Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Robert Lehnert

Entwicklung von headerbasierten Prüfungen zur Verbesserung des Schutzes vor missbräuchlichen E-Mails

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Robert Lehnert

Entwicklung von headerbasierten Prüfungen zur Verbesserung des Schutzes vor missbräuchlichen E-Mails

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Wirtschaftsinformatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Klaus-Peter Kossakowski
Zweitgutachter: Prof. Dr. Michael Köhler-Bußmeier

Eingereicht am: 18.12.2018

Robert Lehnert

Thema der Arbeit
Entwicklung von headerbasierten Prüfungen zur Verbesserung des Schutzes vor missbräuchlichen E-Mails

Stichworte
IT-Sicherheit, E-Mail, Header, Analyse, Spamschutz, Spoofing, Phishing

Kurzzusammenfassung
Im Rahmen dieser Arbeit werden die Header-Informationen von E-Mails zur Identifizierung von prüfungsrelevanten Kriterien analysiert, um eine bessere Klassifizierung von un- und erwünschten E-Mails, im Vergleich zu inhaltsbasierten Filtern zu ermöglichen. Hierfür werden kategorisierte Test-Mails getrennt voneinander betrachtet, um Unterschiede zu erkennen. Nach Abschluss der Analyse und der Identifizierung geeigneter Kriterien werden diese in ein lauffähiges Filterungsverfahren, unter Verwendung der Programmiersprache Python umgesetzt. Abschließend wird das entwickelte Filterungsverfahren in ein operatives System integriert und mit einer Vergleichsmenge von un- und erwünschten E-Mails evaluiert.

Robert Lehnert

Title of the paper
Development of header-based checks to improve the protection against unsolicited emails

Keywords
IT security, Email, Header, Analysis, Spam protection, Spoofing, Phishing

Abstract
This bachelor thesis analyzes the information of email headers for the identification of test criteria that ensure better classification of unsolicited and desired emails as compared to content-based filters. The categorized test mails are regarded separately to identify differences. After the analysis and the identification of relevant test criteria those are implemented using the programming language Python. In conclusion, the developed filtering procedure is integrated into an operative system and evaluated with a set of unsolicited and desired e-mails.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Quellcode-Verzeichnis

Abkürzungsverzeichnis

DDoS	Distributed-Denial-of-Service 28
DNS	Domain Name System 14
DNSBL	Domain Name System Black/Block List 22
DNSWL	Domain Name System White List 22
DynDNS	Dynamic Domain Name System 48
ESP	E-Mail Service Provider 5
FN	False Negative 21
FP	False Positive 21
IMAP4	Internet Message Access Protocol Version 4 12
IMF	Internet Message Format 9
IoT	Internet of Things 1
IP	Internet Protocol 7
MDA	Mail Delivery Agent 13
MSA	Message Submission Agent 13
MTA	Mail Transfer Agent 13
MUA	Mail User Agent 16
MX-RR	Mail Exchange Resource Records 14
POP3	Post Office Protocol Version 3 12
RBL	Realtime-Blackhole-List 22
rDNS	reverse Domain Name System 22
RFC	Request for Comments 7
SMTP	Simple Mail Transfer Protocol 13
TCP	Transmission Control Protocol 7
TLD	Top-Level-Domain 5
TN	True Negative 21
TP	True Positive 21

1. Einleitung

Die Nutzung der E-Mail beschränkt sich längst nicht mehr auf die Kommunikation mit einer anderen Person. In Zeiten der Digitalisierung wird ein Benutzer stets per E-Mail über aktuelle Angebote, seinen Bestellstatus oder Neuigkeiten in sozialen Netzwerken informiert.

So werden aktuell etwa 281 Milliarden E-Mails täglich von 3,8 Milliarden E-Mail-Accounts versendet. Für 2022 wird erwartet, dass etwa 333 Milliarden E-Mails täglich von 4,3 Milliarden Accounts versendet werden. Dies würde einen Zuwachs von 18,5% an täglich versendeten E-Mails und 11,5% an neuen Accounts bedeuten. [vgl. 46, S. 3]

Der Internet Security Threat Report der Firma Symantec Corporation legt dar, dass der Anteil von unerwünschten Mails im Jahr 2017 die Marke von 54% geknackt hat und das ein weiterer Anstieg zu erwarten ist. Symantec schätzt ein, dass ein Mitarbeiter, vorausgesetzt das Unternehmen hat keinen Spamfilter, für das Sortieren und Löschen von unerwünschten Mails ca. 10 Minuten pro Tag benötigt. Demnach entstehen einem Unternehmen (pro 100 Mitarbeiter) jährliche Kosten in Höhe von ca. 117.000 US-Dollar. [vgl. 33, S. 18] Für den Privatanwender bedeutet dies, dass er ca. 2,5 Tage im Jahr damit beschäftigt ist unerwünschte Mails auszusortieren.

Hinzukommt die Menge von ca. 8,4 Milliarden IOT-Geräten (2017), die mit dem Internet verbunden sind [vgl. 51] und zusätzlich noch die steigende Verbreitung und Nachfrage von IoT-Geräten [vgl. 18, S. 17 f.], welche es potenziellen Angreifern ermöglicht, die Menge an unerwünschten E-Mails zu steigern. Die ITU[1] definiert "Internet of Things" (kurz IoT) als:

> "A global infrastructure for the information society, enabling advanced services by interconnecting (physical and virtual) things based on existing and evolving interoperable information and communication technologies." [23, S. 1]

[1] International Telecommunication Union.

In diesem Kontext sind die "physischen Dinge", also die IoT-Geräte (bspw. vernetzte Haushaltsgeräte und mittlerweile auch Autos) relevant, da diese durch bösartige Software (siehe Kapital 2.4.2) kompromittiert werden und unerwünschte E-Mails versenden können. Für Angreifer sind IoT-Geräte interessant, weil diese im Vergleich zu Computern aktuell weniger Sicherheit bieten.

> "Das Internet der Dinge ist für Unternehmen mittlerweile zu einer zentralen Komponente der Digitalisierung geworden. [...] Es zeichnet sich ab, dass das IoT im gewerblichen Einsatz, weit mehr als im privaten Bereich, zu einer großen Erfolgsgeschichte wird. [...] IoT als Technologie ist noch ziemlich jung und dementsprechend noch lange nicht ausgereift. [...] Wie bei fast jeder jungen Technologie steht bei der Entwicklung der jeweiligen Komponenten die Funktionalität im Vordergrund und ist oft Herausforderung genug. Sicherheitsaspekte bleiben außen vor oder werden nur am Rande behandelt; wichtig ist vielmehr, dass das Ding überhaupt zum Laufen kommt." [27]

Die Firma Proofpoint[2] entdeckte 2014 den ersten nachweisbaren auf IoT-Geräte basierenden Angriff, bei der haushaltsübliche Geräte wie Router, Fernsehgeräte und auch ein Kühlschrank involviert waren. Bei diesem Angriff wurden weltweit mehr als 750.000 infizierte E-Mails von mehr als 100.000 dieser Geräte verschickt, dass bedeutet durchschnittlich ca. 7,5 E-Mails pro Gerät. [vgl. 37] Die Anzahl der infizierten Geräte dieses Angriffs ist im Verhältnis zu der Anzahl der heute verbundenen Geräte ziemlich gering, wodurch ein deutlich höheres Potenzial besteht.
Die IDC[3] rechnet im Jahr 2020 mit 30 Milliarden vernetzten Geräten [vgl. 22]. Wird diese Schätzung mit dem Durchschnitt aus dem Jahr 2014 multipliziert, würde dies eine mögliche Kapazität von zusätzlich 225 Milliarden unerwünschten Mails bedeuten. Dies entspricht ca. 90% des heutigen E-Mail-Verkehrs und hätte zur Folge, dass der Endverbraucher fast doppelt solange zum Sortieren von E-Mails bräuchte.

Um diese Flut an unerwünschten E-Mails zu bewältigen, können Spamfilter eingesetzt werden, damit die wichtigen von den unerwünschten E-Mails getrennt werden und somit eine Kostenersparnis für die Unternehmen und eine Zeitersparnis für den Mitarbeiter/Privatanwender erreicht wird.

[2] Ist ein führender "Security as a Service" (SaaS) Anbieter.
[3] International Data Corporation.

Ansätze zum Filtern von Spam sind sender-, regel-, inhaltsbasierte und hybride Filterungsverfahren (siehe Abschnitt 2.5). Letzterer ist eine Kombination aus den vorher genannten Ansätzen und spielt in heutigen Filterungsverfahren die größte Rolle.

Die Convios Consulting GmbH[4] führte im Jahr 2017, im Auftrag von "WEB.DE"[5] und "GMX"[6] eine Umfrage im Bezug auf Datenschutz und Verschlüsselung durch [vgl. 1]. Aus dieser Umfrage geht hervor, dass 60% der Deutschen glauben, dass ihre E-Mails mitgelesen werden und drei Viertel aller Befragten halten E-Mail-Verschlüsselung für wichtig. Obwohl so viele der Befragten glauben, das ihre E-Mails mitgelesen werden und Verschlüsselung wichtig ist, nutzen erst 16% der Befragten eine Verschlüsselung. Gründe für die Nichtnutzung sind ein hoher Installationsaufwand oder mangelnde Kenntnisse. Unter Betrachtung dieser Umfrage und der aktuell, benutzerfreundlichen Einrichtung einer Verschlüsselung (bspw. bei "GMX"[7]) kann mit einer steigenden Nutzung einer E-Mail-Verschlüsselung gerechnet werden.

In Hinblick auf das Umfrageergebnis und unter Betrachtung rechtlicher Aspekt bezogen auf inhaltsbasierte Filterung (siehe Unterabschnitt 2.6) wird deutlich, dass die Klassifizierung von Spam-Mails anhand des Inhaltes der E-Mail zukünftig problematisch werden könnte. Deshalb setzt sich diese Arbeit nicht mit dem Inhalt einer E-Mail, sondern mit der Analyse der Informationen, welche für die Übertragung einer E-Mail unerlässlich sind, auseinander.

[4] URL: http://convios.com/.
[5] E-Mail Service Provider - URL: https://web.de.
[6] E-Mail Service Provider - URL: https://gmx.net.
[7] URL: https://gmx.net/mail/sicherheit/pgp/.

1.1. Problembeschreibung

Die Verbreitung von Spam belastet nicht nur die Infrastruktur, die für den Transport, die Zustellung oder die Prüfung auf verdächtige Inhalte zuständig ist, sondern kostet die Unternehmen auch eine Menge Geld, ganz zu schweigen von der täglichen Belastung für den Endverbraucher durch nerviges Sortieren der E-Mails.

Der Begriff Spam ist auf ein Produkt der US-amerikanischen Firma "Hormel Foods"[8] zurückzuführen. Diese produzierte 1937 erstmals das Produkt "SPAM©"[9] ("spiced ham" = gewürzter Schinken), welches 1970 in einem Sketch der britischen Komikergruppe "Monty Python" verwendet wurde.
Dieser Sketch spielt in einem Restaurant, in welchem es nur Gerichte mit SPAM© gibt. Ein Gast will ein Gericht ohne SPAM© und diskutiert mit dem Kellner. Ein paar Wikinger, die am Nachbartisch sitzen, unterbrechen immer wieder das Gespräch und heben hervor, wie toll SPAM© ist. Am Ende singen sie ein Loblied auf SPAM© und verhindert damit jegliche Kommunikation zwischen dem Gast und dem Kellner, wodurch der heutige Begriff geprägt wurde (siehe Kapitel 2.4.1).
Damit diese Art der Störung uns nicht beeinträchtigt, muss Spam erfolgreich abgewehrt werden und dafür gibt es bereits eine Vielzahl von technischen Ansätzen (siehe Abbildung 1.1).

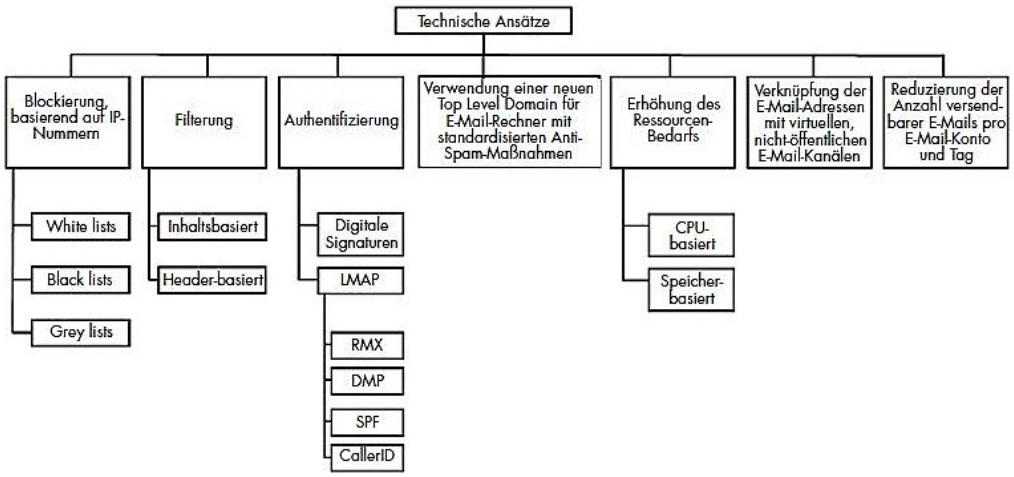


Abbildung 1.1.: Technische Anti-Spam-Ansätze [42, S. 283]

[8]URL: https://hormelfoods.com.
[9]URL: http://spam.com.

Schryen zeigt in seiner Arbeit für die einzelnen Ansätze die Grenzen und Nachteile auf. Ansätze, die sich mit der Authentifizierung, der Verwendung einer neuen Top-Level-Domain (kurz TLD) für E-Mail-Rechner, der Erhöhung des Ressourcenbedarfs oder der Verknüpfung der E-Mail-Adressen mit virtuellen, nicht öffentlichen E-Mail-Kanälen beschäftigen, müssen neue Infrastruktur schaffen oder hätten einen starken Einfluss auf derzeitige Infrastrukturen. Die Begrenzung der versendbaren E-Mails pro Tag und E-Mail-Konto würde die E-Mail Service Provider (kurz ESP) verpflichten, dies zu kontrollieren. Des Weiteren würde dies zum Nachteil für Unternehmen werden, welche ein großes Aufkommen an E-Mails haben. [vgl. 42, S. 287]
Die Blockierung und Filterung sind die bevorzugten Ansätze, meist in Kombination, da sie in jede Infrastruktur integriert werden können und in der Abwehr von Spam, mit Erkennungsraten von bis zu 99% [vgl. 3] effektiv sind.

Die Blockierung einer E-Mail kann mit Hilfe von sogenannten "Black-/ White- und Greylists" (siehe Kapitel 2.5.1) erreicht werden. Beim Versand einer E-Mail wird die IP-Adresse des Absenders mit IP-Adressen der jeweils integrierten Liste abgeglichen. Bei einer Blacklist wird die E-Mail abgewiesen, bei einer Whitelist akzeptiert, wenn die IP-Adresse des Absenders enthalten ist. Bei beiden Verfahren besteht das Problem, dass E-Mails nicht zugestellt werden und der Anwender von der Existenz nie etwas erfährt.
Bei einer Greylist ist dies nicht der Fall. Der erste Zustellversuch wird vom empfangenden Mailserver für eine definierte Zeit verweigert und der sendende Mailserver muss erneut versuchen die E-Mail zuzustellen. Das "Simple Mail Transfer Protocol" (siehe Kapitel 2.2) besagt, dass ein sendender Mailserver die E-Mail bei einem fehlerhaften Zustellversuch erneut zustellen muss. Hierbei sollte ein Intervall für den erneuten Zustellversuch mindestens 30 Minuten betragen. In der Regel wird das Zustellen einer E-Mail solange versucht, bis die Nachricht zugestellt werden konnte oder eine maximale Wartezeit von 4 bis 5 Tagen erreicht wird. [vgl. 26, S. 66] Bei Greylisting wird angenommen, dass der Absender von Spam (ugs. "Spammer") eine erneute Zustellung nicht versucht oder diese binnen kurzer Zeit durchführt, um seine Ressourcen nicht unnötig zu binden. [vgl. 21]
Zusammenfassend kann gesagt werden, dass durch die Blockierung auf der einen Seite (Black-/Whitelist) E-Mails verloren gehen können und auf der anderen Seite (Greylist) in ihrer Zustellung verzögert werden. Das Blockieren stellt eine andere Art von Filterung dar.

Ein weiterer Ansatz zur Abwehr von Spam ist die Filterung von E-Mails. Eine E-Mail teilt sich in ein E-Mail-Header und -body (siehe Kapitel 2.1) auf und kann mit nachfolgenden Filterungsverfahren analysiert werden. Neben dem "Listing" wird unterschieden zwischen der regelbasierten (siehe Kapitel 2.5.2) und der inhaltsbasierten Filterung (siehe Kapitel 2.5.3). Letztere analysiert den Inhalt der E-Mail, also den E-Mail-Body, bspw. nach auffälligen Wörtern ("Keyword checking") oder klassifiziert diesen unter Zuhilfenahme eines lernenden Systems ("Bayes-Filter"), worum es sich handelt. Durch eine zunehmende Nutzung von E-Mail-Verschlüsselung könnte eine zentrale inhaltsbasierte Filterung ihren Zweck nicht mehr erfüllen.
Der E-Mail-Header wird beim Versenden jeder E-Mail erstellt und enthält unter anderem Informationen zum Absender, Empfänger und zum zurückgelegten Weg der E-Mail. Durch eine regelbasierte Filterung kann dieser analysiert und die E-Mail klassifiziert werden. [vgl. 8, S. 7 f.]

1.2. Ziel der Arbeit

Ziel dieser Bachelorarbeit ist es, unter Einbeziehung und Analyse der Informationen eines E-Mail-Headers, bessere Prüfungen zu entwickeln, um den Schutz vor unerwünschten E-Mails zu verbessern. Der eigentliche Inhalt der E-Mail wird nicht mit in das Verfahren einbezogen.
Der E-Mail-Header wird auf geeignete Kriterien evaluiert, die eine bessere Klassifizierung von (un-)erwünschten E-Mails ermöglicht. Hierbei werden sowohl unerwünschte, als auch erwünschte E-Mails getrennt voneinander betrachtet, um Gemeinsamkeiten und Unterschiede zu erkennen. Nach der Identifizierung möglicher Kriterien, werden diese auf (un-)erwünschte E-Mails angewendet, um ihre Wirksamkeit zur Klassifizierung festzustellen. Im nächsten Schritt werden die identifizierten Kriterien zu headerbasierten Prüfungen zusammengefasst und in Programmcode umgesetzt. Abschließend wird das entwickelte Verfahren mit (un-)erwünschten E-Mails evaluiert.

1.3. Abgrenzung

Die "Public-Key-Infrastruktur" (kurz PKI) ist die häufig benutzte Methode zur E-Mail-Verschlüsselung, dabei kommt einer der Standards "Secure / Multipurpose Internet Mail Extensions" (kurz S/MIME) oder "Open Pretty Good Privacy" (kurz OpenPGP) zum Einsatz. S/MIME ist zum Zeitpunkt dieser Arbeit in der RFC[10] 2633 [vgl. 38] und OpenPGP in der RFC 4880 [vgl. 7] spezifiziert. Da eine inhaltsbasierte Filterung nicht Teil dieser Arbeit ist, werden PKI, wie auch S/MIME und OpenPGP nicht weiter erläutert. Zur Vertiefung können die RFC´s oder [43] genutzt werden.

1.4. Zielgruppe der Arbeit

Diese Arbeit richtet sich an jeden, der ein Interesse an der Funktionsweise von E-Mail-Systemen und im speziellen der Kategorisierung von E-Mails durch Spamfilter hat. Um dem Inhalt der Arbeit folgen zu können, sind grundlegende Kenntnisse in den Bereichen von Rechnernetzen und Betriebssystemen sinnvoll. Dabei stehen das OSI-Referenzmodell und die TCP/IP-Protokollfamilie im Mittelpunkt. Eine besondere Rolle spielen dabei das "Simple Mail Transfer Protocol" und das "Internet Message Format".

1.5. Struktur der Arbeit

Diese Arbeit strukturiert sich wie folgt: Das zweite Kapitel erläutert grundlegende Verfahren und Begriffe. Es wird der Aufbau einer E-Mail sowie die Funktionsweise eines E-Mail-Systems beschrieben. Die zentralen Begriffe, z.B. Spam, Malware, Spoofing, Social Engineering und Phishing, werden veranschaulicht und erklärt. Des Weiteren werden bekannte Erkennungs- und Korrelationsmechanismen, wie z.B. sender-, regel- und inhaltsbasierte Filterung aufgezeigt und erläutert. Der Abschluss dieses Kapitels verdeutlicht die rechtlichen Aspekte, die bei einer inhaltsbasierten Filterung von E-Mails relevant sind.
Im dritten Kapitel werden die Header-Informationen von E-Mails auf mögliche Kriterien analysiert. Aus den daraus resultierenden Prüfkriterien wird ein Filterungsverfahren entwickelt, welches im vierten Kapitel mit der Programmiersprache Python umgesetzt wird. Das fünfte Kapitel zeigt die Möglichkeiten der Platzierung eines Spamfilters auf und beschreibt eine beispielhafte Integration des in Kapitel 4 umgesetzten Filterungsverfahrens

[10] Request for Comments.

in ein operatives System. Anschließend werden die analysierten und die über "WEB.DE" empfangenen E-Mails mit dem entwickelten Filterungsverfahren evaluiert.
Die Zusammenfassung, das Fazit und der Ausblick bilden im sechsten Kapitel den Abschluss der Arbeit.

2. Grundlagen

Das Kapitel erläutert den Aufbau einer E-Mail und die Funktionsweise eines E-Mail-Systems. Es beschreibt zentrale Begriffe, wie z.B. Spam, Malware, Spoofing, Social Engineering und Phishing und stellt einige gängige Erkennungsmechanismen, bspw. "Bayes-Filter", dar. Zum Schluss werden die rechtlichen Aspekte für eine zentrale Filterung des Inhalts einer E-Mail erläutert.

2.1. Aufbau einer E-Mail

Eine E-Mail ("electronic mail") besteht grundsätzlich aus zwei Teilen: dem Header (Kopf) und dem Body (Textkörper).
Der Header einer E-Mail kann verglichen werden mit einem Umschlag eines Briefes, welcher sämtliche Informationen enthält, die zur Übertragung und Lieferung nötig sind. Der Body stellt in diesem Kontext den Brief mit seinem Inhalt dar.

Die RFC 5322 beschreibt das "Internet Message Format" (kurz IMF) und besagt, dass eine Nachricht aus einem Header und einen optionalen Body besteht. Der Header setzt sich aus Kopfzeilenfeldern, nachfolgend Feld genannt, zusammen.
Der Body einer E-Mail besteht aus Zeilen mit Zeichen nach ASCII[11]. Es werden einige Einschränkungen spezifiziert, bspw. das ein Zeilenumbruch nur zusammen mit einem Zeilenvorschub im Body genutzt werden darf. Da der Body einer E-Mail in dieser Arbeit nicht Gegenstand ist, wird dieser nicht weiter betrachtet. [vgl. 39, S. 6]

Die RFC 5322 spezifiziert, dass der Header eine Folge von Zeilen, bestehend aus Zeichen mit einer speziellen Syntax, ist. Der Body hingegen besteht nur aus einer Folge von Zeichen und wird durch eine leere Zeile vom Header getrennt.
Ein Feld beginnt mit einem Feldnamen, wird durch ein Doppelpunkt vom Feldkörper getrennt und endet mit einem Zeilenumbruch. Der Feldname muss sich aus druckbaren Zeichen nach ASCII (Werte zwischen 33-126) zusammensetzen. Der Doppelpunkt darf

[11] American Standard Code for Information Interchange.

nicht verwendet werden, da er als Trennung zwischen Feldnamen und Feldkörper genutzt wird. Für den Feldkörper gilt die gleiche Regelung, allerdings darf der Doppelpunkt, zusätzlich das Leerzeichen (Wert 32) und der Tabulator (Wert 9) verwendet werden. Für Felder, welche die Zeichenlimitierung pro Zeile überschreiten, gibt es die Möglichkeit der mehrzeiligen Darstellung. Diese wird als "Folding" beschrieben und zu Beginn einer Kopfzeile durch ein Leerzeichen kenntlich gemacht.
Es gibt zwei Arten von Feldern, unstrukturierte und strukturierte Felder. Beide müssen aus druckbaren Zeichen bestehen, aber nur strukturierte Felder sind nach bestimmten syntaktischen Regeln aufgebaut. Für diese Arbeit sind strukturierte Felder relevant, da sie besser analysiert werden können. [vgl. 39, S. 7]

Des Weiteren legt die RFC 5322 fest, dass die einzig erforderlichen Felder das Ursprungsdatum und die Absenderadresse(n) sind, alle weiteren sind optional. [vgl. 39, S. 18]
Das Ursprungsdatum ("Date:") enthält einen Zeitstempel, welcher nach syntaktischen Regeln folgendermaßen aufgebaut sein kann: " Mon, 01 Jan 2018 07:00:00 +0200". Das Leerzeichen vor dem Wochentag und dem Tag, sowie die Angabe der Sekunden, sind optional. Des Weiteren kann der Tag ein- oder zweistellig dargestellt werden, somit ist bspw. "1" und "01" erlaubt. Die Zone "+0200" kann auch mit der Abkürzung, hier "GMT"[12] dargestellt werden. [vgl. 39, S. 14] Das "Date:"-Feld beschreibt den Zeitpunkt, in der die Nachricht vom Verfasser vollendet wurde und dem E-Mail-System zum Versenden bereitgestellt wird. Es beschreibt nicht den Zeitpunkt des Versands durch das E-Mail-System. [vgl. 39, S. 21]
Für das Versenden einer Nachricht stehen dem Absender folgende Felder für Absenderadressen zur Verfügung [vgl. 39, S. 21 f.]:

- "From:" - enthält eine oder mehrere, durch Komma getrennte, E-Mail-Adresse(n), welche den oder die Verfasser der E-Mail darstellen.
- "Sender:" - sind im "From:"-Feld mehrere Adressen enthalten, wird das "Sender:"-Feld angehängt. In diesem muss eine Absenderadresse stehen, welche den Verantwortlichen der Nachricht repräsentiert.
- "Reply-To:" - in beiden Fällen kann dieses Feld optional hinzugefügt werden. Dieses enthält eine oder mehrere, durch Komma getrennte, E-Mail-Adresse(n), welche im Falle einer Antwort als Zieladresse(n) genutzt wird/werden.

[12] Greenwich Mean Time.

Für die Absenderadresse ist folgende Syntax spezifiziert: Sie besteht aus einem optionalen Bezeichner und einer E-Mail-Adresse, die von den Zeichen "<" und ">" umschlossen wird (z.B. "bezeichner <e-mail-adresse@domain.de>"). [vgl. 39, S. 15]

Für die Arbeit ist ein weiteres Feld relevant, das "Received:"-Feld. Dieses wird sowohl in der RFC 5322, als auch in der RFC 5321 spezifiziert. Im Rahmen der Arbeit ist die Spezifikation der RFC 5322 relevant. Demnach besteht das "Received:"-Feld aus einer Liste von "Received-Token" und einem Zeitstempel. Die beiden Bestandteile werden durch ein Semikolon getrennt. In der Abbildung 2.1 wird ein "Received:"-Feld dargestellt.

```
Received: from mout.web.de ([212.227.15.3]) by mx-ha.web.de (mxweb112
 [212.227.17.8]) with ESMTPS (Nemesis) id 1MGAoI-1gFsos1mxw-00Gbo2 for
 <spoof-phish@web.de>; Fri, 16 Nov 2018 13:20:30 +0100
```

Abbildung 2.1.: Beispiel eines "Received:"-Feldes

Der Beginn wird mit einer "from"-Klausel, welche die Daten des zustellenden Mailservers enthält, initiiert. Voran gestellt ist der Name des Mailservers und in den runden Klammern folgt der Name des Quellhosts, wie er im HELO-/EHLO-Befehl (siehe Unterabschnitt 2.2.2) angegeben wurde und dessen IP-Adresse in eckigen Klammern. Wird kein Name im HELO-/EHLO-Befehl angegeben, bleibt dieser wie in der Abbildung leer. Danach folgt die "by"-Klausel, welche die Daten des annehmenden Mailservers enthält. Diese Daten werden in der gleichen Syntax, wie beim zustellenden Mailserver angegeben. Anschließend folgen zusätzliche Klauseln, wie bspw. "Via", "With", "Id" oder "For". Diese werden nicht weiter betrachtet, da sie für diese Arbeit nicht relevant sind. [vgl. 39, S. 29] [vgl. 26, S. 58 f.]

Die RFC 5322 spezifiziert weitere Gruppen von Feldern, zum Beispiel Zieladress-, Identifikations- und Informationsfelder. Diese sind nicht relevant und in der RFC auf den Seiten 21-30 dokumentiert.

2.2. Protokolle für den E-Mail-Verkehr

Für die Funktionsweise eines E-Mail-Systems werden Protokolle zum Versenden, Transportieren und Empfangen und physische Server zum Weiterleiten, Speichern und Bereitstellen einer E-Mail benötigt.

2.2.1. POP3 / IMAP4

Das Protokoll "Post Office Protocol Version 3" (kurz POP3) ist in der RFC 1939 [32] und das Protokoll "Internet Message Access Protocol Version 4" (kurz IMAP4) in der RFC 3501 [9] zum Zeitpunkt dieser Arbeit spezifiziert. Beide Protokolle sind für die Verwaltung und das Abrufen von E-Mails von einem Mailserver zuständig, doch sie unterscheiden sich stark voneinander.

POP3 ermöglicht den Zugriff und den Abruf von E-Mails am Mailserver, dabei wird die E-Mail heruntergeladen und anschließend, wenn gewünscht, gelöscht. Operationen zum Verändern der E-Mails am Mailserver sind daher nicht vorgesehen und die Verwaltung der E-Mails findet lokal statt.

Im IMAP4 wiederum wird dem Benutzer der Zugriff und die Manipulation von E-Mails am Mailserver erlaubt, wodurch die E-Mails auf dem Server existent bleiben und von jedem Computer aus abgerufen werden können. Ein Benutzer kann somit auf dem Mailserver neue E-Mails verfassen, bestehende suchen, markieren oder löschen und in einer Ordnerstruktur verwalten.

Die Schnittstelle zwischen dem Benutzer und dem Mailserver übernehmen E-Mail-Anwendungen wie bspw. "Mozilla Thunderbird"[13] oder "Microsoft Outlook"[14], welche beide Protokolle nutzen können. Dem Benutzer wird empfohlen, sich für eine E-Mail-Anwendung auf all seinen Endgeräten zu entscheiden.

[13] URL: https://thunderbird.net/de/.
[14] URL: https://account.microsoft.com/account/outlook.

2.2.2. SMTP

Für diese Arbeit relevanter ist das Protokoll "Simple Mail Transfer Protocol" (kurz SMTP), welches aktuell in der RFC 5321 [26] spezifiziert ist und den Transport und die Übermittlung von E-Mails im Internet regelt. Das Ziel des Protokolls ist es, eine E-Mail zuverlässig und effizient bis zum Ziel zu übertragen.

In der Basisstruktur des SMTP-Modells (siehe Abbildung 2.2) sind vereinfacht der SMTP-Client des Senders und der SMTP-Server des Empfängers dargestellt. Der SMTP-Client ist dafür verantwortlich, dass die Nachricht zu einem oder mehreren SMTP-Server(n) übertragen wird und wenn ein Fehler auftritt, diesen zu melden.

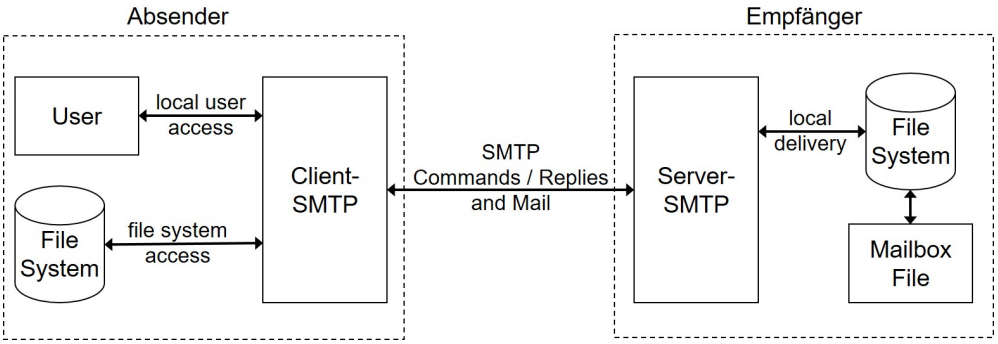


Abbildung 2.2.: SMTP-Modell - Basisstruktur [vgl. 26, S. 6]

Der SMTP-Server kann drei verschiedene Rollen einnehmen [vgl. 17, S. 4 f.]:

- er ist ein "Gateway" (MSA). Er erhält die E-Mail von einem SMTP-Client und liefert sie aus oder fungiert als SMTP-Client und leitet sie weiter.
- er ist ein "Relay" (MTA). Er erhält die E-Mail entweder von einem MSA oder von einem anderen MTA. Er liefert die E-Mail aus oder fungiert als SMTP-Client und leitet sie weiter. Der MTA kann der Zieldomain angehören.
- er gehört der Zieldomain an und die Nachricht wurde übermittelt und per "local delivery" abgelegt, dann spricht man von einem MDA.

Die Erläuterung der Abkürzungen MSA, MTA und MDA erfolgen im Kapitel 2.3, wenn das E-Mail-System als Ganzes betrachtet wird.
Eine Verbindung zum SMTP-Server des Empfängers kann direkt oder über mehrere SMTP-Server erfolgen, die zurückgelegten Instanzen werden als Sprünge (engl. "Hops")

bezeichnet. Die Kommunikation zwischen beiden Instanzen wird von der Clientseite mit Befehlen (engl. "Commands") und von der Serverseite mit Antworten (engl. "Replies") durchgeführt. [vgl. 26, S. 6-8]

Wenn der SMTP-Client eine E-Mail zum Versand bereithält, muss dieser zuerst die zugehörige IP-Adresse des SMTP-Servers der Empfänger E-Mail-Adresse ermitteln. Er stellt eine Anfrage an das "Domain Name System" (kurz DNS), welches nach "Mail Exchange Resource Records" (kurz MX-RR) für die angefragte Domain sucht und sendet die IP-Adresse des SMTP-Servers der Zieldomain zurück.
Daraufhin stellt der SMTP-Client eine bidirektionale TCP-Verbindung zum SMTP-Server her. Nach Abschluss des Handshakes und der erfolgreich aufgebauten Verbindung sendet der SMTP-Server eine Antwort in Form von "220 service ready".
Infolgedessen schickt der SMTP-Client seine Identität mit dem Befehl "HELO" oder "EHLO" (Extended HELO) zum SMTP-Server. Die Wahl dieses Befehls ist abhängig davon, ob der SMTP-Client Serviceerweiterungen verarbeiten kann oder nicht, letzterer gibt an, dass der SMTP-Client mit den Erweiterungen umgehen kann. Jeder Befehl an den SMTP-Server wird von diesem beantwortet. Die Antworten erstrecken sich von der Akzeptanz des Befehls, über die Erwartung weiterer Befehle bis hin zu Fehlern. Hat der SMTP-Server dem SMTP-Client mit "250 OK" geantwortet, wurde der Befehl akzeptiert und der SMTP-Client kann eine Transaktion mit dem Befehl ("MAIL") initiiert. Dieser Befehl gibt die E-Mail-Adresse des Absenders an. Nach bestätigender Antwort kann der SMTP-Client den Empfänger mit ("RCPT") festlegen. Dieser Befehl kann vom SMTP-Client mehrfach gesendet werden, wenn die E-Mail an mehrere Empfänger gesendet werden soll. Nach wiederholter Akzeptanz durch den SMTP-Server, kann der SMTP-Client den Inhalt mit ("DATA") senden. Zum Abschluss des "DATA"-Befehls enthält die letzte Zeile ausschließlich einen Punkt. Wurde dieser Befehl ebenfalls vom SMTP-Server bestätigt, ist die Transaktion abgeschlossen. Der SMTP-Client kann nun die Verbindung mit ("QUIT") beenden oder eine neue Transaktion initiieren. Nach erfolgreicher Transaktion ist der SMTP-Server für die Zustellung oder Fehlerbehandlung der Nachricht verantwortlich. [vgl. 26, S. 17-20]
In der Abbildung 2.3 wird eine beispielhafte Kommunikation zwischen einem SMTP-Client/-Server dargestellt.

```
pi@raspberrypi:~ $ telnet raspberry.localdomain 25
Trying 192.168.2.180...
Connected to raspberry.localdomain.
Escape character is '^]'.
220 raspberry.localdomain ESMTP Postfix (Debian)
EHLO phishi@analyse.ddnss.de
250-raspberry.localdomain
250-PIPELINING
250-SIZE 50000000
250-VRFY
250-ETRN
250-STARTTLS
250-ENHANCEDSTATUSCODES
250-8BITMIME
250-DSN
250 SMTPUTF8
MAIL FROM: phishi@analyse.ddnss.de
250 2.1.0 Ok
RCPT TO: spoofi@analyse.ddnss.de
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Subject: Testnachricht

Dies ist der Inhalt der Testnachricht.

.
250 2.0.0 Ok: queued as A67DB801F7
QUIT
221 2.0.0 Bye
Connection closed by foreign host.
```

Abbildung 2.3.: Beispielhafte SMTP-Kommunikation

Der SMTP-Server sendet einen Fehler als Antwort an den SMTP-Client, wenn die Reihenfolge der festgelegten Befehle nicht stimmt, ein Befehl fehlt oder die Nachricht nicht an den adressierten Empfänger zugestellt werden konnte. Die Antwort stellt einen dreistelligen Code ("reply code") dar. Die erste Ziffer beschreibt den Grad der Antwort, d.h. der Server hat den Befehl akzeptiert (2xx) oder der Befehl war unvollständig (3xx/4xx). Wurde der Befehl vom Server akzeptiert, aber fehlen noch weitere Informationen, antwortet er mit dem Code 3xx. Der Code 4xx besagt, dass der Server den Befehl temporär nicht akzeptiert und die Aktion erneut angefordert werden kann. Wenn der Server ein Code mit 5xx sendet, bedeutet dies, dass der Befehl weder akzeptiert noch ausgeführt wurde. Es handelt sich dabei um einen permanenten Fehler und der SMTP-Client sollte denselben Befehl nicht erneut ausführen.
Die zweite Ziffer kategorisiert die Antwort in spezifische Gruppen und die Dritte gibt detaillierte Informationen zu diesen Gruppen. Eine Vertiefung der zweiten und dritten Ziffer ist für den Kontext dieser Arbeit nicht notwendig. [vgl. 26, S. 47 f.]

2.3. Funktionsweise eines E-Mail-Systems

Nachdem die relevanten Protokolle erläutert wurden, folgt nun die Erläuterung deren Zusammenspiels.

Mit dem "Mail User Agent" (kurz MUA), welcher eine E-Mail-Anwendung darstellt, werden Nachrichten verfasst, gesendet und übermittelte Nachrichten bearbeitet. Das Senden erfolgt wie oben beschrieben über SMTP, unter Berücksichtigung der IMF und das Empfangen über POP3 oder IMAP4. Nach dem Versand durch den MUA gelangt die Nachtricht an den "Message Submission Agent" (kurz MSA), er fungiert als Übermittlungsserver und ist im MUA hinterlegt. Dieser liefert entweder die Nachricht aus oder fungiert als SMTP-Client und leitet sie an einen MTA weiter. Ein "Mail Transfer Agent" (kurz MTA) fungiert als ein SMTP-Server, welcher Nachrichten von einem MSA oder anderen MTA´s akzeptiert und diese entweder ausliefert oder ebenfalls als SMTP-Client agiert, um die Nachricht weiterzuleiten. Bei jeder Weiterleitung wird dem Header ein "Received:"-Feld angehängt, damit der Weg der E-Mail nachvollzogen werden kann. Für den Fall, das eine E-Mail nicht zugestellt werden kann oder weitere Befehle benötigt werden, kann anhand der "Received:"-Felder im Header die Route der Zustellung rekonstruiert werden, um den Absender darüber zu informieren. Wurde eine Nachricht am MTA des Empfängers zugestellt, wird diese durch den "Mail Delivery Agent" (kurz MDA) dem richtigen E-Mail-Konto

("Message Store", kurz MS) zugeordnet. Im Anschluss kann der empfangende MUA mittels POP3 oder IMAP4 die Nachricht abrufen und dem Endbenutzer zur Verfügung stellen. Die nachfolgende Abbildung 2.4 verdeutlicht die Funktionsweise eines E-Mail-Systems. [vgl. 17, S. 4 f.] [vgl. 10, S. 20-22]

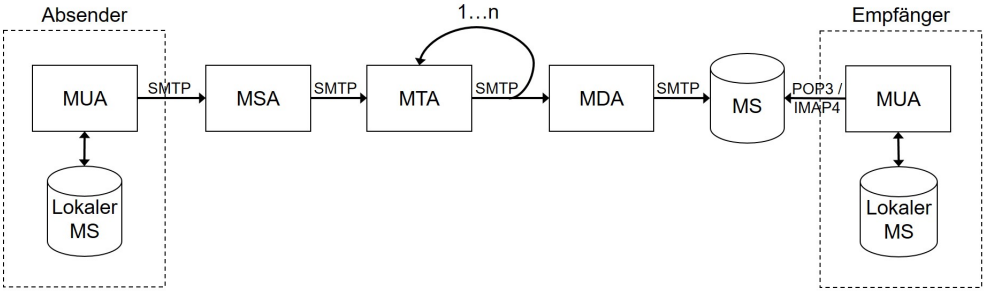


Abbildung 2.4.: Beispielhafte Darstellung eines E-Mail-Systems

2.4. Betrug - verschiedene E-Mail-basierte Strategien

2.4.1. Spam

Für den Begriff Spam gibt es keine einheitliche Definition, häufig verwendete Definitionen sind:

> "An electronic message is "spam" if (A) the recipient's personal identity and context are irrelevant because the message is equally applicable to many other potential recipients; AND (B) the recipient has not verifiably granted deliberate, explicit, and still-revocable permission for it to be sent." [47]

> "Spam is the term now generally used to refer to unsolicited electronic messages, usually transmitted to a large number of recipients. They usually, but not necessarily, have a commercial focus, promoting or selling products or services [...]" [34, S. 6]

In Anlehnung an die obigen Definitionen kann Spam definiert werden als eine elektronische Nachricht, welche unaufgefordert an eine breite Masse von Empfängern versendet wird. Die daraus resultierenden Probleme sind neben der enormen Belastung der weltweiten

IT-Infrastruktur, auch steigende Kosten für die Sicherheit der unternehmerischen IT-Infrastruktur sowie ein Produktivitäts-/Zeitverlust der Endbenutzer. Des Weiteren wird der Grad zwischen Spam und seriösen Werbemails für den Endbenutzer schmaler, wodurch E-Mails seriöser Unternehmen schneller als Spam abgestempelt werden und somit an Akzeptanz verlieren.
Zusätzlich steigt aufgrund der Masse an Spam-Mails auch die Gefahr, sich mit Malware zu infizieren, da das Potenzial schadhafte Anhänge oder Links zu schadhaften Webseiten anzuklicken steigt. [vgl. 6, S. 23 f.]

2.4.2. Malware

Malware setzt sich aus den Wörtern "malicious" und "software" zusammen und wird von der OECD[15] beschrieben als eine Software, die das zu attackierende Informationssystem in seiner Funktionalität beeinträchtigt, beschädigt oder die eine nicht vom Anwender gewollte Nutzung beabsichtigt. Durch Malware können z.B. Fernzugriffe ermöglicht, Daten gesammelt und an Dritte weitergeben werden, ohne das der Betroffene etwas davon mitbekommt bzw. überhaupt sein Einverständnis dazu gegeben hat. Des Weiteren kann die Software ihre Existenz verbergen und gegebenenfalls vorhandene Sicherheitsmechanismen deaktivieren. Malware wird in verschiedene Typen differenziert: Viren, Würmer, Trojaner, Backdoors, Keylogger, Rootkits und Spyware. Diese Typen werden nicht näher erläutert, da ihre Unterschiede keine Relevanz für diese Arbeit haben. [vgl. 35, S. 19 ff.]

2.4.3. Spoofing

Der Begriff Spoofing beschreibt das Vortäuschen einer falschen Identität.
In Bezug auf eine E-Mail bedeutet das, dass versucht wird, die echte Absenderadresse im Header durch eine falsche Absenderadresse auszutauschen (IP-Spoofing). Der Angreifer kann durch die manipulierte IP-Adresse eine falsche Identität vortäuschen und somit seine wahre IP-Adresse verschleiern. Eine Rückverfolgung der E-Mail ist somit nicht mehr möglich. Die gefälschte Absenderadresse soll dem Empfänger suggerieren, dass es sich um eine seriöse Absenderadresse handelt. Dadurch soll der Empfänger zum Öffnen der E-Mail verleitet werden. Ein Ziel dieser Vortäuschung kann die Aussicht auf sensible Daten sein. IP-Spoofing wird im Kontext dieser Arbeit eine große Rolle spielen, da das Erkennen einer gefälschten IP-Adresse auf Spam hindeutet. [vgl. 14, S. 19] [vgl. 43, S. 5]

[15] Organisation for Economic Co-operation and Develop

2.4.4. Social Engineering

Social Engineering ist die Kunst, eine Person auf zwischenmenschlicher Ebene zu manipulieren, unter Zuhilfenahme von Überredungskünsten oder dem Vortäuschen falscher Tatsachen, um sich Informationen zu beschaffen.
Social Engineering wird im Bezug auf E-Mails im Body angewendet und stellt somit keine Relevanz für diese Arbeit dar. [vgl. 31, S. XII]

2.4.5. Phishing

Das Bundesamt für Sicherheit in der Informationstechnik (kurz BSI) beschreibt die Zusammensetzung des Wortes "Phishing" wie folgt:

> "Das Wort setzt sich aus "Password" und "fishing" zusammen, zu Deutsch "nach Passwörtern angeln"." [5]

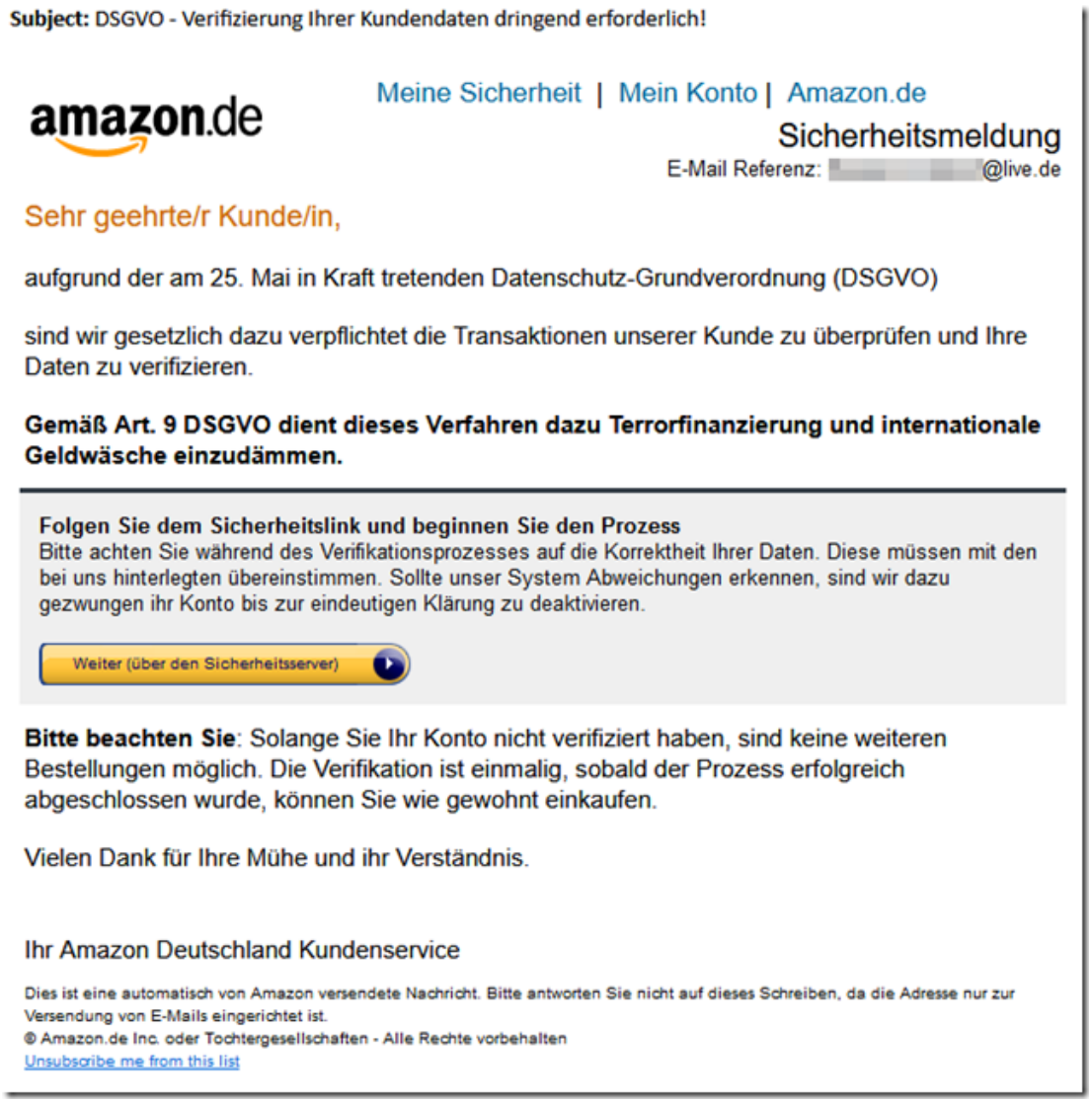
Die APWG[16] definiert den Begriff folgendermaßen:

> "Phishing is a criminal mechanism employing both social engineering and technical subterfuge to steal consumers' personal identity data and financial account credentials." [2, S. 2]

Der Absender versucht mit dem Versand von Phishing-Mails dem Empfänger, unter Verwendung von Social Engineering und technischer Täuschung, persönliche bzw. sensible Daten zu stehlen. Die Phishing-Mail wird mit einer falschen Absenderadresse so manipuliert, dass sie den Anschein erweckt, von einer namentlich bekannten Firma (bspw. Amazon[17]) abgesendet worden zu sein. So wurde bspw. im Rahmen des Inkrafttretens der neuen DSGVO[18] am 25.05.2018 im Namen von Amazon eine Phishing-Mail (siehe Abbildung 2.5) versendet.
Der Empfänger soll durch den Inhalt so unter Druck gesetzt werden, teils durch Androhung von z.B. Kontoeinschränkungen, dass er den darin befindlichen Link klickt. Durch das Klicken auf den Link wird der Nutzer auf meist optisch identische Nachbauten der Originalseiten weitergeleitet und zur Eingabe seiner Account-Daten, Kreditkarteninformationen oder ähnlich sensiblen Daten aufgefordert, die für weitere Aktionen missbraucht werden sollen. Wenn der Nutzer diese bestätigt, ist der Phishing-Versuch erfolgreich und

[16] Anti-Phishing Working Group.
[17] Amazon.com, Inc.
[18] Datenschutz-Grundverordnung.

der Nutzer wird entweder auf die Originalseite weitergeleitet, in welcher er mit dem von ihm bereitgestellten Account-Daten eventuell automatisch eingeloggt wird oder er bekommt einen Fehler angezeigt. [vgl. 2, S. 2] [vgl. 36, S. 10] [vgl. 48]
Eine weitere Phishing-Attacke bietet versteckte Malware, welche angehängt, eingebettet oder über einen angeklickten Link auf den Rechner des Nutzers gelangt. [vgl. 2, S. 2]



Subject: DSGVO - Verifizierung Ihrer Kundendaten dringend erforderlich!

amazon.de

Meine Sicherheit | Mein Konto | Amazon.de

Sicherheitsmeldung

E-Mail Referenz: [redacted]@live.de

Sehr geehrte/r Kunde/in,

aufgrund der am 25. Mai in Kraft tretenden Datenschutz-Grundverordnung (DSGVO)

sind wir gesetzlich dazu verpflichtet die Transaktionen unserer Kunde zu überprüfen und Ihre Daten zu verifizieren.

Gemäß Art. 9 DSGVO dient dieses Verfahren dazu Terrorfinanzierung und internationale Geldwäsche einzudämmen.

Folgen Sie dem Sicherheitslink und beginnen Sie den Prozess
Bitte achten Sie während des Verifikationsprozesses auf die Korrektheit Ihrer Daten. Diese müssen mit den bei uns hinterlegten übereinstimmen. Sollte unser System Abweichungen erkennen, sind wir dazu gezwungen ihr Konto bis zur eindeutigen Klärung zu deaktivieren.

Weiter (über den Sicherheitsserver)

Bitte beachten Sie: Solange Sie Ihr Konto nicht verifiziert haben, sind keine weiteren Bestellungen möglich. Die Verifikation ist einmalig, sobald der Prozess erfolgreich abgeschlossen wurde, können Sie wie gewohnt einkaufen.

Vielen Dank für Ihre Mühe und ihr Verständnis.

Ihr Amazon Deutschland Kundenservice

Dies ist eine automatisch von Amazon versendete Nachricht. Bitte antworten Sie nicht auf dieses Schreiben, da die Adresse nur zur Versendung von E-Mails eingerichtet ist.
© Amazon.de Inc. oder Tochtergesellschaften - Alle Rechte vorbehalten
Unsubscribe me from this list

Abbildung 2.5.: Phishing-Mail am Beispiel der Firma Amazon.com, Inc. [55]

2.5. Erkennungs- und Korrelationsmechanismen für Spam

Erkennungs- und Korrelationsmechanismen werden eingesetzt um eine E-Mail als (un-)erwünscht zu klassifizieren. Der Spamfilter kann eine E-Mail als Spam (positiv) oder Nicht-Spam (negativ) einordnen. Der Endbenutzer kann der Entscheidung des Spamfilters zustimmen (wahr) oder nicht (falsch). Daraus ergeben sich vier Kategorien (siehe Tabelle 2.1), in welche eine E-Mail klassifiziert werden kann.
Wird eine E-Mail sowohl vom Spamfilter, als auch vom Endbenutzer als Spam eingestuft, wird von "True Positive" (kurz TP) gesprochen. Kategorisieren beide die E-Mail als Nicht-Spam, wird diese als "True Negative" (kurz TN) bezeichnet. Bei "False Positive" (kurz FP) klassifiziert der Spamfilter die E-Mail als Spam, der Endbenutzer jedoch nicht. Im umgekehrten Fall wird von "False Negative" (kurz FN) gesprochen.
Das Ziel eines Spamfilters ist es, die Erkennungsrate von Spam gegen 100% zu konvergieren und eine fehlerhafte Klassifizierung (FP/FN) zu vermeiden. Der Fokus liegt dabei auf der Vermeidung von FP, da eine Kategorisierung einer Nicht-Spam-Mail als Spam-Mail weitaus kritischer ist, als andersherum. [vgl. 40, S. 80]

		Benutzer	
		Spam (positive)	Nicht-Spam (negative)
Spamfilter	Spam (positive)	True Positive	False Positive
	Nicht-Spam (negative)	False negative	True negative

Tabelle 2.1.: Klassifizierung einer E-Mail

2.5.1. Senderbasierte Filterung

Im Jahr 1997 begannen Paul Vixie und Dave Rand, zwei bekannte Internet-Software-Ingenieure, eine Liste von IP-Adressen zu führen. Diese IP-Adressen verschickten Spam oder E-Mails mit störenden Inhalt. Sie veröffentlichen die Liste als "Border Gateway Protocol feed" (kurz BGPf) für Personen, die den gesamten Verkehr der aufgelisteten IP-Adressen an ihren Routern blockieren wollten. Die Liste wurde als "Realtime-Blackhole-List" (kurz

RBL) bekannt. Viele Netzwerkadministratoren waren nicht auf das "Border Gateway Protocol feed" vorbereitet und konnten die Liste nicht nutzen. Daraufhin entwickelten Vixie und Rand ein DNS-basiertes Verteilungsschema, was deutlich populärer wurde. Dieses ist unter den Namen "DNS-Blacklist" oder "DNS-Blocklist" (kurz DNSBL) bekannt. Im Gegensatz dazu werden IP-Adressen von erwünschten E-Mail-Absendern in eine "Whitelist" eingetragen. Das DNS-basierte Verteilungsschema nennt sich dann "DNS-Whitelist" (kurz DNSWL). [vgl. 29, S. 1]
Eine DNS-basierte Liste ist eine Zone im DNS. Ein Host wird durch eine Umwandlung der IP-Adresse (manchmal der Hostname), welche an die Struktur des "reverse DNS"[19] (kurz rDNS) angepasst ist, in eine Zone codiert.
Ein Beispiel: der Name der DNSBL ist "blacklist.beispiel.de" und von der IP-Adresse "211.14.56.99" werden Spam-Mails empfangen. Um den Empfang zu blockieren wird die IP-Adresse mittels rDNS in "99.56.14.211" umgewandelt und der Zone hinzugefügt. Der Eintrag in der Zone der DNSBL sieht folgendermaßen aus "99.56.14.211.blacklist.beispiel.de". Sendet diese IP-Adresse erneut eine E-Mail, wird die IP-Adresse mit rDNS umgewandelt und überprüft, ob diese in der DNSBL enthalten ist. [vgl. 29, S. 2]
Der Nachteil beider Listen ist das Entfernen von Einträgen, die nicht den Zweck der Liste erfüllen, d.h. wurde bspw. Spam von einer IP-Adresse geschickt, welche sich in der Whitelist befindet, dann muss diese manuell entfernt werden. Des Weiteren erhöht sich bei DNS-basierten Listen die Belastung des DNS durch die rDNS-Abfragen [vgl. 24].
Der Vorteil einer Blacklist ist, dass bei einer Übereinstimmung die Verbindung abgebaut wird, bevor der SMTP-Dialog startet. Bei der Nutzung einer Whitelist könnten nachfolgende Spamfilter entlastet werden, da der Absender erwünscht ist. Problematisch wird es, wenn ein Spammer einen solchen Absender herausbekommt, da er ungehindert die Whitelist passieren kann (IP-Spoofing). Eine Blacklist oder Whitelist sollte nur in Kombination mit anderen Filterungsverfahren genutzt werden.

[19] Für IPv4-Adressen "IN-ADDR.ARPA" (RFC2317) und für IPv6-Adressen "IP6.ARPA" (RFC3596).

2.5.2. Regelbasierte Filterung

Für die Analyse und Kategorisierung der E-Mails werden Regeln definiert. Diese Regeln analysieren den Header oder den Body auf bestimmte Wörter oder Muster und bewerten die E-Mail. Ein Algorithmus entscheidet nach jeder bewerteten Regel, wie viele Punkte auf einen Wert addiert oder subtrahiert werden. Wird ein vordefinierter Wert überschritten, wird die E-Mail als Spam klassifiziert. Der Nachteil dieser Filterung ist, dass die definierten Regeln ggf. aufwändig angepasst werden müssen, wenn neue Spam-Techniken entwickelt werden. [vgl. 8, S. 8]

2.5.3. Inhaltsbasierte Filterung

Als Spam noch am Anfang stand, konnte auf Grund von eindeutiger Schlüsselwörter (z.B. "Viagra", "Penis", "Porn") oder Inhalte mit gebrochener Sprache, das Kategorisieren einer E-Mail schnell und effizient erfolgen. Da sich Spamfilter stetig verbessern, hat dies zur Folge, dass auch die Spammer ihre Techniken und Methoden überarbeiten und verbessern. Ein Ansatz war es bspw. den Inhalt in Kombination mit "Leetspeak" darzustellen, um Spamfilter zu täuschen. "Leetspeak" beschreibt das Ersetzen von Buchstaben durch ähnlich aussehende Zahlen oder Symbole, so wurde z.B. "V1AGR4" statt "Viagra" geschrieben. [vgl. 44, S. 102]
Mittlerweile ist Spam inhaltlich schwer zu erkennen, da die Ausführung der Inhalte stets verbessert wird. Die Schlüsselwörter werden gut umschrieben, durch textuelle Methoden oder durch Grafiken versteckt. Um in der heutigen Zeit Inhalte schnell und effizient zu analysieren, werden selbstlernende Systeme eingesetzt. Das meist benutzte System dafür ist der "Bayes-Filter".

Im Vergleich zu regelbasierten Spamfiltern muss das System, vor seinem Einsatz, mit bereits kategorisierten (un-)erwünschten E-Mails trainiert werden. Ein weiterer Nachteil ist, dass die Wörterdatenbank jederzeit aktuell gehalten werden muss, damit Wörter in den E-Mails erkannt werden können. Bei lernenden Systemen, wie diesem, müssen keine Regeln definiert werden. Ein weiterer Vorteil ist, dass das System durch eine falsche Klassifizierung aus diesem Fehler selbst lernt und sich so schnell an neue Arten von Spam-Mails anpassen kann.

Der "Bayes-Filter" basiert auf der Formel von Thomas Bayes zur Berechnung der bedingten Wahrscheinlichkeit. Die Formel ermittelt die Wahrscheinlichkeit für ein Ereignis, unter der Bedingung, dass ein anderes Ereignis eingetreten ist. Mathematische Formeln werden an dieser Stelle nicht betrachtet, da diese für die Arbeit nicht relevant sind [vgl. 41].
Der "Bayes-Filter" analysiert nach vollständiger Übermittlung den Inhalt der E-Mail. Hierbei werden sowohl das "Subject:"-Feld des Headers, als auch der Body der E-Mail durchsucht. Er berechnet dabei für jedes Wort die Wahrscheinlichkeit für das Vorkommen in einer (un-)erwünschten E-Mail und speichert diese in einer Datenbank ab. Nach der Trainingsphase wird beim Analysieren einer neuen E-Mail ermittelt, wie wahrscheinlich es ist, dass es sich dabei um Spam handelt. Hierfür werden n Wörter der E-Mail ausgewählt, die häufig in (un-)erwünschten E-Mails vorkommen. Dabei sollte n nicht zu groß sein, da sonst eine fehlerhafte Kategorisierung auftreten kann. Anhand dieser gewählten Wörter wird die Wahrscheinlichkeit berechnet, dass es sich bei dieser E-Mail um Spam handelt. [vgl. 16, S. 400]
Sahami u.a. erläutern, dass eine Klassifizierung von Nicht-Spam als Spam (FP) mehr ins Gewicht fällt, als wenn Spam als Nicht-Spam (FN) eingeordnet wird. Eine E-Mail sollte daher nur als Spam klassifiziert werden, wenn die Wahrscheinlichkeit für Spam mehr als 99,9% beträgt. [vgl. 41, S. 58 f.]
Damit wird der Schwellenwert auf 99,9% festgelegt. Ein Unterschreiten würde bedeuten, dass mindestens eine E-Mail als FP eingestuft wurde, was zu vermeiden ist.

Zur Visualisierung von FP eingeordneten E-Mails im Verhältnis zu allen TP klassifizierten E-Mails und zur optischen Bestimmung eines optimalen Schwellenwertes kann eine "Receiver-Operating-Characteristic-Kurve" (kurz ROC-Kurve) eingesetzt werden. Diese beschreibt auf der Ordinatenachse die Treffergenauigkeit für eine Spam-Mail und auf der Abszissenachse die Fehlerrate. Für die Treffergenauigkeit werden alle positiv klassifizierten E-Mails des Spamfilters im Verhältnis zu allen tatsächlich positiv eingeordneten E-Mails gesetzt. Die Fehlerrate beschreibt die E-Mails, die fälschlich als Spam klassifiziert wurden. Die Kurve beginnt im Koordinatenursprung, da alle E-Mails als Nicht-Spam kategorisiert werden können und somit keine Fehlerrate im Bezug auf TP aufweisen. Das Ende der Kurve ist in der rechten oberen Ecke des Koordinatensystems, da alle E-Mails als Spam eingeordnet werden können und somit auch alle Nicht-Spam-Mails betroffen sind. Der Verlauf der Kurve wird aus den relativen Werten der Treffergenauigkeit und der Fehlerrate

unter Verwendung verschiedener Schwellenwerte bestimmt.
Zur Ermittlung des optimalen Schwellenwertes wird eine Diagonale vom Koordinatenursprung zur rechten oberen Ecke des Koordinatensystems gezogen. Von der Diagonale aus wird im Lot der weit entfernteste Punkt auf der Kurve ermittelt. Der ermittelte Punkt stellt den optimalen Schwellenwert dar, weil er das beste Verhältnis zwischen Treffergenauigkeit und Fehlerrate hat. [vgl. 30, S. 288]

2.5.4. Hybride Filterung

Die Kombination aus sender-, regel- und inhaltsbasierter Filterung beschreibt eine hybride Filterung. Apache SpamAssassin ist die bekannteste Open-Source Anti-Spam-Plattform, welche einen Filter zur Klassifizierung für Spam anbietet. Dieser wird wie folgt beschrieben:

> "It uses a robust scoring framework and plug-ins to integrate a wide range of advanced heuristic and statistical analysis tests on email headers and body text including text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases." [45]

2.6. Rechtliche Aspekte bei der inhaltsbasierten Filterung von E-Mails

Die inhaltliche Analyse einer E-Mail zählt zu den häufig verwendeten und effektivsten Mechanismen zur Klassifizierung von Spam. Wird diese allerdings auf Basis einer zentralen Filterung betrachtet, kann es zu erheblichen Auseinandersetzungen mit der Rechtsprechung kommen.

Im Behörden- oder Unternehmensumfeld kann der private Umgang mit E-Mails gestattet, geduldet oder untersagt werden. Unabhängig davon, kann der Einsatz eines Spamfilters nur mit Zustimmung des Betriebsrates eingesetzt werden. Im §87 Abs. 1 Nr. 6 BetrVG wird das Mitbestimmungsrecht des Betriebsrates hinsichtlich technischer Einrichtungen, wie folgt geregelt:

> "(1) Der Betriebsrat hat, soweit eine gesetzliche oder tarifliche Regelung nicht besteht, in folgenden Angelegenheiten mitzubestimmen: [...] 6. Einführung und Anwendung von technischen Einrichtungen, die dazu bestimmt sind, das Verhalten oder die Leistung der Arbeitnehmer zu überwachen; [...] "

Unter technische Einrichtungen fallen somit auch Spamfilter, da sie nicht nur eingehende sondern auch ausgehende E-Mails klassifizieren und somit das Verhalten und die Leistung des Arbeitnehmers überwachen. Somit könnte bspw. festgestellt werden, ob ein Arbeitnehmer Spam intern bzw. nach extern versendet.
Der Umgang mit E-Mails kann für den Arbeitnehmer durch eine Betriebsvereinbarung, klar und transparent geregelt werden. Diese kann ebenfalls die Verwendung einer zentralen Filterung, unabhängig ob ein Betriebsrat existiert oder nicht, regeln.
Wird die private Nutzung gestattet oder geduldet, verstößt eine zentrale, inhaltsbasierte Filterung, egal ob automatisiert oder durch einen Administrator durchgeführt, gegen die Wahrung des Fernmeldegeheimnisses (siehe § 88 Abs. 2 TKG). Daher ist eine Filterung des Inhalts trotz Betriebsvereinbarung nicht möglich, da dies gegen das Grundgesetz verstößt (siehe Art. 10 GG). Jegliche Verletzung des Fernmeldegeheimnisses kann bestraft werden (siehe § 206 StGB). In diesem Fall wird eine E-Mail nur markiert und an den Empfänger weitergeleitet.
Der Einsatz einer inhaltsbasierten Filterung auf Ebene der E-Mail-Anwendung ist möglich. Der Empfänger der E-Mail erhält dadurch die vollste Kontrolle, ob und wie gefiltert wird. [vgl. 15] [vgl. 28]

3. Entwicklung von headerbasierten Prüfungen

In diesem Kapitel wird der Header unter Beachtung der Plausibilität auf mögliche Kriterien untersucht. Aus den daraus resultierenden Kriterien, die sich für headerbasierte Prüfungen eignen, wird ein Filterungsverfahren entwickelt. Dieses dient als Grundlage für die darauf folgende Umsetzung.

3.1. Analyse des E-Mail-Headers zur Ermittlung von Prüfkriterien

Damit geeignete Prüfungen entwickelt werden können, um Spam möglichst effektiv zu filtern, müssen häufig auftretende Gemeinsamkeiten oder Unterschiede zwischen Nicht-Spam- und Spam-Mails identifiziert werden. Im Rahmen dieser Arbeit werden die Header von 70 E-Mails, welche sich in 24 Nicht-Spam- und 46 Spam-Mails kategorisieren, analysiert (siehe Anhang C.1 und C.2). Die E-Mails repräsentieren den typischen Erhalt von Spam- und Nicht-Spam-Mails im Alltag, auf der einen Seite Spam-Mails, die über internationale ESP´s oder eigene Domains inkl. eines Mailservers mit bzw. ohne Verwendung eines Open Relays verschickt werden und auf der anderen Seite Nicht-Spam-Mails, die nationale und internationale Newsletter darstellen.

Aus dem Abschnitt 2.1, dem Unterabschnitt 2.2.2 und dem Vergleich einiger Spam- und Nicht-Spam-Mails (siehe Anhang A) resultieren nachfolgende Kriterien.

Kriterium 1 (K1)

Der erste Ansatz ist das Identifizieren von E-Mails mit gefälschten Adressen. Unterscheidet sich die Hauptdomain des Feldes "Reply-To:" (falls vorhanden), von der des Feldes "From:", handelt es sich um einen möglichen Phishingversuch. Weicht die Hauptdomain des Feldes "Return-Path:" von der des Feldes "From:" ab, handelt es sich möglicherweise um eine

DDoS-Attacke[20] auf die eingetragene Adresse. Die Adresse in dem Feld "Return-Path:" wird verwendet, um den vermeintlichen Absender darüber in Kenntnis zu setzen, dass seine E-Mail nicht zugestellt werden konnte. Diese Nachricht wird als "Bounce-Nachricht" bezeichnet.
Bei Betrachtung der zu analysierenden E-Mails wird deutlich, das alle Nicht-Spam-Mails, die über ein "Reply-To:"-Feld verfügen, dieselbe Subdomain bzw. mindestens dieselbe Hauptdomain in den Feldern "From:" und "Reply-To:" aufweisen. Ein fehlendes "Reply-To:"-Feld ist kein Ausschlusskriterium, da es optional ist. Drei der als Spam klassifizierten E-Mails erfüllen die oben genannte Anforderung nicht und werden als Phishing-Mails mit gefälschten Absenderadressen identifiziert. Der Vergleich mit dem Feld "Return-Path:" zeigt auf beiden Seiten deutliche Abweichungen, im Bezug auf die Hauptdomain des Feldes "From:" und wird daher nicht weiter berücksichtigt.

Kriterium 2 (K2)
Ein weiterer Ansatz ist der Vergleich der geographischen Daten, der in dem Feld "From:" enthaltenen Domain und dem ersten öffentlichen MTA ("Received:"-Feld). In der Regel ist die Domain sowie der dafür zuständige Mailserver beim selben Provider, wodurch die geographische Lage mindestens das selbe Land aufweisen sollte. Die Analyse aller E-Mails ergibt, dass 63% der Spam-Mails und 79% der Nicht-Spam-Mails dieses Kriterium erfüllen. Damit ist das Kriterium nicht aussagekräftig genug, um es als Prüfung zu verwenden.

Kriterium 3 (K3)
Der erste öffentliche MTA ist ein mögliches Kontrollelement. Dieser wird im Header mit seinem Namen und der IP-Adresse im "Received:"-Feld eingetragen. Die IP-Adresse ist zugewiesen und nicht fälschbar. Der Name des MTA´s wird vom Betreiber festgelegt und kann gefälscht werden. Der Grund hierfür ist, ähnlich wie bei Phishing-Mails, einen seriösen Anbieter vorzutäuschen und ggf. vorhandene Spamfilter zu täuschen. Es ist möglich, dass der Name auch eine IP-Adresse darstellt, um ein anderes Herkunftsland vorzutäuschen. Ein gefälschter Name ist ein Indiz für ein sogenanntes "Open Relay", d.h. der MTA wurde bewusst falsch konfiguriert. Er ist weder für die Absender- noch für die Empfängeradresse zuständig und leitet die E-Mails weiter. Spammer nutzen sehr bewusst "Open Relays", um ihre Identität zu verschleiern.

[20] Distributed-Denial-of-Service beschreibt einen verteilten Angriff auf eine IT-Komponente, mit dem Ziel die Verfügbarkeit einzuschränken oder zu verhindern.

Dieses Kriterium untersucht den Namen, welcher aus der Namensauflösung der IP-Adresse resultiert. Stimmt der aufgelöste Name mit dem angegebenen Namen des MTA´s überein, liegt kein manipulierter MTA vor. Problematisch sind komplett gefälschte "Received:"-Felder, bei denen die Namensauflösung der IP-Adresse mit dem angegebenen Namen des MTA´s übereinstimmt. Diese wurden dem Header "manuell" angehängt, ohne den MTA je durchlaufen zu haben und können nicht erkannt werden.
Infolgedessen kann der Spammer ein gefälschtes, aber syntaktisch korrektes "Received:"-Feld mit einer öffentlichen IP-Adresse, vor der Nutzung eines "Open Relays" hinzufügen. Würde ausschließlich der erste öffentliche MTA überprüft, hätte die E-Mail dieses Kriterium erfüllt. Damit der Fall nicht eintritt, wird bei allen sendenden MTA´s, die eine öffentliche IP-Adresse besitzen, die aufgelöste IP-Adresse mit dem angegebenen Namen des MTA´s verglichen.
Die Analyse ergibt, dass alle Nicht-Spam-Mails und nur 33% der Spam-Mails dieses Kriterium erfüllen.

Kriterium 4 (K4)

Unter Berücksichtigung der geographischen Lage des MTA´s besteht die Möglichkeit, nach Herkunftsland zu filtern. Die Abbildung 3.1 stellt ein Kreisdiagramm mit den Top 15 Herkunftsländern von Spam-Mails dar, welches sich aus den durchschnittlichen Anteilen der Jahre 2015 [53], 2016 [19], 2017 [20] und dem ersten Halbjahr von 2018 (Q1 [12], Q2 [52]) zusammensetzt. Die Vermutung, dass in diesen Ländern vorallem "Open Relays" genutzt werden, liegt nah.
Die Auswertung der E-Mails ergibt, dass eine Filterung nach Herkunftsland nur sinnvoll ist, wenn das eigene Land ausgeschlossen wird. Der gewollte Empfang von E-Mails aus dem Ausland muss ebenfalls berücksichtigt werden. Aufgrund dieser Einschränkungen ist eine Filterung, ausschließlich nach diesem Kriterium, nicht realitätsnah.

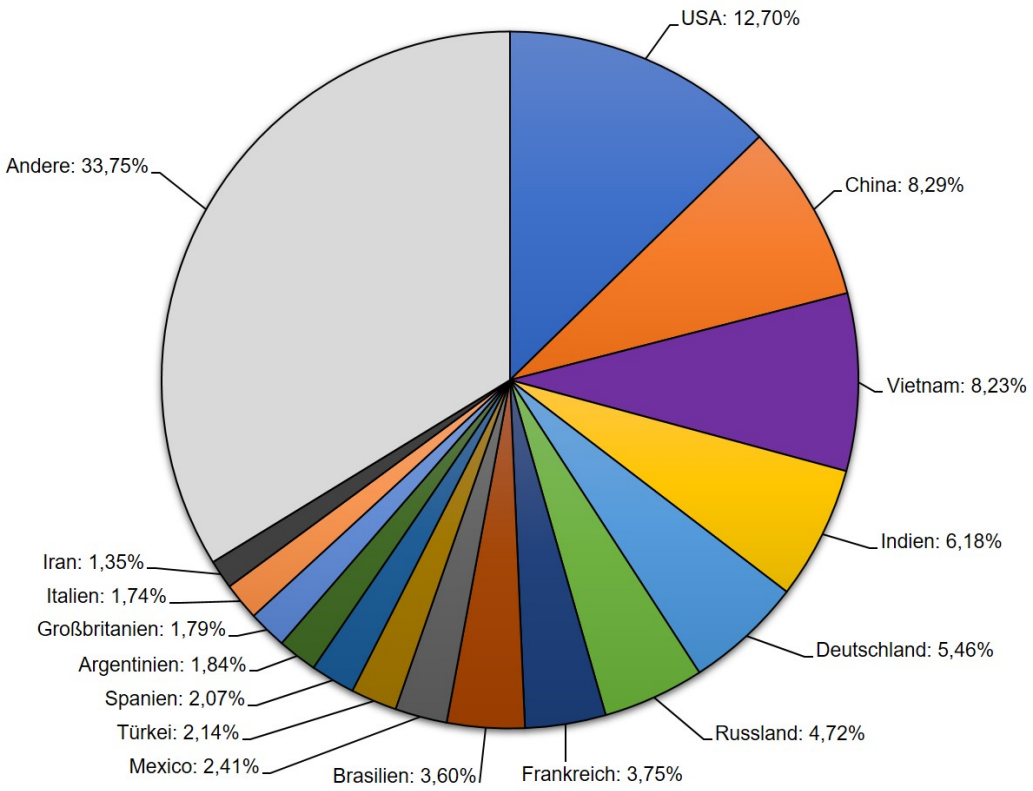


Abbildung 3.1.: Top 15 - Herkunftsländer von Spam-Mails (2015-2018)

Kriterium 5 (K5)

Das Kriterium vergleicht die geographische Lage der Subdomain, aus dem "From:"-Feld, mit der Lage der übergeordneten Domain, z.B. wird die Subdomain "account.microsoft.com" mit der übergeordneten Domain "microsoft.com" verglichen. In diesem Fall ist "account" die Subdomain der Domain "microsoft.com". Beide Domains sollten mindestens das gleiche Land, im Bezug auf ihre geographische Lage, aufweisen. Durch den Vergleich soll verhindert werden, das eine Subdomain seine Herkunft verschleiert, indem die übergeordnete Domain in einem "vertrauenswürdigen" Land registriert/gekauft wurde und die Subdomain im Ursprungsland verwendet wird.
Bei der Untersuchung der E-Mails wird deutlich, dass 94% der Spam- und 71% der Nicht-Spam-Mails eine Übereinstimmung der geographischen Lage aufweisen. Die Tatsache kann damit begründet werden, dass vor allem internationale Unternehmen für ihre Internetpräsenz länderspezifische TLD´s nutzen, welche über einen Anbieter registriert werden. Die Subdomain verweist dann wiederum auf ein anderes Land, in welchem der Betreiber seine IT-Infrastruktur zentral (z.B. in einem Rechenzentrum) für weitere Länder verwaltet. Dies trifft im bereits oben verwendeten Beispiel für Microsoft[21] zu. Die Subdomain "account.microsoft.com" befindet sich in der Niederlande, während sich die übergeordnete Domain "microsoft.com" in den USA befindet.
Aufgrund dieses Resultats entfällt das Kriterium.

Kriterium 6 (K6)

Jede Domain die einen MX-RR besitzt, hat mindestens einen für ihn zuständigen MTA. Dieser kann über die Konsole mit dem Befehl "nslookup" und den zusätzlichen Parametern "set type=MX" und "Domain oder IP-Adresse" abgefragt werden. Die Abfrage zeigt eine Liste mit den zuständigen MTA´s an. Wird über die Domain eine E-Mail versendet, nimmt einer der aufgelisteten MTA´s diese an. Die Annahme wird im Header der E-Mail mit einem "Received:"-Feld quittiert. Ein Vergleich zwischen dem ersten öffentlichen MTA, der die E-Mail angenommen hat, mit der Liste der zuständigen MTA´s der Absenderdomain soll einen Treffer ergeben. Gibt es keine Übereinstimmung liegt der Verdacht nahe, dass die Absenderadresse gefälscht ist.
Die Auswertung der Untersuchungsmenge ergibt, dass 46% der Nicht-Spam- und 11% der Spam-Mails dieses Kriterium erfüllen. Die Ursache für das Ergebnis ist, das ein- und

[21] URL: https://microsoft.com/.

ausgehende E-Mails voneinander getrennt werden, d.h. es existiert jeweils ein MTA. Der MTA der für die Domain zuständig ist, entspricht nicht dem ersten öffentlichen MTA. Das Kriterium kann somit nicht alleinig zum Filtern verwendet werden.

Kriterium 7 (K7)

Mit dem Kriterium wird der zeitliche Ablauf der E-Mail untersucht. Insbesondere wird die Dauer der Zustellung und die Plausibilität der Zeitstempel ("Date:", "Received:") betrachtet. Der Zeitstempel im "Date:"-Feld wird bei der Bereitstellung der E-Mail und nicht beim Versand durch den MUA gesetzt. Eine E-Mail, die zu einem späteren Zeitpunkt versendet werden soll, wird solange zwischengespeichert und erst zu diesem Zeitpunkt bereitgestellt. Bei Annahme der E-Mail dokumentiert der MTA den Zeitstempel im "Received:"-Feld. Die Zeitstempel werden in Abhängigkeit zur Systemuhr gesetzt und sind daher fälschbar. Das Vertrauen an den zustellenden MTA dient als Basis zur Berechnung für die Dauer der Zustellung.
Die Analyse der nicht als Spam kategorisierten E-Mails ergibt, dass alle E-Mails, ausgenommen vier E-Mails, nicht länger als 1-2 Minuten für die Zustellung benötigten. Die vier ausreißenden E-Mails teilen sich in folgende Zustellungszeiten auf: zwei innerhalb von 10 Minuten, eine innerhalb von 20 Minuten und eine benötigte 82 Minuten. Grund für diese enorme verzögerte Zustellung kann ein genutztes Greylisting des MTA´s der Zieldomain sein. Der Ausreißer dient als Richtlinie für die weitere Analyse der Spam-Mails. Damit FP vermieden werden, wird ein Schwellenwert von 90 Minuten festgelegt.
Unter Berücksichtigung des Schwellenwertes erfüllen 89% der Spam-Mails dieses Kriterium, somit lässt sich eine geringe Anzahl an Spam deutlich filtern.

Kriterium 8 & 9 (K8/K9)

Die Kriterien acht und neun setzen sich mit dem Zeitpunkt der Bereitstellung und dem Zeitpunkt der Zustellung der E-Mail auseinander. Es wird betrachtet, ob die Zeitpunkte in den üblichen Arbeitszeiten (07:00 - 21:00 Uhr) liegen. Ist dies nicht der Fall, so deutet das auf Spam hin.
Die Untersuchungen ergeben, dass Spam-Mails zu 80% und Nicht-Spam-Mails zu 92% die beiden Kriterien erfüllen. Eine realistische Ausweitung der oben genannten Arbeitszeiten und unter Berücksichtigung des Wochentages verbessert das Resultat nicht. Somit können die beiden Kriterien als alleinige Prüfungen nicht eingesetzt werden.

Kriterium 10 (K10)

Es wird untersucht, ob die Zeitzone des "Date:"-Feldes mit der des ersten öffentlichen MTA´s übereinstimmt. Anwender nutzen in der Regel nationale Dienste, wodurch sich die Zeitzone nicht unterscheidet.
Die Analyse der Test-Mails ergibt, dass nur 4% der Spam- und 75% der Nicht-Spam-Mails das Kriterium erfüllen. Es kann ein Großteil an Spam gefiltert werden, mit dem Nebeneffekt, das ein Viertel FP klassifiziert wird. Daraus folgt, dass dieses Kriterium im Rahmen der Arbeit nicht allein zu betrachten ist.

Kriterium 11 (K11)

Das Kriterium analysiert die Zeilenlänge und die Formatierung der "Received:"-Felder. Dem IMF (siehe Abschnitt 2.1) nach wird eine maximale Länge von 80 Zeichen pro Zeile empfohlen und bei Benutzung von "folding" soll das erste Zeichen der Zeile ein Leerzeichen sein.
Das Resultat dieser Untersuchung ist, dass 85% der Spam- und 71% der Nicht-Spam-Mails dieses Kriterium erfüllen. Aufgrund der ähnlichen Ergebnisse kann das Kriterium nicht für die Filterung betrachtet werden.

Kriterium 12 (K12)

Jeder MTA dokumentiert seine Annahme und ggf. Weitergabe der E-Mail im Header. Die RFC 5322 spezifiziert das ein "Received:"-Feld nicht gelöscht oder geändert werden darf. Dennoch kann ein MTA dahingehend konfiguriert werden, um bspw. interne IT-Strukturen zu verschleiern. Dadurch kann der Weg einer E-Mail nur bedingt lückenlos nachvollzogen werden. Das Kriterium analysiert die Plausibilität der Weiterleitungen. Wenn vorhanden, dann vom ersten öffentlichen MTA bis zum zustellenden MTA.
Das Ergebnis ist, dass 92% der Nicht-Spam- und 28% der Spam-Mails das Kriterium erfüllen. Es werden dennoch 8% FP erzeugt, wodurch eine alleinige Filterung nicht in Betracht kommt.

Kriterium 13 (K13)

Das Kriterium setzt sich mit dem Erstellungszeitpunkt der Domain und dem Bereitstellungszeitpunkt der E-Mail auseinander. Es wird betrachtet, ob aus der Differenz zwischen den Zeitpunkten signifikante Abweichungen, zwischen Spam- und Nicht-Spam-Mails auftreten.

Das Datum der Erstellung kann mit der Kombination "whois" und der Domain oder IP-Adresse via Konsole (unter Linux) abgefragt werden. Die Untersuchung der Test-Mails zeigt, dass bei der Abfrage für einige TLD´s (z.B. ".de") keine "Creation time" existiert. Stattdessen wird das letzte Datum, an dem die Informationen der Domain geändert wurden, dargestellt. An diesem Datum könnte geschlussfolgert werden, dass die Domain länger existiert. Allerdings könnte bei einer kürzlich erstellten Domain, welche für den Versand von Spam missbraucht werden soll, die ein oder andere Information verändert werden. Dadurch wird die "Change time" aktualisiert und die "Creation time" verschleiert. Daraus folgt, dass dieses Kriterium anhand eines ermittelten Schwellenwertes der Test-Mails nicht sinnvoll erscheint.
Bei Betrachtung der Test-Mails fällt auf, dass drei E-Mails (als Spam klassifiziert) zwischen den beiden oben genannten Zeitpunkten eine Abweichung von null Tagen aufweisen. Das bedeutet, dass am selben Tag die Domain erstellt und zum Versand von E-Mails genutzt wurde. Die Vorstellung, dass am selbigen Tag der Erstellung einer Domain Nutzer sich bspw. für einen Newsletter anmelden, scheint unrealistisch. Infolgedessen kann nach diesem Kriterium klassifiziert werden.

weitere Kriterien

Wang und Chen [vgl. 54] untersuchen in ihrer Arbeit u.a. zwei Header-Felder, welche durch den MUA erstellt werden. Dabei handelt es sich um die Felder "X-Mailer:" und "Message-ID:".
Das "X-Mailer:"-Feld wird aufgrund des "X" als ein nicht erforderliches Feld markiert, d.h. es kann muss aber nicht im Header angegeben werden. Die Untersuchung ergibt, dass die Angabe oder Nichtangabe des Feldes keine eindeutige Klassifizierung ermöglicht, ohne FP zu verursachen. [vgl. 54, S. 386]
Die "Message-ID:" wird in der Regel vom MUA generiert, ist dies nicht der Fall, wird sie vom ersten MTA erzeugt. Generell ist die Domain des Senders ein Bestandteil der "Message-ID:". Nach Wang und Chen stimmt bei Nicht-Spam-Mails die enthaltene Domain der "Message-ID:" zu 11,02% nicht mit der Domain des Senders überein. Bei Spam-Mails gibt es zu 82,66% keine Übereinstimmung. Folglich ist eine Klassifizierung unter der Berücksichtigung, dass FP vermieden werden nicht möglich. [vgl. 54, S. 386 f.]

3.2. Entwicklung eines Filterungsverfahrens auf Grundlage der zuvor identifizierten Kriterien

Aus den zuvor identifizierten Kriterien werden in diesem Abschnitt prüfungsrelevante Kriterien sondiert. Zur grafischen Unterstützung dienen Mengendiagramme.

Aus den untersuchten Kriterien geht hervor, das sich K1, K3, K7 und K13 besonders gut für eine Filterung eignen, da die Gemeinsamkeiten bzw. die Unterschiede zwischen Spam- und Nicht-Spam-Mails am deutlichsten sind. Des Weiteren wird nur mit diesen Kriterien eine Klassifizierung von kritischen FP vermieden. Ein Spamfilter der eine hohe Spam-Erkennung, aber auch viele FP kategorisiert, hat keinen Mehrwert für den Benutzer. Um die Relevanz der vier Kriterien nachzuweisen, werden diese mit den anderen Kriterien verglichen.

K1 vergleicht die Hauptdomains auf Abweichungen, wodurch eindeutig Spam erkannt werden kann. Da dieser Vorgang einen geringen Aufwand zur Folge hat, bietet sich dieses Kriterium als erste Wahl an. Durch die Benutzung des Kriteriums können 3 von 70 E-Mails als Spam klassifiziert werden (siehe 3.2, oberes linkes Mengendiagramm und die dazugehörige Klassifizierung).

Um die Relevanz von K3 nachzuweisen, werden die Kriterien K2-K6 (ausgenommen K5), welche sich auf den ersten öffentlichen MTA beziehen, verglichen. Wie im vorherigen Abschnitt festgestellt, eignet sich K4 nicht für eine alleinige Kategorisierung. Daraus resultiert, dass die Treffermenge, welche K1 erfüllt hat, im Bezug auf K2, K3 und K6 betrachtet wird. Es wird deutlich, dass 11 E-Mails keine dieser Kriterien erfüllen. Des Weiteren fällt auf, dass alle als Nicht-Spam kategorisierten E-Mails K3 erfüllen, wodurch alle, die nicht K3 erfüllen, als Spam klassifiziert werden können. Insgesamt können durch K3 30 Spam-Mails erfolgreich kategorisiert werden (siehe 3.2, oberes rechtes Mengendiagramm und die dazugehörige Klassifizierung).

An der verbleibenden Treffermenge von K3 werden die Kriterien K7-K9, welche sich mit den zeitlichen Aspekten einer E-Mail auseinandersetzen, angewendet. Es wird deutlich, dass wie im vorherigen Abschnitt festgestellt, K7 ein weiteres Kriterium zur Klassifizierung von Spam ist. So werden weitere vier E-Mails erfolgreich als Spam erkannt (siehe 3.2, unteres linkes Mengendiagramm und die dazugehörige Klassifizierung).

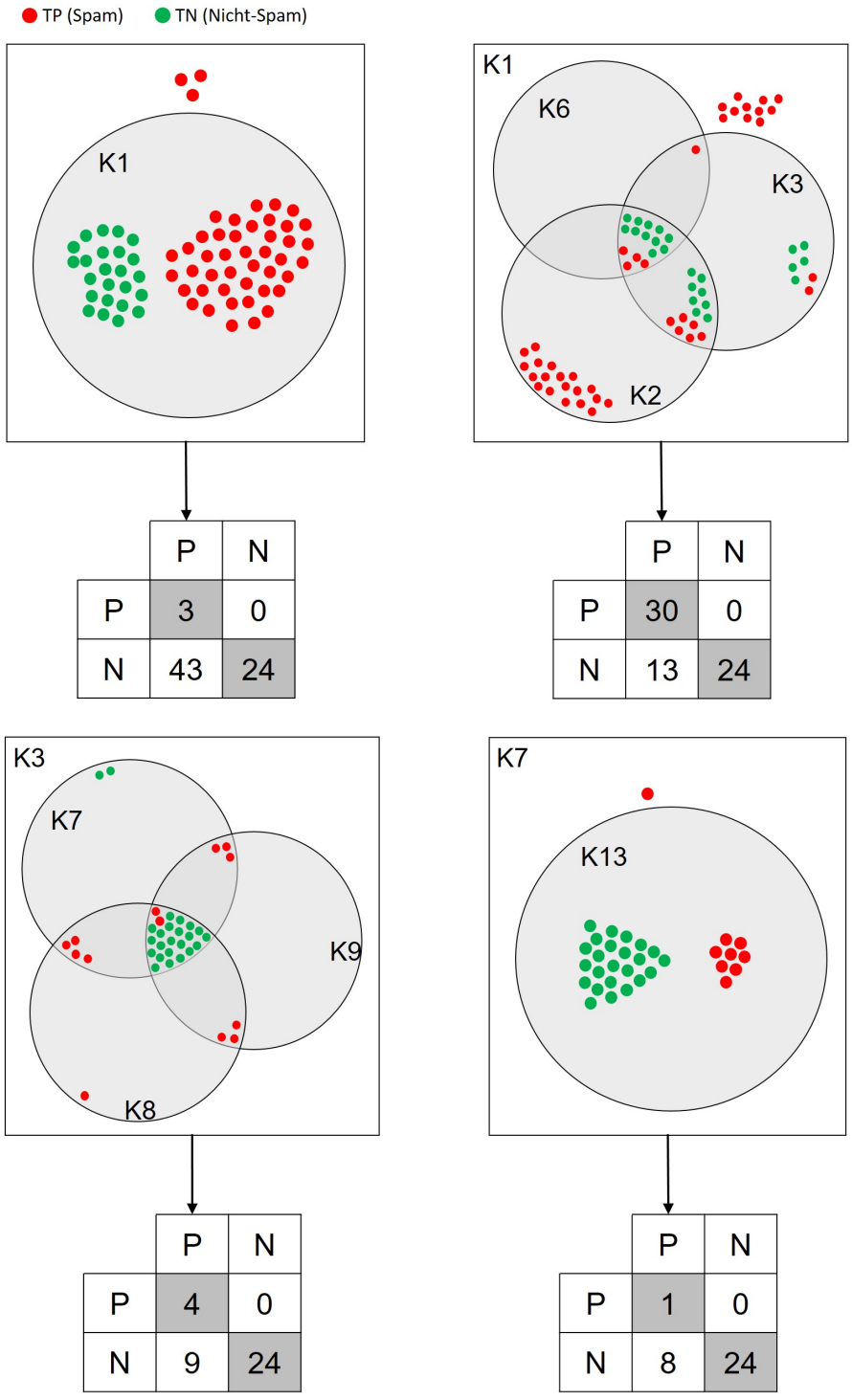


Abbildung 3.2.: Grafische Darstellung zur Identifizierung prüfungsrelevanter Kriterien und die daraus resultierende Klassifizierung

Wird dieses Mengendiagramm in Bezug auf die Treffermenge von K7 betrachtet, fällt auf, dass die Teilmengen K7 und K8 sowie K7 und K9 aus Spam-Mails bestehen. Das Festlegen des Kriteriums "(K8 und nicht K9) oder (nicht K8 und K9)" kann weitere sieben E-Mails als Spam kategorisieren. Die Untersuchung der gesammelten E-Mails (siehe Abschnitt 5.4.2), bestätigt diese Hypothese nicht. Es werden auf die Menge der gesammelten E-Mails 3% FP klassifiziert, weshalb sich das Kriterium nicht für die Umsetzung eignet.
Das 13. Kriterium wird auf die Treffermenge von K7 angewendet, wodurch eine E-Mail als Spam kategorisiert werden kann (siehe 3.2, unteres rechtes Mengendiagramm und die dazugehörige Klassifizierung). Aufgrund der geringen Filterwirkung und der Nutzung eines externen Dienstes erfolgt dieses Kriterium am Ende.

Die Vermutung, dass K1, K3, K7 und K13 sich zur erfolgreichen Kategorisierung von Spam-Mails eignen, wurde bestätigt. Durch die sequenzielle Abfolge der vier Kriterien können 38 von 46 Spam-Mails (82,6%) klassifiziert werden, ohne einen FP zu erzeugen.

Die Analyse der verbleibenden Spam-Mails (siehe Anhang C.3) im Bezug auf die verbleibenden Kriterien zeigt, dass die E-Mails die Kriterien K10 zu 13% und K12 zu 88% erfüllen. Die restlichen Kriterien werden im Durchschnitt zu 68% erfüllt. Der Vergleich der Kriterien K10 und K12 mit den der Nicht-Spam-Mails ergibt, dass keine eindeutige Klassifizierung durchgeführt werden kann. Die Erfüllung der restlichen Kriterien liegt im Durchschnitt bei 74% und entspricht nahezu dem Durchschnitt der Spam-Mails, was eine eindeutige Klassifizierung durch eine Gewichtung nicht ermöglicht.

In der Abbildung 3.3 wird die allgemeine Funktionsweise des Filterungsverfahrens veranschaulicht. Auf der linken Seite wird das aus diesem Abschnitt resultierende regelbasierte Filterungsverfahren dargestellt und auf der Rechten ein mögliches hybrides Filterungsverfahren. Letzteres kann eine Black-/ Whitelist und eine inhaltsbasierte Filterung integrieren, um die Erkennungsrate von Spam zu steigern.

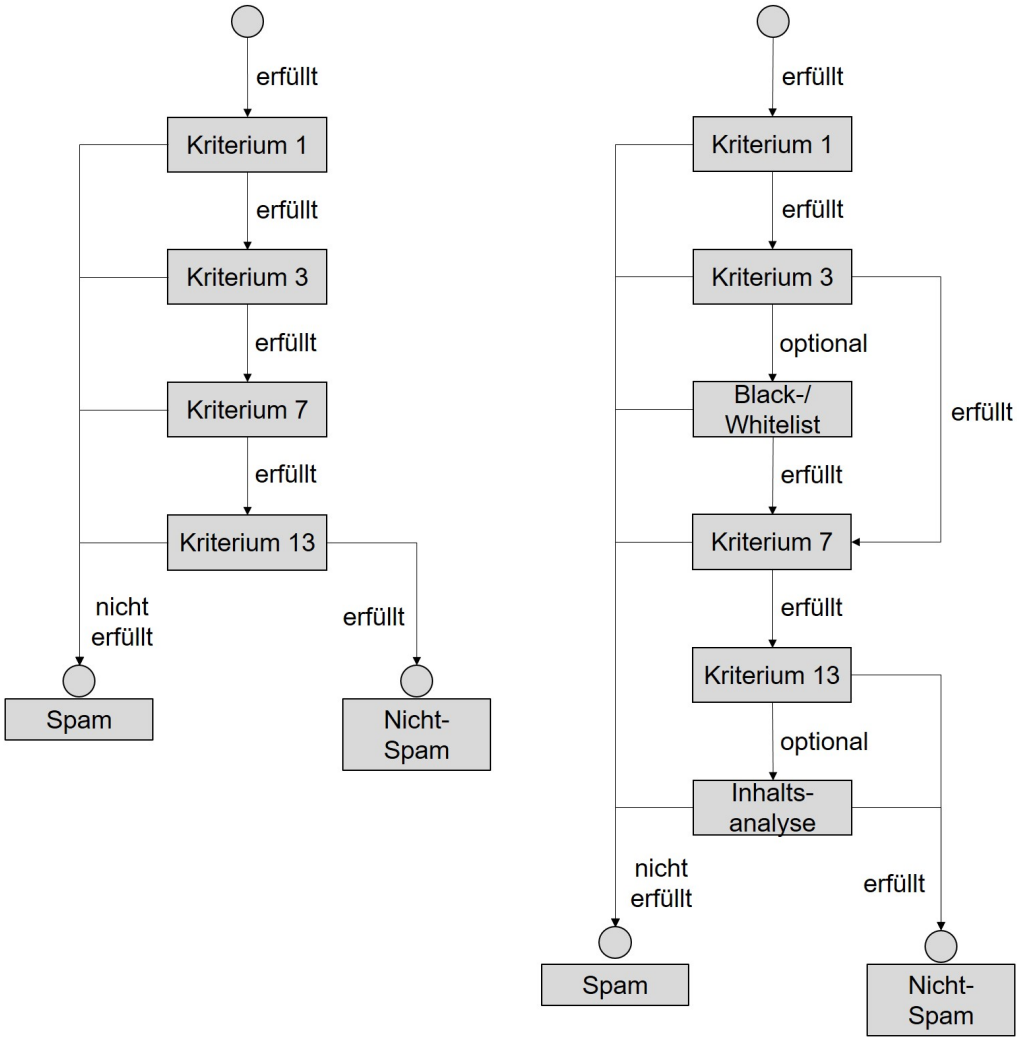


Abbildung 3.3.: Funktionsweise des Filterungsverfahrens ohne und mit optionalen Kriterien

4. Umsetzung des Filterungsverfahrens in Python

Das vorherige Kapitel hat mögliche Prüfkriterien erläutert und identifiziert. Dieses Kapitel beschäftigt sich mit der Umsetzung der prüfungsrelevanten Kriterien sowie des Verfahrens in ausführbaren Programmcode.
Im Rahmen dieser Umsetzung wird die Skriptsprache "Python" (Version 2.7.13) genutzt. Sie ermöglicht eine einfache und leserliche Umsetzung und bietet eine effiziente Nutzung auf allen Betriebssystemen. Zum Vertiefen der Programmierung in Python kann [25] genutzt werden.

Das entwickelte Skript (siehe Anhang D) zur Klassifizierung von E-Mails setzt sich allgemein aus folgenden Teilen zusammen: Datenaufbereitung, Filterung und Hilfsmethoden. Die Schnittstellen für die Integration in das operative System werden gesondert im Abschnitt 5.3 erläutert.

4.1. Datenaufbereitung

Der erste Teil (Programmcode Zeile 36-136) bereitet die Daten der übertragenen E-Mail für die weitere Analyse vor. Zuerst werden der Header und der Body voneinander getrennt, da der Body für die weitere Verarbeitung nicht von Relevanz ist.

Im nächsten Schritt wird jede Zeile des Headers angepasst (nicht inhaltlich), ausgelesen und ggf. in einer Variablen zwischengespeichert. Bei der Anpassung wird bspw. die ASCII-Codierung für Zeilenumbrüche entfernt, da diese zu einem späteren Zeitpunkt das Ergebnis eines Vergleiches verfälschen würde. Beim Auslesen wird das erste Wort jeder Zeile auf Schlagwörter, wie bspw. "From:", "Received:", überprüft und bei Übereinstimmung in einer Variablen zwischengespeichert. In der Regel werden die "Received:"-Felder mehrzeilig dargestellt, deswegen werden die nachfolgenden zugehörigen Zeilen anhand eines Schalters und dem am Zeilenanfang stehenden Leerzeichen erkannt und angehängt.

Die vollständigen "Received:"-Felder werden mit Hilfe eines regulären Ausdrucks einer syntaktischen Prüfung gemäß RFC 5321 unterzogen. Dadurch werden nicht relevante Einträge, die durch den MTA aufgrund eines internen Durchlaufs (Spam- oder Virenfilter) eingetragen wurden, entfernt. Erfüllt ein Eintrag diese Prüfung, wird die enthaltene IP-Adresse des sendenden MTA´s auf Korrektheit (0.0.0.0 - 255.255.255.255) überprüft. Kann ein Eintrag diese Prüfung ebenfalls erfüllen, wird er in eine Liste für die spätere Verwendung gespeichert.

4.2. Filterung: Umsetzung der Kriterien

Die Zeilen 142-284 beschreiben den zweiten Teil. Dieser besteht aus dem eigentlichen Verfahren, dem Analysieren der relevanten Daten anhand der zuvor festgelegten Kriterien und der aufbereiteten Daten. Das Nichterfüllen eines Kriteriums beendet die Analyse und klassifiziert unmittelbar die E-Mail als Spam-Mail. Das Verfahren wird dann nicht weiter durchlaufen.

Zu Beginn wird untersucht, ob die Spezifikation des Feldes "Date:" (RFC 5322) erfüllt wird. Die Verletzung dieser Spezifikation klassifiziert die E-Mail als Spam-Mail. Die darauf folgenden Kriterien werden nicht mehr durchlaufen.
Die Überprüfung wird über einen regulären Ausdruck umgesetzt, welcher aufgrund der optionalen Elemente in der Spezifikation viele Kombinationen aufweist. Die Webseite "regular expressions 101" [vgl. 13] bietet eine Möglichkeit den erstellten regulären Ausdruck auf eigene Beispiele anzuwenden. Eine Markierung stellt übersichtlich dar, welches Beispiel mit dem erstellten regulären Ausdruck übereinstimmt. Des Weiteren werden Gruppierungen, welche im regulären Ausdruck mit "(?<name>...)" maskiert werden, unterschiedlich farbig hervorgehoben (siehe Abbildung 4.1). Die letzten fünf Beispiele in der Abbildung stimmen nicht mit dem regulären Ausdruck überein, da sie nicht der Spezifikation der RFC entsprechen.
Die "Received:"-Felder wurden im Rahmen der Datenaufbereitung einer syntaktischen Überprüfung unterzogen. An dieser Stelle wird nur sichergestellt, dass mindestens ein "Received:"-Feld existiert, welches diese Überprüfung erfüllt hat. Ist dies nicht der Fall, erfolgt eine Klassifizierung als Spam-Mail, da die Spezifikation nach RFC 5321 nicht erfüllt wird und K3 nicht überprüft werden kann.

Abbildung 4.1.: Vergleich des erstellten regulären Ausdrucks mit möglichen Kombinationen des Feldes "Date:" nach RFC 5322 [vgl. 13]

Das erste Kriterium (K1) überprüft die Übereinstimmung der Hauptdomains des "From:"- und "Reply-To:"-Feldes. Damit diese Überprüfung stattfinden kann, wird zunächst sichergestellt, dass genau ein "From:"-Feld existiert. Durch das Manipulieren des Header besteht die Möglichkeit weitere "From:"-Felder hinzuzufügen, um ggf. den Spamfilter zu täuschen. Ist genau ein Feld enthalten, wird die darin enthaltene Hauptdomain auf ihre Existenz überprüft. Eine nicht existente Domain verweist auf eine gefälschte Domain und muss daher nicht weiter analysiert werden. Die Anzahl der "Reply-To:"-Felder entscheidet das weitere Vorgehen. Existiert kein Feld, entspricht diese Zeile dem "From:"-Feld und das Kriterium wurde erfüllt. Existiert genau ein Feld, wird die darin enthaltene Hauptdomain mit der Hauptdomain des "From:"-Feldes verglichen. Bei Übereinstimmung gilt das Kriterium als erfüllt. Ist mehr als ein "Reply-To:"-Feld enthalten, liegt eine Manipulation vor und das Kriterium wird nicht erfüllt.

Das nächste Kriterium (K3) analysiert die "Received:"-Felder der sendenden MTA´s, die eine öffentliche IP-Adresse besitzen. Hierfür wird die Liste mit den Einträgen, die die syntaktische Prüfung und die Überprüfung der IP-Adresse auf Korrektheit erfüllt haben, genutzt. Die Aufbereitung dieser Daten erfolgte im Header von "oben nach unten". Bei der Übertragung der E-Mail werden die "Received:"-Felder jedoch von "unten nach oben" dem Header angehängt. In der Regel werden gefälschte "Received:"-Felder als erstes dem Header angehängt, daher wird die Liste in umgekehrter Reihenfolge abgearbeitet. Bei jedem "Received:"-Feld wird die IP-Adresse des sendenden MTA´s dahingehend überprüft, ob es sich um eine öffentliche IP-Adresse handelt. Damit die Überprüfung effizient ist, wird die IP-Adresse in ihre Oktette aufgesplittet. Für die privaten Adressbereiche 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 wird das erste Oktett auf Übereinstimmung geprüft. Bei dem "10er" Adressbereich ist dies ausreichend, bei den anderen beiden nicht. Für diese Adressbereiche werden die Oktette, wenn nötig, mit voranstehenden Nullen aufgefüllt und anschließend zu einer Zeichenkette zusammengefügt. Anschließend wird überprüft, ob die IP-Adresse im Adressbereich enthalten ist. Hierfür werden die Bereichsgrenzen ebenfalls in Zeichenketten überführt. Handelt es sich bei der IP-Adresse um eine private IP-Adresse wird das nächste "Received:"-Feld überprüft, bis eine öffentliche IP-Adresse ermittelt wurde.
Ist das der Fall wird die öffentliche IP-Adresse mit dem Befehl "nslookup" ausgeführt. Die Antwort wird mit einem regulären Ausdruck auf den aufgelösten Namen untersucht. Die daraus resultierende Liste kann einen oder mehrere aufgelöste Namen enthalten. Die Namen werden mit dem im "Received:"-Feld angegebenen Namen des sendenden

MTA´s auf Übereinstimmung überprüft. Das Ergebnis entscheidet über die Erfüllung des Kriteriums. Dieser Vorgang wird für alle in der Liste enthaltenen "Received:"-Feld durchgeführt.

Die Zeilen 238-263 (K7) analysieren die E-Mail auf Plausibilität der Zeitstempel und die Dauer der Zustellung. Die Zeitstempel des "Date:"- und "Received:"-Feldes können unterschiedliche Zeitzonen enthalten und erschweren somit den Vergleich. Ein Sonderfall, bei der Angabe der Zeitzone, ist z.B. "GMT" statt "+0000" am Ende des Zeitstempels. Der Name einer Zeitzone wird durch einen regulären Ausdruck erkannt und durch die Zeit (hier "+0000") ersetzt.
Im weiteren Verlauf wird das "Date:"- und das letzte "Received:"-Feld (das des annehmenden, eigenen MTA‘s) mit einem regulären Ausdruck gruppiert. Die Gruppierungen ermöglichen das Erzeugen eines "Datetime"-Objektes, welches die Verarbeitung erleichtert. Anschließend wird die dazugehörige Zeitzone, als "Timedelta" vom "Datetime"-Objekte subtrahiert, wodurch das resultierende "Datetime"-Objekt, der Zeitzone "+0000" entspricht. Nachdem beide "Datetime"-Objekte an die selbe Zeitzone angepasst wurden, kann das "Datetime"-Objekt des "Date:"-Feldes vom "Datetime"-Objekt des letzten MTA´s subtrahiert werden. Das resultierende "Timedelta", welches die Dauer der Zustellung widerspiegelt, kann in Form von Tagen, Sekunden oder Millisekunden weiterverarbeitet werden. Da im vorherigen Kapitel der Schwellenwert auf 90 Minuten festgelegt wurde, bietet sich die Nutzung von Sekunden an. Daraus folgt, dass bei der Überschreitung eines "Timedeltas" von 5400 Sekunden das Kriterium nicht erfüllt wird und die E-Mail als Spam-Mail klassifiziert und markiert wird.

Das nachfolgende Kriterium (K13) untersucht den Zeitpunkt der Domainerstellung und vergleicht diesen mit dem Zeitpunkt, an dem die E-Mail zum Versand bereitgestellt wurde. Informationen über die Domain werden über den Befehl "whois" in Kombination mit der Domain abgefragt. Die Antwort der Anfrage wird mit einem regulären Ausdruck auf den Eintrag "Creation Date:" überprüft. Ist die Treffermenge leer, wird das Kriterium übersprungen, da kein Vergleich stattfinden kann. Bei einem Treffer wird von dem Datum und dem des "Date:"-Feldes je ein "Datetime"-Objekt erstellt und das "Timedelta" berechnet. Ergibt das "Timedelta" eine Abweichung von null Tagen, wurde die E-Mail am selben Tag zum Versand bereitgestellt, an dem auch die Domain erstellt wurde. Daraus resultiert die Nichterfüllung des Kriteriums und die E-Mail wird als Spam-Mail kategorisiert und markiert.

4.3. Hilfsmethoden

Der dritte Teil (Zeile 290-469) stellt Hilfsmethoden für den zweiten Teil bereit, damit dieser lesbarer ist und redundanter Code vermieden wird. Einige Hilfsmethoden, wie bspw. die Erzeugung eines "Datetime"-Objekts oder die Überprüfung der IP-Adresse auf private Adressbereiche, wurden schon in Verbindung mit der Umsetzung der Kriterien genannt. Erstere Methode erfordert zwei Parameter, ein Objekt und ein Wahrheitswert (True/False). Das Objekt ist ein Listenelement und beinhaltet einen Zeitstempel. Der Wahrheitswert entscheidet darüber, ob das "Datetime"-Objekt nur Tag, Monat und Jahr (False) oder zusätzlich Stunden, Minuten und Sekunden (True) enthalten wird. Auf das Objekt wird ein regulärer Ausdruck angewendet, wodurch ein gruppiertes Objekt erzeugt wird. Die Gruppierungen dienen als Parameter in Abhängigkeit des übergebenen Wahrheitswerts für das "Datetime"-Objekt. Es wird eine gemeinsame Methode für K7 und K13 geschaffen und redundanter Code vermieden.

Eine weitere Methode ist das Kürzen der E-Mail-Adresse auf die Hauptdomain. Damit die Hauptdomains durch das erste Kriterium verglichen werden können, müssen die Domains des "From:"- und "Reply-To:"-Feldes aus den E-Mail-Adressen herausgelöst werden. Hierfür wird die E-Mail-Adresse, anhand des @-Zeichens, in zwei Teile gesplittet. Der zweite Teil stellt die Domain der E-Mail-Adresse dar, welche Subdomains enthalten kann. Damit der Vergleich erfolgen kann, wird die Domain wiederum in ihre Bestandteile, anhand der Trennung durch den/die Punkt/e, gesplittet. Aus der resultierenden Liste werden die letzten beiden Elemente, mit einem Punkt, zu einer Zeichenkette verbunden und von der Methode zurückgegeben.

Weitere Methoden sind bspw. das Anlegen eines Logfiles und das Hinzufügen einer Markierung im Header, falls es sich um Spam handelt. Die Abbildung 4.2 veranschaulicht ein Logfile und eine mögliche Auswertung.

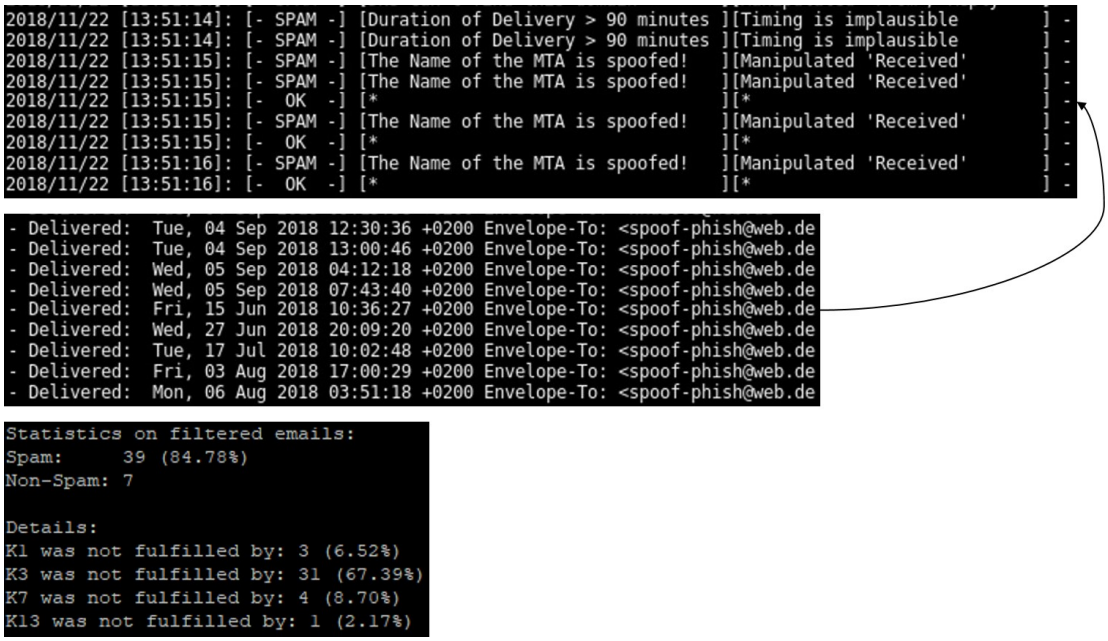


Abbildung 4.2.: Darstellung und Auswertung eines beispielhaften Logfiles

4.4. Laufzeitanalyse der umgesetzten Kriterien

Um einen möglichst effizienten Algorithmus für das Filterungsverfahren umzusetzen, wird die Laufzeit der umgesetzten Kriterien untersucht. Hierzu wird die "O-Notation" verwendet, welche es ermöglicht, die Komplexitätsklasse eines Algorithmus zu bestimmen und mit anderen zu vergleichen [vgl. 4, S. 101 f.]. Es wird der "Worst Case" betrachtet, da dieser einfach zu bestimmen ist und die maximale Laufzeit des Algorithmus garantiert.

Das Resultat der Laufzeitanalyse (siehe Anhang B) ergibt für die Kriterien folgende Komplexitätsklassen: K1, K7, K13 $\in O(n)$ und K3 $\in O(n^2)$.
Infolgedessen, dass K1, K7 und K13 der selben Komplexitätsklasse angehören, wird die Analyse der Spam-Mails (siehe Anhang C.2), zur Bestimmung der Reihenfolge berücksichtigt. Die Wahl des ersten Kriteriums fällt auf K7, da es ca. 5% mehr Spam-Mails erkennt als K1 oder K13. Des Weiteren sind die Kriterien K1 und K13 von einem externen Dienst abhängig, was zu Verzögerungen führen kann.
Die Menge der externen Daten ist bei K13 größer als bei K1. Dadurch benötigt K13 mehr Zeit, um die relevanten Daten zu finden. Deshalb wird K1 als nächstes überprüft.
Obwohl K3 eine höhere Komplexitätsklasse als K13 aufweist, wird dieses Kriterium an

die dritte Stelle platziert. Der Grund hierfür ist, dass K3 65% der analysierten Spam-Mails klassifizieren kann. Folglich muss K13 eine deutlich geringere Anzahl an Spam-Mails, einer aufwändigen Abfrage und Durchsuchung der externen Daten unterziehen. Außerdem ist die Erkennungsrate von K13 so gering, dass eine Ausführung dieses Kriteriums vor K3 kaum einen Unterschied macht.

Für das entwickelte Filterungsverfahren ergibt sich, aus der Kombination der Laufzeitanalyse und der Analyse der Spam-Mails, der folgende sequenzielle Ablauf der Kriterien: K7, K1, K3 und K13.
Das Kriterium K3 legt für den gesamten Filterungsprozess eine Komplexität von $O(n^2)$ fest. In Bezug auf die übrige Implementation ergibt sich für das entwickelte Filterungsverfahren eine Komplexität von $O(n^2)$.

5. Beispielhafte Integration in ein operatives System

In diesem Kapitel wird ein operatives System beschrieben, in welches das zuvor entwickelte Filterungsverfahren integriert wird. Bevor die Integration näher erläutert wird, werden die verschiedenen Möglichkeiten der Platzierung des Spamfilters aufgezeigt und erläutert. Abschließend wird das entwickelte Filterungsverfahren mit einer Menge an (un-)erwünschten E-Mails getestet.

5.1. Beschreibung des operativen Systems

Die Idee ist es eine minimale und effiziente Infrastruktur auf der Grundlage einer Open Source Lösung zu schaffen, die den Schutz vor missbräuchlichen E-Mails verbessert.
Um diese Idee umzusetzen wird ein "Raspberry Pi 3 Model B+"[22] verwendet. Als Betriebssystem wird das dafür angepasste Linux, namens "Raspbian" (basierend auf Debian) mit dem Release "stretch"[23] (Version 9), genutzt.

Um die Funktion des Mailservers und des entwickelten Filterungsverfahrens sicherzustellen, wird ein DNS-Server installiert und konfiguriert. Dadurch wird eine eigene DNS-Tabelle aufgebaut und zukünftige DNS-Abfragen können beschleunigt werden, welche ggf. die Effizienz des Filterungsverfahrens steigern.

Anschließend werden "Postfix"[24] (Version 3.1.8) und "Dovecot"[25] (Version 2.2.27) als Komponenten für den Mailserver installiert und konfiguriert. Postfix ist ein MTA und ermöglicht die Integration eines Spamfilters, während Dovecot ein MDA und IMAP-Server ist.

[22] URL: https://raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
[23] Download-URL: https://downloads.raspberrypi.org/raspbian_latest [Abruf: 2018-06-27].
[24] URL: https://de.postfix.org/index.html.
[25] URL: https://dovecot.org/.

Der Raspberry Pi befindet sich für diese Arbeit in einem Heimnetzwerk mit einer privaten IP-Adresse, d.h. mehrere Geräte im Heimnetzwerk haben die selbe öffentliche IP-Adresse, welche durch den Router bereitgestellt wird. Damit die Erreichbarkeit des Raspberry Pi und der Empfang von E-Mails sichergestellt ist, muss "DynDNS"[26] im Router konfiguriert werden. Hierfür gibt es drei Möglichkeiten:

- der Besitz einer Fritzbox ermöglicht die Nutzung des Dienstes "MyFRITZ",
- das Erwerben einer eigenen Domain oder
- das Nutzen einer Lösung eines (kostenlosen) DynDNS-Anbieters.

Im Rahmen dieser Arbeit wird auf die letzte Möglichkeit zurückgegriffen und der Anbieter "ddnss.de"[27] genutzt. Dieser ermöglicht die Eintragung von MX-RR, damit E-Mails empfangen werden können. Nach der Erstellung eines Accounts kann bei dem Anbieter eine Subdomain (hier "analyse.ddnss.de") erstellt werden. Anschließend werden die vom Anbieter bereitgestellten Daten im eigenen Router, unter der Rubrik DynDNS, konfiguriert. Nach erfolgreicher Einrichtung wird jede Aktualisierung der IP-Adresse vom Router an den DynDNS Anbieter übermittelt (A). Der Anbieter verknüpft die übermittelte IP-Adresse mit der eingerichteten Subdomain. Durch das Einrichten eines DynDNS kann der Router jederzeit über eine feste Domain, trotz ändernder IP-Adresse, erreicht werden. Um die Nutzung des Mailservers zu gewährleisten, wird auf dem Router "Port Forwarding", für den Port 25, auf den Raspberry Pi konfiguriert. Das bedeutet, dass der Router alle Pakete, die er über SMTP bekommt, an den Raspberry Pi weiterleitet.
Eine versendete E-Mail mit dem Empfänger "spoofi@analyse.ddnss.de" (spoofi ist ein angelegter Nutzer mit einer Mailbox) wird vom zuständigen MTA des DynDNS Anbieters entgegengenommen (1). Ist dieser für die Domain des Empfängers zuständig, wird die E-Mail, an die im DynDNS hinterlegte IP-Adresse (2,3), weitergeleitet (4). Der Router empfängt die E-Mail und leitet sie, aufgrund der Port-Forwarding-Regel, an den MTA (hier Postfix) des Raspberry Pi weiter (5). In der Abbildung 5.1 wird der Ablauf dargestellt.

[26] Dynamic Domain Name System.
[27] URL: https://ddnss.de/.

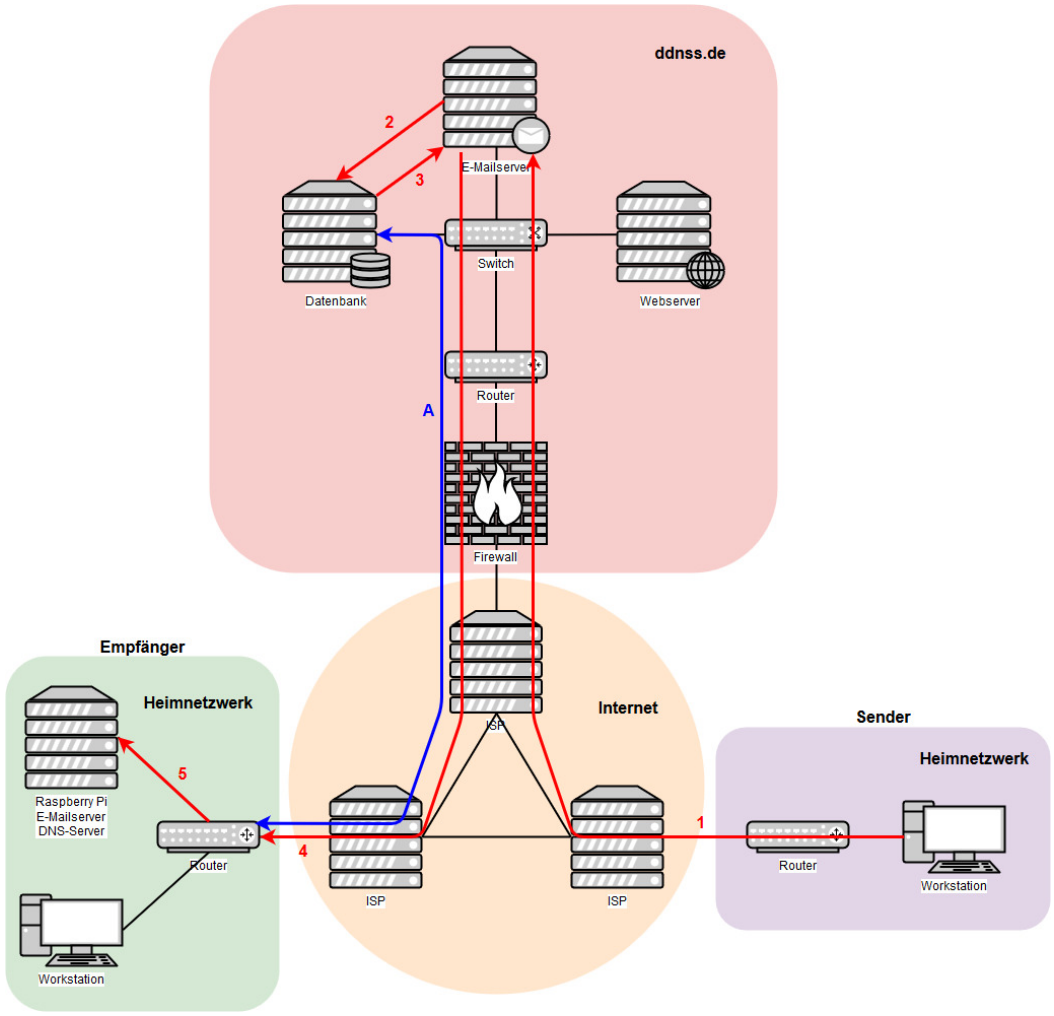


Abbildung 5.1.: Versand einer E-Mail unter Verwendung von DynDNS

5.2. Möglichkeiten der Platzierung eines Spamfilters

Dieser Abschnitt setzt sich mit den Vor- und Nachteilen der Platzierung eines Spamfilters auseinander. Es gibt drei Möglichkeiten einen Spamfilter zu integrieren: erstens auf der Ebene der Anwendung bei einem Clienten, zweitens auf dem Mailserver oder drittens auf einem Proxy-Server. Bei den letzten beiden Möglichkeiten handelt es sich um eine zentrale Filterungslösung, während die Erste eine individuelle Lösung ist.

Die Platzierung eines Spamfilters auf der Ebene der Anwendung (wie bspw. Mozilla Thunderbird) bietet ein hohes Maß an Individualität und Privatsphäre für den Benutzer. Der Benutzer kann die Software nach seinen Kriterien einstellen und hat die Kontrolle über seine Daten. Im Unternehmensumfeld dient dies eher der Feinfilterung, da nicht jeder Mitarbeiter die Relevanz für den Einsatz sieht oder das Know-How für die Konfiguration besitzt. Außerdem steht ein zentrales Vorgehen für alle im Vordergrund.
Eine zentrale Filterungslösung bietet eine einheitliche Filterung und entlastet die Mitarbeiter, entzieht ihnen jedoch die Kontrolle über bestimmte Aspekte, bspw. wie die E-Mails bewertet werden. Die Auswirkungen einer Platzierung auf einem Mailserver oder Proxy-Server sind für die Mitarbeiter nicht von Relevanz, viel mehr geht es hier um Ressourcen und Verantwortlichkeiten des MTA´s.

Wird ein Spamfilter auf einem Proxy-Server platziert, wird im Bezug auf Postfix von einem "Before-Queue Content Filter" [vgl. 50] gesprochen. Die ankommende E-Mail wird erst gefiltert, bevor der MTA diese annimmt. Dadurch kann eine E-Mail, die vom Spamfilter nicht akzeptiert wird, weil sie bspw. auf einer Blacklist steht, abgelehnt werden, bevor die SMTP-Übertragung abgeschlossen ist. Da die SMTP-Übertragung nicht abgeschlossen wurde, werden dem MTA auch keine Verantwortlichkeiten übertragen und der SMTP-Client muss sich um die Bounce-Nachricht kümmern. Nachteil dieser Platzierung ist, dass ein erhöhtes Aufkommen an E-Mails, in Abhängigkeit der Effizienz des Spamfilters, die Ressourcen des Proxy-Servers auslasten kann. Dies hat zur Folge, dass das parallele Filtern von E-Mails auf ein Minimum reduziert werden muss, was wiederum bei einem "normalen" E-Mail-Aufkommen das Verfahren verlangsamt. Durch die steigende Ressourcenauslastung verlängert sich ebenfalls die Antwortzeit des Spamfilters und somit auch die Antwort an den SMTP-Client. Nach dem Erreichen einer Frist beendet dieser, unabhängig vom Resultat des Spamfilters, die SMTP-Übertragung. Die E-Mail gilt als nicht zustellbar. Damit eine Antwort den SMTP-Client rechtzeitig erreicht, muss entweder die Annahme von E-Mails gestoppt oder der Spamfilter deaktiviert werden.

Von einem "After-Queue Content Filter" [vgl. 49] wird gesprochen, wenn der Spamfilter direkt in Postfix integriert wird. Im Vergleich zum vorherigen Filter, kehren sich die Vor- und Nachteile um. Der MTA nimmt die ankommende E-Mail erst an und übernimmt somit alle Verantwortlichkeiten, vorausgesetzt er ist für diese zuständig, bevor diese gefiltert wird. Die Filterung der E-Mails erfolgt sequenziell und in Abhängigkeit der parallel zugelassenen Instanzen, unter Berücksichtigung der Ressourcenauslastung. Die Annahme von E-Mails und somit auch die Antwort an den SMTP-Client wird nicht beeinflusst. Durch ein erhöhtes Aufkommen von E-Mails wird dahingehend nur die Weiterleitung oder Zustellung verzögert.

5.3. Integration des Filterungsverfahrens

Im vorherigen Abschnitt wurden die Möglichkeiten zur Platzierung des Spamfilters aufgezeigt. Der Raspberry Pi verfügt über geringe Ressourcen und muss dennoch eine große Menge an E-Mails nahezu zeitgleich annehmen. Infolgedessen wird im Rahmen dieser Arbeit das entwickelte Filterungsverfahren als "After-Queue Content Filter" in Postfix integriert. Dadurch werden Ressourcen nur für die Filterung der E-Mails beansprucht, für welche der MTA auch zuständig ist.

Das Filterungsverfahren ist eigenständig und wird beim Systemstart ausgeführt. Die Ausführung des Skripts wird mit folgenden Eintrag in die Datei "/etc/rc.local" sichergestellt.

```
1 # Start mail-filtering
2 python /etc/postfix/filter_via_smtp.py
```

Listing 5.1: Ausführen des Skripts beim Systemstart

Das Filtern der E-Mail kann über eine temporäre Kopie und dessen Aufruf oder durch eine Weiterleitung an den Spamfilter durchgeführt werden. Die Umsetzung der ersten Variante ist simpel, beansprucht aber beim Anlegen der temporären Kopie Ressourcen und vor allem Speicherplatz. Ein Spammer könnte gezielt E-Mails mit großen Dateianhängen schicken, um den Speicherplatz auszulasten. Dies hätte zur Folge, dass sich die Lesegeschwindigkeit verringert und sich somit auch die Laufzeit des Filterungsverfahrens verschlechtert.
Die zweite Variante leitet die E-Mail an das Skript weiter. Hierbei wird die E-Mail im Arbeitsspeicher abgelegt. Aufgrund der schnellen Lesegeschwindigkeit des Arbeitsspeichers wird die Laufzeit nicht negativ beeinflusst. Deshalb wird im Rahmen dieser Arbeit die zweite Variante genutzt.

Nach der Ausführung des Skripts wird ein lokaler SMTP-Server erzeugt, der auf dem Port 10025 auf eingehende SMTP-Verbindungen wartet. Eine eingehende E-Mail wird vom MTA (Postfix) an den lauschenden Port weitergeleitet, wenn die Postfix Konfigurationsdatei "master.cf" folgendermaßen (Zeile 2-7) erweitert wird:

```
1 # service type  private unpriv  chroot  wakeup  maxproc command + args
2 smtp      inet  n       -       y       -       -       smtpd
3     -o content_filter=filter:localhost:10025
4     -o receive_override_options=no_address_mappings
5
6 filter    unix  -       -       n       -       10      smtp
7     -o smtp_send_xforward_command=yes
```

Listing 5.2: Postfix: Erweiterung der Konfigurationsdatei "master.cf" zur Weiterleitung der E-Mail an den entwickelten Filter [vgl. 49]

Die Zeilen 2-4 beschreiben die Weiterleitung von eingehenden SMTP-Verbindungen an den angegebenen Filter (filter:localhost:10025), ohne die originale E-Mail-Adresse abzuändern. Die Zeilen mit "-o", beschreiben die Befehlsoptionen für den ausführenden Daemon und überschreiben die globalen Parameter der Konfigurationsdatei "main.cf". Die Befehlsoptionen werden in der Postfix Dokumentation[28] ausführlich erläutert. Bevor die E-Mail an den lokalen Port 10025 weitergeleitet wird, durchläuft sie den Service "filter" (Zeile 6,7). Durch die Angabe des Typs "unix", kann der Service "filter" nur lokal erreicht werden. Die maximale Anzahl an Prozessen, die gleichzeitig den Service ausführen dürfen, wird auf 10 beschränkt, um die Ressourcen nicht auszulasten. Die Befehlsoption in Zeile 7 veranlasst Postfix, die ursprüngliche IP-Adresse und den Hostnamen der E-Mail an das entwickelte Filterungsverfahren weiterzuleiten. Würde dieser Befehl fehlen, wird stattdessen die IP-Adresse und der Hostname von Postfix übermittelt.
Die E-Mail wird nach der abgeschlossenen Klassifizierung, durch das Filterungsverfahren, mit der Methode "send_message" des Skripts, an den lokalen Port 10026 gesendet. Der nachfolgende Service in der Konfigurationsdatei "master.cf" ermöglicht, das Entgegennehmen der E-Mail an Port 10026 und das Zurückführen in den Zustellprozess von Postfix.

[28] URL: http://www.postfix.org/master.5.html [Abruf: 2018-11-06].

```
1  # service          type  private unpriv  chroot  wakeup  maxproc command + args
2  localhost:10026    inet      n       -       n       -       -       smtpd
3      -o content_filter=
4      -o receive_override_options=no_header_body_checks,no_unknown_recipient_checks
5      -o smtpd_relay_restrictions=
6      -o smtpd_client_restrictions=
7      -o smtpd_sender_restrictions=
8      -o smtpd_helo_restrictions=
9      -o mynetworks=127.0.0.0/8
10     -o smtpd_recipient_restrictions=permit_mynetworks,reject
11     -o smtpd_authorized_xforward_hosts=127.0.0.0/8
```

Listing 5.3: Postfix: Erweiterung der Konfigurationsdatei "master.cf" zur Weiterleitung der E-Mail an Postfix [vgl. 49]

Die zweite Zeile entfernt den Verweis auf den Filter und verhindert eine Endlosschleife. Zeile 3 verhindert die Überprüfung von Header und Body, da diese explizit vom entwickelten Filterungsverfahren überprüft wurden. Des Weiteren soll vermieden werden, das eine E-Mail mit einem unbekannten Empfänger verworfen wird, da dies Aufgabe des MTA´s ist. Die E-Mail wurde vor der Filterung vom MTA angenommen und auf vorkonfigurierte Beschränkungen überprüft und muss nicht erneut daraufhin geprüft werden (Zeilen 4-7).

Nachdem die E-Mail zurück im Zustellprozess von Postfix ist, wird die Zuständigkeit geprüft. Ist der MTA für die Zieldomain zuständig, wird die E-Mail an den MDA (hier Dovecot) übergeben. Anschließend ordnet Dovecot die E-Mail dem richtigen Konto zu und verteilt sie in den entsprechenden Ordner. Um die Auswahl des Ordners zu beeinflussen, wird in der Datei "90-sieve.conf" die Variable "sieve_global" aktiviert und durch folgenden Eintrag ergänzt:

```
1  sieve_global = /var/lib/dovecot/sieve/global/spam-global.sieve
```

Listing 5.4: Dovecot: Aktivieren des globalen Filterskripts

Eine E-Mail die vom entwickelten Filterungsverfahren als Spam-Mail klassifiziert wird, erhält die Markierung "X-hbE_Spam: YES". Diese wird von Dovecot in den Spam-Ordner verschoben, wenn die Datei "default.sieve" im angegebenen Pfad angelegt ist und folgenden Inhalt besitzt:

```
1  require "fileinto";
2  if header :contains "X-hbE_Spam" "YES" {
3    fileinto "Spam";
4  }
```

Listing 5.5: Dovecot: Skript zur Verteilung der E-Mail in einen definierten Ordner [vgl. 11, S. 6-11]

5.4. Evaluation des entwickelten Filterungsverfahrens

In diesem Abschnitt wird in zwei Durchläufen, unter Verwendung empfangener E-Mails, die Wirkungsweise des entwickelten Filterungsverfahrens veranschaulicht und bewertet. Die Durchläufe werden nicht am operativen System durchgeführt, da der konfigurierte MTA für die Empfängeradressen nicht zuständig ist. Dies hätte zur Folge, dass der MTA die E-Mails nicht annimmt. Stattdessen erfolgen die Durchläufe direkt am entwickelten Filterungsverfahren, d.h. die E-Mails werden lokal abgespeichert, durch das Ausführen des Skripts aufgerufen und gefiltert. Die Ergebnisse werden anhand des erstellten Logfiles evaluiert.

5.4.1. Der erste Durchlauf - Test-Mails aus der Analyse

Für die Identifizierung der Kriterien wurden 70 E-Mails analysiert. Im ersten Durchlauf werden diese vom entwickelten Filterungsverfahren untersucht. Bei der Ableitung des Filterungsverfahrens (siehe Abschnitt 3.2) wurden 38 von 46 Spam-Mails (82,61%) erkannt. Nach dem Durchlauf ergibt die Auswertung des Logfiles, dass 39 von 46 Spam-Mails (84,78%) erkannt wurden. Das Ergebnis weicht um eine Spam-Mail von der Erwartung ab. Die Ursache hierfür ist, dass sich im Zeitraum zwischen der Analyse und dem Durchlauf der Name des MTA´s geändert hat. Wird angenommen, dass die Filterung zum Zeitpunkt der Analyse erfolgte, werden 82,61% Spam-Mails (TP) erkannt, ohne einen FP zu klassifizieren. Die Nichterfüllung der Kriterien teilt sich wie folgt auf: K1 4,35%, K3 65,22%, K7 10,87% und K13 2,17%. Es wird deutlich, dass alle Spam-Mail die Spezifikation der RFC 5322 erfüllen und der größte Anteil durch die Erkennung von "Open Relays" kategorisiert werden kann.

5.4.2. Der zweite Durchlauf - erhaltene E-Mails bei "WEB.DE"

Im Rahmen der Arbeit wurde beim ESP "WEB.DE" eine E-Mail-Adresse erstellt. Diese wurde für Newsletter-Anmeldungen und für die Verbreitung auf unseriösen Webseiten verwendet. Für die E-Mail-Adresse wurden in einem Zeitraum von fünf Monaten 1203 E-Mails entgegengenommen. Die manuelle Klassifizierung ergibt 119 TP, 761 TN, 323 FP und 0 FN. Die Kategorien durchlaufen das entwickelte Filterungsverfahren voneinander unabhängig, um ggf. Unterschiede festzustellen. Der Schwerpunkt liegt auf den TP und FP, um die Effizienz des Filterungsverfahrens festzustellen.

Das entwickelte Filterungsverfahren kommt zu folgenden Ergebnissen:

- TP: es werden 98 der 119 Spam-Mails richtig erkannt (82,35%),
- TN: es gibt keine Abweichung, d.h. alle 761 werden als Nicht-Spam-Mails kategorisiert,
- FP: es werden alle als TN klassifiziert, d.h. alle werden als Nicht-Spam-Mails eingestuft.

Im Vergleich zum genutzten Filterungsverfahren von "WEB.DE"[29] werden weniger Spam-Mails erkannt, jedoch keine FP kategorisiert (siehe Abbildung 5.2). Die Nichterfüllung der Kriterien teilt sich wie folgt auf: Spezifikation 4,20%, K1 9,24%, K3 56,30%, K7 11,77% und K13 0,84%. Es wird deutlich, dass K3 die meisten Spam-Mails erkennt und das einige nicht die Spezifikationen erfüllen. Die Untersuchung der Spam-Mails, die Letzteres nicht erfüllen, zeigt, dass sich keine an die Spezifikation des "Date:"-Feldes hält.

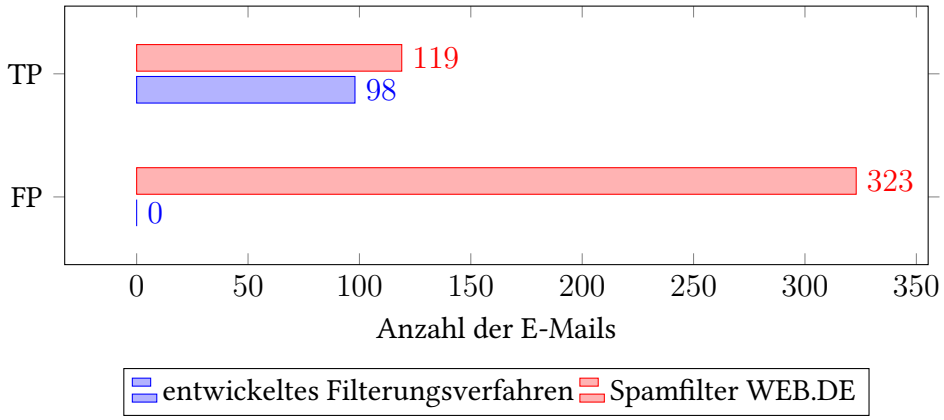


Abbildung 5.2.: Effizienzvergleich: Spamfilter von "WEB.DE" vs. entwickeltes Filterungsverfahren

Zusammenfassend ergibt sich aus beiden Durchläufen eine Spam-Erkennungsrate von knapp 83,00%, ohne das ein FP klassifiziert wird. Das bedeutet, dass 323 E-Mails durch nachfolgende Filterkomponenten, z.B. Anti-Virenfilter nicht untersucht werden müssen, wodurch die Inanspruchnahme wichtiger Ressourcen verhindert wird.
Durch ein manipuliertes "Received:"-Feld werden die meisten Spam-Mails (durchschnittlich 60,76%) mit Hilfe von K3 kategorisiert. Die Kriterien K1 und K7 klassifizieren im

[29] Es gibt keine Informationen, welche Art von Filterung (vermutlich eine hybride Filterung) genutzt wird.

Durchschnitt 6,80% und 9,29% der E-Mails als Spam-Mails. Mit 1,5% erkennt K13 die wenigsten Spam-Mails. Des Weiteren wird im zweiten Durchlauf deutlich, dass 4,20% der Spam-Mails nicht die Spezifikation nach RFC 5322 für das "Date:"-Feld erfüllen. Die Ergebnisse der beiden Durchläufe sind im Anhang C.4 veranschaulicht.

6. Abschluss

Das Kapitel fasst die Arbeit zusammen, setzt sich kritisch mit dem Ergebnis auseinander und gibt einen Ausblick auf weitere Verbesserungsmöglichkeiten.

6.1. Zusammenfassung

Im Rahmen dieser Arbeit wurden 70 E-Mails (24 Nicht-Spam- und 46 Spam-Mails) im Hinblick auf ihre Header-Informationen auf mögliche Prüfkriterien analysiert. Insgesamt wurden 13 Kriterien identifiziert. Die Analyse ergab, dass sich vier Kriterien davon (K1, K3, K7, K13) zur Klassifizierung von Spam eignen, ohne einen False Positiv zu kategorisieren.

Das Kriterium K1 vergleicht die Hauptdomain der Felder "From:" und "Reply-To:", sofern letzteres vorhanden ist. Unterscheiden sich die Hauptdomains voneinander, besteht die Möglichkeit, dass die "From:"-Domain vorgetäuscht ist und es sich um einen Phishing-Versuch handelt.

K3 untersucht die "Received:"-Felder aller MTA´s, die eine öffentliche IP-Adresse besitzen. Die IP-Adresse des sendenden MTA´s wird via DNS aufgelöst und mit dem angegebenen Namen des MTA´s verglichen. Gibt es keine Übereinstimmung, wurde der Name des MTA´s gefälscht, was auf ein Open Relay hindeutet.

Der zeitliche Ablauf von der Bereitstellung bis zur Zustellung der E-Mail wird mit K7 analysiert. Zur Ermittlung der Zustelldauer werden die Zeitstempel des "Date:"-Feldes und des "Received:"-Feldes des zustellenden MTA´s auf UTC[30] (+0000) standardisiert. Anschließend wird der Zeitstempel des "Date:"-Feldes vom Zeitstempel des zustellenden MTA´s subtrahiert. Überschreitet die Differenz den Schwellenwert von 90 Minuten, welcher nach der Analyse der Nicht-Spam-Mails festgelegt wurde, wird die E-Mail als Spam klassifiziert.

[30] Universal Time, Coordinated.

Das Kriterium K13 vergleicht den Bereitstellungszeitpunkt der E-Mail mit dem Erstellungszeitpunkt der sendenden Domain. Das Versenden von E-Mails ohne unerwünschten Inhalt am Tag der Domainerstellung scheint recht unwahrscheinlich. Eine Übereinstimmung deutet auf Spam hin.

Die Überprüfung der Spezifikationen nach der RFC 5322 für die Felder "Date:" und "Received:" und die oben genannten Kriterien wurden mit Python umgesetzt.
Nach der Umsetzung dieser Kriterien in Python wurde eine Laufzeitanalyse durchgeführt. Die Kombination aus dem Ergebnis der Laufzeitanalyse und der Analyse der Spam-Mails ergab die Reihenfolge für die Filterung. Demzufolge arbeitet das entwickelte Filterungsverfahren die Kriterien K7, K1, K3 und K13 nacheinander ab, sofern es zu keiner Spam-Klassifizierung kommt. Anschließend wurde das entwickelte Filterungsverfahren in ein operatives System integriert.

Die Evaluation der analysierten E-Mails (46 Spam-Mails- und 24 Nicht-Spam-Mails) ergab, dass 82,61% der Spam-Mails als True Positive erkannt und keine False Positive klassifiziert wurden. Die Nichterfüllung der Kriterien teilt sich wie folgt auf: K1 4,35%, K3 65,22%, K7 10,87% und K13 2,17%. Alle Nicht-Spam-Mails wurden korrekt als True Negatives kategorisiert.

Im Rahmen der Arbeit wurden 1203 E-Mails in einem Zeitraum von fünf Monaten unter Verwendung des E-Mail Service Providers "WEB.DE" entgegengenommen. Dieser kategorisierte die E-Mails wie folgt: 119 True Positives, 761 True Negatives, 323 False Positives und keine False Negatives. Die False Positives wurden nach Erhalt nicht manuell klassifiziert (keine Spam-Markierung), damit ein ggf. vorhandener Bayes-Filter nicht angelernt wird. Dadurch könnten nachfolgende E-Mails nicht als False Positive sondern als True Positive kategorisiert werden und folglich die Evaluation verfälschen. Die Evaluation dieser E-Mails ergab, dass der entwickelte Filter 82,35% der True Positives als Spam-Mails erkannt hat. Sowohl die True Negatives, als auch die False Positives wurden zu 100% als Nicht-Spam-Mails erkannt. Die Ursache für eine Spam-Klassifizierung teilt sich prozentual folgendermaßen auf: das Nichterfüllen der Spezifikation nach RFC 5322 4,20%, von K1 9,24%, von K3 56,30%, von K7 11,77% und von K13 0,84%.

Der Vergleich beider Durchläufe wird in der Tabelle 6.1 dargestellt. Demnach erreichen beide eine Spam-Erkennungsrate von knapp 83,00%. In der Analyse wurde eine Filterung in der Reihenfolge der Kriterien K1, K3, K7 und K13 betrachtet. Aufgrund der Laufzeitanalyse dieser Kriterien wurde die Reihenfolge optimiert (K7, K1, K3 und K13). Dadurch kann sich die Abweichung beider Durchläufe in Bezug auf die Nichterfüllung der Kriterien ergeben.

Klassifizierung True Positives		entwickeltes Filterungsverfahren Nichterfüllung der Kriterien					
Evaluation	Anzahl	Spezifikation	K1	K3	K7	K13	Spam
Analyse	46	0,00%	4,35%	65,22%	10,87%	2,17%	82,61%
WEB.DE	119	4,20%	9,24%	56,30%	11,77%	0,8%	82,35%

Tabelle 6.1.: Vergleich der Durchläufe bezogen auf die Nichterfüllung der Kriterien

6.2. Fazit

Bei dem entwickelten Filterungsverfahren handelt es sich um eine regelbasierte Filterung. Die enthaltenen Kriterien werden sequenziell überprüft, sofern das vorherige Kriterium erfüllt wurde. Dementsprechend wird eine E-Mail als Spam-Mail klassifiziert, wenn sie ein Kriterium nicht erfüllt. Dies hat den Vorteil, dass eine Spam-Mail nicht alle Kriterien durchläuft und Ressourcen bindet. Der Nachteil ist, dass eine Nicht-Spam-Mail bei Nichterfüllung als False Positive kategorisiert wird, welche eventuell alle nachfolgenden Kriterien erfüllen könnte.

Aus der Arbeit geht hervor, dass eine Filterung anhand der identifizierten Prüfkriterien möglich ist. Es können knapp 83,0% der Spam-Mails erkannt werden, ohne einen False Positive zu klassifizieren. Es wird allerdings auch deutlich, dass die alleinige Filterung auf Basis der Header-Informationen an ihre Grenzen stößt. 60,8% der fälschlicherweise 119 zugestellten Spam-Mails wurden von E-Mail-Konten internationaler E-Mail Service Provider ("Gmail"[31], "Hotmail"[32], "Yahoo"[33]) verschickt. Besonders "Hotmail" rückt in den Fokus, da es mit 54,2% von 60,8% den größten Anteil ausmacht. Das entwickelte Filterungsverfahren konnte 2,5% der Spam-Mails, welche über E-Mail-Konten von E-Mail Service Providern und weitere 4,2%, welche über andere Domains versendet wurden, nicht als True Positives klassifizieren. Der Versand einer Spam-Mail über einen E-Mail Service Provider oder einer Domain, ohne Nutzung eines "Open Relays" oder der Manipulation des Headers, kann mit dem entwickelten Filterungsverfahren nicht als Spam-Mail klassifiziert werden. Folglich wäre eine Kategorisierung nur in Kombination mit einer inhaltsbasierten Filterung möglich. Dies steht jedoch im Konflikt mit den in Kapitel 2.6 genannten An-

[31]URL: https://google.com/intl/de/gmail/about/.
[32]URL: https://outlook.live.com/.
[33]URL: https://overview.mail.yahoo.com/?lang=de-DE.

forderungen, welche besagen, dass eine zentrale inhaltsbasierte Filterung im Behörden- oder Unternehmensumfeld nur zulässig ist, wenn die private Nutzung des E-Mail-Systems untersagt und die Filterung durch den Betriebsrat genehmigt wurde. Für den Fall, dass kein Betriebsrat existiert, muss dies aus der Betriebsvereinbarung hervorgehen.

Das Ziel dieser Arbeit war es, headerbasierte Prüfungen zu entwickeln, welche den Schutz vor unerwünschten E-Mails verbessern. Es konnten Kriterien auf Basis des Headers identifiziert werden, welche eine eindeutige Klassifizierung von Spam-Mails ermöglichen ohne auf den Inhalt der E-Mail zuzugreifen. Der Schutz vor unerwünschten E-Mails konnte vergleichsweise (siehe Abschnitt 5.4.2) nicht verbessert werden. Jedoch wurde die Anzahl der False Positive klassifizierten E-Mails auf null reduziert. Dadurch werden weniger Ressourcen für nachfolgende Komponenten, wie bspw. Anti-Virenfilter, inhaltsbasierte Filter oder Komponenten die "ZIP-Dateien" untersuchen, benötigt.
Daraus resultiert, dass das entwickelte Filterungsverfahren kein vollständiger Ersatz für aktuell eingesetzte hybride Filterungsverfahren ist, sondern eine wichtige Komponente in mehrstufigen Systemen darstellt.

6.3. Ausblick

In dieser Arbeit wurde deutlich, dass der Aufwand geeignete Verfahren zur Klassifizierung von Spam-Mails zu entwickeln, enorm ist. Die Mehrheit der Spam-Mails kann durch diese zwar erkannt werden, dennoch zeigen sich oben genannte Grenzen auf, z.B. der Versand von Spam-Mails über E-Mail-Konten von E-Mail Service Providern oder über die Erstellung und Nutzung eigener Domains.

Neben der Weiterentwicklung von Filterungsverfahren muss auch der Missbrauch von E-Mail-Konten großer E-Mail Service Provider betrachtet werden. Unter Angabe falscher Personendaten kann z.B. bei "Hotmail" ein E-Mail-Konto innerhalb einer Minute erstellt werden. Der einzige Mechanismus um eine automatisierte Erstellung zu verhindern, ist eine Captcha-Abfrage. Allerdings gibt es spezielle Dienste (bspw. "2Captcha"[34]), welche die Lösung von Captchas anbieten und so wiederum eine Automatisierung auch des Spamversands ermöglichen. Andere E-Mail Service Provider z.B. "Gmail" nutzen eine mobile Nummer zur Authentifizierung. Über diese wird ein Code per SMS verschickt, womit die Registrierung abgeschlossen werden kann. Dies hat zur Folge, dass der Ersteller des E-Mail-Kontos eine mobile Nummer angeben muss, auf die er Zugriff hat. Hierfür gibt

[34]URL: https://2captcha.com/.

es ebenfalls Dienste, die mobile Nummern zur Verfügung stellen (bspw. "Spoofbox"[35]). Die Nutzung solcher Dienste wird durch die E-Mail Service Provider dahingehend eingeschränkt, dass die zur Verfügung gestellten Nummern auf einer Blacklist landen.
Mögliche Schutzmechanismen wie Captchas oder die Authentifizierung mit einer mobilen Nummer sind erste Schritte, um das automatisierte Erstellen von E-Mail-Konten zu erschweren und einen möglichen Missbrauch für den Versand von Spam-Mails zu verhindern.

Resümierend ergibt sich, dass nicht nur die Weiterentwicklung von Filterungsverfahren notwendig ist, sondern vorallem die Ursachenbekämpfung, welche zum Beispiel bei der Einrichtung eines E-Mail-Kontos beginnt.
Zur Vermeidung von Spam-Mails, aber auch der Abwehr einhergehender Straftaten bspw. Diebstahl von persönlichen Daten, sollte der Zugang zu einem E-Mail-Konto für Privatpersonen nur durch eine Legitimation in Form eines Ausweisdokumentes möglich sein. Diese sollte am besten via Webcam durchgeführt werden, um den Missbrauch eines fremden Ausweisdokumentes vorzubeugen.
Der Benutzer könnte somit von einer schnellen Authentifizierung und der sinkenden Wahrscheinlichkeit, dass er ein Opfer solcher Spam-Mails wird, profitieren. Jedoch unter Preisgabe seiner Identität, Stichwort "Gläserner Mensch".
Für den E-Mail Service Provider ergibt sich neben dem Schutz gegen eine automatisierte Erstellung von E-Mail-Konten ebenfalls die Entlastung seiner IT-Infrastruktur, speziell der/des Filterungsverfahren/s. Wurden zuvor SMS-Codes zur Authentifizierung genutzt, können diese Kosten eingespart werden. Im Gegensatz dazu müsste der E-Mail Service Provider einen 24h-Service zur Legitimation anbieten, um seine aktuelle Verfügbarkeit beizubehalten.

Durch die Nutzung oder Veröffentlichung von entwickelten Filterungsverfahren werden die Spammer ggf. auf neue Filtermethoden aufmerksam und passen ihre Praktiken an. Insbesondere regelbasierte Filterungsverfahren können durch kleine Veränderungen seitens des Spammers eine falsche Klassifizierung und somit Schaden für den Empfänger zur Folge haben. Letztendlich findet ein "Wettrüsten" zwischen den Entwicklern von Filterungsverfahren und den Spammern statt.

[35] URL: https://spoofbox.com/de/tool/trash-handy.

Die Synergie zwischen E-Mail Service Providern, der Entwicklung bzw. Weiterentwicklung von Filterungsverfahren, aber auch die Bereitschaft eines Jeden, neue Mechanismen oder Verfahren zu akzeptieren, ist der Schlüssel für einen erfolgreichen Kampf gegen Spam.

Literaturverzeichnis

[1] Convios Consulting: DATENSCHUTZ UND VERSCHLÜSSELUNG Repräsentative Umfrage im Auftrag von WEB.DE und GMX. 2017. – Umfrage. – Online verfügbar unter: http://www.convios.com/wp-content/uploads/2017/07/12305-convios-datenschutz-2017-final.pdf [Abruf: 2018-08-10]

[2] Aaron, Greg ; Manning, Ronnie: Phishing Activity Trends Report 2nd Quarter 2016 / APWG. 2016. – Report. – Online verfügbar unter: https://docs.apwg.org/reports/apwg_trends_report_q2_2016.pdf [Abruf: 2018-06-08]

[3] Bambusch, Fabian: *Spam-Filter im Test: Die besten Tools für Ihr E-Mail-Postfach.* 2016. – Online verfügbar unter: https://www.pc-magazin.de/vergleich/spam-filter-test-bestenliste-e-mails-sicherheit-3196550.html [Abruf: 2018-07-31]

[4] Barth, Armin P.: *Algorithmik für Einsteiger.* 2., überarb. Aufl. Wiesbaden : Springer Spektrum, 2013. – ISBN 978–3–658–02281–5. – Online verfügbar unter: https://doi.org/10.1007/978-3-658-02282-2 [Abruf: 2018-11-26]

[5] BSI - Bundesamt für Sicherheit in der Informationstechnik: *Phishing Gefährliche Umleitung für Ihre Passwörter.* – Online verfügbar unter: https://www.bsi-fuer-buerger.de/BSIFB/DE/Risiken/SpamPhishingCo/Phishing/phishing_node.html [Abruf: 2018-07-13]

[6] Büllingen, Franz ; Hillebrand, Annette ; Stamm, Peter: *Transaktionskosten der Nutzung des Internet durch Missbrauch (Spamming) und Regulierungsmöglichkeiten.* Bad Honnef : WIK, 2006 (Diskussionsbeiträge / Wissenschaftliches Institut für Infrastruktur und Kommunikationsdienste, ISSN 0943-0288 ; ZDB-ID: 2207862-9 ; 272)

[7] Callas, J. ; PGP Corporation ; Donnerhacke, L. ; IKS GmbH ; Finney, H. ; Shaw, D. ; Thayer, R.: *OpenPGP Message Format.* November 2007. – Online verfügbar unter: https://tools.ietf.org/html/rfc4880 [Abruf: 2018-08-09]

[8] Christina, V. ; Karpagavalli, S. ; Suganya, G.: A Study on Email Spam Filtering Techniques. In: *International Journal of Computer Applications* 12 (2010), Dezember, Nr. 1, S. 7–9. – Online verfügbar unter: http://doi.org/10.5120/1645-2213 [Abruf: 2018-06-20]

[9] Crispin, M. ; University of Washington: *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1.* März 2003. – Online verfügbar unter: https://tools.ietf.org/html/rfc3501 [Abruf: 2018-07-19]

[10] Crocker, D. ; Brandenburg InternetWorking: *Internet Mail Architecture.* Juli 2009. – Online verfügbar unter: https://tools.ietf.org/html/rfc5598 [Abruf: 2018-08-03]

[11] Daboo, C. ; Stone, A.: *Sieve Email Filtering: Include Extension draft-ietf-sieve-include-05.* Juli 2010. – Online verfügbar unter: https://tools.ietf.org/html/draft-ietf-sieve-include-05 [Abruf: 2018-11-14]

[12] Demidova, Nadezhda ; Shcherbakova, Tatyana ; Vergelis, Maria: Spam and phishing in Q1 2018 / Kaspersky Lab. 2018. – Report. – Online verfügbar unter: https://securelist.com/spam-and-phishing-in-q1-2018/85650/ [Abruf: 2018-09-26]

[13] Dib, Firas: *regular expressions 101.* – Online verfügbar unter: https://regex101.com/ [Abruf: 2018-12-06]

[14] Eckert, Claudia: *IT-Sicherheit: Konzepte, Verfahren, Protokolle.* 9. Aufl. München [u.a.] : Oldenbourg, 2014. – ISBN 978–3–486–77848–9

[15] Fox, Dirk: Leiden oder löschen? Praxisanforderungen und Rechtsfolgen zentraler Spam-Filterung. In: *KES - die Zeitschrift für Informations-Sicherheit* (2004), Nr. 3, S. 6–10. – Online verfügbar unter: http://2014.kes.info/archiv/online/04-3-006.htm [Abruf: 2018-08-09]

[16] Garcia, Flavio D. ; Hoepman, Jaap-Henk ; Nieuwenhuizen, Jeroen v.: Spam Filter Analysis. In: Deswarte, Yves (Hrsg.) ; Cuppens, Frédéric (Hrsg.) ; Jajodia, Sushil (Hrsg.) ; Wang, Lingyu (Hrsg.): *Security and Protection in Information Processing Systems IFIP 18th World Computer Congress TC11 19th International Information Security Conference 22–27 August 2004 Toulouse, France.* Boston, MA : Springer, 2004. – ISBN 978–1–4020–8143–9, S. 395–410. – Online verfügbar unter: https://doi.org/10.1007/b98992 [Abruf: 2018-06-12]

[17] Gellens, R. ; QUALCOMM Incorporated ; Klensin, J.: *Message Submission for Mail.* November 2011. – Online verfügbar unter: https://tools.ietf.org/html/rfc6409 [Abruf: 2018-08-03]

[18] Gentner, Andreas: Global Mobile Consumer Survey 2017 – Mobile Evolution / Deloitte Touche Tohmatsu Limited. 2017. – Studie. – Online verfügbar unter: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/technology-media-telecommunications/Global%20Mobile%20Consumer%20Survey%202017%20Study%20Deloitte1.pdf [Abruf: 2018-07-09]

[19] Gudkova, Darya ; Vergelis, Maria ; Demidova, Nadezhda ; Shcherbakova, Tatyana: Spam im Jahr 2016 / Kaspersky Lab. 2016. – Report. – Online verfügbar unter: https://de.securelist.com/kaspersky-security-bulletin-spam-and-phishing-in-2016/72383/ [Abruf: 2018-09-26]

[20] Gudkova, Darya ; Vergelis, Maria ; Shcherbakova, Tatyana ; Demidova, Nadezhda: Spam and phishing in 2017 / Kaspersky Lab. 2017. – Report. – Online verfügbar unter: https://securelist.com/spam-and-phishing-in-2017/83833/ [Abruf: 2018-09-26]

[21] Harris, Evan: *The Next Step in the Spam Control War: Greylisting.* 2003. – Online verfügbar unter: http://projects.puremagic.com/greylisting/whitepaper.html [Abruf: 2018-07-13]

[22] IDC - International Data Corporation: *Connecting the IoT: The Road to Success.* – Online verfügbar unter: https://www.idc.com/infographics/IoT/ATTACHMENTS/IoT.pdf [Abruf: 2018-07-13]

[23] International Telecommunication Union: Recommendation ITU-T Y.2060 Overview of the Internet of things / ITU. 2012. – Studie. – Online verfügbar unter: https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-Y.2060-201206-I!!PDF-E&type=items [Abruf: 2018-07-09]

[24] Jung, Jaeyeon ; Sit, Emil: An Empirical Study of Spam Traffic and the Use of DNS Black Lists. In: *IMC '04 Proceedings of the 4th ACM SIGCOMM conference on Internet measurement.* New York, NY, USA : ACM, 2004. – ISBN 1–58113–821–0, S. 370–375. – Online verfügbar unter: 10.1145/1028788.1028838 [Abruf: 2018-11-19]

[25] Klein, Bernd: *Einführung in Python 3.* 3., überarb. Aufl. München : Hanser, 2017. – ISBN 978–3–446–45208–4

[26] Klensin, J.: *Simple Mail Transfer Protocol.* Oktober 2008. – Online verfügbar unter: https://tools.ietf.org/html/rfc5321 [Abruf: 2018-07-17]

[27] Koch, Christian: *IoT-Security: Im Netz der unsicheren Dinge.* 2018. – Online verfügbar unter: https://www.internetworld.de/technik/internet-dinge/iot-security-im-netz-unsicheren-dinge-1567969.html [Abruf: 2018-07-31]

[28] Konferenz der unabhängigen Datenschutzbehörden des Bundes und der Länder: *Orientierungshilfe der Datenschutzaufsichtsbehörden zur datenschutzgerechten Nutzung von E-Mail und anderen Internetdiensten am Arbeitsplatz.* 2016. – Online verfügbar unter: https://www.baden-wuerttemberg.datenschutz.de/wp-content/uploads/2016/02/OH_E-Mail_Internet_Arbeitsplatz.pdf [Abruf: 2018-08-30]

[29] Levine, J. ; Taughannock Networks: *DNS Blacklists and Whitelists.* Februar 2010. – Online verfügbar unter: https://www.rfc-editor.org/rfc/rfc5782.txt [Abruf: 2018-11-19]

[30] Metz, Charles E.: Basic Principles of ROC Analysis. In: *Seminars in Nuclear Medicine* (1978), Nr. 4, S. 283–298. – Online verfügbar unter: https://doi.org/10.1016/S0001-2998(78)80014-2 [Abruf: 2018-08-31]

[31] Mitnick, Kevin D. ; Simon, William L.: *The Art of Deception: Controlling the Human Element of Security.* 1. Aufl. New York : Wiley, 2002. – ISBN 978–0471237129

[32] Myers, J. ; Mellon, Carnegie ; Rose, M. ; Dover Beach Consulting, Inc.: *Post Office Protocol - Version 3.* Mai 1996. – Online verfügbar unter: https://tools.ietf.org/html/rfc1939 [Abruf: 2018-07-19]

[33] Nahorney, Ben: ISTR Email Threats 2017 / Symantec Corporation. 2017. – ISTR Special Report. – Online verfügbar unter: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/istr-email-threats-2017-en.pdf [Abruf: 2018-06-07]

[34] National Office for the Information Economy (NOIE): SPAM : final report of the NOIE review of the spam problem and how it can be countered / NOIE. 2003. – Spam Report. – Online verfügbar unter: http://pandora.nla.gov.au/pan/42105/20040520-0000/www2.dcita.gov.au/__data/assets/file/13050/SPAMreport.pdf [Abruf: 2018-06-11]

[35] OECD Organisation for Economic Co-operation and Develop: *Computer Viruses and Other Malicious Software: A Threat to the Internet Economy*. Paris : OECD Publishing, 2009. – ISBN 9264056505. – Online verfügbar unter: http://dx.doi.org/10.1787/9789264056510-en [Abruf: 2018-06-11]

[36] OECD Organisation for Economic Co-operation and Develop: *Online Identity Theft*. Paris : OECD Publishing, 2009. – ISBN 9264056580. – Online verfügbar unter: http://dx.doi.org/10.1787/9789264056596-en [Abruf: 2018-06-08]

[37] Proofpoint, Inc.: *Proofpoint Uncovers Internet of Things (IoT) Cyberattack*. 2014. – Online verfügbar unter: https://www.proofpoint.com/us/proofpoint-uncovers-internet-things-iot-cyberattack [Abruf: 2018-07-13]

[38] Ramsdell, B. ; Worldtalk: *S/MIME Version 3 Message Specification*. Juni 1999. – Online verfügbar unter: https://tools.ietf.org/html/rfc2633 [Abruf: 2018-08-09]

[39] Resnick, P. ; Qualcomm Incorporated: *Internet Message Format*. Oktober 2008. – Online verfügbar unter: https://tools.ietf.org/html/rfc5322 [Abruf: 2018-07-17]

[40] Rossmann, Nils L.: Anti-Spam Techniken. In: Juling, Prof. Dr. Wilfried (Hrsg.) ; Hartenstein, Prof. Dr. Hannes (Hrsg.) ; Dinger, Jochen (Hrsg.): *IT-Management in der Praxis Seminar – WS 2004/05*. Karlsruhe, 2005, S. 77–90. – Online verfügbar unter: https://dsn.tm.kit.edu/medien/publication-seminar/seminar-it-mgmt-ws0405.pdf [Abruf: 2018-08-13]

[41] Sahami, Mehran ; Dumais, Susan ; Heckerman, David ; Horvitz, Eric: A Bayesian approach to filtering junk e-mail. In: *Learning for Text Categorization: Papers from the AAAI workshop*. Madison, Wisconsin : The AAAI Press, 1998, S. 55–62. – Online verfügbar unter: https://aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-009.pdf [Abruf: 2018-08-30]

[42] Schryen, Guido: Effektivität von Lösungsansätzen zur Bekämpfung von Spam. In: *WIRTSCHAFTSINFORMATIK* 46 (2004), August, Nr. 4, S. 281–288. – ISSN 1861–8936. – Online verfügbar unter: https://link.springer.com/article/10.1007/BF03250945 [Abruf: 2018-07-12]

[43] Schwenk, Jörg: *Sicherheit und Kryptographie im Internet: Von sicherer E-Mail bis zu IP-Verschlüsselung*. 3., überarb. Aufl. Wiesbaden : Vieweg + Teubner, 2010. – ISBN 978–3–8348–0814–1

[44] Tavosanis, Mirko: A Causal Classification of Orthography Errors in Web Texts. In: Knoblock, Craig (Hrsg.) ; Lopresti, Daniel (Hrsg.) ; Roy, Shourya (Hrsg.) ; Subramaniam, L. Venkata (Hrsg.): *Proceedings of IJCAI-07 Workshop on Analytics for Noisy Unstructured Text Data (AND-07)*. Hyderabad, India, 2007, S. 99–106. – Online verfügbar unter: http://research.ihost.com/and2007/cd/Proceedings_files/p99.pdf [Abruf: 2018-08-07]

[45] The Apache Software Foundation: *Welcome*. – Online verfügbar unter: https://spamassassin.apache.org/ [Abruf: 2018-11-19]

[46] The Radicati Group, Inc.: Email Statistics Report, 2018-2022 / The Radicati Group, Inc. 2018. – Studie. – Online verfügbar unter: https://www.radicati.com/wp/wp-content/uploads/2017/12/Email-Statistics-Report-2018-2022-Executive-Summary.pdf [Abruf: 2018-06-07]

[47] The Spamhaus Project: *The Definition of Spam*. – Online verfügbar unter: https://www.spamhaus.org/consumer/definition/ [Abruf: 2018-06-11]

[48] THM - Technische Hochschule Mittelhessen: *Was ist Phishing?* 2016. – Online verfügbar unter: https://www.its.thm.de/phishing/phishing.html [Abruf: 2018-07-13]

[49] Unbekannt: *Postfix After-Queue Content Filter*. – Online verfügbar unter: http://www.postfix.org/FILTER_README.html [Abruf: 2018-10-28]

[50] Unbekannt: *Postfix Before-Queue Content Filter*. – Online verfügbar unter: http://www.postfix.org/SMTPD_PROXY_README.html [Abruf: 2018-10-28]

[51] van der Meulen, Rob: Gartner Says 8.4 Billion Connected "Things" Will Be in Use in 2017, Up 31 Percent From 2016 / Gartner, Inc. 2017. – Studie. – Online verfügbar unter: https://www.gartner.com/newsroom/id/3598917 [Abruf: 2018-07-31]

[52] Vergelis, Maria ; Demidova, Nadezhda ; Shcherbakova, Tatyana: Spam and phishing in Q2 2018 / Kaspersky Lab. 2018. – Report. – Online verfügbar unter: https://securelist.com/spam-and-phishing-in-q2-2018/87368/ [Abruf: 2018-09-26]

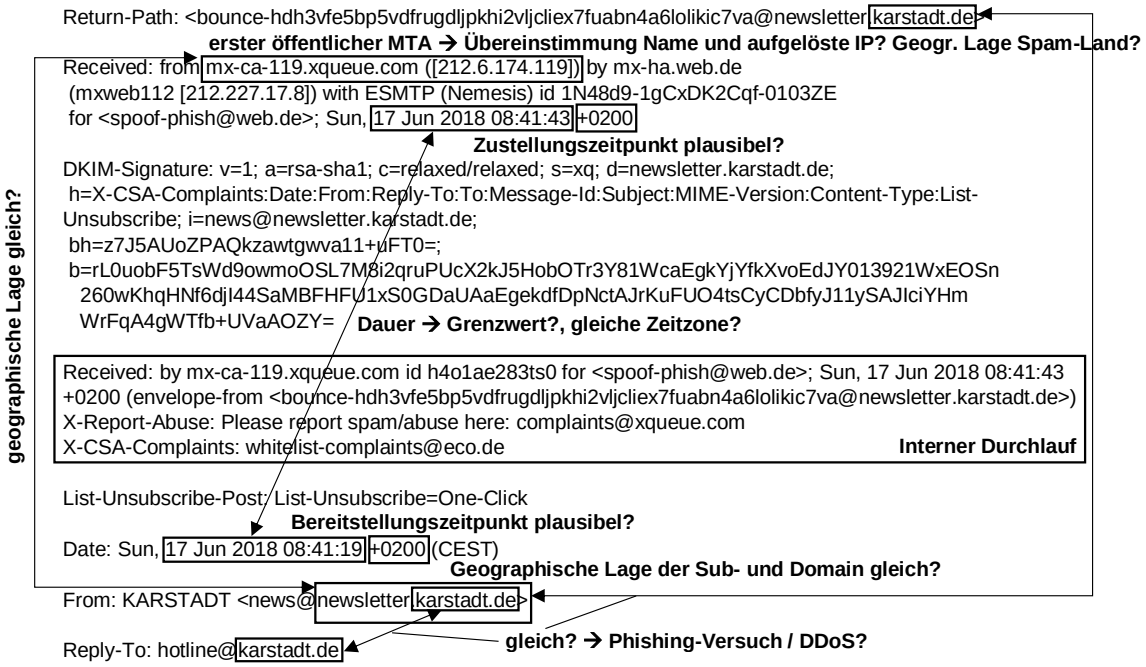
[53] Vergelis, Maria ; Shcherbakova, Tatyana ; Demidova, Nadezhda ; Gudkova, Darya: SPAM AND PHISHING IN 2015 / Kaspersky Lab. 2015. – Report. – Online verfügbar unter: https://securelist.com/kaspersky-security-bulletin-spam-and-phishing-in-2015/73591/ [Abruf: 2018-09-26]

[54] Wang, Chih-Chien ; Chen, Sheng-Yi: Using header session messages to anti-spamming. In: *Computers & Security* 26 (2007), 08, Nr. 5, S. 381–390. – Online verfügbar unter: https://doi.org/10.1016/j.cose.2006.12.012 [Abruf: 2018-12-11]

[55] Wolf, Andre: *Phishing-Radar: Nicht auf DSGVO-Fake hereinfallen!* Mai 2018. – Online verfügbar unter: https://www.mimikama.at/allgemein/phishing-radar-2/ [Abruf: 2018-06-08]

Anhangsverzeichnis

A. Zusammenfassung möglicher Prüfkriterien, resultierend aus der Untersuchung einiger Spam- und Nicht-Spam-Mails



To: spoof-phish@web.de

Message-Id: <hdh3vfe5bp5vdfrugdljpkhi2vljcliex7fuabn4a6lolikic7va@newsletter.karstadt.de>

Subject: =?UTF-8?Q?Noch_3_Tage_11=E2=82=AC_geschenkt!?=

MIME-Version: 1.0

Content-Type: multipart/alternative;
boundary="----=_Part_17200476_809743754.1529217679400"

List-Unsubscribe: list-unsubscribe@newsletter.karstadt.de,
http://www.karstadt.de/

X-Mailer: Maileon

X-Maileon-FingerPrint: hdh3vfe5bp5vdfrugdljpkhi2vljcliex7fuabn4a6lolikic7va

X-Campaign: M.4082|630

Feedback-ID: Typeregular:MailingID630:Sender4082

Envelope-To: <spoof-phish@web.de>

B. Laufzeitanalyse der umgesetzten Kriterien

K1:

```
def check_crit_one(short_from, short_reply):
    global failure_msg
    if short_from.__len__() > 0:
        answer = use_console('nslookup', short_from)
        result = re.findall(r'\*\* server', answer)
        if result.__len__() == 0:
            if short_reply is None:
                return True
            elif short_reply.__len__() > 0:
                if short_from == short_reply:
                    return True
                else:
                    failure_msg = "\'From:\' Field != \'Reply-To:\' Field"
                    return False
            else:
                failure_msg = "Amount of \'Reply-To:\' Field > 1:"
                return False
        else:
            failure_msg = "DNS can't find this domain        "
            return False
    else:
        failure_msg = "Amount of \'From:\' Field != 1    "
        return False
```

Zeile	Ebene 1	Ebene 2	Ebene 3	Gesamt
global failure_msg	O(1)			O(n)
if short_from.__len__() > 0:	O(1)		O(n)	
answer = …	O(n)			
result = …	O(n)			
if result.__len__() == 0:	O(1)	O(1)		
if short_reply is None:	O(1)			
return True	O(1)			
elif short_reply.__len__() > 0:	O(1)			
if short_from == short_reply:	O(1)			
return True	O(1)			
else:	O(1)			
failure_msg = …	O(1)			
return False	O(1)			
else:	O(1)			
failure_msg = …	O(1)			
return False	O(1)			
else:	O(1)			
failure_msg = …	O(1)			
return False	O(1)			
else:	O(1)	O(1)		
failure_msg = …	O(1)			
return False	O(1)			

K3:

```
def check_crit_three(group_received):
    global failure_msg
    if group_received.__len__() > 0:
        result = False
        for mta in group_received[::-1]:
            if check_ip_public(mta[2]):
                answer = use_console('nslookup', mta[2])
                match = re.findall(r'name = (.+\..+)\.', answer)
                if match:
                    for x in match:
                        if mta[0] == x:
                            result = True
                            break
                        else:
                            result = False
                    if not result:
                        failure_msg = 'The Name of the MTA is spoofed!   '
                        break
                else:
                    failure_msg = "IP-Address don't exists!          "
                    result = False
                    break
        return result
    else:
        failure_msg = "No valid 'Received:' Field        "
        return False
```

Zeile	Ebene 1	Ebene 2	Ebene 3	Ebene 4	Gesamt
global failure_msg	O(1)				O(n²)
if group_received.__len__() > 0:	O(1)				
result = False	O(1)				
for mta in group_received[::-1]:	O(n)			O(n²)	
if check_ip_public(mta[2]):	O(n)		O(n)		
answer = …	O(n)				
match = …	O(n)				
if match:	O(1)	O(n)			
for x in match:	O(n)				
if mta[0] == x:	O(1)				
result = True	O(1)				
break	O(1)				
else:	O(1)				
result = False	O(1)				
if not result:	O(1)				
failure_msg = …	O(1)				
break	O(1)				
else:	O(1)				
failure_msg = …	O(1)				
result = False	O(1)				
break	O(1)				
return result	O(1)				
else:	O(1)				
failure_msg = …	O(1)				
return False	O(1)				

K7:

```
def check_crit_seven(field_date, group_received):
    global failure_msg
    sender_match = re.match(get_regex_date(), ''.join(field_date))
    receiver_match = re.match(get_regex_date(), group_received[0][8])
    s = get_datetime_obj(field_date, True)
    r = get_datetime_obj(group_received[0][8], True)
    s_zone = get_utc(sender_match.group('zone1'), sender_match.group('zone2'))
    r_zone = get_utc(receiver_match.group('zone1'), receiver_match.group('zone2'))
    s_utc = s - datetime.timedelta(seconds=(s_zone * 60 * 60))
    r_utc = r - datetime.timedelta(seconds=(r_zone * 60 * 60))
    diff = r_utc - s_utc
    if diff.seconds > 5400:
        failure_msg = 'Duration of Delivery > 90 minutes '
        return False
    else:
        return True
```

Zeile	Ebene 1	Ebene 2	Gesamt
global failure_msg	O(1)	O(n)	O(n)
sender_match = …	O(n)		
receiver_match = …	O(n)		
s = …	O(n)		
r = …	O(n)		
s_zone = …	O(n)		
r_zone = …	O(n)		
s_utc = …	O(1)		
r_utc = …	O(1)		
diff = r_utc - s_utc	O(1)		
if diff.seconds > 5400:	O(1)	O(1)	
failure_msg = …	O(1)		
return False	O(1)		
else:	O(1)		
return True	O(1)		

K13:

```
def check_crit_thirteen(short_from, field_date):
    global failure_msg
    answer = use_console('whois', short_from)
    match = re.findall(r'Creation Date: (?P<year>\d{4})-(?P<month>\d{2})-(?P<day>\d{2})', answer)
    if match.__len__() > 0:
        date_header = get_datetime_obj(field_date, False)
        date_creation = datetime.datetime(int(match[0][0]), int(match[0][1]), int(match[0][2]))
        diff = date_header - date_creation
        if diff.days == 0:
            failure_msg = "Domain Creation == \'Date:\' Field  "
            return False
    return True
```

O(1)		O(n)
O(n)		
O(n)		
O(1)	O(n)	
O(n)		
O(1)		
O(1)		
O(1)		
O(1)		
O(1)		
O(1)		

Hilfsmethoden:

```
def use_console(command, temp_str):
    process = subprocess.Popen([command, temp_str], stdout=subprocess.PIPE)
    process.wait()
    return process.stdout.read()
```

O(n)		O(n)
O(1)		
O(n)		

```
def check_ip_public(ip):
    oct_list = []
    oct = ''
    for char in ip:
        if char == '.':
            oct_list.append(oct)
            oct = ''
        else:
            oct += char
    oct_list.append(oct)
    if oct_list[0] in ['10']:
        return False
    if oct_list[0] in ['172', '192']:
        ip_str = ''
        for numb in oct_list:
            ip_str += numb.zfill(3)
        if ('172016000000' <= ip_str <= '172031255255') | \
                ('192168000000' <= ip_str <= '192168255255'):
            return False
        else:
            return True
    else:
        return True
```

O(1)		O(n)
O(1)		
O(n)		
O(1)		
O(1)	O(n)	
O(1)		
O(1)		
O(1)		
O(1)		
O(1)		
O(1)		
O(1)	O(1)	
O(1)		
O(1)		
O(1)		
O(1)		
O(1)		

```
def get_datetime_obj(obj, bool_time):
    switcher = {
        "Jan": 1,
        "Feb": 2,
        "Mar": 3,
        "Apr": 4,
        "May": 5,
        "Jun": 6,
        "Jul": 7,
        "Aug": 8,
        "Sep": 9,
        "Oct": 10,
        "Nov": 11,
        "Dec": 12
    }

    hour = 0
    minute = 0
    sec = 0

    match = re.match(get_regex(), ''.join(obj))
    year = int(match.group('year'))
    month = switcher.get(match.group('month'), "Invalid month")
    day = int(match.group('day_num'))
    if bool_time:
        hour = int(match.group('hour'))
        minute = int(match.group('min'))
        sec = int(match.group('sec'))
    return datetime.datetime(year, month, day, hour, minute, sec)
```

O(1)		O(n)
O(1)		
O(1)		
O(1)		
O(n)		
O(n)		
O(n)		
O(n)		
O(1)	O(n)	
O(n)		
O(n)		
O(n)		
O(1)		

C. Tabellen

Legende:

- x: erfüllt das Kriterium
- K1: das "From:"- und "Reply-To:"-Feld (falls vorhanden) haben die gleiche Hauptdomain
- K2: die geographische Lage der "From:"-Domain entspricht der, des ersten öffentlichen MTA´s ("Received:"-Feld)
- K3: die aufgelöste IP-Adresse des öffentlichen MTA´s entspricht dem angegebenen Namen (alle "Received:"-Felder)
- K4: das Herkunftsland des ersten öffentlichen MTA´s entspricht keinem Spam-Herkunftsland (Top 15)
- K5: die geographische Lage der Subdomain, stimmt mit der der Hauptdomain überein ("From:"-Feld)
- K6: der erste öffentliche MTA stimmt mit dem zuständigen MTA der "From:"-Domain überein
- K7: die Zeitpunkte der Bereitstellung und Zustellung sind plausibel und überschreiten nicht die Dauer der Zustellung
- K8: der Zeitpunkt der Bereitstellung liegt zwischen 07:00 - 21:00 Uhr
- K9: der Zeitpunkt der Zustellung liegt zwischen 07:00 - 21:00 Uhr
- K10: die Zeitzone des "Date:"-Feldes und des ersten öffentlichen MTA´s stimmen überein
- K11: die Länge und Formatierung der "Received:"-Felder entspricht den Empfehlungen
- K12: die MTA-Weiterleitungen sind plausibel (falls vorhanden)
- K13: die E-Mail-Zustellung und die Erstellung der "From:"-Domain erfolgte nicht am selben Tag

C.1. Übersicht der analysierten Nicht-Spam-Mails (TN)

Nr.	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13
1	x	x	x		x	x	x	x	x	x	x	x	x
2	x	x	x		x	x	x	x	x	x		x	x
3	x		x	x			x	x	x	x	x	x	x
4	x		x	x	x		x	x	x	x	x	x	x
5	x	x	x		x		x	x	x	x	x	x	x
6	x	x	x		x	x	x	x	x	x	x	x	x
7	x	x	x		x	x	x	x	x	x	x	x	x
8	x		x		x		x	x	x		x	x	x
9	x	x	x		x	x	x	x	x	x	x	x	x
10	x	x	x		x	x	x					x	x
11	x	x	x		x		x	x	x	x	x	x	x
12	x	x	x		x	x	x	x	x	x		x	x
13	x	x	x	x	x		x	x	x	x	x	x	x
14	x	x	x	x		x	x	x	x		x	x	x
15	x		x	x	x		x	x	x	x	x	x	x
16	x	x	x		x		x	x	x	x	x	x	x
17	x	x	x				x	x	x				x
18	x	x	x		x	x	x	x	x	x	x	x	x
19	x		x				x	x	x	x	x	x	x
20	x	x	x		x	x	x	x	x	x	x	x	x
21	x	x	x		x		x			x			x
22	x	x	x			x	x	x	x	x		x	x
23	x	x	x				x	x	x			x	x
24	x	x	x				x	x	x		x	x	x
	100%	79%	100%	21%	71%	38%	100%	92%	92%	75%	77%	92%	100%
		↓				↓		↓	↓		↓		
	74%												

C.2. Übersicht der analysierten Spam-Mails (TP)

Nr.	K1	K2	K3	K4	K5	K6	K7	K8	K9	K10	K11	K12	K13
1					x		x	x	x		x		x
2	x	x	x		x		x		x	x	x	x	x
3	x		x		x	x	x		x		x	x	x
4	x	x	x		x		x	x	x			x	x
5	x	x			x		x	x	x		x		x
6	x	x			x		x				x		x
7	x	x	x		x	x	x	x			x	x	x
8	x	x			x		x	x	x		x		x
9	x			x	x		x	x	x		x		x
10	x	x			x		x	x	x		x		x
11	x			x	x		x	x	x		x		x
12	x	x	x		x		x	x				x	x
13	x			x	x		x	x			x		x
14	x	x			x		x	x	x		x		x
15	x	x			x		x	x	x		x		x
16	x			x	x		x	x	x		x		x
17	x			x	x		x				x		x
18	x			x	x		x	x	x		x		x
19	x	x	x		x	x	x	x					x
20	x			x	x		x				x		x
21	x			x	x		x	x	x		x		x
22		x	x	x	x	x		x	x	x		x	x
23	x	x			x		x	x	x		x		x
24	x			x	x		x	x	x		x		x
25	x	x			x		x	x	x		x		x
26	x			x	x		x	x	x		x		x
27	x			x	x		x	x	x		x		x
28	x	x			x		x	x	x		x		x
29	x	x		x	x		x	x	x		x		x
30	x	x		x	x		x	x	x		x		x

31		x	x		x		x	x	x				x
32	x	x			x		x	x	x		x		x
33	x	x	x		x			x	x		x	x	
34	x	x			x		x	x	x		x		x
35	x	x			x		x	x	x		x		x
36	x	x	x		x			x			x	x	
37	x	x			x		x	x	x		x		x
38	x	x	x		x		x	x			x	x	
39	x	x	x			x	x		x		x	x	x
40	x	x			x		x	x	x		x		x
41	x	x			x		x	x	x		x		x
42	x	x	x			x	x	x	x		x	x	x
43	x		x		x			x	x			x	x
44	x		x		x			x	x			x	x
45	x	x			x		x				x		x
46	x	x			x		x		x		x		x
	94%	63%	33%	30%	94%	11%	89%	83%	78%	4%	85%	28%	93%
		↓				↓		↓	↓		↓		
	64%												

C.3. Treffermenge (TP) nach der Anwendung von K1, K3, K7 und K13

Nr.	K2	K4	K5	K6	K8	K9	K10	K11	K12
2	x		x			x	x	x	x
3			x	x		x		x	x
4	x		x		x	x			x
7	x		x	x	x			x	x
12	x		x		x				x
19	x		x	x	x				
39	x			x		x		x	x
42	x			x	x	x		x	x
	88%	0%	75%	63%	63%	63%	13%	63%	88%
	↓			↓	↓	↓		↓	
	68%								

C.4. Evaluation des entwickelten Filterungsverfahrens

1. Durchlauf:

web.de Filter		entwickelter Filter				
Klassifizierung	Anzahl	TP	FN	TN	FP	Spam
TP	46	39	7			84,78%
TN	24			24		0,00%
Summe	70					

2. Durchlauf:

web.de Filter		entwickelter Filter				
Klassifizierung	Anzahl	TP	FN	TN	FP	Spam
TP	119	98	21			82,35%
TN	761			761		0,00%
FP	323			323		0,00%
Summe	1203					

Ursachen für die Spam-Klassifizierung:

Klassifizierung True Positives		entwickeltes Filterungsverfahren Nichterfüllung der Kriterien					
Evaluation	Anzahl	Spezifikation	K1	K3	K7	K13	Spam
1. Durchlauf	46	0,00%	4,35%	65,22%	10,87%	2,17%	82,61%
2. Durchlauf	119	4,20%	9,24%	56,30%	11,77%	0,84%	82,35%

D. Quellcode - Entwickeltes Filterungsverfahren

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# Author: Robert Lehnert
# Date: 2018-12-01
# Project: bachelor thesis / HAW Hamburg

import smtplib
import smtpd
import asyncore
import socket
import re
import subprocess
import time
import datetime

HOST = '127.0.0.1'
LISTEN_PORT = 10025
REPLY_PORT = 10026
PATH_LOG = '/etc/postfix/filter/logfile'

failure_msg = "*                                    "


# This class implements a SMTP server which is waiting for incoming emails (LISTEN_PORT).
# The email is prepared for analysis and then ckecked for some criteria. After filtering
# the email is classified as non-spam or spam. When the email is marked as spam, the header
# will be extended with a spam flag. The result is logged in a log file to make statistics.
# At the end the email is transferred back to the MTA (REPLY_PORT).
class Filter(smtpd.SMTPServer):
    def process_message(self, peer, mailfrom, rcpttos, data):
        # ************************************************************
        # ******************    PREPARATION    ***********************
        # ************************************************************

        # separates the header from the body of the email (d) and returns a list
        # [0] = header, [1] = body
        # @param string
        # @return list
        def separate_header(d):
            temp = []
```

```
42              if re.match(r'.*\r\n.*', ''.join(d)):
43                  msg = d.split('\r\n')
44              else:
45                  msg = d.split('\n')
46              for x in msg:
47                  if x == '':
48                      temp.append(msg[:msg.index(x)])  # header
49                      temp.append(msg[msg.index(x):])  # body
50                      break
51              return temp
52
53          # Prepares the header for the following analysis.
54          # Removes line breaks and assigns the fields which are needed to
                variables.
55          # Returns the relevant variables for analysis.
56          # @param list
57          # @return list, list, list, list, list, list, string, string
58          def prepare_fields(header):
59              field_from, field_to, field_reply, group_received, field_date = [],
                    [], [], [], []
60              field_received, field_received_begin, field_received_multi = [], [],
                    []
61              inside_received = False
62              previous_line = None
63
64              # assignment
65              for line in header:
66                  first_word = ''
67                  for char in line:
68                      if char != ' ':
69                          first_word += char
70                      else:
71                          break
72                  if re.findall(r'.*\r\n', line):
73                      line = line[:-2]
74                  if re.findall(r'.*\n', line):
75                      line = line[:-1]
76                  if re.findall(r'.*>$', line):
77                      line = line[:-1]
78                  if re.findall(r'^\S', line):
79                      inside_received = False
80                  if inside_received:
81                      field_received_multi.append(line)
82                  if first_word == 'Received:':
83                      field_received_begin.append(line)
84                      field_received_multi.append('NEXT')
85                      inside_received = True
```

```
86                  if first_word == 'From:':
87                      field_from.append(line)
88                  if first_word == 'Envelope-To:':  # used for identifier in
                        logfile
89                      field_to.append(line)
90                  if first_word == 'Reply-To:':
91                      field_reply.append(line)
92                  if first_word == 'Date:':
93                      field_date.append(line)
94                  if re.findall(r'^\s', line):  # for folding case
95                      if previous_line == 'From:':
96                          del field_from[-1]
97                          field_from.append(line)
98                      if previous_line == 'Reply-To:':
99                          del field_reply[-1]
100                         field_reply.append(line)
101                 previous_line = first_word
102
103             # prepares the "Received:" fields
104             for x in field_received_begin:
105                 full = [x]
106                 del field_received_multi[0]
107                 pos = 0
108                 for y in field_received_multi:
109                     if y == 'NEXT':
110                         del field_received_multi[:pos]
111                         break
112                     else:
113                         full.append(y)
114                         pos += 1
115                 field_received.append(full)
116
117             # finds "Received:" fields with correct syntax
118             for received_obj in field_received:
119                 received_str = ''.join(received_obj)
120                 # match IPv4
121                 match = re.match(get_regex_received(), received_str, re.S)
122                 if match:
123                     # check if IPv4 address exists
124                     if valid_ip(match.group('s_ip')):
125                         group_received.append(match.groups())
126
127             # truncates the domain of the available fields 'From:' and 'Reply-To
                    :'
128             short_from, short_reply = '', ''
129             if field_from.__len__() == 1:
130                 short_from = get_short_domain(field_from)
```

```
131             if field_reply.__len__() == 0:
132                 short_reply = None
133             elif field_reply.__len__() == 1:
134                 short_reply = get_short_domain(field_reply)
135 
136             return field_from, field_to, field_reply, field_received,
                    group_received, field_date, short_from, short_reply
137 
138         # ************************************************************
139         # ********************    CRITERIA    ************************
140         # ************************************************************
141 
142         # Checks if the fields 'Date:' and 'Received:' meet the specification of
                the
143         # RFC 5321 and 5322.
144         # True = the criterion has been fulfill, False = classified as spam
145         # @param list, list
146         # @return True if both fields fulfill the specification, else False
147         def check_specification(field_date, group_received):
148             regex_date = r'^Date:(\s?(?P<day_alpha>\w{3}),)?\s?' \
149                          r'(?P<day_num>\d{1,2})\s' \
150                          r'(?P<month>\w{3})\s' \
151                          r'(?P<year>\d{4})\s' \
152                          r'(?P<time>(?P<hour>\d{2}):(?P<min>\d{2})(:(?P<sec>\d
                                {0,2}))?)\s' \
153                          r'((?P<utc>(?P<zone>(?P<sign>\D)\d{4}|\w{2,3})))(\s
                                \(.*\))?$'
154             return re.match(regex_date, ''.join(field_date)) and (group_received.
                    __len__() > 0)
155 
156         # Compares the domains of the 'From:' and the 'Reply-To:' fields.
157         # Checks if the 'From:' field and its domain exist. If both are exist
158         # the 'Reply-To:' field is checked.
159         # If there is no 'Reply-To:' field, the email is categorized as non-spam.
160         # The email is also classified as non-spam, if both fields exist and
                identical.
161         # In all other cases, the email is classified as spam.
162         # True = the criterion has been met, False = classified as spam
163         # @param string, string
164         # @return True if both fields are equal or only the 'From:' field exist,
                else False
165         def check_crit_one(short_from, short_reply):
166             global failure_msg
167             if short_from.__len__() > 0:
168                 # checks if the domain exists
169                 answer = use_console('nslookup', short_from)
170                 result = re.findall(r'\*\* server', answer)
```

```
171                 # domain exists
172                 if result.__len__() == 0:
173                     if short_reply is None:
174                         return True
175                     elif short_reply.__len__() > 0:
176                         # checks if "From:" and "Reply-To:" has the same domain
177                         if short_from == short_reply:
178                             return True
179                         else:
180                             failure_msg = "'From:' Field != 'Reply-To:' Field
                                "
181                             return False
182                     else:
183                         failure_msg = "Amount of 'Reply-To:' Field > 1:"
184                         return False
185                 # domain doesn't exist
186                 else:
187                     failure_msg = "DNS can't find this domain        "
188                     return False
189             else:
190                 failure_msg = "Amount of 'From:' Field != 1    "
191                 return False
192 
193         # Each "Received:" field is checked to see if the provisioning MTA
194         # has a public IP address. Is the IP address a public address it checks
195         # to see if the resolved name matches the specified name of the
196         # providing MTA.
197         # True = the criterion has been met, False = classified as spam
198         # @param list
199         # @return True if no manipulation of 'Received:' fields, else False
200         def check_crit_three(group_received):
201             global failure_msg
202             if group_received.__len__() > 0:
203                 result = False
204                 # Reverse order because manipulated 'Received:' fields are more
205                 # likely to be included at the beginning of the transfer.
206                 for mta in group_received[::-1]:
207                     if check_ip_public(mta[2]):
208                         answer = use_console('nslookup', mta[2])
209                         match = re.findall(r'name = (.+\..+)\.', answer)
210                         if match:
211                             for x in match:
212                                 if mta[0] == x:
213                                     result = True
214                                     break
215                                 else:
216                                     result = False
```

```
217                             if not result:
218                                 failure_msg = 'The Name of the MTA is spoofed!
                                        '
219                                 break
220                         else:
221                             failure_msg = "IP-Address don't exists!          "
222                             result = False
223                             break
224                 return result
225                 # checks internal email access / private IP
226                 # code here
227             # e.g. IP address instead of a mta name
228             else:
229                 failure_msg = "No valid 'Received:' Field        "
230                 return False
231 
232         # Checks the plausibility of timestamps and determined the difference
              between the
233         # time of email provision and the time of delivering. If the difference
              higher
234         # then 90 minutes the email is marked as spam.
235         # True = the criterion has been met, False = classified as spam
236         # @param list, list
237         # @return
238         def check_crit_seven(field_date, group_received):
239             global failure_msg
240             # date of mailing
241             sender_match = re.match(get_regex_date(), ''.join(field_date))
242             # [0] = delivering MTA (our provider), [8] = group date
243             receiver_match = re.match(get_regex_date(), group_received[0][8])
244 
245             # gets Datetime object of 'Date:' and the delivering MTA
246             s = get_datetime_obj(field_date, True)
247             r = get_datetime_obj(group_received[0][8], True)
248 
249             # normalizes the timezone to UTC (+0000)
250             s_zone = get_utc(sender_match.group('zone1'), sender_match.group('
                  zone2'))
251             r_zone = get_utc(receiver_match.group('zone1'), receiver_match.group(
                  'zone2'))
252 
253             # Datetime object with UTC (+0000)
254             s_utc = s - datetime.timedelta(seconds=(s_zone * 60 * 60))
255             r_utc = r - datetime.timedelta(seconds=(r_zone * 60 * 60))
256 
257             # calculates the timedelta
258             diff = r_utc - s_utc
```

```
259             if diff.seconds > 5400:
260                 failure_msg = 'Duration␣of␣Delivery␣>␣90␣minutes␣'
261                 return False
262             else:
263                 return True
264 
265         # Checks if the date of email provision ('Date:' field) is the same as
266         # the date of the domain creation.
267         # True = the criterion has been met, False = classified as spam
268         # @param string, list
269         # @return True if the dates are different, else False
270         def check_crit_thirteen(short_from, field_date):
271             global failure_msg
272             answer = use_console('whois', short_from)
273             regex_creation = r'Creation␣Date:␣(?P<year>\d{4})-(?P<month>\d{2})-(?
                P<day>\d{2})'
274             match = re.findall(regex_creation, answer)
275             if match.__len__() > 0:
276                 date_header = get_datetime_obj(field_date, False)
277                 day, month, year = int(match[0][0]), int(match[0][1]), int(match
                    [0][2])
278                 date_creation = datetime.datetime(day, month, year)
279                 # calculate the timedelta
280                 diff = date_header - date_creation
281                 if diff.days == 0:
282                     failure_msg = "Domain␣Creation␣==␣\'Date:\'␣Field␣␣"
283                     return False
284             return True
285 
286         # ************************************************************
287         # ******************    HELPER METHODS    ********************
288         # ************************************************************
289 
290         # Separates the mail account, sub-, domain & TLD and returns the domain+
                TLD.
291         # example: [sub.domain.com] -> "domain.com"
292         # @param list (FQDN)
293         # @return string
294         def get_short_domain(mailadd):
295             split_at = ''.join(mailadd).rsplit('@')
296             string_domain = split_at[split_at.__len__() - 1]
297             split_domain = string_domain.rsplit('.')
298             domain = split_domain[split_domain.__len__() - 2]
299             tld = split_domain[split_domain.__len__() - 1]
300             return domain + "." + tld
301 
```

```
302         # Creates a subprocess of the transferred command and parameter and
                returns the answer.
303         # @param string, string
304         # @return string
305         def use_console(command, temp_str):
306             process = subprocess.Popen([command, temp_str], stdout=subprocess.
                    PIPE)
307             process.wait()
308             return process.stdout.read()
309 
310         # Checks if the transferred IP address is a valid IP address (octet
                between 1-255).
311         # @param string
312         # @return True if valid, else False
313         def valid_ip(ip):
314             try:
315                 socket.inet_aton(ip)
316                 return True
317             except:
318                 return False
319 
320         # Examines the transferred IP address if its a public address or not.
321         # private 10.0.0.0 - 10.255.255.255 // 172.16.0.0 - 172.31.255.255 //
322         #         192.168.0.0 - 192.168.255.255
323         # @param list
324         # @return True if public, else False
325         def check_ip_public(ip):
326             oct_list = []
327             oct = ''
328             for char in ip:
329                 if char == '.':
330                     oct_list.append(oct)
331                     oct = ''
332                 else:
333                     oct += char
334             oct_list.append(oct)
335             if oct_list[0] in ['10']:
336                 return False
337             if oct_list[0] in ['172', '192']:
338                 ip_str = ''
339                 for numb in oct_list:
340                     ip_str += numb.zfill(3)
341                 if ('172016000000' <= ip_str <= '172031255255') | \
342                         ('192168000000' <= ip_str <= '192168255255'):
343                     return False
344                 else:
345                     return True
```

```
346             else:
347                 return True
348 
349         # Returns a regular expression to check the syntax of 'Date:' field.
350         # @return regular expression
351         def get_regex_date():
352             return r'^(Date:)?(\s?\w{3},)?\s?' \
353                    r'(?P<day_num>\d{1,2})\s' \
354                    r'(?P<month>\w{3})\s' \
355                    r'(?P<year>\d{4})\s' \
356                    r'(?P<hour>\d{2}):(?P<min>\d{2})(:(?P<sec>\d{0,2}))?\s' \
357                    r'((?P<zone1>\D\d{2})\d{2}|(?P<zone2>\w{2,3}))' \
358                    r'(\s\(.*\))?$'
359 
360         # Returns a regular expression to check the syntax of 'Received:' fields.
361         # @return regular expression
362         def get_regex_received():
363             return r'^Received:\sfrom\s(?P<s_name>[\w\d\S]+)\s' \
364                    r'\((?P<s_helo>[\w\d]*)\s?' \
365                    r'\[(?P<s_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\])(\s\(.*\))
                           ?\s' \
366                    r'by\s(?P<r_name>[\w\d\S]+)\s\((?P<r_helo>[\w\d]*)\s?' \
367                    r'\[(?P<r_ip>\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})\])' \
368                    r'(\svia.*|\swith.*|\sid.*|\sfor.*|\(.*\));' \
369                    r'(?P<date>\s?(\w{3},)?\s?(?P<day_num>\d{1,2})\s' \
370                    r'(?P<month>\w{3})\s(?P<year>\d{4})\s' \
371                    r'(?P<hour>\d{2}):(?P<min>\d{2})(:(?P<sec>\d{0,2}))?\s' \
372                    r'((?P<zone1>\D\d{2})\d{2}|(?P<zone2>\w{2,3}))(\s\(.*\))?)$'
373 
374         # Normalizes the transferred timezone to UTC (+0000).
375         # @param string, string
376         # @return int
377         def get_utc(zone_one, zone_two):
378             if zone_one is not None:
379                 return int(zone_one)
380             if zone_two is not None:
381                 # spezification rfc 5322 p.32
382                 if zone_two == 'UT' or 'GMT':
383                     return 0
384                 if zone_two == 'EDT':
385                     return -4
386                 if zone_two == 'EST' or 'CDT)':
387                     return -5
388                 if zone_two == 'CST' or 'MDT)':
389                     return -6
390                 if zone_two == 'MST' or 'PDT':
391                     return -7
```

```
392                 if zone_two == 'PST':
393                     return -8
394 
395         # The transferred object is grouped by a regular expression to create a
396         # datetime object with day, month and year. If bool_time is True it
397         # still contains hours, minutes and seconds.
398         # @param object with timestamp, boolean
399         # @return datetime object
400         def get_datetime_obj(obj, bool_time):
401             switcher = {
402                 'Jan': 1,
403                 'Feb': 2,
404                 'Mar': 3,
405                 'Apr': 4,
406                 'May': 5,
407                 'Jun': 6,
408                 'Jul': 7,
409                 'Aug': 8,
410                 'Sep': 9,
411                 'Oct': 10,
412                 'Nov': 11,
413                 'Dec': 12
414             }
415 
416             hour = 0
417             minute = 0
418             sec = 0
419 
420             match = re.match(get_regex_date(), ''.join(obj))
421             year = int(match.group('year'))
422             month = switcher.get(match.group('month'), 'Invalid month')
423             day = int(match.group('day_num'))
424             if bool_time:
425                 hour = int(match.group('hour'))
426                 minute = int(match.group('min'))
427                 sec = int(match.group('sec'))
428             return datetime.datetime(year, month, day, hour, minute, sec)
429 
430         # Splits header and body of the email and insert a spam mark.
431         # @param string, string
432         # @return  string
433         def advance_header(mail_data, result):
434             msg = mail_data.split('\n\n')
435             if msg.__len__() == 1:
436                 msg = mail_data.split('\r\n\r\n')
437             if re.match(r'\[- SPAM -\]', result):
438                 insert = 'YES\n\n'
```

```
439             else:
440                 insert = 'NO\n\n'
441             return msg[0] + '\nX-hbE_Spam: ' + insert + msg[1]
442 
443         # Creates an identifier for the log file consisting of the recipient and
                the
444         # timestamp of the delivering MTA.
445         # @param list, list -> both not None
446         # @return string
447         def create_identifier(field_to, group_received):
448             date = group_received[group_received.__len__()-1][8]
449             return 'Delivered: ' + date + ' ' + ''.join(field_to)
450 
451         # Opens the logfile and creates a entry with timestamp, failure, details,
452         # result message and an identifier.
453         # @param string
454         def open_write_file(message):
455             timestamp = time.strftime('%Y/%m/%d [%H:%M:%S]: ')
456             try:
457                 file = open(PATH_LOG, 'a')
458                 file.write(timestamp + message + '\n')
459                 file.close()
460             except IOError:
461                 print('--> Create Logfile ... FAILED ... check permission!')
462 
463         # Sends the email back to the MTA.
464         def send_message(sender, receiver, data):
465             smtp_server = smtplib.SMTP(HOST, REPLY_PORT)
466             try:
467                 smtp_server.sendmail(sender, receiver, data)
468             finally:
469                 smtp_server.quit()
470 
471         # ************************************************************
472         # **********************    MAIN    ************************
473         # ************************************************************
474 
475         def main(d):
476             global failure_msg
477             temp_data = d
478             failure_msg = '*                                    '
479             detail_msg = '*                                    '
480             result_msg = '[-  OK  -] '
481             identifier = ''
482 
483             data_list = separate_header(temp_data)
484             if data_list.__len__() > 0:
```

```
485                 field_from, field_to, field_reply, field_received, group_received
                        , field_date, \
486                     short_from, short_reply = prepare_fields(data_list[0])
487
488                 if check_specification(field_date, group_received):
489                     if check_crit_seven(field_date, group_received):
490                         if check_crit_one(short_from, short_reply):
491                             if check_crit_three(group_received):
492                                 if check_crit_thirteen(short_from, field_date):
493                                     True
494                                 else:
495                                     detail_msg = 'Domain was created today     '
496                                     result_msg = '[- SPAM -] '
497                             else:
498                                 detail_msg = "Manipulated 'Received'       "
499                                 result_msg = '[- SPAM -] '
500                         else:
501                             detail_msg = "Manipulated 'From'/'Reply'   "
502                             result_msg = '[- SPAM -] '
503                     else:
504                         detail_msg = 'Timing is implausible        '
505                         result_msg = '[- SPAM -] '
506                 else:
507                     detail_msg = 'Specifications not fulfilled'
508                     result_msg = '[- SPAM -] '
509
510                 identifier = create_identifier(field_to, group_received)
511             else:
512                 detail_msg = 'E-Mail unreadable            '
513
514             logfile_msg = result_msg + '[' + failure_msg + '][' + detail_msg + ']
                    - ' + identifier
515             open_write_file(logfile_msg)
516             temp_data = advance_header(temp_data, result_msg)
517             send_message(mailfrom, rcpttos, temp_data)
518
519         main(data)
520         return
521
522
523 server = Filter((HOST, LISTEN_PORT), None)
524 asyncore.loop()
```

Listing D1: Entwickeltes Filterungsverfahren mit Python

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 18.12.2018 Robert Lehnert