BRNO UNIVERSITY OF TECHNOLOGY

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

FAKULTA ELEKTROTECHNIKY
A KOMUNIKAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF TELECOMMUNICATIONS

ÚSTAV TELEKOMUNIKACÍ

SPAM DETECTION METHODS

METODY DETEKCE NEVYŽÁDANÉ ELEKTRONICKÉ POŠTY

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR
AUTOR PRÁCE

Michal Rickwood

SUPERVISOR
VEDOUCÍ PRÁCE

Ing. Václav Oujezský, Ph.D.

BRNO 2022

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

Bachelor's Thesis

Bachelor's study program **Information Security**

Department of Telecommunications

Student: Michal Rickwood ***ID:*** 221568

Year of study: 3 ***Academic year:*** 2021/22

TITLE OF THESIS:

Spam detection methods

INSTRUCTION:

The bachelor thesis objective is to propose a method for outgoing spam detection without the possibility of directly analyzing the content of electronic mail messages (e-mail). This problem concerns ISPs (Internet Service Providers) penalized for such unsolicited traffic. The purpose of spam detection is to identify the source of such traffic by using MX records in the DNS (Domain Name System) in combination with analysis of unusual SMTP (Simple Mail Transfer Protocol) traffic from information messages. In the theoretical part of the bachelor thesis, describe the possibilities of spam detection from information messages captured in NetFlow, Argus, or BiFlow format. In the practical part of the bachelor thesis, propose a method for such detection and implement it in the chosen programming language. The output of the thesis is a functional method for detecting spam messages from a selected sample of data and an evaluation of the chosen approach. The preferred programming language is Python.

RECOMMENDED LITERATURE:

[1] Introduction to Cisco IOS NetFlow - A Technical Overview. (2012). CISCO. Retrieved January 22, 2016, from http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[2] BIDGOLI, Hossein. Handbook of Information Security. 2. New Jersey: John Wiley & Sons, 2006. ISBN 9780471648314.

Date of project specification: 7.2.2022 ***Deadline for submission:*** 31.5.2022

Supervisor: Ing. Václav Oujezský, Ph.D.

doc. Ing. Jan Hajný, Ph.D.
Chair of study program board

ABSTRACT

The main goal of this thesis is to build a spam detection algorithm that uses solely traffic flow logs in the form of Netflow messages. Internet service providers must detect spam in order for their entire subnets not to be marked as spamming stations. The algorithm was drafted based on an analysis of various datasets containing Netflow records. These datasets consist of valid e-mails, spam and common non e-mail related traffic. The algorithm uses domain name system blacklist verification as the first step of identifying a spamming station. All flagged communications are dropped immediately. Only if a station is not marked are filtering criteria subsequently applied. These criteria have been divided into acceptance and ordering criteria. An acceptance criterion has been drafted to select potentially significant stations. Five ordering criteria have been formulated to sort these selected IP addresses by the probability of them being spamming stations. Behind each criterion is a mathematical equation that returns a value between 0 and 1. The total sum of such returned values are close to 5 with spamming stations, while legitimate stations have noticeably lower values. The output of the developed algorithm is a list of potential spamming stations sorted probability of them being spamming stations.

KEYWORDS

Netflow traffic, privacy protection, unsolicited electronic mail

ROZŠÍŘENÝ ABSTRAKT

Tato práce se zabývá tématem, se kterým se setkává v podstatě každý uživatel internetu, což v dnešním světě tvoří přibližně 50 % populace. Tímto tématem je spam. Spam je zasílání nevyžádané pošty na velké množství příjemců. Přestože poměr takových e-mailů vůči legitimní komunikaci vykazuje v posledních letech klesající trend, tvoří stále spam přibližně 45 % e-mailového provozu. S neustálou expanzí internetu také samozřejmě stále stoupá absolutní počet zaslaných spammových e-mailů.

Existují mnohé velmi propracovaně vyvinuté a naimplementované systémy, které spam úspěšně filtrují. Drtivá většina takových systému však vyžaduje zásah do obsahu komunikace. Tento přístup je relativně jednoduchý a velice účinný, mnohé systémy využívají umělé inteligence, které jsou schopné se přizpůsobovat změnám spammerů. Komplikace však přichází s ohledem na zmíněný zásah do soukromí a spojitostí *General Data Protection Regulation* (GDPR). Toto se týká například poskytovatelů internetových služeb, kteří poskytují e-mailové služby a musí odchozí provoz filtrovat. Pokud nezastihne poskytovatel takovou komunikaci včas a daný uživatel, spammer, z jeho sítě je označen například projektem UCEPROTECT, dojde k zablokování veškeré další komunikace z dané sítě. To samozřejmě ovlivňuje i legitimní uživatele, kteří nemohou využívat služeb. Poskytovatelé poté nesou právní a ve většině případů i finanční následky.

Cílem této práce je vytvoření algoritmu na detekci spamu ze záznamu běžného provozu sítě bez nutnosti zásahu do soukromí čtením obsahu zpráv. Algoritmus byl navržen na základě analýzy záznamů, které byly posbírány na *České vysoké učení technické* (ČVUT). Tyto datové sady obsahují validní e-maily, spam, ale i jiný běžný provoz. Podkladová data jsou ve formátu Netflow a obsahují například informaci o zdrojové adrese, cílové adrese, časové razítko a jiné. Vývoj algoritmu byl prováděn ve webovém prostředí Kaggle. Jedná se o sociální síť pro developery, která nabízí využívání vzdálených prostředků zdarma. Mimo to, je možné se účastnit soutěží, ukládat datasety a jiné. Implementace algoritmu byla provedena pomocí programovacího jazyku Python.

Algoritmus v prvním kroku využívá domain name system blacklistů k ověření, zda není daná IP adresa již označena jako spamovací. Následují samotná kritéria, dle kterých dochází k filtrování provozu. Kritéria jsou dělena na akceptační a seřaďovací. Akceptační kritérium tvoří poměr příchozích a odchozích *Simple mail tranfer protocol* (SMTP) spojení. Prvotní podmínka zařazení byla, aby tato hodnota byla nižší než 0.005. Během testování bylo nutné podmínku upravit na hodnotu 0.003. Všechny přijaté stanice jsou poté seřazeny pomocí 5 kritérií. Za každým kritériem stojí matematická funkce přiřazující hodnotu mezi 0 a 1. Stanice jsou poté seřazeny

dle výsledného součtu. Spamující klienti budou mít součet blížící se k 5, zatímco legitimní stanice budou mít výrazně nižší celkový součet.
Filtrační kritéria jsou definovaná následovně:

Příchozí spojení Dle akceptačního kritéria již víme, že stanice má mnohonásobně více odchozích než příchozích spojení. Vzhledem k povaze spamovacích stanic je velmi pravděpodobné, že mnohé stanice nebudou mít příchozí spojení žádná. Z tohoto důvodu je první kritérium stanoveno následovně:

$$a = \begin{cases} 1 & \text{počet příchozích SMTP spojení} = 0 \\ 0 & \text{Ostatní případy} \end{cases} \tag{1}$$

Počet různých cílových serverů Druhé kritérium vychází také z povahy spammera, tentokráte z předpokladu, že stanice bude vysílat emaily na vícero serverů. Zasílat veškerou komunikaci na pár nebo dokonce na pouze na jednu by bylo vysoce neefektivní. Druhé kritérium:

$$b = \begin{cases} 1 & \text{Počet různých cílových serverů } > 10 \\ 0 & \text{Ostatní případy} \end{cases} \tag{2}$$

Doba bez aktivity Vysoké procento času bez aktivity a naopak mnohá spojení v rozmezí malého časového rozpětí je znak podezřelé *Internet protocol* (IP) adresy. Velice často dochází k zasílání veškeré komunikace v podstatě v jeden moment. Stanice je považována za aktivní v momentě, kdy se účastní komunikace. Nejmenší perioda aktivity je 5 minut. Zbytek času je stanice považována za neaktivní.

$$\text{c} = \text{procento času bez aktivity} \tag{3}$$

Směrodatná odchylka Cílem tohoto kritéria je vybrat stanice s velkou směrodatnou odchylkou odchozích SMTP spojení v čase. Odchylka by mohla ukázat nárazové zasílání zpráv a bude spojena s vyšším procentem času bez aktivity.

$$d = \begin{cases} 1 & \sigma > 10 \\ 0 & \text{Ostatní případy} \end{cases} \tag{4}$$

,kde σ je směrodatná odchylka.

Nárazová aktivita Páté a poslední kritérium rozšiřuje směrodatnou odchylku a zesiluje detekci nárazové aktivity zasílání emailů. K nalezení takové aktivity jsou sledovány body, které jsou vysoce nad standardem. Tím je v této práci stanoven

součet čtyřnásobek směrodatné odchylky a aritmetického průměru. Stanice, které takovýchto bodů mají více než 10 splňují toto kritérium a jsou tedy považovány za podezřelé.

$$e = \begin{cases} 1 & P(datapoints > (4\sigma + \mu)) > 10 \\ 0 & \text{Ostatní případy} \end{cases} \tag{5}$$

,kde σ je směrodatná odchylka a μ je aritmetický průměr.

Z časových důvodů nebyla provedena analýza kompletních datasetů, ale pouze prvních 10 000 záznamů. Záznamy obsahují 1 703 unikátních IP adres. Tyto adresy byly v první části zaslány dotazem na *Domain name system* (DNS) Blacklist. 27,66 % z těchto adres bylo uvedeno alespoň na jednom seznam. V případě pozitivní odpovědi, byla pak IP adresa uvedena průměrně na 1,9378 seznamech. Následovaly matematické vzorce kritérií. Akceptačním kritériem prošlo 29 IP adres, tedy přibližně 1,7 %. Adresám, které prvotní podmínku splnily, byl poté vypočítán celkový součet ohodnocení kritérií. V průměru vyšel u těchto IP adres celkový součet 3,01.

Hlavní cíl této práce, kterým bylo v praktické části návrh a implementace metody filtrace spamu, byl splněn. Kritéria, která jsou v algoritmu použita mohou být dále rozvinuta a optimalizována v budoucích analýzách. Vhodným pokračováním by byla úprava konstant u vzorců a sledování vlivu na detekci. Kritériem B prošlo 28 z 29 IP adres. Zde by bylo vhodné kritérium zpřísnit zvýšením počtu různých cílových adres a snížit tím počet stanic, které ho splňují. Naopak kritérium A splnily pouze 4 stanice – vícero stanic mělo přibližně 2 příchozí spojení. Příhodnou modifikací by mohlo být zvolnění počtu příchozích spojení z 0 na zmíněné 1 až 3 spojení.

RICKWOOD, Michal. *Spam detection methods*. Brno: Brno University of Technology, Faculty of Electrical Engineering and Communication, Department of Telecommunications, 2022, 58 p. Bachelor's Thesis. Advised by Ing. Václav Oujezský, Ph.D.

Author's Declaration

Author:	Michal Rickwood
Author's ID:	007
Paper type:	Bachelor's Thesis
Academic year:	2021/22
Topic:	Spam detection methods

I declare that I have written this paper independently, under the guidance of the advisor and using exclusively the technical references and other sources of information cited in the paper and listed in the comprehensive bibliography at the end of the paper.

As the author, I furthermore declare that, with respect to the creation of this paper, I have not infringed any copyright or violated anyone's personal and/or ownership rights. In this context, I am fully aware of the consequences of breaking Regulation § 11 of the Copyright Act No. 121/2000 Coll. of the Czech Republic, as amended, and of any breach of rights related to intellectual property or introduced within amendments to relevant Acts such as the Intellectual Property Act or the Criminal Code, Act No. 40/2009 Coll. of the Czech Republic, Section 2, Head VI, Part 4.

Brno ....

author's signature*

*The author signs only in the printed version.

ACKNOWLEDGEMENT

I would like to thank the advisor of my thesis, Ing. Václav Oujezský, Ph.D. for his/her valuable comments etc.

Contents

List of Figures

List of Tables

Listings

1 Introduction

This work is concerned with a topic virtually every e-mail user knows, which in today's society means roughly 50 % of the world's population: spam. Spam is a method of sending unsolicited e-mail messages to a large number of users. Even though the proportion of spam compared to legitimate communications is showing a downwards trend, it still accounts for approximately 45 % of traffic [1].

There have been many systems developed and implemented to fight spam. Most of these systems however consist of going through the contents of the sent message. This approach is relatively simple yet, as systems using *Artificial intelligence* (AI) nowadays are able to learn and recognise spam almost without mistake, highly effective. This is, however, complicated with regards to legality and invasion of privacy of the users. This concerns mostly the *Internet service provider* (ISP) that provides the e-mail servers. When the user uses the services of the provider with the intent of sending unsolicited messages, the user is flagged as a spammer in UCEPROTECT, which can affect all the other users in the subnet and the provider usually bears all the legal and financial accountability. For this reason, the ISP has to monitor the outgoing traffic, in a way other than going through the contents of the messages, ideally to discover spam stations before anyone else does.

One of the most commonly used methods of detection is by the use of NetFlow messages. These records contain only metadata about the messages and users, but not the contents of the communications. The records, which are part of the headers of packets, contain for example the IP address of both the sender and receiver of the message, port numbers and more. This work also addresses traffic monitoring based on Netflow records

Instead of creating our own traffic and collecting the data, previously collected datasets have been used [2]. The primary reason for this was so that we would be able to identify and model the behaviour of spamming devices. The first step of the developed algorithm is to send a query to *Domain name system blacklist* (DNSBL) to determine whether or not the sender IP is listed. The following steps consist of various mathematical comparisons to previously created models of spamming users and legitimate users and defining criteria based on which the algorithm will filter ongoing traffic.

2 Netflow, Spam and SMTP

The first part of this chapter is devoted to spam, its history and variations. Netflow protocol is described in the second part, along with its connection to the e-mail protocol SMTP

2.1 Spam and its etymology

The etymology of the name „spam“ is not quite so straightforward and simple as it might seem at first glance. The original meaning of the word spam was a conserved meat product from a company named Hormel [3], which was revolting but very popular during the second world war. What each letter in SPAM stands for is known only to a few people from Hormel. Popular theory points to „Spiced ham“ or „Special processed American meat“. The meaning used and known and use today comes from a 1970's BBC comedy sketch show from Monty Python's Flying Circus.

Spam was, is and most likely will continue to be used for many different purposes. The most common reasons are unsolicited offers, marketing and similar. One of the most well-known types of spam is Scam419, commonly known as Nigerian Letters. This kind of spam is primarily used for tricking victims into sending funds to the scammer, similar to today's „phishing“. Scam419 was very popular, especially in the 90's, however, it is still used in many forms today.

2.2 NetFlow

Netflow is a protocol which was developed by the company Cisco and is used by network administrators as a provider of information on the flow of data on their networks. Exported data can be used in multiple ways, for example, network administration, billing ISP or as a defence against *Distributed denial of service* (DDOS) attacks and others.

Historically, the first version of the NetFlow protocol was developed in 1994 [4]. This version is no longer used today and was reduced to IPv4 without an IP mask and *Autonomous system* (AS) numbers. The next 3 versions (v2-v4) were never published to the general public and were used only by Cisco internally. The following version that was released and implemented was v5. This version is still widely popular, especially for router manufacturers. At the same time, it is still limited to IPv4. Versions 6-8 did not bring any significant changes compared to the updates that were brought in v5 and are no longer used. The latest version is v9, which is used by some manufacturers primarily to monitor IPv6 flows. A big advantage is the fact that this version is based on the use of presets, which allows space for future

improvements of services without any significant changes of the format. Another build of NetFlow is IPFIX, IPFIX is IETF standard based on NetFlow v9. See table 2.1.

Tab. 2.1: NetFlow verze 9 packet header format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Version																Count															
System Uptime																															
UNIX Seconds																															
Package Sequence																															
Source ID																															

Monitoring the network can be done by watching the data at the level of packets with the use of tools such as TCPDUMP [5]. Due to time requirements for data processing together with computing memory requirements, this solution is not considered practical. A popular protocol for monitoring is called *Simple Network Management Protocol* (SNMP). However, SNMP has very low granularity, resulting in large data loss. The Netflow protocol lays somewhere in between the two poles resulting in data with enough details to allow easier and quicker processing while maintaining enough information for our purposes [6]. Cisco [7] defines data flow as a one-way flow of a packet between a set source and destination. Specifically, flow is defined by the following seven key poles:

- Source IP address
- Destination IP address
- Source port number
- Destination port number
- Layer 3 protocol type
- *Type of Service* (ToS) byte
- Input logical interface

2.3 SMTP

This section describes the e-mail protocol SMTP and its connection to unsolicited electronic mail. SMTP, *Post office protocol version 3* (POP3) and *Internet message access protocol* (IMAP) are all used for the use of e-mail communications. All 3 of the above-mentioned protocols are application-layer Internet standard protocols in the TCP/IP suite and *Open Systems Interconnection* (OSI) model [8]. However, Protocols POP3 and IMAP are used for receiving e-mails whereas protocol SMTP is used for sending e-mails. Because sending spam requires the same apparatuses

as sending traditional e-mails SMTP is considered to be a lot more relevant to this work [6].

SMTP protocol is an application layer protocol from the TCP/IP protocol suit and is used as a common mechanism for the transfer of e-mail communication [9]. When sending an e-mail, the client's process establishes a TCP connection with the server-side and attempts to send the e-mail. The SMTP server listens to port 25 and the client initialises the connection on this port. In the case of a successful connection, the processes start a simple „request-response" dialogue, defined by the SMTP protocol, in which the client's process sends e-mail addresses to which the e-mail should be sent to. Supposing that the server process accepts these addresses, the client process sends an instant e-mail message. This message must contain a header and text formatted by RFC 822 [10].

The SMTP protocol predominantly uses port 25 between two servers [11]. This port is also the default for sending e-mail messages. For the purpose of security, the *Internet Assigned Numbers Authority* (IANA) started to develop a version of SMTP with the use of TLS and SSL protocols. Due to insufficient *Request for Comments* (RFC), it was never approved by the *Internet Engineering Task Force* (IETF) and its use is not recommended in its current state. Nowadays, port 587 is the best solution for the client-side, mostly given the native support of TLS. In the event that other ports are blocked, it is possible to use port 2525, which also supports TLS, but never gained official status from neither IANA nor IETF.

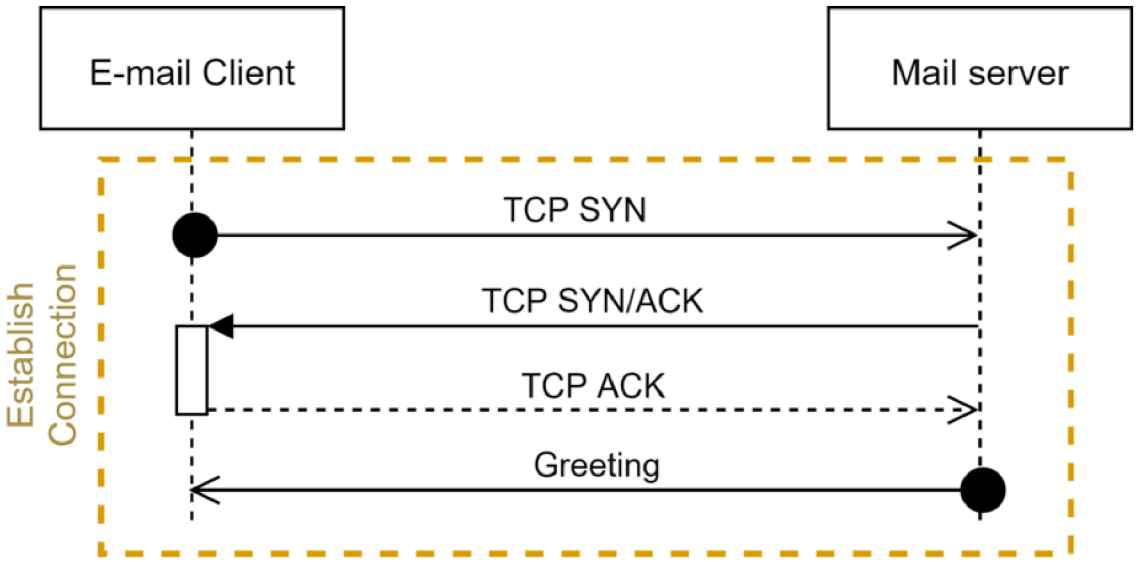

Fig. 2.1: Establishing SMTP communication [12]

2.3.1 Email communication

Electronic mail is sent through the help of multiple dialogues of the request-response type between the client and server. An SMTP transaction is composed of two parts – the header and the body. The headers are composed of field/value pairs, whereas the body, which does not have to be structured, follows *Multipurpose Internet Mail Extensions* (MIME). The header is composed of the source address (defining where errors should be sent), mode of delivery and one or more addresses of the receivers [13].

SMTP session

```
After a TCP connection setup between the sender (Snd) and the receiver (Rcv):
1.  Rcv: 220 mailserver.MrX.com
2.  Snd: HELO springfield.net
3.  Rcv: 250 Hello springfield.net, pleased to meet
4.  Snd: MAIL FROM: <bart@springfield.net>
5.  Rcv: 250 bart@springfield.net ... Sender ok
6.  Snd: RCPT TO: homer@MrX.com
7.  Rcv: 250 homer@MrX.com ... Recipient ok
8.  Snd: DATA
9.  Rcv: 354 Enter mail, end with "." on a line by itself
10. Snd: From: bart@springfield.net
11. Snd: To: homer@MrX.com
12. Snd: Are we there yet?
13. Snd: .
14. Rcv: 250 Message accepted for delivery
15. Snd: QUIT
16. Rcv: 221 MrX.com closing TCP connection
```

Fig. 2.2: Console output example

Figure 2.2 above is an example of an SMTP session, sending an e-mail to one destination address. It is, naturally possible to use more destination addresses for the sending of a single message. Lines 1-6 from the sender side are the header. The following lines 9-15 are the message itself. It is easily visible that the address of the receiver is seen in both parts of the e-mail.

3 Current state of spam detection

This chapter describes currently implemented methods of spam detection. In the first part, so-called blacklists and their use by ISPs are explained, the next part is dedicated to the UCEPROTECT project[14] and the last part is about current spam solutions that do not require interference with the contents of messages.

3.1 Blacklist

DNSBL are lists of blocked addresses, which have been flagged as spamming stations in the past [15]. These lists make it possible for ISPs to verify IP addresses with relative ease. One of the downsides is the possible false flagging of addresses during the classification of the whole subnet. As soon as the subnet in question is added to the list, ISPs which check the blacklist will start to block all incoming communication from all the IP addresses on the subnet, even the legitimate ones that are not/were not sending spam. The principle of a blacklist is based on 3 basic components:

- Domain name under which the list is hosted
- The server
- A list of blocked addresses

RBL

The first blacklist was *Real-time blacklist* (RBL), created in 1997. The primary purpose was to block spam, but also to improve the knowledge of ISPs on the topic of spam. The vast majority of blacklists contemporarily exist only as a list of blocked addresses.

DNS query

The server of the provider of e-mail services, which uses some DNSBL (for example dnsbl.info), sends the query in the following steps:

1. The bit order of the IP address is reversed. For instance, when the questioned IP address is 100.64.209.32, the result will be 32.209.64.100
2. The domain name of the server is added to the IP address, for example, 32.209.64.100.dnsbl.info
3. The query is sent and subsequently compared to the list. If the IP address is flagged as a spamming station by the blacklist, the response contains one or more records, by whom the IP queried IP address has been. In the opposite case, the result response is "NXDOMAIN" ('No such domain').

3.2 What is UCEPROTECT?

UCEPROTECT is operated by a subject presenting under the name (nickname) Claus von Wolfhausen. It is well known that he takes a rather aggressive attitude and has a tendency to indiscriminately add whole subnetworks and address blocks to the list of blocked addresses if even only one of the IP addresses is suspected as a source of spam. Monetary compensation is required to be removed from the list.

If you wish to have your address (addresses) added to or removed from a white list, you will be redirected to a web page http://www.whitelisted.org/, which is registered and hosted in Germany.

Officially, the presented objective of this project is to stop global misuse of e-mail. The project penalises stations where it detects spam. The project however creates a bigger problem for internet service providers. The providers have to watch all outgoing traffic from their servers and detect spam faster than it will be marked by UCEPROTECT. In the case that the spam is caught by UCEPROTECT, very often the whole subnet where the spamming station is placed will get flagged. The ISP then has to prevent further spam and then wait a time period to be automatically removed from the list, or pay a penalty for faster removal. Aside from that, the ISP will likely be required to pay for damages to their clients who were using their services correctly and, due to the whole subnet being flagged, their services are limited. While the project appears very similar to blacklists and states that services provided by blacklists might be better in some respects on their own website, they also mention the following issues with classical blacklists:

- Reaction time is usually hours, sometimes even days, while the time to flag a spammer at UCEPROTECT is considerably lower.
- IP address owners who do not show interest in and do not invest time and finances into securing their systems are often simply removed from blacklists and the same situations repeat.
- Providers are not forced to implement preventative measures against abuse of their services.
- Most RBLs do not have clearly stated policies – they do not know who exactly they are blocking.
- Most RBLs run on a single host. In the case of an attack or an outage, they stop working completely.

3.2.1 Level policy

The rules for removal from the list after being flagged depend on the tier and number of offences. UCEPROTECT divides to levels 1, 2 and 3.

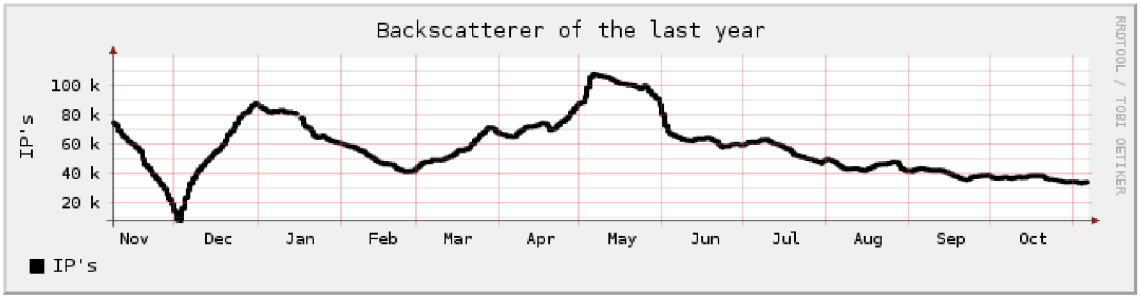

Fig. 3.1: Retrospective IP count on 7. 11. 2021 [14]

Level 1

To be removed from this category, the only thing that needs to be done is for UCEPROTECT to stop detecting spam from the IP address in question. After that, the station is automatically removed after 7 days. A second option offers immediate removal from the list, a payment of 89 CHF (about 2,100 CZK) is required.

Level 2

Level 2 is designated for network ranges from which spam is sent repeatedly. The collision numbers required per 7 days is determined by the size of the subnet as per table. 3.1

Tab. 3.1: Impact count to subnet size [14]

Subnet size	Impact count
/27	1
/26	2
/25	3
/24	4
Netmask - 1	(netmask value + 1) + (netmask value + 3)

For the subnet and all its IP addresses to be removed from level 2, 4 steps, as specified by UCEPROTECT, must to be taken. If these steps are not implemented, the alternative is to pay a penalty of 249 CHF (about 6,000 CZK). However, the administrator will be warned about the possibility of being listed again if the offences are repeated.

Level 3

AS with a SPAMSCORE of 50 or higher are added to this tier. At the same time, they must have at least 50 collisions for 7 consecutive days. This precaution is

so that mini providers are not affected due to 1 or 2 spammers. SPAMSCORE is calculated based on the following eq. 3.1:

$$\text{SPAMSCORE} = \frac{\text{level 1 impact count}}{\text{total IPs in this ASN}} \times 100,000 \tag{3.1}$$

According to UCEPROTECT, there are approximately 105,000 e-mail service providers. The providers listed in level 3 are responsible for 50-75 % of global spam.

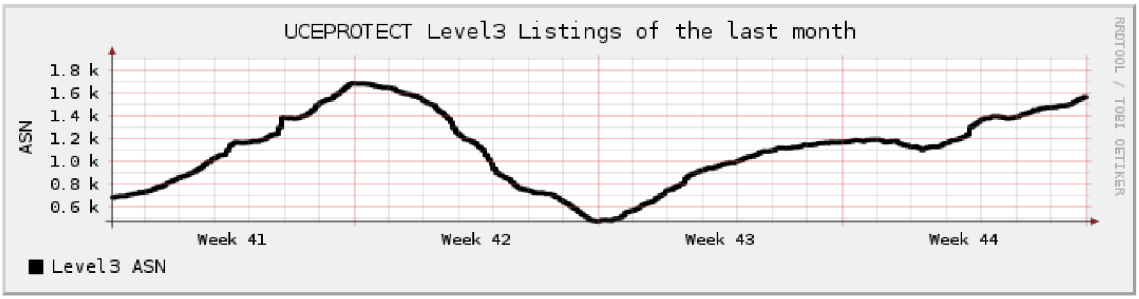

Fig. 3.2: UCEPROTECT Level3 listing of the last month on 8. 11. 2021, UCEPROTECT

For an ASN to be removed from level 3, once again the 4 steps defined by UCEPROTECT must be implemented by the provider. The option to pay a penalty to be removed exists again, in this case, 443 CHF (approximately 10,700 CZK).

3.3 Currently used methods

Many of the algorithms for the detection of unsolicited mail these days still function based on the message contents, even if only the subject of the message and not the text itself. Among the providers who interact with the message contents are to be found even leading companies such as Google, Yahoo and Outlook [16]. These providers use various methods of machine learning such as neuron networks, k-nearest neighbours algorithm and others. Methods then learn on wide sets of both spam and legitimate messages. In addition to using already established rules, these algorithms can create their own new rules based on what they learn. The algorithm for spam detection by Google has advanced to the stage where it is capable of detecting up to 99.9 % of unsolicited mail. Before flagging spam, common machine learning algorithms compare every message with valid and spam data. There is also a proposal for systems which use an artificial immune system as an inspiration and use special functions to generate detectors to cover the space for spam [17]. Providers use methods that have the highest percentage of success but the problem still lies in the interference with the message contents. Nevertheless, there are many algorithms and proposals that detect spam without the need for reading the message

contents. These methods collect and monitor data about the traffic on a network, then aggregate the data based on the source addresses [6]. An example of this can be seen in the following table, where the observer watches the numbers of incoming and outgoing connections on port 25 and the numbers of different addresses in bot.

IP	dist out	out	dist in	in
1	1,980	334,356	36,354	675,381
2	3,227	247,588	36,354	17,645
3	11,459	11,459	36,354	745,408
4	**39,460**	**244,117**	**0**	**0**
5	11,280	240,733	153,275	675,632
6	3,512	238,665	788	27,738
7	7,943	195,573	132,616	539,297
8	**2**	**184,698**	**0**	**0**
9	2,252	136,847	10	187
10	**24,213**	**116,898**	**1**	**2**
11	7,774	115,746	8	24,972
12	8,376	68,413	24	172,464
13	**175,32**	**64,685**	**0**	**0**
14	341	57,251	66,237	901,280
15	443	54,212	10	578

Tab. 3.2: Data aggregation per IP address [6]

In Table 3.2, it can be seen that most of the IP addresses which have a high number of outgoing connections also have a roughly similar number of incoming connections. In this case, Vliek assumes that those are active but valid stations. On the other hand, the highlighted stations have a large number of outgoing connections, but very few or even no incoming. By Vliek's assumptions, these stations are credible candidates for spam clients. However, at the same time, he also notes that there are valid stations that are used only for outgoing messages. In this category, there are, for example, accounts used and operated by banks for sending account information and online news, which often send e-mails many times a day. Most of these examples will have a relatively low number of varied IP addresses to which the e-mails are sent.

4 Practical part

The following half of this work is devoted to the practical solution of spam detection solely from the records of network traffic. First, the entire development environment is described in detail, including the underlying data to which the algorithm has been applied. Next, the model of a spammer is defined, as well as a legitimate SMTP client along with criteria for filtering. In the last part, the algorithm itself which looks for spamming in the underlying data station is described.

4.1 The proposed solution for the SPAM detection

In this chapter, all the issues in developing the algorithm to detect spam traffic based on Netflow records are presented.

4.1.1 Development environment

The entire development was done in Python, specifically version 3. The main advantages of Python include its clarity, numerous support libraries and more. The development of the detection algorithm can be performed in several different environments that have different software and hardware resources at their disposal. The main hardware requirement will be a relatively high computing capacity. In the case of small computing capacity, all calculations would take a very long time and it would not be possible to achieve sufficient results. Included among the variants that are easily available without larger monetary investments are:

University laboratories

The university has computer laboratories with large computing capacities, which can be used to browse datasets. However, due to the current situation around COVID-19, this solution could be problematic and was therefore rejected.

Home PC

At a first glance, this solution is definitely the simplest. The only essential requirement is an *Integrated development environment* (IDE) and installation of Python itself. The problem comes with computing capacity. An ordinary PC is not designed for calculations of this size and therefore this is an impractical solution.

Cloud solution

The most practical solution turned out to be the use of cloud software. One possibility of this solution is from the company Kaggle [18], which offers an environment

Tab. 4.1: Average home PC – Acer Swift SF315-51

Component	Type
Operating system	Windows 10 Home
Processor	Intel(R) Core(TM) i7-8550U CPU@1.80 GHz 1.99 GHz
Installed memory (RAM)	16 GB (15.9 GB usable)
System type	64-bit Operating System, x64-based processor

for developers in which they can write projects without the need for setup, browse other projects, take part in free courses and much more. Programming can be done in languages Python and R. It is possible to access *Graphics processing unit* (GPU)s free of charge. Kaggle also offers free space for saving datasets that the algorithm then uses. Without phone number verification, it is not possible for the algorithm to access the internet, which is necessary in order to use the pip system for package management. At the same time, with internet access, it is possible to set up an automatic startup of the code at specified time intervals.

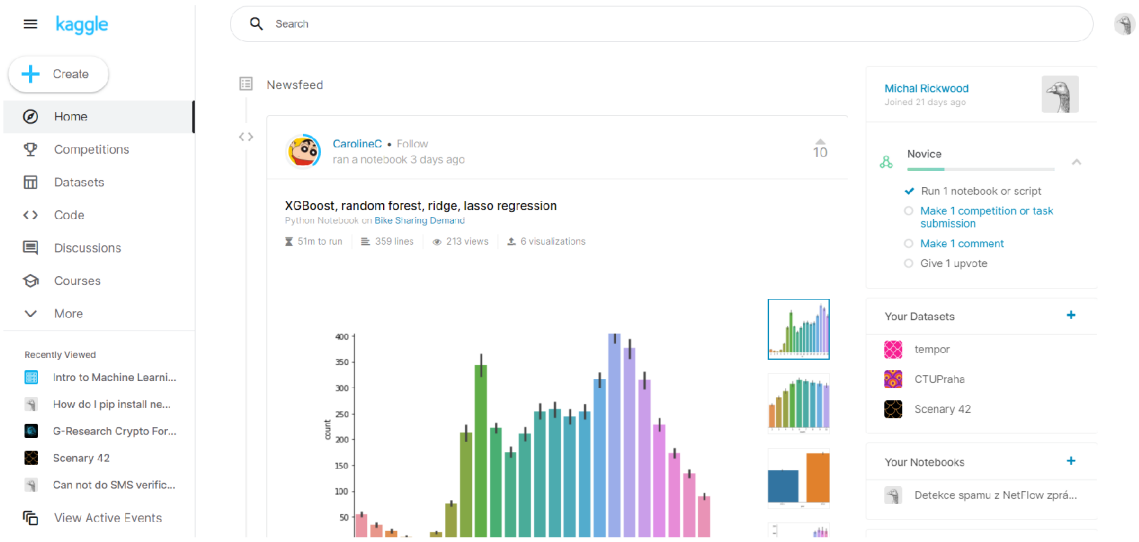

Fig. 4.1: Front page of Kaggle.com

[18]

Kaggle essentially works as a social network for developers. The main page is the newsfeed, where one can find popular analyses, datasets and projects. The service also creates a platform for the organisation of competitions in programming where thousands of teams compete and prizes often reach tens or even hundreds of thousands of US dollars.

4.1.2 Base data

The detection algorithm must be applied to a dataset. This consists of data entries of the same type, in our case packets, which are sent through a monitored network. Datasets have a large number of entries, many of which are not relevant for this work. Some of the monitored parameters are:

- Time of sending
- Protocol type
- Source IP address and port
- Destination IP address and port
- Others

The dataset can be created by recording and saving SMTP traffic. This option would require setting up an environment, creating traffic and recording it. Another option is using already created datasets. Because the details of new sample data in new datasets would be unknown, existing datasets were chosen as the better option. Wide datasets are available, recorded by *Czech technical University* (CTU) in Prague [2]. The complete set consists of 13 scenarios with varying samples. Table 4.2 shows the characteristics of the datasets in question.

ID	IRC	SPAM	CF	PS	DDoS	P2P	HTTP	Note
1	x	x	x					
2	x	x	x					
3	x			x				
4	x				x			UDP and ICMP DDoS
5		x		x			x	Scan web proxies
6				x				Proprietary C&C. RDP
7							x	Chinese hosts
8				x				Proprietary C&C. NetBIOS, STUN
9	x	x	x	x				
10	x				x			UDP DDoS
11	x				x			ICMP DDoS
12						x		Synchronization
13		x		x			x	Captcha. Web mail.

Tab. 4.2: Data aggregation per IP address [6]

Table 4.2 shows that the only scenarios relevant for spam filtering are 1,2,5,9 and 13. All these scenarios were used as underlying data. Scenario 4, which does not contain spam, was used to validate the falsely flagged stations. The other scenarios

were discarded from the set. The original scenarios contained botnet[1], normal traffic and background traffic. However, for privacy protection reasons, complete scenarios are not available.

ID	Duration(hrs)	#Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	166,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,121,077	53GB	Rbot	1
5	11.63	4,481,167	129,883	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,779	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

Tab. 4.3: Amount of data in each scenery.

Table 4.3 lists the duration, number of packets, number of NetFlow and size of the scenarios. The scenario data itself is stored in .pcap files; that is, the file that contains the packets. For the sake of simplicity of implementation and to save memory the datasets were first converted to .csv format. The .pcap format is an abbreviation of Packet Capture [20]. The .csv format is a type of simple text, where the comma character is used to separate values. Each line specifies one record [21]. Files of .pcap type are primarily associated with Wireshark [22], which is used to analyse network traffic. The conversion was done manually using Wireshark, which can read and display packets from the .pcap file and export them to .csv file. Other options are Netresec NetworkMiner [23] and TCPDump [5][24]. Scenario 1 was too large for Wireshark to load the whole set. It was necessary to first use the editcap tool [22] of Wireshark to split the file into multiple parts. Even while loading the other smaller scenarios, the Wireshark application crashed very often. If an algorithm was used, it would be necessary to retrieve the data directly in .csv format or implement an automatic conversion.

[1]Botnet is a network of interconnected devices used to carry out commands of the attacked. Often used for DDOS [19].

Another dataset format that was used for the based data from the records CTUis Argus file type. The dataset contains traffic in both directions.

Scenario	.pcap	.csv
1	5.75 GB	1.3 GB
2	34.5 MB	23.2 MB
5	29.5 MB	6.26 MB
9	1.04 GB	300 MB
13	109 MB	63.4 MB

Tab. 4.4: Size comparison

Table 4.4 shows a comparison of the sizes of different scenarios before and after conversion. Table reftable:dataset shows the first 7 records of scenario number 1. The number of columns was reduced to show only some information about each record. The columns show the start time of the communication, the duration, the source address, the direction of the communication and the destination address. In addition to this information, the original datasets also contain protocol type, source and destination port numbers and more.

StartTime	Dur	SrcAddr	Dir	DstAddr
49:35.7	2,069.973	203.253.8.233	<->	147.32.84.229
49:35.7	895.9893	81.47.154.13	<?>	147.32.84.229
49:35.7	0.00012	147.32.84.229	->	78.42.25.171
49:35.7	3,561.927	147.32.84.229	<->	113.128.219.130
49:35.7	0	147.32.84.229	->	60.50.167.24
49:35.7	3,389.645	147.32.84.229	->	68.193.182.77
49:35.7	3,511.665	147.32.84.229	<?>	155.56.68.217

Tab. 4.5: Record example

4.1.3 Detection algorithm

In this chapter, the design of an algorithm for detecting spam from network traffic is drafted. Data import, processing, output and suggestions for continuation are depicted.

Figure A below shows the flowchart of the proposed detection algorithm. The first step is the conversion from .pcap to .csv format and then loading using the pandas package [25]. The next step is to query DNS blacklists using the pydnsbl package [26]. If the IP addresses are in the list, the provider will drop the message.

In our case, only the data from the Netflow message is added to the aggregation of illegitimate traffic. If the response from the DNS blacklist is OK, the inclusion of the information from the Netflow record is added to the aggregation of legitimate communications. A check is then performed to determine if the IP address from which the communication is sent still falls into the legitimate category station and whether it has crossed one of the established thresholds, which further determines whether it is a spamming station or not. If no threshold is crossed, the process is terminated.

Algorithm code

The algorithm is implemented in Python in the Kaggle web environment. The code used in Listing 4.1 with a description is listed below.

Listing 4.1: List of used packages

```
import csv
!pip install pydnsbl
!pip install nest-asyncio

from ast import Try
import pydnsbl
import csv
import nest_asyncio
import pandas as pd
import numpy
import sys
nest_asyncio.apply()
```

The first part of the code are packages that need to be imported in order to use their methods. The pandas package is used to read .csv format files. Pip call is a package manager for Python [27]. To query DNS Blacklists the package pydnsbl [26], which supports multiple blacklists, is used. Nest asyncio allows the use of nested events in loops.

Listing 4.2: IP address format check

```
def isIPAdress(cell):
    try:
        splited = cell.split('.')
        if len(splited)!=4: return False
        return all((int(ele) >= 0 and int(ele)<256) for ele in splited)
    except Exception:
        return False
```

The `isIPAdress` 4.2 function used ensures that the value that is considered to be the IP address is indeed the IP address of version 4. Records in multiple datasets contain text in a different format compared to the IP address, so it was necessary to implement the check function for this reason. The function returns True if the argument is in the form of four numbers in the range 0-255 separated by a period. Otherwise, if the argument is not in IPv4 format, it returns False.

Listing 4.3: Dataset import and print

```
with open('../input/testing1/Book1.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=',')
    for row in csv_reader:
        if (isIPAdress(row[3])):
            print(ip_checker.check(row[3]))
```

The fourth column of the dataset contains the source IP address field. Code 4.3 retrieves the dataset from the Kaggle online repository at `"../input/testing/capture20110810.binetflow"`, starts iterating through the file line by line and sends a query to the DNS blacklist with an IP address as an argument. The server's responses to the IP addresses that were given in Figure 4.2 can be seen below.

DNSBL query result

```
DNSBLResult: 203.253.8.233 {[}BLACKLISTED{]} (1/50)
DNSBLResult: 81.47.154.13  (0/50)
DNSBLResult: 147.32.84.229  (0/50)
DNSBLResult: 147.32.84.229  (0/50)
DNSBLResult: 147.32.84.229  (0/50)
DNSBLResult: 147.32.84.229  (0/50)
DNSBLResult: 147.32.84.229  (0/50)
```

Fig. 4.2: Console output example

The blacklist's response to the first entry is positive, i.e. it contains the keyword 'BLACKLISTED', which indicates that one of the queried lists contains an IP address. Following, the count of such lists in which the IP address has been found is printed (in parentheses). In the other cases, it is visible that there were no findings in any of the lists.

As the program iterates through the dataset, several statistics are simultaneously being tracked, e.g., how many unique IP addresses out of the total are listed on at least one of the blacklists, the average number of lists on which an address appears if it is found on at least one, and others.

Code 4.4 represents one iteration in a dataset loop. The statistics are gathered into a collection of Directionary type. This type is used to store data in key:value pairs [28]. In this case, the key is a string determined by the IP address and its value is a list of the collected statistics.

Listing 4.4: Dataset iteration

```
row_ipadress = row[3]
        if isIPAdress(row_ipadress):
            if row_ipadress in statistics:
                statistics[row_ipadress][0]+=1
                statistics[row_ipadress][1].add(row[6])
            else:
                #statistics[row_ipadress] = [len(ip_checker.check(row_ipadress).
    detected_by),1,{row[6]},0,set()]
                statistics[row_ipadress] = [1,{row[6]},0,set()]
            add_dest(row_ipadress,row[6])
```

Querying DNS can be a relatively slow process if the internet connection is slow. For this reason, the queries weren't actually sent, only statistics were collected, namely the number of outgoing connections, the number of different IP addresses that were contacted and the number of incoming connections from which traffic was accepted.

Station address	# outgoing	outgoing addresses	#incoming	incoming addresses
147.32.84.138	530,299	18	648	13
147.32.84.59	263,150	27,919	14,971	7,041
147.32.84.229	143,712	67,934	1,061,741	510,790
70.37.98.60	64,001	1	0	0
147.32.85.25	44,863	5	91	26

Tab. 4.6: Scenery 13 characteristic

Table 4.6 shows the 5 stations with the highest number of outgoing connections in scenario 13. Data from the show shows that the second and third stations with the highest number of outgoing connections have a high number of incoming connections as well, while the first, fourth and fifth have a categorically smaller number. At the same time, the number of stations the communications were sent to was also low. Without further analysis, it is not possible to determine whether it truly is a spam station, but it is a sufficient enough indicator to suggest that more parameters be investigated.

4.2 Criteria

The criteria for detecting spam machines using Netflow are proposed in this section. The criteria are separated into two groups. There are two explanations for this. To begin, due to the enormous number of IPs to be evaluated, it would be beneficial to make a preliminary selection on which to do a more detailed study. Only devices that exhibit clear suspicious behaviour will be further investigated, minimising processing time. Second, screening only the suspicious machines according to the first criteria

narrows the result set by omitting non-suspect machines. By calculating a likelihood, the remaining criteria can be utilised to order the previously questionable devices. If this were done without a first elimination round on all machines that handle SMTP traffic, the result would be an excessively huge result set and an excessively long processing time.

As a result, the following two types of criteria have been drafted:

Acceptance criteria
These are the standards by which suspect machines are identified. If these requirements are not met, the IPs are not considered for further investigation.

Ordering criteria
Acceptance criteria are used to order the machines, and these are the criteria used to order them. The purpose is to rank the machines in order of most suspicious to least suspicious. A ranking will be determined by the ordering criteria. It is feasible to broaden the acceptance criteria; some machines will easily pass through them, while others will struggle. This can be used to sort machines into suspicious and non-suspicious categories.

4.2.1 Acceptance criteria

In order to mass filter stations the following acceptance criterion has been defined:

The proportion of inbound and outbound SMTP connections
This was the initial hypothesis for detecting suspicious machines. This criterion was discussed in the preceding chapter. Only machines with a lot of outbound SMTP traffic but little or no inbound SMTP traffic are chosen.

$$\frac{\text{incoming SMTP connections}}{\text{outgoing SMTP connections}} < 0.005 \tag{4.1}$$

Real Traffic Examples
In this section, an actual SMTP traffic trace acquired in the campus network of company X which cannot be disclosed for privacy reasons, is utilised to observe the reality of the discussion. The trace was taken at the gateway of the campus network to the Internet from 03:44 to 23:56 on May 1st, 2022 (the data of the last few minutes was lost due to a technical difficulty). It only provides the commencement of each outbound SMTP session from the college network to the Internet, including SMTP

client identities. It does not contain any other data such as email sender/recipient addresses and message body. Table 1 includes the basic statistics of the trace data.

Total Number of Sessions (outbound)	714,043
Unique IP Addresses	614
Unique Client Names	3,444

Tab. 4.7: The number of sessions, IP addresses, and client names in the session trail

In addition, an irregularity is readily observed in Table 1. There are too many (3444) client names transmitted via HELO/EHLO commands compared with the number of unique IP addresses (614). It has been analysed how many names were used by each IP address and identified one IP address that had sent 2,101 email messages using 1,932 distinct names. That means that, by studying the client names, a rogue SMTP client has been discovered unintentionally. The client had not been spotted by other monitoring systems such as an IDS on our network. The session trace of the malicious client has been reviewed manually, and it has been observed that the SMTP client changed its name on every SMTP connection, perhaps for camouflage. The names were not completely random, as they included many famous service providers like netscape.net, hotmail.com, excite.com, and soon. As a result, it has been verified that a basic validity check of customer names could sometimes reveal aberrant clients by an actual case. On the other hand, not all the harmful clients mask its identity by changing their names. A Trojan executing on a computer can easily retrieve the valid domain name of the network and send an email using a valid client name. In such a circumstance, the approach is not effective. Next, Table 4.8 further classification of client names detected in the session trace (excluding a spammer spotted in the preceding discussion) (except a spammer spotted in the previous discussion). The table shows that only less than half of customers have correctly configured their names in terms of the SMTP protocol specification. That means that it is tough to detect odd customers merely by checking whether the name is a valid *Fully qualified domain name* (FQDN) and inside our domain or not.4

Unique IP Addresses	614
Unique Client Names	1233
Domains Inside	**xxxxx-u.ac.jp**
Hostname only	smtp.office365.com
Domains Outside	**hjfhsjfh-u.ac.jp**
.local domain	xxx@office365.com

Tab. 4.8: Details of client names discovered in the trail

As expected, several clients have been detected using domain names outside **xxxxx-u.ac.jp**. By human inspection, the name has been verified and it has it was decided that most of them were authentic (not malicious) SMTP clients, which host outside services. It is difficult to accept these hosts as authentic automatically, hence a whitelist is needed to exclude these hosts from questionable hosts.

There were 35 IP addresses that used multiple client names, and the maximum number of names per a single IP address was 8. Manual inspection indicated that most of them were names without a domain, private IP addresses, and a loop-back address (127.0.0.1). The assumption stands that these IP addresses worked as a *Network address and port translation* (NAPT) box which was a gateway to a private IP network with several customers. The number of messages transmitted by these hosts was quite little, thus it was not possible to verify if the clients were malicious or not. Currently, a longer traffic trail gathering is being carried out in order to investigate alternative approaches to detect rogue clients, including a study of clients' temporal behaviour.

In this section, a detection approach employing a client's behaviour such as pauses between connections is described. The host sent 720 messages from 10:28 to 10:51, 178 messages from 12:54 to 13:00, and 304 messages from 14:01 to 14:07. Insufficient traffic traces of subsequent days are available, but it can be estimated that the host had been attacked by malware at a specific point, and then started transmitting spam. Therefore, a note of a sudden debut of an email sender must be taken, which had not been sending an email before, or rapid rise in the amount of emails from an IP address. A challenge is how to identify a malicious client from a new mailing-list server that might also suddenly starts sending many email messages. To find a way to distinguish a malware-induced mail client from a normal client, the behaviour of how the malicious client contacted other mail servers has been investigated. The virus program delivered many email messages automatically, which might have distinct features from a regular sender that distributes email messages sent by humans.

The variety of destinations available

Some false positives were discovered while playing with Criterium 1 with one or two separate destination IPs. These were discovered to be logging methods that emailed log messages. Additionally, more computers were positively validated with a large number of diverse destination IPs while verifying using DNS blacklists and SpamAssassin blacklists. This can be explained in part by the fact that botnets transmit a small number of messages to a large number of distinct domains in order to avoid being caught or banned. Because the institution has five load-balanced SMTP servers, it was decided to set this criterion to a value greater than 7 distinct

destinations for the Kaggle dataset, to at least exclude most users who will only send email using the company's SMTP servers. This criterion can be summarised as follows for the Kaggle dataset, eq. 4.2

$$\text{distinctDestinations} < 7 \tag{4.2}$$

The Kaggle dataset was queried to group the traffic by source IP throughout the whole Kaggle dataset to find the computers where the assumption held true. The following metrics were calculated for each IP: To discover the computers where the assumption held true, the Kaggle repository was searched to group the traffic by source IP throughout the whole Kaggle dataset. For each IP, the following metrics were calculated:

- The total number of destination servers. These are the separate destination servers for outgoing connections using port 25 as the destination.
- The total number of connections made. These are all the outgoing connections with port 25 as the destination.
- The number of different machines arriving. These are all the unique IP addresses that connect to the present system over port 25.
- The total number of connections received. These are all the incoming connections with port 25 as the destination.
- Timestamps

Listing 4.5 depicts the entire data aggregation implemented in Python in the web environment Kaggle.

Listing 4.5: Data aggregation

```
with open('../input/basedata/a.csv') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=';')

    for row in csv_reader:
        row_ipadress = row[2]
        if isIPAdress(row_ipadress):
            time = timeToSeconds(row[1])
            if row_ipadress in statistics:
                statistics[row_ipadress][0]+=1 #outgoing connection count
                statistics[row_ipadress][1].add(row[3])  #outgoing connections
                statistics[row_ipadress][3].add(time) #timestamps
            else:
                statistics[row_ipadress] = [1,{row[3]},0,{time}] #new record
            add_dest(row_ipadress,row[3],timeToSeconds(row[1]))

            if maxTime<time: maxTime = time
            if minTime>time: minTime = time


for row in statistics:

    if statistics[row][0]==0:ratio = -1#no outoing
    else: ratio = statistics[row][2]/statistics[row][0]#ratio

    temp = numpy.array(list(statistics[row][3]))#convert to needed format
    sd = numpy.std(temp,dtype=numpy.float32)#calculate sd
    avg = numpy.average(temp)#average of times
    results[row] = [statistics[row][2],statistics[row][0],ratio,len(statistics[row
    ][1]),sd,avg]#add to table
```

Listing 4.5 above shows the iteration through the dataset and data aggregation. Some defined functions (for example timeToSeconds)are not shown here, see D for further details.

It's worth mentioning that each direction of a connection is recorded as a separate flow in Netflow. Furthermore, the assumption just requires the incoming and outgoing SMTP connections. The number of sources and destination IP addresses has been raised. This could also provide some light on the behavioural variations between legitimate and illegitimate gadgets. Finally, because the purpose is to identify obvious spam machines, the list is sorted (descending) by the number of outgoing connections to port 25.
The remaining columns reflect the number of distinct destinations, outgoing connections, distinct arrival machines, and incoming connections, as previously stated. The machines in bold correlate to the concept that spam machines will send a lot of traffic but receive very little.

Because *Transmission Control Protocol* (TCP) has a three-way handshake, SMTP should send some packets in the other direction if the target port is open. The system is most likely performing a port search if there are far fewer answers than outgoing connections. There are several connections to various places. However, because the

IP	dist out	out	dist in	in
1	1,980	334,356	36,354	675,381
2	3,227	247,588	36,354	17,645
3	11,459	11,459	36,354	745,408
4	**39,460**	**244,117**	**0**	**0**
5	11,280	240,733	153,275	675,632
6	3,512	238,665	788	27,738
7	7,943	195,573	132,616	539,297
8	**2**	**184,698**	**0**	**0**
9	2,252	136,847	10	187
10	**24,213**	**116,898**	**1**	**2**
11	7,774	115,746	8	24,972
12	8,376	68,413	24	172,464
13	**17,532**	**64,685**	**0**	**0**
14	341	57,251	66,237	901,280
15	443	54,212	10	578

Tab. 4.9: Data aggregation per IP address [6]

line indicating the number of outgoing connections is a little smaller than the line representing the number of various destination servers, the computer creates several connections to a variety of different destinations. The reasons for this are unclear; SMTP allows all mail to be delivered in a single connection, eliminating the need for separate connections for each mail. It's also possible that the Netflow timers are causing the connections in many flows to break.

In summary, this computer exhibits highly suspicious activity, since it sends a large amount of traffic to a variety of different locations while getting only two incoming connections. The first spam machine has most certainly been discovered! Most other IPs with a large number of outbound connections compared to incoming connections showed the same tendency.

4.2.2 Ordering criteria

Spam protection is a critical component of any email security solution. Spam is profitable, that is why all mailboxes are continuously bombarded. Deciphering spam will thus likely remain an „open field“ in the future.

The comprehensive spam detection algorithm will use the ordering criteria to order the results. To do this, each ordering criterion has been assigned a value between 0 and 1. The variables a through e, which will be used to show the whole

procedure, reflect those values. The greater the value, the more likely a computer is spamming. As a result of the ordering criteria, a ranking is generated.

Incoming Connections

Acceptance criterion 1 already specifies that suspect workstations will have fewer incoming SMTP connections than outgoing SMTP connections. This ordering criterion will consider the total number of incoming SMTP connections. Only a few machines have connections coming in, which was also seen in the Kaggle dataset. Since the point of spam machines is to send out mail, it doesn't make much sense for them to be able to receive mail. On the other hand, mail servers that are legitimate will want to accept mail.

The result of this argument is that machines with incoming SMTP connections are less suspicious than those without incoming SMTP connections (of course, this logic must be tested experimentally). This is why this criterion is defined as follows, and implemented as code 4.6:

$$a = \begin{cases} 1 & \text{incoming SMTP connections} = 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.3}$$

Listing 4.6 below shows implementation of criterion a in Kaggle.

Listing 4.6: Ordering criterion a

```
if results[current_ipAddress][0]==0: #incoming connection count
    a = 1
else:
    a = 0
```

Obviously, this is the behaviour that the vast majority of individuals will exhibit if they use a mail client application like Outlook or Thunderbird configured to use SMTP to send email. Because they did not install a mail server, they will not receive e-mail over SMTP; instead, they will utilise protocols such as POP3 or IMAP to receive messages from the mail server provided by their internet provider, for example. This conduct indicates that the computers used by these individuals are not spamming. Note that this criterion is evaluated based on the result set of the combined acceptance criteria, therefore machines that meet this criterion must also have a large number of outgoing SMTP connections to multiple distinct destination IP addresses. This will likely omit IP addresses that are not malicious.

Number of Distinct Destinations

The second acceptance criterion stipulates that suspicious devices must have at least a few target servers. This ordering criterion attempts to clarify the situation. To accomplish this, a graph depicting the number of unique destination IPs per source IP was created. For this plot, the following considerations were taken into account:

- This plot will, of course, have to be on the result set of computers that meet the combined tolerance requirements, because the ordering will take place on this set of IP addresses.
- Aside from that, because the purpose is to evaluate a seven-day (full-week) period of time, the plot will need to encompass seven days.
- Furthermore, because the last data capture of the Kaggle dataset occurred one year prior to the time of the analysis, the behaviour could have changed as a result of network changes or changes in spam behaviour.

Given these considerations, it was decided to conduct a new full week of data collection for this plot as well as the other plots required for the criteria analysis. This capture was carried out on the 2nd of May 2022 and ran for a total of seven days. As can be seen, only a small number of the 10,000 selected IPs have more than ten destinations (about 1 per cent). These are the IPs that are thought to be the most suspicious. suspicious ones, especially when combined with the criteria for acceptance (only IPs that The ordering criteria are based on how well the items meet the acceptance criteria. Also, remember Botnet machine, will send spam to a lot of IPs to avoid being caught. Several things can be said:

- The university has five mail servers that spread out the work. This means that getting in touch with for the average PC, having about 5 mail servers isn't that strange.
- This doesn't mean, though, that IPs that were sent to fewer than 10 IPs are not valid.

$$b = \begin{cases} 1 & \text{distinct destination servers } > 10 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

Listing 4.7 below shows implementation of criterion b in python using with the use of a simple if statement.

Listing 4.7: Ordering criterion b

```
if results[current_ipAddress][3]>10:#diff destination count
    b = 1
else:
    b = 0
```

Idle time

A lot of downtime is a sign of a suspicious IP. Like most other plots involving

suspicious machines, all of the traffic in this one is made in short bursts with sudden peaks. This gives space for a possibility of spotting suspicious behaviour based on idle time. To do this, the idle time for the same data capture was plotted. Note that „active time“ is defined as the number of 5-minute periods in a week when a connection has been made. The rest is „idle time“. This percentage will also be high if acceptance criterion 2 uses a low number of minimum outgoing SMTP connections. Based on this criterion, it is likely that this is not the case. Of course, this also depends on the router(s) used in the Netflow dataset. For a router on a large network, this is probably less likely than for a router on a small network. Since the top 10,000 machines chosen are the ones that send the most traffic out, it seems odd that the traffic is only going out for a short time. The plots of the official mail servers show that a normal mail server will have a kind of baseline. Because of this, if a machine has a lot of idle time, it is more likely to be a scam, eq. 4.5 :

$$\text{c} = \text{percentage of idle time} \tag{4.5}$$

Implementation of criterion c is of course more complex than those of criterion a and b. The algorithm it self can be seen in listing 4.8 below.

Listing 4.8: Ordering criterion *c*

```
while j <len(times): #all timestamps split
      start = times[j] #current timestamp

      count=0
      if not j==len(times)-1:
          while times[j]+300 >times[j+1]:
              j+=1
              count+=1
              if j==len(times)-1:break

      if count == 0: nonidletime+=300#single timestamp in 5 minute range
      else: nonidletime+= times[j] - times[j-count]#more timestamps less than 5
  mins apart
      j+=1
  c = 1 - nonidletime/timeframe
```

Standard Deviation

Using the standard deviation on the 5 minute plot points in the outgoing SMTP connections plot is a simple way to figure this out. The goal is to find machines with a large standard deviation, which means that the points on the plot are far from the mean. On the other hand, a small standard deviation means that most of the points are close to the mean. Clearly, points that are close together show a baseline, which is what a normal mail server should do. Because of this, values that are greater than 10 are thought to be suspicious, which is about 10 per cent of the

time, eq. 4.6.

$$d = \begin{cases} 1 & \text{if } \sigma > 10 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

,where σ is the standard deviation.
Python's package numpy contains many sophisticated tools, one of which is std, which calculates the standard deviation of the input array. Below the implementation is shown in 4.9

Listing 4.9: Ordering criterion d

```
if  results[current_ipAddress][4] > 10:#SD criterion
    d = 1
else:
    d = 0
```

Peak Behaviour

In the last criterion, the standard deviation was used to see if there was a nice baseline like a normal e-mail server should have. So far, this doesn't find sudden peaks as seen in previous studies.

A complicated peak detection system could be used but because of the large number of IPs that will have to be analysed (manually, this is limited to 20,000 machines, which takes about three days to do), this won't be possible because of how long it will take to process. So a simple solution was picked. To find sudden peaks, the number of data points that are bigger than four times the standard deviation plus the mean value over a five-minute time span is counted. Compared to the average, this should find important peaks (=5-minute time spans). Based on some testing, the value of 4 was chosen so that this mechanism is not too sensitive, but is sensitive enough to pick up on relatively high peaks. Based on these assumptions, machines with more than 10 peaks were chosen as the suspicion parameter, eq. 4.7.

$$e = \begin{cases} 1 & \text{if } P(datapoints > (4\sigma + \mu)) > 10 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

,where σ is the standard deviation and μ is the mean value.
Listing 4.10 depicts the code used for calculating criterion E.

Listing 4.10: Ordering criterion e

```
for time in times:
        countSub =  times.count(time) # count of time in list
        if countSub > 4*results[current_ipAddress][4] + results[current_ipAddress
    ][5]: #condition
            count+=1
    if count>10:
        e = 1
    else:
        e = 0
```

The Efficient Automation of Data Extraction

To detect suspicious computers, as shown in the preceding section, a few actions must be taken. Manually doing this takes a long time and is inefficient in terms of machine resource utilisation. After experimenting with technologies such as flow-tools, it was immediately determined that a database is more adaptable for research.

Queries take a long time to perform due to the enormous dataset in the database. It takes about 1.5 hours to go through the entire dataset with a single query. When you have to do this for each suspect machine, it becomes impossible. The process should be more efficiently mechanised.

The simplest and most obvious approach to improve performance is to remove all non-relevant material from the repository. All flows with port 25 (SMTP) in the source or destination port field are considered non-relevant. As a result, the first step is to make a new datatable that only contains those flows. Because the number of entries was lowered from 1,004,370,893 to 13,666,533, SMTP now accounts for only roughly 12 % of total flows. The SMTP traffic table is then used to aggregate a few statistics per IP (number of outgoing connections, distinct destination servers, and so on). A second table is created from this traffic table, with just those IPs that satisfy the premise that spam machines generate a lot of outgoing traffic but receive little traffic. The following criteria were selected:

$$\frac{\text{incoming SMTP connections}}{\text{outgoing SMTP connections}} < 0.003 \tag{4.8}$$

As a result, only the computers highlighted in bold in the list of IPs are chosen. Because of GDPR information about IP address cannot be provided. As a result, only IPs with a large number of outbound SMTP connections but few or no incoming SMTP connections are acquired. Only 0.3 per cent of the number of outgoing connections is allowed as an incoming connection when the value is 0.003. The fundamental idea is to limit the number of incoming connections to a fraction of the outgoing connections. This behaviour has been seen a lot in Netflow data. Another option is to pick just IPs that have no incoming connections but a large number of

outgoing connections. Then IPs with a small number of incoming connections would be excluded from the list, despite the fact that many machines with a small number of incoming connections were discovered to be acting suspiciously. As a result, the ratio is chosen instead. The number of outbound connections to port 25 is used to sort the results. The top 100 machines in this group are used to generate time plot data that can be fed into a charting tool like gnuplot.

4.3 The Results of Testing

An automated data extraction strategy based on simple principles was built in the preceding section. The end result is a list of hosts with a large number of outgoing connections to port 25, but no (or only a small number of) connections to port 25, sorted by the number of outgoing connections (descending). Data points have also been generated as input for plot generation. Large intervals of no traffic were found in the Kaggle dataset, followed by unexpected jumps in the number of outbound connections to port 25. The SURFnet and Geant data both showed the same pattern. To decide how to proceed with a spam detection technique based on Netflow, more analysis of the questionable IPs is required. The most crucial consideration is to minimise false positives. An SMTP server set up for genuine mass mailings, for example, can be a false positive with this approach. The SMTP server will send a lot of traffic, receive very little, and have sharp peaks with this configuration. The question is if such SMTP servers are frequently utilised in practice.

At the very least, the first experimental efforts presented in this chapter indicate a potential future for detecting spam robots. The next step is to choose an algorithm that accomplishes this.

4.3.1 Validation

Finding spam is not an exact science, and there are a lot of exceptions. There are false positives and false negatives with every method that is currently used. This is true for both the validation methods and the algorithm presented in this work. Still, finding ways to back up the results is an important part of this research. Since only Netflow data is available, it is hard to say for sure why strange things are happening in the repositories. Only flow-level data is looked at, and spam messages can't be picked out by looking at the mail bodies themselves. This chapter talks about ways to prove that the conclusions drawn in the earlier steps of this research are correct. DNS blacklists and SpamAssassin log files were chosen as the two ways to do it. First, we'll talk about both methods, and then we'll show validation results.

4.3.2 Validation Methods

This section starts by talking about DNS blacklists and naming a few that were used for this research. Also, the bad things about using DNS blacklists to validate are talked about. In the second part of this section, SpamAssassin and how it works using SpamAssassin log files as a way to check if something is right is discussed.

DNS Blacklists

Using a blacklist is a simple way to stop spam from being sent. A blacklist is a list of IPs that are known to send spam. Most of the time, these blacklists are run through the DNS system.

A simple DNS lookup on the machine that keeps the blacklist can be used to check an IP to see if it is on the blacklist. Backlists vary a lot in how reliable they are. Some blacklists block half of the internet, while others don't get updated very often.

DNSBLs are an important part of any toolkit for fighting spam. Because so many people on the Internet are updating them, you get the benefit of blocking a spammer before you even get any spam. To know how DNSBLs help, you need to know what kinds there are and how they work. Most DNSBLs are IP-based, which means they look at the IP address of the mail server. Every host connected to the Internet, like e-mail servers, has its own unique IP address. This IP address is checked against a database to see if it belongs to a spammer, an open relay, or an open proxy that is already known.

Validation Results

The first experimental steps, which are described, have been checked and found to be correct. This is a problem right away because the data was from a long time ago. This makes it hard to verify with DNS blacklists, which, as stated above, are time-sensitive.

Because of this, the results of this first validation will not be discussed furthermore in this chapter. So, from the live Kaggle data, two sets of 100 random IPs with outgoing SMTP connections were chosen. These were checked against the 25 blacklists and the 5 conservative blacklists. Table 4.10 shows the number of IPs that did not pass validation

Run	Optimistic	Conservative
1	1/100	5/100
2	7/100	11/100

Tab. 4.10: Validation Fail Rate

Validation Fail Rate

So it seems like picking IPs at random is a good way to get a high percentage of IPs validated. But keep in mind that the goal is not to get a high percentage of IPs validated. Instead, the goal is to find spam machines reliably. If you pick IPs at random, you might get a high percentage of them to work, but it won't stop legal machines from sending mail. But it will be hard to validate with DNS blacklists when such a large number of machines have already been validated with random IPs. There isn't much more that can be done to make the algorithm better. Does this work reliably? Also, is it true that more than 90 % of all IPs with SMTP connections that send emails are spammers? Of the 1.064.363 IPs with outgoing SMTP connections, nearly 40 % have only one in 7 days, and nearly 80 % have less than 10. So just 10% have more than 10 SMTP connections.

Because most IPs with outbound SMTP connections have less than 10 connections, an analysis cannot be performed on them. The algorithm's acceptance criterion solves this. This criterion's influence is now understood; it affects just a small percentage of mail sending IPs.

As a result, when a bot is beyond the domain of the router(s) supplying Netflow data, it is difficult (perhaps impossible) to detect them using solely Netflow data. SMTP destinations for bots on our domain are likely to be numerous, which is factored into ordering criteria.

Based on these findings, another nice criterion to classify spammers is the ratio: Number of outgoing connections vs Number of distinct destinations

4.3.3 Communication Patterns

The mail transfer-delivery process has three main communication patterns based on the protocols and components involved. Similar to the previous email systems, it is feasible to distinguish between communication between the client and the mail server, and the mail server to the recipient. When an email message is sent over the Internet, a new communication pattern occurs between mail servers. If the customer chooses to use an end-to-end secure email system (encrypting all traffic), it is viable to detect another communication pattern that is critical when evaluating secure email solutions.

Connection Establishment

For successful connections, the server responds with SMTP response code 220 (service ready) or 421 (service unavailable) for unsuccessful connections. The client then sends a HELO SMTP command to the server with its domain name as an input. The server answers with a code, usually 250 for a successful request. The client

then issues an AUTH instruction to the server, waiting for a server authentication challenge to arrive as a parameter of a 334 response code.

Email Transfer

Email messages can be sent from the client to the server and subsequently to their intended recipients if the connection is successful. The client sends a MAIL FROM instruction to indicate the sender of the email message (mailbox and domain name address) for error or reporting messages. The server will respond with a code, usually 250, indicating successful command reception.

The client then sends the RCPT TO command using the recipient's email address. To send an email message, the server will answer with 250 and wait for the client to deliver DATA. The server responds to the DATA instruction with 354, meaning it is ready to receive mail.

Connection Termination

To close the connection to the server, the client sends the QUIT SMTP command, which the server must acknowledge. A successful connection response code is 221 (service closed).

Mail Server to mail server

Communication between mail servers is essentially a server's *Message transfer agent* (MTA) client with another's MTA server. So it is It is clear that mail server communication follows the same pattern. as in the previous section. Notable particularity is the role of DNS in email routing. improving security between mail servers and email transmissions. Emails are being forwarded, the former must locate the latter. verify its identity Identity spoofing is a serious security concern.

Proposed DNS checks The recipient mail server can first check the sender's IP address (through the SMTP TCP connection) against the sender's *Mail exchange* (MX) record (the recipient knows the sender's domain name via his/her mailing address and can run an nslookup to retrieve the relevant MX record).

Additionally, reverse *Pointer* (PTR) records may make it easier to fight against spammers and identity spoofing attacks in the future. Malicious individuals frequently attempt to impersonate genuine websites by using a forged mailing address. When a mail server gets an email message, it can check to see if the IP address from which the message was received matches the IP address of the *Domain name system pointer* (DNSPTR) record for the domain to which the mailing address belongs. The resolution of a PTR record is the process of resolving an IP address to the hostname associated with it, and it is effectively the inverse process of a DNS lookup in that it does not require the use of a domain name. A failure of the process will occur if

there is no PTR record in the DNS server connected with the sender's mail server, and the email message will not be delivered, e.g., it can be safely classified as spam.

Using this method, the recipient of an email can rapidly verify that the sending host is listed in the SPF record before deciding whether or not to accept an email from that sender. In order to maintain backward compatibility, SPF records are either kept in *Text* (TXT) files or in SPF-formatted files. Evidently, all of these procedures to protect against identity spoofing are vulnerable to the open, unprotected nature of the Domain Name System (DNS).

Mail Server to Server (recipient)

POP3 and IMAP are the protocols used to communicate between the recipient of an email message and his or her mail server. Assuming that the recipient of the email message is using the TCP/IP port 110, the recipient will connect to its designated POP3 server. Following completion, the recipient sends a message containing both the username and password for its POP3 server mailbox, and both of these messages are sent while waiting for positive acknowledgement. Once connected to its mailbox, the recipient can list all of the email messages contained within it by issuing the LIST command, or retrieve a specific email message by issuing the RETRIEVE command, which takes the email identification as an argument, from the mailbox.

4.3.4 Algorithm output

Due to time restrains it was not possible to test entire datasets. Therefore, only the first 100,000 records were analysed. These records contained 1,703 unique IP addresses. Out of the 1,703 addresses only 29 passed the acceptance criterion (approx. 1,7 %. Table B shows collected data of these 29 IP addresses. Table C shows calculated values of each criteria of these IP addresses. On average, the total sum of the observed parameters resulted at 3,01.

Conclusion

The main goal of this work was to design and develop an algorithm that can detect and filter spam messages in normal network traffic. The primary reason why it is necessary to filter using logs without examining the messages themselves is to protect privacy and personal information.

As part of the design, an analysis of the current spam filtering solution was first performed with and without examining the contents of the communication. In the practical part, a working environment for the development of a spam detection algorithm was created. The algorithm sends a query to DNS Blacklists as a first step. If the station is not listed, the flow is processed and added to the aggregation per IP address. The algorithm then checks whether or not the station in question meets the established acceptance criterion and should be considered potentially dangerous. Further details per IP address are collected if the station is considered suspicious. Such collected details are then used by the 5 filtration criteria to sort all suspicious addresses.

For reasons of time computing capacity constraints, not all IP addresses in the dataset used have been examined, only the first 10,000 records for DNS queries and 100,000 for the criteria. Of those addresses, 27.66 % were listed on at least one blacklist. On average, each address was then listed on 1,9378 lists. The first 100,000 records contain 1,703 unique IP addresses. Only 29 of those passed the acceptance criterion (approx. 1,7 %). The average sum of the filtration criteria of these IP addresses resulted at 3,01.

In conclusion, the main objective of this work, which was to develop a filtration algorithm, was met in the practical part. The criteria that are used in the algorithm can be further developed and optimised in future analyses. A good continuation would be to adjust the constants in the criteria formulae and to observe the influence of such modifications. For example, 28 out of 29 IP addresses passed through criterion B. Here, it would be appropriate to tighten the criterion by increasing the number of different destination addresses, thus reducing the number of stations that meet it. Conversely, criterion A was met by only 4 stations – some stations had, for example, 1-3 incoming connections. An opportune alteration might be to allow a few incoming connections.

Bibliography

[1] Joseph Johnson. *Global spam volume as percentage of total e-mail traffic from January 2014 to March 2021, by month*. URL: https://www.statista.com/statistics/420391/spam-email-traffic-share/ (visited on 11/06/2021).

[2] Sebastian Garcia et al. *An empirical comparison of botnet detection methods*. URL: http://dx.doi.org/10.1016/j.cose.2014.05.011 (visited on 11/05/2021).

[3] *What does SPAM stand for?* URL: https://www.mailguard.com.au/blog/what-does-spam-stand-for/ (visited on 11/05/2021).

[4] *NetFlow for Cybersecurity*. 2017. URL: https://www.ciscopress.com/articles/article.asp?p=2812391%5C&seqNum=3 (visited on 11/06/2021).

[5] *TCPDUMP&LIBPCAP*. 2010. URL: https://www.tcpdump.org/ (visited on 12/09/2021).

[6] Gert Vliek. "Detecting spam machines, a Netflow-data based approach". Magisterská práce. University of Twente, The Netherlands: University of Twente, 2009.

[7] *Netflow Services Solutions Guide*. 2007.

[8] Yadong Li et al. "Research based on OSI model". In: *2011 IEEE 3rd International Conference on Communication Software and Networks*. 2011, pp. 554–557. DOI: 10.1109/ICCSN.2011.6014631.

[9] Calton Pu and Mustaque Ahamad. "Resisting SPAM Delivery by TCP Damping." In: Jan. 2004.

[10] Jaeyeon Jung and Emil Sit. "An empirical study of Spam traffic and the use of DNS black lists". In: Jan. 2004, pp. 370–375. DOI: 10.1145/1028788.1028838.

[11] Alex Phillip. *What's an SMTP port & how to choose the right one*. URL: https://postmarkapp.com/blog/choosing-the-right-smtp-port (visited on 12/08/2021).

[12] Dominik Dancs. "Detecting active abuse of leaked personal data". PhD thesis. May 2020. DOI: 10.13140/RG.2.2.23461.93927.

[13] J. Klensin et al. *SMTP Service Extensions*. URL: https://www.hjp.at/doc/rfc/rfc1869.html (visited on 11/07/2021).

[14] *UCEPROTECT®-Network*. 2021. URL: http://www.uceprotect.net/en/index.php (visited on 11/07/2021).

[15] *What is a DNSBL?* URL: https://www.dnsbl.info/ (visited on 12/08/2021).

[16] Emmanuel Gbenga Dada et al. “Machine learning for email spam filtering: review, approaches and open research problems”. In: *Heliyon* 5.6 (2019), e01802.

[17] Ismaila Idris and Ali Selamat. “Improved email spam detection model with negative selection algorithm and particle swarm optimization”. In: *Applied Soft Computing* 22 (2014), pp. 11–27.

[18] *Kaggle*. URL: https://www.kaggle.com/ (visited on 12/08/2021).

[19] *What is a Botnet?* URL: https://www.paloaltonetworks.com/cyberpedia/what-is-botnet (visited on 05/22/2022).

[20] Dr. Leslie F. Sikos. *The pcap Packet Capture Format*. URL: https://www.lesliesikos.com/pcap/%5C#:~:text=The%5C%20pcap%5C%20file%5C%20format%5C%20is,support%5C%20for%5C%20nanosecond%5C%2Dprecision%5C%20timestamps.%5C&text=Each%5C%20captured%5C%20packet%5C%20starts%5C%20with,actual%5C%20length%5C%20of%5C%20the%5C%20packet. (visited on 12/10/2021).

[21] Y. Shafranovich. *Common Format and MIME Type for Comma-Separated Values (CSV) Files*. RFC 4180. RFC Editor, Oct. 2005. URL: https://www.rfc-editor.org/rfc/rfc4180.txt.

[22] The Wireshark team. *Wireshark*. Version latest. 1998. URL: https://www.wireshark.org/.

[23] NETRESEC AB. *Tetresec*. Version latest. URL: https://www.netresec.com/.

[24] *.PCAP File Extension*. URL: https://www.reviversoft.com/en/file-extensions/pcap (visited on 12/10/2021).

[25] The pandas development team. *pandas-dev/pandas: Pandas*. Version latest. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[26] dmippolitov. *Pydnsbl*. Version latest. Sept. 2017. URL: https://github.com/dmippolitov/pydnsbl/.

[27] Python Packaging Authority. *pip*. Version latest. URL: https://pypi.org/project/pip/.

[28] *Python Dictionaries*. URL: https://www.w3schools.com/python/python_dictionaries.asp (visited on 12/11/2021).

Symbols and abbreviations

AI Artificial intelligence

AS Autonomous system

CTU Czech technical University

ČVUT České vysoké učení technické

DOS Denial of service

DNS Domain name system

DNSBL Domain name system blacklist

DDOS Distributed denial of service

DNSPTR Domain name system pointer

FQDN Fully qualified domain name

GDPR General Data Protection Regulation

GPU Graphics processing unit

POP3 Post office protocol version 3

PTR Pointer

IDE Integrated development environment

IMAP Internet message access protocol

IANA Internet Assigned Numbers Authority

IETF Internet Engineering Task Force

IP Internet protocol

ISP Internet service provider

MIME Multipurpose Internet Mail Extensions

MTA Message transfer agent

MX Mail exchange

NAPT Network address and port translation

OSI Open Systems Interconnection

RBL Real-time blacklist

RFC Request for Comments

SMTP Simple mail tranfer protocol

SNMP Simple Network Management Protocol

TCP Transmission Control Protocol

ToS Type of Service

TXT Text

List of appendices

A Proposed detection algorithm scheme

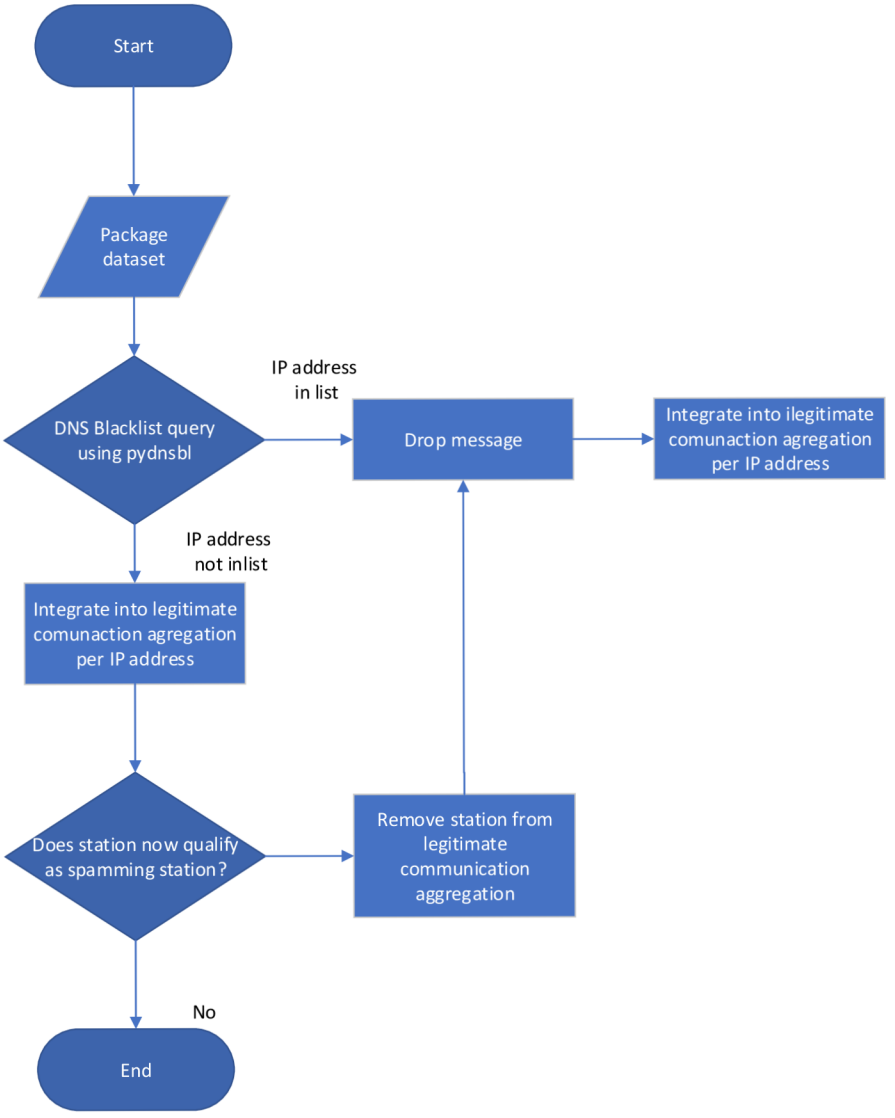

Fig. A.1: Proposed detection algorithm scheme

B Data collection sample

ID	Incoming	Outgoing	In/Out ratio	Distinct	Standard deviation
1	1	1666	0,0006	28	7,43
2	7	7777	0,0009	146	18,69
3	3	1000	0,0030	12	3,91
4	0	1377	0,0000	33	13,24
5	3	1428	0,0021	26	6,79
6	5	1923	0,0026	22	12,99
7	4	1428	0,0028	16	6,92
8	1	769	0,0013	7	8,12
9	4	1904	0,0021	34	6,86
10	4	1481	0,0027	15	4,41
11	3	1304	0,0023	11	14,38
12	6	2500	0,0024	69	19,63
13	2	3333	0,0006	85	24,69
14	0	2715	0,0000	30	20,78
15	4	1538	0,0026	39	20,89
16	6	8571	0,0007	74	21,7
17	1	1428	0,0007	17	10,54
18	8	2857	0,0028	36	20,62
19	1	625	0,0016	17	19
20	6	6666	0,0009	70	20,74
21	4	2000	0,0020	22	4,04
22	4	1428	0,0028	34	5,32
23	0	3758	0,0000	45	3,44
24	8	3636	0,0022	90	7,71
25	2	869	0,0023	18	24,77
26	7	5000	0,0014	44	21,42
27	5	5000	0,0010	51	18,42
28	0	3731	0,0000	35	9,67
29	3	1304	0,0023	14	3,04

Tab. B.1: Collected data

C Algorithm output

ID	A	B	C	D	E	Σ
1	0	1	0,8	0	1	4,8
2	0	1	0,85	1	1	3,85
3	0	1	0,85	0	0	3,85
4	1	1	0,84	1	0	3,84
5	0	1	0,82	0	1	3,82
6	0	1	0,8	1	1	3,8
7	0	1	0,79	0	1	3,79
8	0	0	0,76	0	0	3,76
9	0	1	0,75	0	1	3,75
10	0	1	0,74	0	1	3,74
11	0	1	0,72	1	1	3,72
12	0	1	0,7	1	1	3,7
13	0	1	0,84	1	1	2,84
14	1	1	0,82	1	1	2,82
15	0	1	0,81	1	1	2,81
16	0	1	0,8	1	1	2,8
17	0	1	0,77	1	1	2,77
18	0	1	0,74	1	0	2,74
19	0	1	0,73	1	0	2,73
20	0	1	0,72	1	0	2,72
21	0	1	0,8	0	1	2,72
22	0	1	0,85	1	1	2,71
23	0	1	0,85	0	0	2,7
24	1	1	0,84	1	0	2,7
25	0	1	0,82	0	1	1,81
26	0	1	0,8	1	1	1,78
27	0	1	0,79	0	1	1,76
28	0	0	0,76	0	0	1,76
29	0	1	0,75	0	1	0,7

Tab. C.1: Algorithm output

D Content of the electronic attachment

The algorithm itself is an inseparable part of this thesis. It can be found in a .ipynb file that was exported from the Kaggle notebook and attached to this thesis. Some parts of the implementation are also possible to find as listings. By default, Kaggle uses Python 3.7, so the algorithm was tested on that version. Due to the size, used datasets are not attached but are available freely online.