-structured analyzing of model and representation". In: *Soft Computing* 21.18, S. 5413–5423 (siehe S. 11, 31).

He, Z., T. Zhang und R. B. Lee (2017). "Machine Learning Based DDoS Attack Detection from Source Side in Cloud". In: *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, S. 114–120 (siehe S. 21, 22).

Hendler, D., S. Kels und A. Rubin (2018). "Detecting Malicious PowerShell Commands using Deep Neural Networks". In: *Proceedings of the 2018 on Asia Conference on Computer and Communications Security - ASIACCS '18*. New York, New York, USA: ACM Press, S. 187–197 (siehe S. 25).

Homoliak, I. u. a. (2013). "ASNM: Advanced security network metrics for attack vector description". In: *Proceedings of the International Conference on Security and Management (SAM)*. The Steering Committee of The World Congress in Computer Science, Computer~..., S. 1 (siehe S. 29).

Javed, A., P. Burnap und O. Rana (Mai 2019). "Prediction of drive-by download attacks on Twitter". In: *Information Processing & Management* 56.3, S. 1133–1145 (siehe S. 33).

Jayaprakash, S. und K. Kandasamy (Apr. 2018). "Database Intrusion Detection System Using Octraplet and Machine Learning". In: *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*. Icicct. IEEE, S. 1413–1416 (siehe S. 29).

Jazi, H. H. u. a. (2017). "Detecting HTTP-based application layer DoS attacks on web servers in the presence of sampling". In: *Computer Networks* 121, S. 25–36 (siehe S. 30).

Jeong, Y.-S., J. Woo und A. R. Kang (Apr. 2019). "Malware Detection on Byte Streams of PDF Files Using Convolutional Neural Networks". In: *Security and Communication Networks* 2019, S. 1–9 (siehe S. 35).

Kumar, A., K. Kuppusamy und G. Aghila (2017). "A learning model to detect maliciousness of portable executable using integrated feature set". In: *Journal of King Saud University - Computer and Information Sciences* 31.2, S. 252–265 (siehe S. 23, 24).

Le, D. C. und A. Nur Zincir-Heywood (2019). "Machine learning based insider threat modelling and detection". In: *2019 IFIP/IEEE Symposium on Integrated Network and Service Management, IM 2019*, S. 1–6 (siehe S. 35).

Leonard, J., S. Xu und R. Sandhu (2009). "A framework for understanding botnets". In: *2009 International Conference on Availability, Reliability and Security*. IEEE, S. 917–922 (siehe S. 27).

Li, Y., Z. Peng u. a. (2016). "Facial age estimation by using stacked feature composition and selection". In: *The Visual Computer* 32.12, S. 1525–1536 (siehe S. 31).

Li, Y., K. Xiong u. a. (2019). "A Machine Learning Framework for Domain Generation Algorithm-Based Malware Detection". In: *IEEE Access* 7, S. 32765–32782 (siehe S. 34).

Maniath, S. u. a. (2017). "Deep learning LSTM based ransomware detection". In: *2017 Recent Developments in Control, Automation & Power Engineering (RDCAPE)*. Bd. 3. IEEE, S. 442–446 (siehe S. 20).

Mathur, L., M. Raheja und P. Ahlawat (2018). "Botnet Detection via mining of network traffic flow". In: *Procedia Computer Science* 132, S. 1668–1677 (siehe S. 27).

More, S. S. und P. P. Gaikwad (2016). "Trust-based Voting Method for Efficient Malware Detection". In: *Procedia Computer Science* 79, S. 657–667 (siehe S. 17).

Nataraj, L. u. a. (2011). "Malware images: visualization and automatic classification". In: *Proceedings of the 8th international symposium on visualization for cyber security*. ACM, S. 4 (siehe S. 32).

Pham, T. S., T. H. Hoang und V. C. Vu (2016). "Machine learning techniques for web intrusion detection — A comparison". In: *2016 Eighth International Conference on Knowledge and Systems Engineering (KSE)*. IEEE, S. 291–297 (siehe S. 19).

Provos, N. u. a. (2007). "The Ghost in the Browser: Analysis of Web-based Malware." In: *HotBots* 7, S. 4 (siehe S. 33).

Qin, Y. u. a. (Sep. 2018). "Attack Detection for Wireless Enterprise Network: a Machine Learning Approach". In: *2018 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*. IEEE, S. 1–6 (siehe S. 30, 31).

Raff, E., J. Sylvester und C. Nicholas (2017). "Learning the PE Header, Malware Detection with Minimal Domain Knowledge". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security - AISec '17*. New York, New York, USA: ACM Press, S. 121–132 (siehe S. 23, 24).

Rhode, M., P. Burnap und K. Jones (2018). "Early-stage malware prediction using recurrent neural networks". In: *computers & security* 77, S. 578–594 (siehe S. 31).

Robin Tommy, Gullapudi Sundeep, H. J. (2017). "Automatic Detection and Correction of Vulnerabilities using Machine Learning". In: *2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, S. 1062–1065 (siehe S. 22).

Sabar, N. R., X. Yi und A. Song (2018). "A Bi-objective Hyper-Heuristic Support Vector Machines for Big Data Cyber-Security". In: *IEEE Access* 6, S. 10421–10431 (siehe S. 27, 28).

Shahzad, R. K. und N. Lavesson (2013). "Comparative analysis of voting schemes for ensemble-based malware detection". In: *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications* 4.1, S. 98–117 (siehe S. 17).

Shijo, P. und A. Salim (2015). "Integrated Static and Dynamic Analysis for Malware Detection". In: *Procedia Computer Science* 46.Icict 2014, S. 804–811 (siehe S. 16, 17).

Shiravi, A. u. a. (2012). "Toward developing a systematic approach to generate benchmark datasets for intrusion detection". In: *computers & security* 31.3, S. 357–374 (siehe S. 26).

Sikorski, M. (2012). *Praise for Practical Malware Analysis*, S. 802 (siehe S. 14).

Siracusano, M., S. Shiaeles und B. Ghita (2018). "Detection of LDDoS Attacks Based on TCP Connection Parameters". In: *2018 Global Information Infrastructure and Networking Symposium (GIIS)*. IEEE, S. 1–6 (siehe S. 30).

Teoh, T. T. u. a. (2018). "Anomaly detection in cyber security attacks on networks using MLP deep learning". In: *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*. IEEE, S. 1–5 (siehe S. 29).

Vinayakumar, R. u. a. (2019). “Robust Intelligent Malware Detection Using Deep Learning”. In: *IEEE Access* 7, S. 46717–46738 (siehe S. 32).

Webster, J. und R. T. Watson (2002). “Analyzing the Past to Prepare for the Future: Writing a Literature Review.” In: *MIS Quarterly* 26.2, S. xiii–xxiii (siehe S. 6, 7).

Yeo, M. u. a. (2018). “Flow-based malware detection using convolutional neural network”. In: *2018 International Conference on Information Networking (ICOIN)*. Bd. 2018-Janua. IEEE, S. 910–913 (siehe S. 28).

Yin, C. u. a. (2017). “A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks”. In: *IEEE Access* 5, S. 21954–21961 (siehe S. 19, 20, 25, 26).

Internetquellen

Bambenek (2019). *OSINT Feeds from Bambenek Consulting*. URL: http://osint.bambenekconsulting.com/feeds/ (besucht am 21. 11. 2019) (siehe S. 34).

Carmen Torrano Giménez, Alejandro Pérez Villegas, G. Á. M. (2010). *CSIC 2010 HTTP dataset*. URL: http://www.isi.csic.es/dataset/ (siehe S. 19).

Kaggle (2015). *Microsoft Malware Classification Challenge*. URL: https://www.kaggle.com/c/malware-classification/data (besucht am 08. 11. 2019) (siehe S. 27).

SmartVisionEurope (2015). *CRISP-DM Methodology*. URL: http://crisp-dm.eu/home/crisp-dm-methodology/ (besucht am 18. 10. 2019) (siehe S. 8, 9).

AV-TEST (2019). *Malware Statistics & Trends Report*. URL: https://www.av-test.org/en/statistics/malware/ (besucht am 08. 10. 2019) (siehe S. 3).

Victoria, U. of (2010). *ISOT Botnet Dataset*. URL: https://www.uvic.ca/engineering/ece/isot/datasets/ (besucht am 08. 11. 2019) (siehe S. 27).

Abbildungsverzeichnis

Tabellenverzeichnis

B. Abkürzungsverzeichnis

AB AdaBoost 18

ACK acknowledge 22

ANN Artificial Neural Network 19

APTs Advanced Persistent Threats 4

ARFF Attribute-Relation File Format 17

ASNM-NPBO Advanced Security Network Metrics & Non-Payload-Based Obfuscations 29

AUC Area Under the Curve 23

AUC ROC area under the receiver operating characteristic curve 26

AWID Aegean WiFi Intrusion Dataset 30

BITKOM Bundesverband Informationswirtschaft,Telekommunikation und neue Medien e.V. 3

BKA Bundeskriminalamt 2

BN Bayesian Network 18

DRBMs Deep Restricted Boltzmann Machines . . . 26

DT Decision Tree . . . 17

ET Extra Random Trees . . . 23

FC Fuzzy Classifier . . . 27

FF Fast-Flux . . . II

FNR False Negative Rate . . . 26

FPR False Positive Rate . . . 16

(FC) Neural Network Fully Connected Neural Network . . . 23

GBT Gradient Boosting Tree . . . 34

GMM-EM Gaussian-Mixture Model for Expectation-Maximization . . . 22

GNBT Gaussian Naive Bayes Tree . . . 27

GUI Graphical User Interface . . . 12

GPU Graphics Processing Unit . . . 20

HTTP Hypertext Transfer Protocol . . . 3

ICMP Internet Control Message Protocol . . . 22

IDS Intrusion Detection System . . . 19

IoC Indicator of Compromise . . . . . B

IoCs Indicators of Compromise 7

IoT Internet of Things 6

k-NN k-nearest-neighbor 17

LANL Los Alamos National Laboratory 26

LB LogitBoost 18

LDDoS Low-rate DDoS II

LDA Linear Discriminant Analysis 24

LightGBM Light Gradient Boosting Machine 32

LR Logistic Regression 18

LSTM Long-Short Term Memory 20

MLAs Machine Learning Algorithmen 3

MLP Multi-Layer Perceptron 19

NB Naïve Bayes 18

NBTree Naïve Bayesian Tree 19

NLP Natural Language Processing 25

Opcode Operation Code 17

PE-Datei Portable Executable Datei

PE-Dateien Portable Executable Dateien 11

RAM Random-Access Memory 13

RBM Restricted Boltzmann Machine 26

RNN Recurrent Neural Network 20

RNNs Recurrent Neural Networks 19

RF Random Forest 16

R2L Remote to Local 19

SMO Sequential Minimal Optimization 18

SSH Secure Shell 22

SVM Support Vector Machine 16

SVMs Support Vector Machines 30

SYN synchronize 22

TCP Transmission Control Protocol 22

TF-IDF Term Frequency - Inverse Document Frequency 18

TPR True Positive Rate 25

UDP User Datagram Protocol 27

U2R User to Root 19

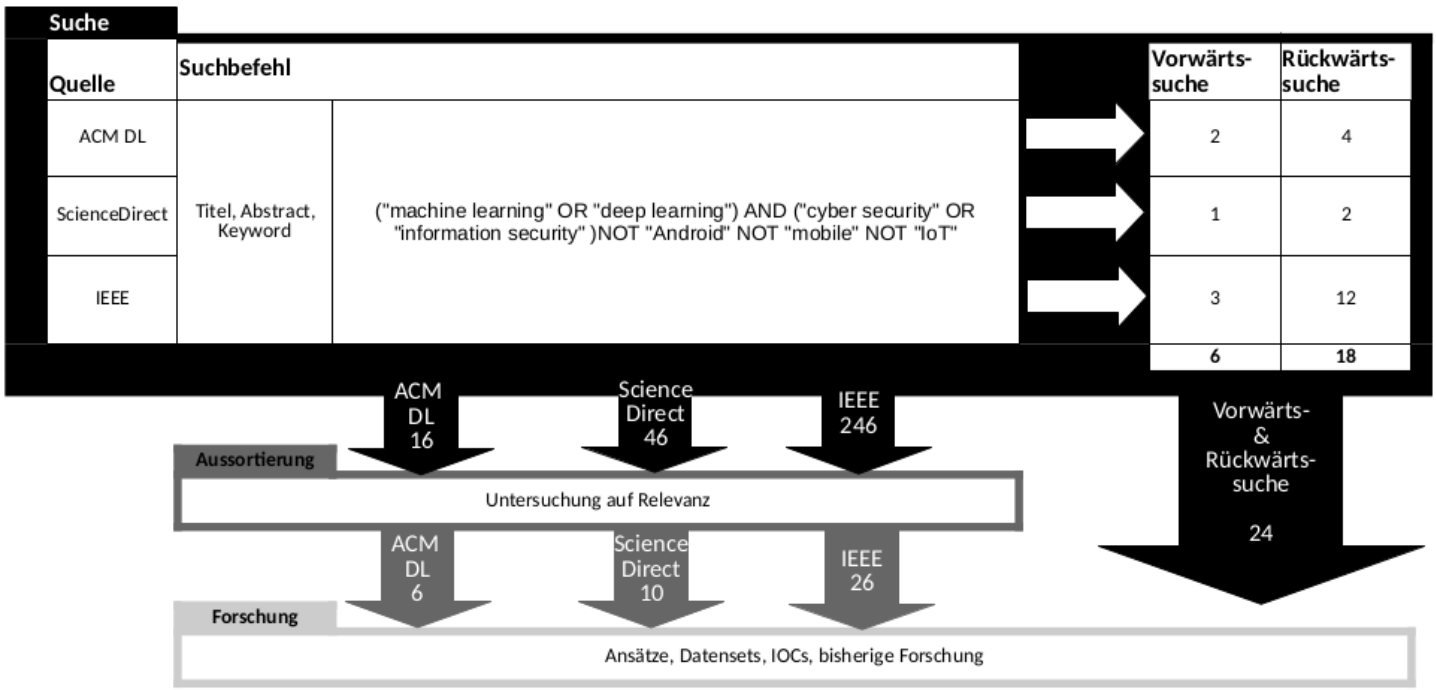
VBA Visual Basic for Applications 17

XGBoost Extreme Gradient Boosting 31

XSS Cross-Site Scripting 19

C. Anlagen

Anlage 1: Research Model



Prozess der Literaturrecherche

Anlage 2: Literaturreview

Legende:						
Related Work	Ansatz+Datensatz	Ansatz ohne Datensatz	Datensatz	IOCs	Motivation	
Autor	Titel	Jahr	Zitate Google Scholar	Publisher	Inhalt	Rating
Raff, Edward and Sylvester, Jared and Nicholas, Charles	Learning the PE Header, Malware Detection with Minima lDomain Knowledge	2017	19	ACM	Malware Detection mit minimalem Domänenwissen wobei ein Teil des PE headers extrahiert wird. Neuronale Netze lernen aus unformatierten Bytes ohne explizite Feature Extrahierung.	3
Sewak, Mohit and Sahay, Sanjay K. and Rathore, Hemant	An investigation of a deep learning based malware detection system	2018	5	ACM	Verbesserungen der Vorhersagen des Malicia Datensets durch Neuronale Netze. Allerdings wurde das Malicia Dataset eingestellt.	2
Hendler, Danny and Kels, Shay and Rubin, Amir	Detecting Malicious PowerShell Commands using Deep Neural Networks	2018	13	ACM	Erkennen bösartiger PowerShell Kommandos mit Hilfe Neuronaler Netze und NLPs	2
Ding, Yalei and Zhai, Yuqing	Intrusion Detection System for NSL-KDD Dataset Using Convolutional Neural Networks	2018	0	ACM	Neuronale Netze wurden auf das NSL-KDD Dataset agewandt welches aus rohen tcpdump Daten besteht	3
Brown, Andy and Tuor, Aaron and Hutchinson, Brian and Nichols, Nicole	Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection	2018	13	ACM	Anomalieerkennung in Systemprotokolle durch RNN (recurrent neural networks)	3
Chen, Xunxun and Li, Gaochao and Zhang, Yongzheng and Wu, Xiao and Tian, Changbo	A Deep Learning Based Fast-Flux and CDN Domain Names Recognition Method	2019	0	ACM	Differenzierung von Fast-Flux domain names und CDN (Content Distriution Network) domain names mit Hilfe von deep Learning	2
Lu Xiaofeng, Zhou Xiao, Jiang Fangshuo, Yi Shengwei, Sha Jing	ASSCA: API based Sequence and Statistics features Combined malware detection Architecture	2018	3	ScienceDirect	Malware Detection von system Files in Windows Systemen durch Machine Learning und Deep Learning. Daten von VirusShare und VirusTotal	3
Quan Le, Oisín Boydell, Brian Mac Namee, Mark Scanlon	Deep learning at the shallow end: Malware classification for non-domain experts	2018	13	ScienceDirect	Daten der Microsoft Malware Classification Challenge von Kaggle werden mit CNN bewertet.	3

Ausschnitt der Literaturreview Liste

Liste der noch zu erledigenden Punkte