A framework for malware analysis in a stand-alone email-server

Daniel Tolboe Handler

Kongens Lyngby 2017

Technical University of Denmark
Department of Applied Mathematics and Computer Science
Richard Petersens Plads, building 324,
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

Summary (English)

100,000,000,000 spam mails are sent and received every day. Even though most email clients are equipped with spam filters, the common user, still receives a severe amount of unwanted emails every day. The problem, to spam filters, is the fact that the user expects the filter to let every genuine email through. When spam filters lower the rate for false positives (genuine emails marked malicious) they increase the rate for false negatives (malicious emails marked genuine). This increases the need for user awareness, to ensure that he do not open any unwanted email.

This project proposes a solution, to which the user can forward an email marked genuine by the spam filter, but looks suspecious to the user. In return, the user receives an exhaustive analysis of the content of the email, whether the content is a link in the email or an attached file. The solution will be implemented as a framework written in Python on a stand-alone emailserver. The framework will include static and dynamic file analysis, passive and active link analysis.

Summary (Danish)

Der sendes og modtages 100.000.000.000 spammails hver dag. Selvom de fleste email klienter har indbygget spamfilter, modtager den almindelige bruger stadig en del spammails. Problemet for spamfiltrene er, at brugeren forventer at alle reelle emails for lov at komme igennem filteret. Når spamfilteret dermed formindsker antallet af falske positiver (reelle mails, klassificeret som ondsindet) forøges antallet af falske negativer (ondsindende emails, klassifieret som reelle). Dette øger behovet for brugerens opmærksomhed når han modtager emails, for at undgå at åbne ondsindende emails.

Dette projekt foreslår en løsning, til hvilken, brugeren kan videresende en mistænkelig email der er sluppet igennem spamfilteret. Brugeren vil dernæst modtage en grundig analyse af indholdet af email, hvad enten det er et link i emailen eller en vedhæftet fil. Løsningen vil blive implementeret som et framework på en selvstændig emailserver. Vores framework vil inkludere statisk og dynamisk filanalyse, samt passiv og aktiv linkanalyse.

Preface

This thesis was prepared at DTU Compute in partial fulfilment of the requirements for acquiring an M.Sc. in Engineering.

As sophisticated modern malware scanners have become, the more sophisticated the creators of the malware has developed, hence some malware is labelled safe by the scanner, but contains malicious code. This is partially due to the fact that the scanners embedded in anti-virus software or mail services has to find a balance between false positives and false negatives, such that no genuine emails are blocked. The aim of the project is to develop a stand-alone email server, to which the user can forward suspicious emails labelled safe by the automated malware scanner. The server will, in a closed environment, analyse links, attached files etc. and return the result to the user. Since the user has already received the suspected email and forwarded it to the service, because she found it suspect, there will be no need to either classify the email safe/unsafe, however the server should develop a more exhaustive description of the content of the email, without worrying about false positives or false negatives. The outcome of the project will be a mail server and an analysing environment. It will contain a framework for integrating scanners. It will be able to create a report to return to the user. Finally, the server will be evaluated.

Lyngby, 28-February-2017

Daniel Tolboe Handler
s113446

Acknowledgements

This project has been developed with support from my supervisor Christian Damsgaard Jensen, DTU Compute.

Contents

Chapter 1

Introduction

The exhaustive use of email as primarily communication form in our society today, makes the importance of working email clients clear. We send and receive several emails per day and has to consider every single one of them. The cyber criminals are as active as they have ever been, and use emails to attack. The sophisticated email clients we use today are equipped with anti-spam filters, to sort out these malicious emails. The filters will e.g. look into the emails for certain string patterns and hereby detect which emails are genuine and which emails are unwanted. Spam mails – named so, after the 1970 sketch by Monty Python – are defined by the Oxford Dictionary as "*Irrelevant or unsolicited messages sent over the Internet, typically to large numbers of users, for the purposes of advertising, phishing, spreading malware, etc.*" According to resent research, more than half of all emails received every day around the world can be classified as spam [AB17].
Throughout the report we will use the term *spam* as an umbrella term for any kind of unwanted emails. We divide *spam mails* into following three categories:

Junk mail Any unharmful – yet annoying - mail received be a user. The content is typically related to advertising.

Phishing mail Any mail send with the aim to harm the receiver. The content requires the receivers interaction to be malicious, e.g. opening a link or executing a file.

Auto-executable mail Any mail send with the aim to the receiver, which will release the malicious content without any user interaction needed. We will not handle this sort of mail in the project.

Spam mails represent an increasing problem, due to the amount of resources used to filter the spam mails from the genuine emails. According to [Fal03] spends 60% of email users more than 5 minutes per day to filter emails. The financial cost of spam mails is not determined exactly, however [Fal03] estimates the cost for American companies to be approximately 50$ per employee per year.

1.1 The false rate problem

As the main concern of anti-spam filters in mail clients is to protect the user against malware and phishing, they encounter a hurdle. The filter should not prevent any legitimate emails to arrive at the user. Hence the developer has to make a well considered threshold between denying as many malicious mails as possible, but allowing as many – preferably all – genuine mails as possible. Due to Danish law, all authorities in Denmark have to journalise all documents related to any proceeding, hence they have to receive and classify all incoming mail[1]. This is one of many reasons to ensuring all genuine mails are allowed through the spam filter.
For user experience reasons the developer wants the false positive rate as low as possible (ideally zero). But when lowering the false positive rate, the false negative rate will increase, which means that some spam mails will be labelled genuine, and accepted through the filter. This means the user has to be aware of the fact that not all emails in her mail box are to be trusted.
Figure 1.1 shows the relationship between the true positive, true negative, false positive and false negative rates, when the error rate is distributed equally. If this setup was adapted to the email filters, the filter would mark a significant amount of genuine mails as spam.

Figure 1.2 shows the relationship between true positive, true negative, false positive and false negative rates, when the error rate is pushed, such that there is zero error rate on false positives. It clearly shows that the amount of false negatives has increased, hence the user will receive more spam mails, labelled as genuine. However ideally the user will receive all genuine mails as well.

However not all users are skilled enough to decipher spam from genuine emails and our interaction with the state, bank, hospital etc. is heavily relying on

[1] In Danish: *Journaliseringspligt*

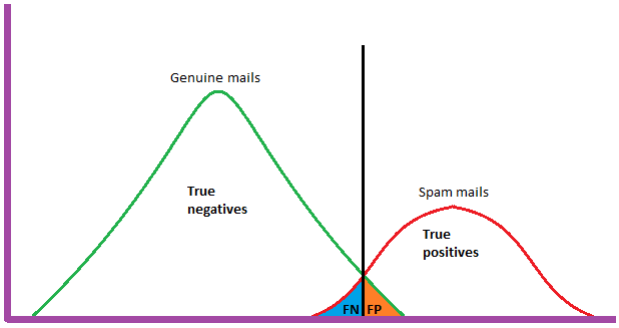

Figure 1.1: False positive and false negative rate, with equal distribution of false positives (orange area) and false negatives (blue area). The black line marks the analysis threshold.

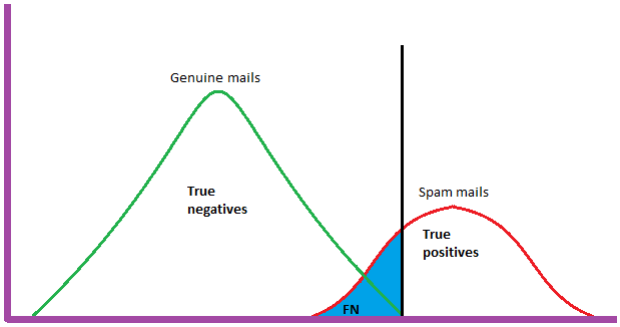

Figure 1.2: False positive and false negative rate, with zero error rate on false positives, however a larger amount of false negatives (blue area). The black line marks the analysis threshold.

digital communication. Phishing campaigns against nemID[2]-users are common and phishing mails pretending to be from the Danish tax authority or postal services are not unheard of.
In a perfect world, spam filters would sort out anything the user find irrelevant or malicious, however as the world is today, the users has to be aware of the content of the emails they are receiving in their inboxes.

[2]NemID is the official Danish digital signature, used by banks, governmental authorities and a number of organisations[Dig17]

1.2 A question of trust

The big issue of spam mails today, is the fact that the ordinary user has a hard time deciding whom to trust and whom not to trust. Even though the user knows that he cannot trust everything arriving in his email inbox, human curiosity and the fact that we still receive an increasing amount of official mails, that has to be opened, entails a risk of opening something malicious.
One thing is when the ordinary user receives an email, that claims his rich, non-existing, American uncle has died, and he has inherited millions of dollars. This would get most user alarmed, and they can then take necessary precautions (probably just deleting the email).
But most user will open an email, if it contains an invoice or similar, that might be genuine – and might risk the user to be of extra expense, if it is a genuine invoice, and is not paid or rejected in time. Rather recently a wave of ransomware attacks, stroke against a range of Danish companies[Dub17]. The attackers had bought .com-domains similar to the .dk-domain of a Danish carpenter company and used this to send of fake invoices. The mails was in perfect Danish and the only suspicious part was the .com-domain. This is a great example of how attackers can use common obligations to target the victims.

1.3 The project

To address the problems described in Sections 1.1 and 1.2, the aim of this project will be to design and develop an automated, user-friendly solution to investigate and help classify email, which the spam filter has marked as genuine, but the user suspects belongs in the blue area of Figure 1.2.
This gives two main challenges to solve:

User awareness The solution is based on the user to detect emails, which looks suspicious. If the user open every emails, without considering it might be spam, the solution will be useless. The Danish national broadcaster, DR, sent, in a internal test, 3000 fake phishing mails to the employees. Around 50% of the recipients open the mail[Boy16]. We get regular warnings from the government, tax authorities, the mail carriers, Nets[3] etc. about phishing campaigns, where the attackers try to imitate the genuine institutions. Many of these phishing campaigns are carried out poorly, such that the users quickly notice the suspect emails. However an increasing number of

[3] The Nordic digital payment administrator

campaigns are looking more like the original institutions, and this makes the need for user awareness grow.

Zero error rate Due to the fact that the user already has detected the suspicious email, we need to ensure that the solution makes a exhaustive analysis of the email, such that the user can rely on the answer. Whereas standard spam mail detectors has to take the false positive and false negative rates in consideration, the solution will merely focus on lowering the false negative rate, cf. Section 1.1. This gives the solution the advantage, that it doesn't have to classify the email *good* or *bad*, but merely graduate the maliciousness and help the user decide whether to trust the mail or not.

1.4 Contributions

The project will conclude with a product, that will handle the problems stated in Sections 1.1 and 1.3. The formal specifications of the product is listed in Table 1.1.

No.	Requirement	Description
01	Automated analysis	The solution should automatic receive emails, retract suspect content and analyse the content for malware or phishing. The examination should be exhaustive and rely on both static and dynamic analysis.
02	User friendly	The tool has to be user friendly. The user should do nothing more than send the suspicious email to the tool and receive the result of the examination back.
03	Reporting	The tool should give an easy-to-access report to the user, such that non-technical users will be able to decipher the result of the examination.

Table 1.1: Product requirements

The report will discuss the current state-of-the-art development in the relevant subjects of malware analysis and sandboxing in Chapter 2. The problem analysis

can be found in Chapter 3. The design and implementation of the automated tool is found in Chapters 4 and 5. Finally the product is evaluated in Section 6 and the report is concluded in Chapter 7.

Chapter 2

State of the art

In our digital life, emails are a very integrated part. However as the use of emails explode[RH11] (269,000,000,000 emails are sent every day) the amount of spam mails is increasing. The intention of spam mails can be everything from clickbait[PKSH16] to blackmailing the user or getting classified/sensitive informations. This chapter will examine some of the most common types of spam mails nowadays and which methods are used to limit the damages from these.
Malicious emails can roughly be split into two main groups, namely mails with malicious content attached, and mails with malicious content in the body of the email. The malicious content can be divided into two groups:

Code Malicious, executable code, that is installed on the victims computer. The code can be attached to the email as an innocent looking attached file or can be hosted at a webpage, to which the email has a link embedded. The behaviour, when opened, can be everything from encrypting the victims computer (ramsomware) to information gathering (spyware). We will discuss these types of malware and how to classify them, in Section 2.1.

Data The content of email will try to get the victim to send classified information to the sender, either by replying the email or following a link to a malicious webpage, where the victim is encouraged to share sensitive information, e.g. social security number, *nemID* or login credentials to

e.g. Google or Facebook. This type of email is harder to protect against, and depends heavily on user awareness to be discovered. The need for user awareness is further discussed in Section 3.1.

When analysing malware, two main approaches are used: Static analysis and dynamic analysis. These approaches will be reviewed in Sections 2.3 and 2.4. But first we will briefly review some of the current sort of malware.

2.1 Malware

A piece of software, which purpose is *intentionally* to harm the victim is known as *malware* – short for *malicious software*. We define the user receiving the malware as the *victim*, and the user developing and distributing the malware as the *attacker*. Malware is fractioned into several groups. This section will briefly review these by comparing propagation mechanisms and purposes of the malware:

Propagation mechanisms

Virus A virus is a piece of code, which cannot propagate on its own. So it has to add itself to other programs or operating systems to survive and spread. Viruses will normally try to spread to other hosts by using shared files, emails or network connections.

Worm A worm is quite similar to the virus, however it is capable of propagating independently. The worm will use file-transport mechanisms or network connections to infect other machines.

Trojan A trojan or a *trojan horse* is a piece of software that pretends to be innocent, but performs malicious activity in the background. Trojans include browser plugins, games etc. Unlike viruses and worms, the trojan will not try to spread to other machines by itself.

Purpose

Spyware This type of malware spies on the infected host, by sending information from the host to the attacker. This information can include sensitive information about the user, banking information, business secrets

etc. Spyware is commonly seen as trojans, i.e. they propagate as innocent looking software.

Bot A bot is a remotely controlled piece of malware. When infected, the host can be controlled to participate in a *botnet*. The attacker will use a *command-and-control* software to manage the botnet, which are used for spamming, Distributed Denial of Service-attacks[1] etc.

Ransomware Ransomware is a more recent type of malware. Ransomware encrypts the victims harddrive, and then ask for an amount of money (a *ransom*) to decrypt the files again. Depending on the time period from infection to the revelation of the ransomware, backups, fileservers and external harddrives might be infected as well.

Doxware Doxware is a new sort of malware, which, like ransomware, requires the victim to pay a ransom. But instead of encrypting the victims computer, doxware will use spyware-like mechanisms to retrieve sensitive files, and use these for blackmailing the victim[Ens17].

Scareware By using the victims anxiety, scareware will lure the victim to harm his own computer, e.g. by claim that the victim is infected, and get the victim to delete some files to remove the infection. Scareware is usually found online, where the user is presented with a pop-up window, which e.g. states the "You have been infected". This can be followed by a link to a trojan, pretending to be a malware scanner.

The amount of malware types and versions is extensive and increasing, and the discussed types are only a fraction of it. New editions of most malware types are discovered frequently, as the attackers develop new ways of attacking and infecting their victims.

2.2 Common infection vectors

To get a victim infected with malware, it usually requires the user to interact more or less direct with the attacker. This section will review the current state of some of the vectors, the attackers use to trick the victim.

[1] A Denial of Service-attack targeting a webpage, will typically sent more requests to the webpage than it can handle, and hereby disrupting the webpage. When *Distributed* the attack will have several sources to send the requests from[Wan06].

2.2.1 Web pages

Compromised webpage A webpage which is partly or fully controlled by an attacker is *compromised*. The attacker will either compromise the web server to add e.g. a Javascript-plugin to the webpage, using known browser vulnerabilities, redirect links or compromise via advertising[MC09] – only the imagination decides what the attacker might do. If the owner of the webpage is unable to detect the compromising, it is hard for the user to protect against.

Fake website In opposition to compromised webpages, fake webpages are designed to trick the victim. Fake webpages are widely used to lure login-credentials from user e.g. by pretending to be a genuine log-in interface from a wellknown service[AC09]. When the user has submitted his credentials, the fake webpage stores the credentials, and redirect the user onto the real service. This is hard to discover by the user and is a highly efficient way to collects credentials. The fake webpages usually uses URL that looks like the genuine (e.g. Google with an extra 'o' etc.).
The previous example of a compromised carpenter company in Section 1.2 is a recent example of an attack using a fake webpage.

2.2.2 Files

Microsoft Office Documents The Microsoft Office-suite is one of the most used software package. It includes software for making documents, spreadsheets, presentations etc. The Office-files are generally objects containers, which, apart from the standard content, can contain executable objects[LSS+07]. These executable objects are typically seen as macros. Macros are – in Microsofts products – a set of actions, usually used to handle data, and automate tasks done frequently. The macros lie in the data stream of the Microsoft Office-file.

Portable Document Format PDF-files are one of the most common filetypes today. The format was developed by Adobe in 1993[BCASMV93]. As with Microsoft Office files, the PDF format is capable of embedding objects. These objects are, by standard, listed in a cross-reference-table, in the beginning of the document, such that the PDF reader is capable of decoding every object correctly.

Compressed files File-compression is widely used, when transporting digital files on network, to decrease the amount of network traffic needed. File compression uses a set of different techniques to reduces the size of the files, e.g. replacing common strings in the original file with a shorter

identifier. When uncompressing, all identifiers will be replaces with the original string[Rob].

The importance of data stream and cross-reference table analysis will be discussed in the following sections.

2.3 Static analysis

Static analysis is where the malware is investigated as static data, hence without running the code. The analysis looks at the executable binaries, to locate malicious patterns or file abnormalities.
One of the most common static malware analysis methods, is to check whether the suspected file has been reported malicious before, known as *signature detection*[SJ05]. This method is used by standard anti-virus software, and is collected at web sites as *virustotal.com*. In practice this is done by computing the hash of the file[2], and looking it up in a database. This method is efficient to detect widespread viruses, but has severe limitations: The malware has to be detected and the classification of the malware has to be stored in the anti-virus developers database, before the signature analysis will trigger, hence new or improved pieces of malware will not be detected by the signature analysis.

2.3.1 *n*-gram analysis

The n-gram analysis method is based on a probability analysis method and was initially presented as a file type fingerprint algorithm. It was presented for malware analysis in 2004[KM04]. The method splits the data stream of the executable file into vectors of n bytes. By computing the average frequency of each of these vectors in the analysed file is it possible to make a classification – a fingerprint – which is merely linked to the filetype, say it is possible to differentiate file types, based on these classifications[NM14a, NM14b]. When a n-gram analysis is performed on a suspicious file, the classification of the analysed file is compared to the known classification of the filetype, hence it is possible to determine whether it is a genuine file or whether it contains non-standard data.

[2]The hash value of a file is computed with a hash-function. A hash-function is a oneway function that takes an arbitrary long input and computes a value of fixed size.

2.3.2 Embedded object analysis

Malware hidden in innocent-looking Microsoft Office-documents, as Word or Excel-files, are common in phishing campaigns targeting companies, disguising as job postings, bills or pay-checks. The malicious file will typically contain a macro which will automatically execute when the file is opened.
By statically analysing the data stream is it possible to extract the macro from the document, to detect how the macro will behave when running the file. If the user expects the macro to behave in a certain way, this can be compared to the data stream analysis, and if the macro will automatically execute or dump a file, it will look suspicious. Microsoft has, however, taken precautions against malicious macros: When opening a document containing a macro, the user will be asked to "Enable macros" before the macro is run. This can prevent automatic execution of macros in a document, the user expects to be without macros. This feature relies, again, on user awareness, and that the user knows the properties a macro *can* have.
Despite the good intentions, this is just another "click OK box", and even though it is better than nothing, it is not much better than nothing.

The Dridex trojan An example of a widespread malicious Microsoft Office-macro campaign is the Cridex/Dridex-trojan[OBr16], first seen in 2012 (Cridex) and then again in 2014 (now renamed to Dridex). The Word-documents was spread in several phishing campaings; when opening the document, a self-executing macro would dump a file on the computer, which would install the trojan. The trojan would add the infected computer to a botnet and begin to harvest banking information, and begin spreading to other computers via network and USB-devices. When peaking, the trojan infected 16,000 machines a month.

While the Microsoft Office applications try to enable some sort of macro security, by blocking automated macro execution, the PDF-format is still vulnerable, due to its open format, and huge number of readers on the market. One of the biggest threat when handling PDF-files is the fact that they can contain executable Javascript-code[TSPM11]. Like Microsoft, Adobe has tried to disable automatic macro execution in their PDF-readers. However as stated above, Adobe is not the only developer of PDF-readers. In addition to this many PDF readers tries to evaluate and show non-standard content of the PDF-files. This "extra service" is often exploited by malware. By comparing the cross-reference table to a static analysis of the objects which are actually found in the file, it is possible to determine whether the file contains hidden objects[TSPM11]. Hidden objects itself is not necessarily malicious, however it should raise some

sort of concern at the user that the document tries to hide information from the PDF-reader.

2.4 Dynamic analysis

Analysing the behaviour when a certain piece of software is run, is known as *dynamic analysis*. This gives a good picture of what sort of malware the investigator is dealing with. However doing this on an open environment will give the malware opportunity to spread or sending data, which supposedly is not the intention of the investigator, hence protected execution environments, where the code is confined and controlled are increasing.
When analysing malware dynamically the investigator will look at certain aspects of the environment in which, the malware is run:

Network traffic By monitoring the network traffic the investigator is capable of analysing if the malware send information about the environment or the user to the internet. In addition it will be clear if the malware downloads more files and install them on the computer.

Processes Monitoring the processes on the computer will give an idea of what changes the malware makes on the infected machine.

Hard drive Monitoring changes on the hard drive will express whether the malware writes to the disc.

As stated earlier, the dynamic analysis of a suspected piece of malware cannot be performed in an open environment, hence the development and research in sandboxes are increasing.

2.4.1 Sandboxing

The concept of sandboxing is when a file is executed in a closed, virtual environment, that appears – to the software – as an ordinary execution environment. This gives the investigator a huge advantage, because he is able to run and observe the malware, and therefore is capable of analysing how the malware is behaving in the environment. By monitoring network traffic, changes in the memory and to the hard drive, the investigator can get a comprehensive picture of how the malware behaves.
The concept of *sandboxing* has been researched heavily the last decade and the

research can be divided into two main groups: Sandboxes isolated totally from the internet and sandboxes with an internet connection [YIM+09].
When using sandboxes with no internet connectivity the risk of running the malware is fairly low. However current forms of malware, like ransomware, botnets and spyware requires an open internet connection to work, hence if no available internet connection is present, the malware will presumably just hibernate until a connection opens, which means the investigator will get nothing from the analysis.
On the other hand, a sandbox with an open internet connection can be fairly tricky as well. When running the malware it is important not to spread the malware to real machines, hence a controlled internet connection is required.

2.4.1.1 Malware detects the sandbox

The problem with sandboxing is that malware developers begins to introduce sandbox detectors in the source code of the malware[CAM+08]. By detecting the CPUID opcode[3] the malware can get an idea of whatever the execution environment is a virtual environment or a real physical machine and refrain from malicious activities if a virtual environment is detected. In addition to this, most virtual environments are dynamically allocating the "physical" memory, i.e. only uses what is required at the time. This means that most virtual environments physical memory is much lower than for a real computer. This is another factor that malware can use to determine if it is executed in a sandbox environment.

2.5 Forensic analysis of IP address

Forensic analysis of an IP address can be split into two main parts: Passive analysis and active analysis. The passive analysis is performed by retrieving known information about the IP address, by making queries to databases etc. The active analysis performs a more direct retraction of the webpage, to locate hidden objects, redirections, etc.

[3] The opcode can be used by software to determine some details of the CPU, e.g. the manifacturer

2.5.1 Passive analysis

Many known organisations keep track of online activity and stores the information, such that it is possible to query the information from their databases. These organisations include Google, IBM, VirusTotal and Bluecoat, to mention a few. The databases of these organisations can be queried for information about registrants, geographical location, passive DNS' etc of IP addresses. Furthermore keep many of these databases track of malicious activity, such that it is possible to learn, if a given IP address has participated in malicious activity before.

The passive analysis of a given IP address or domain will retrieve as many of these information as possible to resemble a picture of the IP address, without making direct contact to the IP address or domain, hence the designation *passive*.

2.5.2 Active analysis

The active analysis requires direct interaction with the IP address or domain being analysed. This includes downloading the content of the webpage to make a analysis of objects on the page, or making a portscan of the IP, to determine if any TCP or UDP ports are open. The latter is mostly used by attackers, to locate any possible ways of accessing the webserver behind the IP address.

Content analysis of the webpage will determine whether the webpage contains Javascript, used to e.g. drop a file on the visitors computer, or using a vulnerability in the users browser.

2.6 Summary of *State of the Art*

The section has discussed the current state of the art in the subjects of malware analysis and IP address forensic.
For malware analysis the section has discussed static analysis, where object analysis and n-gram analysis has been review and discussed, in addition dynamic analysis has been discussed in relation to sandboxing.
For IP addresses we have discussed how passive analysis can retract information of registrant, geographical location etc. from online databases. As supplement

we have discussed how active analysis can retract malicious activity from a given webpage.
The number of methods for malware and IP address analysis is huge and the section has only discussed a limited set. The problem is still that the standard user is not technical capable of performing the analyses, and will rely on automated tools if he has to conduct such an analysis.

Chapter 3

Analysis

Many users relies 100% on their anti-virus software, however not all malicious files are spotted by the anti-virus software (and not all anti-virus software are good enough to keep track of the development of malware).
This means, that when the anti-virus software has labelled an email, attached file or webpage valid, most users trust the mail/file/webpage and might not be watchful enough, and thereby risk a digital infection.

As stated in Chapter 1, the big challenge for anti-virus software is that it has to label the suspected item either good or bad. To ensure a near-to-zero false negative rate, it has to compromise with a significant false positive rate, hence some malware will be falsely be labelled genuine. This gives the user some responsibility, to make a second-line opinion, however many users are not technically capable of doing that, from the information they get from the anti-virus software. So even if the user suspects an email of being malicious, he is not capable of acting on the suspicion.
Another problem is the fact that the anti-virus software does not know what files the user expects to get. A PDF-file with embedded Javascript will automatically trigger most anti-virus solutions and be labelled suspicious, however if the user is using PDF-files with embedded Javascript, it is a problem that the anti-virus software is declaring it *bad*.

3.1 User awareness

One of the general problem with IT security today is the evolution of the digital world is processing in a pace, that the common user cannot keep up with. Due to this, the user is relying – to much – on automated solutions. And even so the anti-virus software developers fight to keep up with the bad guys, the malware is almost always a step ahead.

It is assessed that between 50% and 75% of incidents regarding cyber security in the industry originates from users inside the organisation[DHG09]. Even if we sort out angry employees deliberately trying to harm the organisation, it is still a significant number of incidents that might be non-existing or insignificant if user awareness is increased. This section will analyse how to help the user, to be able to make a determination whether a received email is harmful or not, and hereby decrease the number of security incidents.

3.1.1 Phishing

As mentioned in Chapter 1 the common users are targeted in phishing campaigns. Phishing attacks consists of three main elements[Hon12]:

01 Fake email The first interaction between the user and the attacker is the email. The attacker will try to make the email look as genuine as possible. The subject can be e.g. be a password reset on a well known service (Google) or topless pictures of a celebrity. The aim of the content is to lure the victim either to go to a webpage or to open a file.

02 Malicious content The *trap* is usually a webpage or a file. In the case of a webpage, the attacker will have to make the website look as genuine as possible. This is achieved by using well-known logos and URLs which look like the real one.
Alternatively the user is lured into opening a file. This file will – like the webpage – look genuine (e.g. a job posting or a paycheck), but contain malicious content.

03 Information harvest The last part of the attack is to harvest the information from the victim. This can be done on the fake webpage by luring the victim to enter his credentials to a known service (Google, Facebook or online bank). Or if the user has opened a malicious file, it can dump a piece of malware on the victims computer and harvest the victims information.

Phishing campaigns target users on both private matters as banking or NemID, or corporate matters as salary or job promotions. These campaigns are of various quality and the relevant authorities in Denmark are frequently reminding users to be aware of phishing. However even the most aware user, can be fooled is the phishing mail is looking genuine, are in perfect Danish (which is rarely the case) and links to a genuine looking webpage with a genuine looking URL. In this case the user has to make a forensic analysis of the URL if he is to determine the genuineness of the mail. Most user are not capable of making such an analysis, so the user need a tool for quickly analysing the link, to determine whether it is an IP owned by the apparent sender or it is located in a suspicious location, like Russia or Taiwan. This piece of information would help determine the genuineness of the link.

Spear phishing

Spear phishing is targeted phishing campaigns against specific users, where the attackers research the on victims to make the fake emails more believable[Par12]. The development and propagation of social media has resulted in easier access to personal information about users, which can be used to trick them[Hon12]. An example could be a father receiving an email, which appears to come from the daughters handball association. This looks innocent, however it is from an attacker, who found out about the handball association from a set of pictures on the fathers profile on Facebook[Had11].
Spear phishing is an increasing problem, and users need to be aware of what attackers can used content shared on the social media for. It is hard to protect against the phishing mail itself, however as with normal phishing campaigns, a analysis of the genuineness of the link and web page will help the user to ensure no personal information is given.

3.1.2 Linked software

By linked software download, some software distributors get the users to install more software than the user intended. An annoying – however not malicious – example of this was in 2015, when Java updates included the Ask toolbar[Kei13]. A quick fix to avoid linked software, especially in companies, is to deny downloads at all in the firewall. This is standard procedure in many companies and works well. In addition to this anti-virus will usually scan all downloaded files and warn the user, if the file is known to be malicious.

3.1.3 Drive-by downloads

Unwanted software from webpages is a big problem. And the problem stretches out of the users hands.
The concept of drive-by download, is when a webpage silently dumps malware on the victims computer, while the victim visits the webpage[EKK09]. This concept is hard to contain just by raising the user awareness. Most drive-by downloads are using Javascript to complete the activity. The Javascript can end up on the webpage in two ways:

Embedded in the webpage Malicious webpages, with the only purpose to install the malware on the victims computers. These webpages could be part in a phishing campaign, see Section 3.1.1 or have URL that looks like genuine, well-known URLs, but with a little change. This sort of drive-by download can be handled by increasing the user awareness.
Another way to embed malicious content is to compromise a genuine webpage, using e.g. a vulnerability on the webserver, and plant a piece of malware at the server. This method is hard to protect against.

Embedded in advertising Many webpages are using advertising to raise some money. However these adds can contain malicious code, and this can be hard to avoid. Popular webpages like Facebook has been victims of this malware in advertising[Con11] (known as *malvertising*). No matter how much users raise their awareness, malvertising is impossible to avoid, and the user will have to rely on the security in the browsers and anti-virus software to catch the malware, before it is dumped on the computer

The most efficient counteraction to drive-by-downloads is to ensure that browsers and other software on the users computer is up-to-date.

3.1.4 Watering hole

A recent threat vector is the *watering hole attack*. This sort of attack is a combination of spear phishing and drive-by-download. The attacker will identify a third party web sites, their victims are likely to visit. The attacker will then compromise the webpage, e.g. using vulnerabilities in browsers or similar, and then just wait for the victim to hit the webpage[CDH14].
This attack vector is primarily targeting organisations, where the attacker can ensure that at some point, someone in relation to the organisation will visit the compromised webpage[Azi13]. This makes it hard to protect against and the

protection must rely on updated browsers, with few vulnerabilities and sufficient anti virus software[Kin13].

3.2 Forensic analysis

Raising user awareness is probably the best way of addressing the problem of malware. Users are in general described as *the weakest link* when it comes to security. However just blaming the everyday user will not raise security[SBW01]. Awareness training is a necessary way of addressing the security issues. Combined with some sort of password requirements, and a solid anti-virus software we have come a long way. The problem is what happens when these countermeasures fail, which they surely will.
The common user is still not equipped with tools to help him analyse suspicious content, when a file is labelled genuine by the anti-virus software.
The tool will have to rely on the users awareness, hence it should perform the analysis the user is not capable of making himself. The result of the analysis should be presented to the users, such that the user is capable of deciding whether the suspicious content is malicious or not.

The contribution of this project is to make a tool to help the user make a forensics analysis of the suspicious content, he has received. A graphical representation of the analysis is found in Figure 3.1.
The first problem to solve, is to decide whether the suspicious content is received as attached code or a link in the body of the email. The result of this part of the analysis will determine how the rest of the analysis will be performed. If the content is attached, the tool will have to make a analysis using the methods described in Section 2.3 and 2.4. If the content is a link, the tool will have to make a forensics analysis of the domain and IP address of the link, as discussed in Section 2.5. This gives the tool two main "analysis legs", with different analysis methods.

3.2.1 Suspicious file

If the suspicious content of the mail is an attached file, the file will have to be analysed statical and dynamically such that any malicious activity will be discovered.

01 File type The type of the file will have to be determined:

Firstly to ensure that the file extension match the actual file type. If this is not the case, the user should be warned.
Secondly the second part of the analysis will be determined by the file type.
Thirdly if the file is a compressed file, the analysis will have to include an uncompression, and then a full analysis of the uncompressed file(s).

02 Embedded objects If the file is one of the file types, which can contain objects, these objects will have to be extracted an analysed, such that any malicious or suspicious activity can be found. The analysis will have to include the full behaviour of the embedded objects, such that the user can differentiate between genuine objects, that he expects and malicious objects, that he does not expect.

03 Meta-information Some meta-information about the document will be helpful to the user. Helpful information includes author, number of pages, first revision/creation date, last revision and so on. If the document states to be a report, and it only contains of one page, the user should found it suspicious.

04 Known malicious file If the file previously has been reported malicious, the user should be warned of two reasons: If the same file has actually been received by many people, and it seems to be sent only to him, it seems suspicious.
If other analyses has declared the file malicious, the probability that the file is malicious is quite high.

05 Behaviour when executed The analysis will have to conclude with a behavioural analysis of what happens with the environment the file is executed in. The user will have to need if the file dumps other files, makes network traffic, changes system files etc.

The first four parts are using static analysis methods, whereas the fifth part is using dynamic analysis methods.

3.2.2 Suspicious link

The other leg of the tools analysis is performed if the content is a link to a webpage. The analysis will have to help the user determine whether the link is corresponds to the apparent sender of the email. This can be done by making an analysis of the IP address behind the link.

01 Registrant of IP By investigating who is the owner of the IP (and the domain), we can help the user assess the genuineness of the webpage. E.g. if the mail claims to be from the Danish tax authorities, Skat, and contains a link, which is not registered by Skat, it will make the mail look suspicious, and the user should be warned. Every IP address is linked to a registrant, and this information is public available. It might be relevant, as well, to know how long the given registrant has been registered to the IP address

02 Geographical location of IP In supplement to the registrant of the IP, the geographical location of the IP can help the user decide wether to trust or not trust a webpage. If the apparent sender of the mail is a Danish organisation or authority, the probability that the IP is hosted in a East European or Asian country is tiny, hence the user should be warned if this is the case.

03 Known malicious activity If the domain or IP is taking or has taken part in malicious activities, it will increase the probability that the webpage is non-genuine and the user should be warned.

04 Content of the webpage The content itself of the webpage should be analysed to determine if any hidden scripts or redirections is present. If the page redirects to another webpage, the analysis discussed above should be carried out on the redirected page, such that the user is not lured onto a malicious webpage by redirecting. If the webpage contains Javascript hidden or not hidden, the script should be analysed, to determine whether it is malicious.

The first three parts of the link analysis are using passive analysis methods, and relies on earlier submitted data found in public databases. The fourth part of the analysis is using active methods, and will partly be similar to the file analysis leg.

3.3 Summary of *Analysis*

The section has discussed the necessity for increased user awareness if the rate of successful malware attacks has to be decreased. We have discussed how phishing campaigns in various ways try to trick the victim to install malware or disclose sensitive information, and how compromised or fake webpages, is a threat as well.
The chapter concludes with a abstract description of a forensic tool, that is capable of making an analysis of the malware or suspicious webpage, such that the

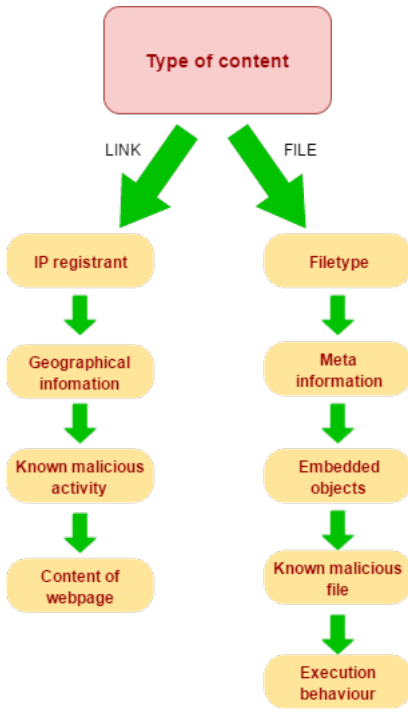

Figure 3.1: Flowchart of the analysis

user, given the information from the tool, can determine whether the malware or website is to be trusted or not.

Chapter 4

Design

This section will describe and discuss the design choices made in the development of the product described in Sections 1.4 and 3.2.
The goal of the tool, will be to present a service to the user, where malicious emails, not captured by the protective mechanisms (cf. Chapter 1), can be forwarded and exhaustively analysed. The result of this analysis should be presented to the user, without to many technical terms, such that a non-technical user can decipher the result, and take action based on it. This chapter will describe the developing process of the product, from the design phase, through the implementation phase. The evaluation of the product is described in Chapter 6.

4.1 The environment

The environment of the tool will be either:

Plugin to an email client This solution will rely on existing email clients. The most widespread email clients for desktop and laptop computers are Microsofts Outlook and Apples Mail[Lit17], and it would be obvious to make the tool to either one or both (presumably Outlook, since it is the most common client in organisations). The advantage of making the tool

embedded in the email client is the user experience – if the user is in a known environment, it will be easier to use. The disadvantage is the constant development of the email clients. Especially Outlook is undergoing a big change, when Microsoft is pushing their online Office-package – Office 365 – onto the market. This implies that the tool would have to be updated in the same pace that Microsoft Office is updated. Additionally would it require that we would integrate a sandbox environment in Outlook, for the dynamic part of the file analysis. This might be challenging.

Stand-alone email server This solution will rely on a dedicated email server linked to an analysing environment. This solution is not relying on the environment of a specific email client, hence it is more independent, which is a great developmental advantage – and we do not need to choose a platform on which the solution has to work on. The disadvantage is of course, that it requires a dedicated email server, which is hard to find in a normal household. The solution to this could be to make it possible to set up in a virtual environment, however it still requires more from the user. In larger organisations this should – however – not be a problem.

We have chosen to go with a stand-alone server. The biggest reason for this is the independence from the email client providers. As stated above the solution will rely on user to set up the server, and since it will be easier in organisations with dedicated IT departments, some of the future design choices will be taken according to this. The whole setup will be developed in a virtual network, which consists of a router with a DNS-server, a mailserver and a client-machine. The network will be connected to the real life internet through the virtual router.

4.2 The front-end

The front-end and user interaction will be fairly simple. The user will have to forward the suspicious email to the server, including any and all attachments, and will receive a report with the result of the analysis in a return email. This only requires a specific email address to the server, which the user is provided with, when the server is installed at the site. Hence no graphical user interface or similar is required.

4.3 The back-end

The back-end includes the email server and the analysis framework.

The email server should be fairly simple, since its only purpose is to receive and read the suspicious email. From here the framework will analyse the email, to detect attached files or links in the email.
When the analysis is completed, the email server will return a result report to the user. As stated earlier, the importance of making an understandable result to the user must be addressed.

The framework will have to include both automated static and dynamic file analysis, and will integrate a selection of malware analysis tools. Since the majority of organisations uses Microsoft Windows and our setup primarily is targeting organisations, the framework will to a great extent analyse for this type of malware. In addition it will use tools like VirusTotal[1] and Malwr[2] and a build-in Linux AntiVirus, which to great extend will be helpful to all kind of malware. The framework will also have to include tools for IP analysis, such that suspicious links can be investigated.
The initial part of the analysis will be to determine whether the suspicious content of the mail is an attached file or a link in the content of the mail.

File analysis

If the analysis addresses an attached file, the first part of the file analysis will be to determine which file type is addressed:

Microsoft Office-file If the file is a Office file (Word, Excel, PowerPoint etc) the tool has to search the document for macros, since macros is the biggest threat in malicious Office-files, cf. Section 2.3.2. If the file contains macros, we will make an exhaustive analysis of the behaviour of the macros. The behaviour analysis will be added to the report, which will be returned to the user. Furthermore will we run the file through VirusTotal and Malwr. The file will be dynamically analysed in the sandbox, which will execute the file in a Windows environment. Finally we will check the file in the anti-virus software embedded in the mailserver for a final check.

Portable Document Format, PDF If the file is of PDF-format the tool has to search for embedded objects, and mismatch of the cross-reference table, cf. Section 2.3.2. If embedded objects are found, the behaviour of these will have to be analysed in the sandbox environment and presented to the user. Like the Microsoft Office-files, we will check the file on VirusTotal, Malwr, in the anti-virus software and in the sandbox.

[1] www.virustotal.com
[2] www.malwr.com

Compressed files Compressed files, like ZIP or TAR files has to be uncompressed, when this is done the analysis will run over again, to check which file types was in the compressed archive. The uncompression part will have to handle recursively compressed files as well.

Other files Other file types will be hard to make a specific static analysis on. The files will be run through VirusTotal, Malwr, in the anti-virus software. They will be tested in the sandbox as well.

Link analysis

If the analysis addresses a link in the email, the analysis will have to determine which IP address the domain is hosted at. When the IP address is determined, a forensics analysis of the IP address is executed. The first part of the analysis is a passive analysis, where the framework will access known databases to retract information about the IP address:

Registrant The registrant linked to the IP address will have to be included in the report to the user, together with the first registration date of the IP address. At some hostsites is it possible to pay for anonymity, hence we cannot ensure that the registrant is revealed.

Geographical location As with the registrant, the IP address' geographical location is public available in online databases. These databased must be visited by the analysis, to harvest this information.

Malicious activity Plenty online databased, e.g. Google and IBM, store data about malicious activity linked to IP address. The final part of the passive analysis will collect information about the history of the IP address, relative to previous malicious activity.

The active part of the link analysis is to download the content of the webpage and analyse this:

Redirecting Is the webpage redirecting the user to another webpage? If this is the case, the user should be notified, and the full link analysis should be applied recursively to the new webpage.

Malicious content If the webpage is hiding content, e.g. Javascript, this content will have to be extracted and analysed.

4.4 Result and reporting

The flow of the analysis can be seen in Figure 4.1.

When the analysis is completed the result will have to be sent back to the user on the same email address the user used to forward the email from.
The report to the user will have to include the complete analysis and a abstract of it. The abstract will be the content of the return email and the full report will be attached to the email, such that the user can see the exhaustive analysis if the user wants to.

The result has to be presented in such a way, that the user can use it for comparing the behaviour of the suspicious content with his expectations, e.g. you do not expect a pay-check to automatically execute a macro and dump a file, or a web page from the Danish tax authorities to be hosted in Russia, hence this information must be presented to the user in a easy-to-read and easy-to-understand sort of way.
This gives a merely abstract challenge of deciding what the user expects the content to be. The analysis will have to return data, such that the user can compare the expectation to reality, and hereby deciding to trust or not to trust the mail.

4.5 Summary of *Design*

The section has taken the abstract description of the forensics tool, discussed in Section 3.2 and have developed an overall architecture for a framework, which solves the challenges found in Chapter 3. The framework will be integrated in a stand-alone emailserver, to which the user can forward a suspicious mail and receive an exhaustive analysis in return. The framework will be able to handle suspicious link, and a wide range of suspicious files.

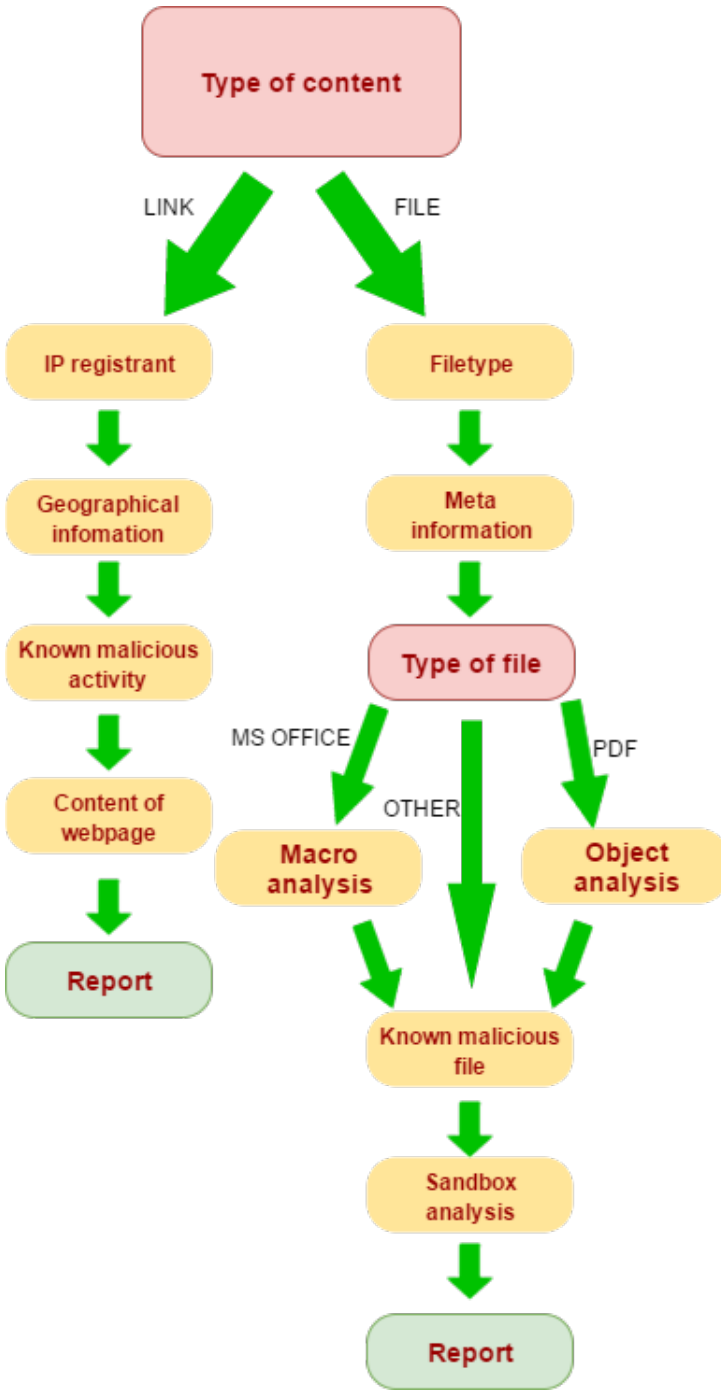

Figure 4.1: Flowchart of the framework design

Chapter 5

Implementation

This section will describe the implementation of the email server and the framework used for malware analysis.

5.1 Operating system

The email server and analysis framework is developed and installed in a Ubuntu version 12.04. Ubuntu is chosen due to its open source nature and diversity, such that both mail server and the analysis framework can run without complications. The virtual network we work in is presented in 5.1.

5.2 Mail server

The email server is set up using Postfix and MySQL. This make a very simple and useful database, that fulfils the requirement. A single user account is setup (`daniel@mailclient.example.com` in our environment). The database will have to contain:

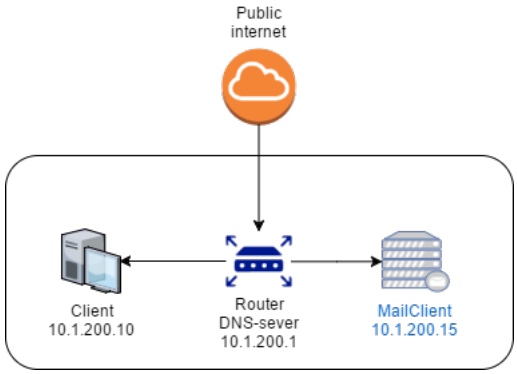

Figure 5.1: Overview of virtual network. The framework will is developed on the `MailClient`-machine.

- The forwarded email
- The attachment (if any)
- The email address of the sender

Since we don not store any sensitive information and due to the scope of the project, we will not consider securing the email database. However MySQL has a protective mechanism, which is used as standard, and we will use it. But we do not encrypt content or mail addresses etc. When the analysis is done and the report is returned to the user, the email and all associated information should be deleted. A future implementation could contain a cache-mechanism such that if two users receive and forward the same mail only the first one will be analysed – the next one will just receive the first analysis. This could be helpful if the mail server is used in a organisation where phishing campaigns targets several users, such that the mailserver will not be overrun be requests of the same malicious mail. However in the current implementation all relevant data are deleted when the report has been sent.

5.3 Framework

The framework is written in Python. Python is chosen due to its very dynamic nature and the fact that Python is easily installed in most Linux-distributions. A lot of the tools chosen for the framework (see Sections 5.3.1 and 5.3.2) are written in Python as well, and it makes it more cooperative to work with. The

framework will include a various selection of software analysis tools and will parse the output of the various tools, into a single result report.

This section review a complete list of the tools used in the framework, both for static analysis and dynamic file analysis and for link/webpage analysis. A graphical representation of the framework can be found in Figure 5.2. The implementation of the framework follows the design described in Section 4.3. The first thing for the framework is to analyse whether it is dealing with a attached file or an embedded link. The implementation of the file analysis is documented in Section 5.3.1. The documentation of the link analyser implementation is found in Section 5.3.2.

5.3.1 File analysis

When the file has been retracted from the email, the framework will begin the analysis of the file, following the design discussed in Section 4.3.

5.3.1.1 File type analysis

First part of the framework will analyse the type of file, by simply checking the file-extension. The next part will verify the file type, by using static analysis methods. We have accessed a selection of pre-existing tools, for this:

TrID TrID is a file type analysing tool, developed by Marco Pontello[Pon03]. TrID uses the files binary signature to determine which file type it is. The file is analysed using the n-gram described in Section 2.3.1 and compare the result to a database of 10,000 file types.

Tika Tika is another file type analysing tool[Apa10]. Tika is developed by Apache and uses a combination of metadata extraction and binary signature detection. The result of the analysis is compared to the Tika database which consists of more than 1,000 file types.

The two output from the two tools are too similar to include both tools in the implementation, hence we will only implement TrID. TrID is merely chosen due to the fact that it is Python-based, which makes the integration with the framework smoother. Additionally is the output from Tika more extensive, yet gives same amount of relevant information as TrID. This means that if the

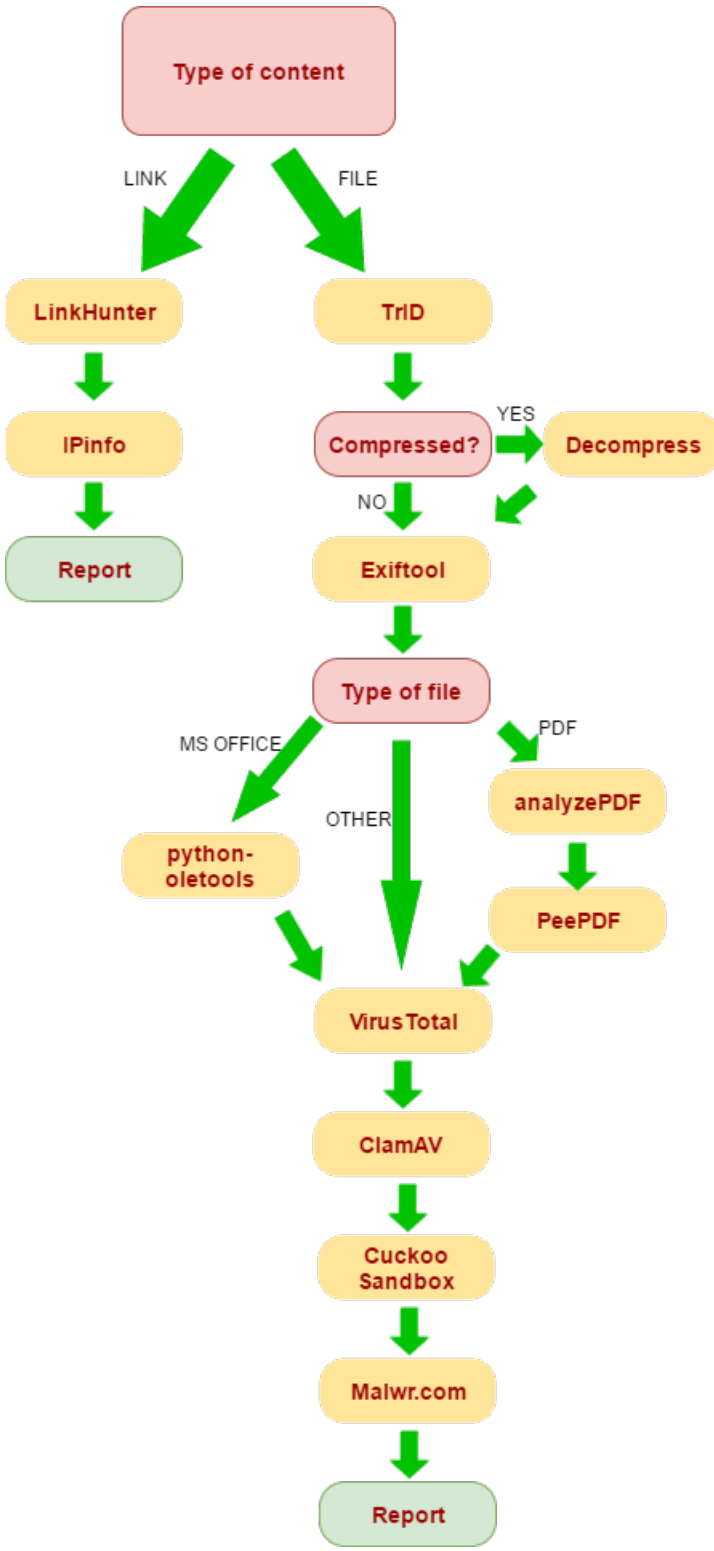

Figure 5.2: Flowchart of the framework implementation

implemented Tika, the framework would have to do a lot of sorting of the output, without getting more relevant data.
The result of the TrID analysis will be compared to the file-extension. If we have a notable difference the user will be notified in the report.

5.3.1.2 Meta data extraction

When the file type has been determined, we will extract as much meta data from the file as possible. Again two tools have been considered for this:

Tika Apart from file type analysis, Tika can be used for metadata extraction.

Exiftool Exiftool is an application for reading, writing and creating meta information in a wide variation of file types. It was developed by Phil Harvey at Queens University in Canada[Har13].
Since we are only analysing files, we are only interested in the reading part of the application, hence Exiftool will analyse the meta-information of the suspicious file. All information found is parsed on to the report.

Due to the fact that we have already dismissed Tika once, and the extensive opportunities in Exiftool (in a future update of the framework), we will use the analysis from Exiftool.
All relevant metadata from the analysis is added to the report.

5.3.1.3 Macro analysis

The macro analysis will be carried out if the file-in-analysis is a Microsoft Office file, either by file-extension or by the TrID-analysis. There are plenty existing tools for macro analysis. We have decided on implementing a selection of tools, all from Python-oletools.
Python-oletools is a set of tools, which are designed to analyse the embedded objects in Microsoft Office files, developed in 2012 by Philippe Lagadec[Lag13].
The tools from the set considered in the implementation are:

OleID decides whether the document is OLE-formatted. If this is the case, the rest of the OLE-analysis is carried out.

OLEdir analyses and displays all directory entries in the OLE-formatted document. The analysis consists of links between the entries and size of the individual entries.

MRaptor is a tool to extract and analyse primarily malicious macros. The outcome of MRaptor is a list of all found macros, and a quick analysis of whether MRaptor finds the macros suspicious.

OLEmap retracts all sectors of the OLE-file.

OLEmeta retracts all standard properties, which are found in the OLE-formatted file, such as information about author, template, number of pages etc. This information is parsed to the user.

OLEtimes collects all timestamps in the document. Timestamps include modification and creation time of the document itself and all the embedded objects in the document.

OLEvba can extract macros in cleartext. This can be used for detecting keywords in the macros, that indicates malicious activity, such as auto-execution or file dump. The result is parsed on to the user.

pyxswf Detects and analyses Flash objects in the document.

The implemention will include the following of the oletools:

- OleID
- MRaptor
- OLEmeta
- OLEtimes
- OLEvba
- pyxswf

Hence OLEdir and OLEmap are not included in the analysis, due to overlapping to much with the rest of the tools. The results of these tool, will overlap as well, hence the parsing of the output will have to take this into account, such that the user don't receive five more or less equal analyses.
The macro analysis will be added to the report, such that the user gets an idea of the behaviour when opening the document. This will make him capable of determine if the document acts as he expect or has unexpected behaviour.

5.3.1.4 Object analysis

The object analysis will be carried out if the file-extension analysis or the TrID analysis declares the file a PDF-file. The following tools has been considered for the analysis:

AnalyzePDF AnalyzePDF is a python script, developed by HiddenIllussions[Ill13]. that reviews the cross reference table of a PDF-file and checks whether the PDF contains Javascript not stated in the cross reference table. If any Javascript is found in the PDF-file, AnalyzePDF will analyse the script, and access whether it seems malicious or not. We will pass this analyse on to the user.

PeePDF PeePDF analyses all objects of PDF-files, and makes and assessment of their validity. PeePDF is developed by Jose Esparza and is written in Python[Esp11]. The output of PeePDF is a list of objects, and the behaviour of the object. As with AnalyzePDF, PeePDF assess the objects and points out malicious objects or behaviour. If a known vulnerability is found (e.g. vulnerabilities stored in the CVE-database), it will return the CVE-indicator, to further analysis.

Origami-pdf Origami-pdf is a open source, analysing framework for PDF-files. It is based in Ruby, and is capable of analysing, modifying and creation of PDF-files. It detects embedded objects, and gives a short analysis behaviour of the object.

pdf-parser The pdf-parser is a Python-based analysing tool, developed by Didier Stevens[Ste]. The tool gives an exhaustive analysis of all objects in the document, describing, amongst other things, size, behaviour and links.

The output from the PDF analysing tools are quite similar, and we will only implement AnalyzePDF and PeePDF in the framework. Origimi-pdf gives almost same result as PeePDF and the output from pdf-parser included to much unuseful information, which had to be sorted out, before sending it to the user. The analyses from AnalyzePDF and PeePDF are merged and sent to the user.

5.3.1.5 Known malicious file

To determine whether the given file previously has been classified malicious, we will implement a anti-virus engine. Two different approaches has been considered for the framework:

VirusTotal API As stated earlier, VirusTotal is a online tool[Tot12], where suspicious files, URL or IP addresses can be uploaded and analysed be a wide range of anti-virus engines, hence it is possible to see how many of the most popular anti-virus software tricker on a certain file. The framework will use the API of VirusTotal to upload the file. If the file has been scanned before by VirusTotal, we will receive the result of that scan. Else we will allow VirusTotal to scan the file and retract the result. The main part of the result is the number of anti-virus engines that declares the file *bad*.

ClamAV ClamAV is a open source anti-virus engine which run on all the major operating systems[Koj04]. The result of the analysis in ClamAV will be presented to the user. Since signature detection, cf. Section 2.4, is a major part of the anti-virus softwares analysis, we will have to ensure regular update of the ClamAV database.

Both approaches will be implemented in the framework, even though ClamAV is a integrated part of VirusTotal, however when running the file through VirusTotal, we only get to know if the anti-virus software classifies it *good* or *bad*. By running it through ClamAV, we get a more exhaustive analysis of the file.

5.3.1.6 Behavioural analysis

The behavioural analysis will be performed exclusively in sandbox environment. We have considered to approaches for the framework.

Cuckoo sandbox Cuckoo sandbox, is a sandbox environment developed and maintained by the Cuckoo Foundation[GTBS12]. The sandbox contains a malware analysis system, where virtualised environments of the most popular operating systems can be run (including mobile operating systems as Android). It is possible to run a malicious file in the sandbox, and get information about behaviour, network traffic, memory analysis etc.
The malicious file is run through the sandbox, and the analysis is added to the report to the user.

Malwr.com API Malwr.com is a online version of the Cuckoo sandbox, which is developed and maintained by the Cuckoo Foundation. As with VirusTotal, if the file has been analysed before, we receive the result of the previously analysis. If this is not the case we upload the file to Malwr.com and receive the fresh analysis.

As stated both solutions use Cuckoo Sandbox. Since we already have used our offline edition, this part of the analysis is only second line analysis. The challenge of the Cuckoo Sandbox is the fact that each operating system must have it own virtual machine. This means, that if a specific piece of malware uses a vulnerability on a very specific, patched edition of Windows, and we are not running it through this specific Windows-edition, the analysis will be useless. Hence, the Malwr.com analysis the file through a wider range of operating systems, and this is a good way of ensuring that we get a dynamic analysis, of the file.
The offline edition of the Cuckoo Sandbox for our framework will be implemented with Microsoft Windows XP, Windows 7 and Windows 10.

5.3.2 Link analysis

If the malicious content of the mail, appears to be a link, the framework will conduct the link analysis. The framework trawls the mail and compares the content to three different regular expressions, such that links are harvested from the email. The link analysis is implemented based on the design described in Section 4.3. First thing is to determine the IP address behind the link. This is done with a quick search by the `traceroute`-command. When the IP address is retrieved, the analysis will focus on this.

5.3.2.1 Registrant

The registrant of the IP address or domain can be found rather simple in Linux based system, using the `whois`-command. This will make a query in the RIPE-database and return information about the registrant.

5.3.2.2 Geographical location

The geographical location of the IP can be using the `geoip-lookup`-command in Linux-based systems. It only requires installation of geoip-bin on the server, and returns the location in a simple one line answer.

5.3.2.3 Known malicious activity

To make a lookup in the databases that collect information about malicious online activity, we have considered the tool IPinfo.
IPinfo is a script developed by HiddenIllussions[Ill12], who also developed AnalyzePDF. The scipt analysis IP addresses or URLs on a wide range of online registers, to determine where the IP address is located, and whether it has been reported of participate in malicious activities. In addition IPinfo retracts info of the geographical location of the IP address, hence the queries for this information, is handled by IPinfo in the framework, and not by the Linux-command, described above.

5.3.2.4 Content on the webpage

`LinkHunter` is the tool that analyses the content of the mail for hyperlinks. `LinkHunter` is a tool, developed by myself for this project.
The first part of the analysis is to download the content of the given webpage, and retract redirections. If a redirection is found, the new webpage will receive the same analysis.
Second part of the LinkHunter will determine whether the website contains hidden objects, e.g. Javascript that performs malicious activity. This is done by a python tool called Thug[Del12]. Thug is developed by Angelo Dell'Aera and is a honeyclient designed to retract and analyse content and objects from websites. Thug is assessing the objects maliciousness and gives a well-descriptive output.

5.4 The report

Reporting is an essential part of the framework – all the information gathered in the analysis has to be parsed, such that the user is receiving a useful report, to help him determine whether the mail is malicious or not.
When all tools has analysed the content, the analysis will have information about:

- Content is a *file*
  - **Microsoft Office-file**
    - Whether the file is a genuine Office-file

    * Whether the file contains macros
        · The behaviour of the macros
    * Metainformation about the document
    * VirusTotal classification
    * ClamAV classification
    * Behaviour when executed
  - **PDF-file**
    * Whether the file is a genuine PDF-file
    * Whether the file contains objects (in particular Javascript objects)
        · The behaviour of the objects
    * Metainformation about the file
    * VirusTotal classification
    * ClamAV classification
    * Behaviour when executed
  - **Other file**
    * Whether the analysed file-type matches the file-extension.
    * VirusTotal classification
    * ClamAV classification
    * Behaviour when executed

- Content is a *link*
  - Owner of IP address
  - Geographical location of IP address
  - Content on the webpage
  - Malicious activity reporting (if any)

The framework parses the output of all the tools, such that a coherent report is delivered to the user, with a readable analysis, such that the user has a tool to determine whether to trust the content of the mail or to discard the mail. An example report can be found in the Appendix B. The report consists the exhaustive analysis of the malicious content, and will be attached as a separate file to the user. A brief abstract of the report will be included in the body of the mail sent to the user, with the most relevant information. Due to simplification the report will be in .txt-format.

5.5 Summary of *Implementation*

The sections has documented the implementation of the framework in the emailserver. We have for each analysis part review the options and discussed which solution has been implemented in the current version of the framework. The final part of section summarises the information in the report, to ensure that all information required for a solid analysis – discussed in Chapter 4 – is gathered and sent to the user.

Chapter 6

Evaluation

The tool is evaluated by analysing a set of emails. To determine the file analysis part we have used a combination of malicious emails, received by myself on a private email account, where my spam filter apparently is lacking. These mails have exclusively had Microsoft Office-files attached (a combination of .xls-files and .doc-files). The PDF-part of the analysis will be tested with a set of malicious files, produced in Metasploit[1], with known vulnerabilities. We will test if the tool is capable of detecting and uncompressing files, if the file receive is compressed.
The link analysis is tested using a set of email I have received on my private email account.
This section will evaluate each part of the analysis tool, described in Chapters 4 and 5 individual, such that we for each part of the analysis will ensure that we get the desired information. Finally we will evaluate the report to ensure the user experience is maintained.

[1] Metasploit is a penetration testing software, developed by Rapid7; https://www.rapid7.com/products/metasploit/

6.1 Evaluation of file analysis

If the mail has an attached file, the file analysis will apply, and will initially make the file-type analysis. We use six different files throughout the testing. The documents are listed in Table 6.1:

No.	Name	Filetype	Malicious content
01	Certificate.xls	Microsoft Excel(.xls)	Contains ransomware macros
02	Bestcomputers.doc	Portable Document Format (.pdf)	Contains malicious Javascript
03	Bestcomputers.pdf	Portable Document Format (.pdf)	Contains malicious Javascript
04	invoice.docx	Microsoft Word (.docx)	No malicious content
05	Confidential.doc	Microsoft Word (.doc)	Contains malicious macros
06	nicegirl.jpeg	Picture (.jpeg)	Contains hidden executables

Table 6.1: Documents used for testing

6.1.1 File type analysis

TrID's file type analysis is compared to the file-extension. The conclusion of the comparison is neatly added to the report. If the result is a compressed file-type, the framework uncompresses the file, and restart the file analysis. The file analysis works with the files tested both on files with correct file-extensions and files with spoofed file-extensions.
We test the file type analysis by running it with a document with correct file extension and a document with wrong file-extension. Results of the analysis is listed in table 6.2.

File	Expected output	Test result
02	Warning	Correct
03	Approval	Correct

Table 6.2: Test results: File type analysis

6.1.2 Meta data extraction

The meta data extraction is conducted by Exiftool, and the output from is readable, and the current version of the framework parses the whole output and add it to the report to the user.

Exiftool works well, and the amount of information is highly dependent of the file type in analysis. The relevance of some of the informations is questionable, however we find in more useful to parse all information to the user, such that the user can make a valid decision based on whatever data he want. This part of the analysis has been tested with a wide range of file types. Test data and results are found in Table 6.3.

File	Expected output	Test result
01	Meta data	Correct
03	Meta data	Correct
04	Meta data	Correct
06	Meta data	Correct

Table 6.3: Test results: Meta data extraction

6.1.3 Macro analysis

The macro analysis part has been tested with the genuine malicious mails I have received, which included both .doc-files and .xls-files. In addition to these malicious files, has it been tested with some non-malicious Microsoft Office-files, to determine what the output is, if the file contains non-harming macros. The macro analysis is exhaustive and gives the user a wide range of information. When a macro-containing document is analysed, the user gets information of the amount of macros, the behaviour of the macros, and the metainformation of the macros. MRaptor gives, in addition, a assessment of the suspiciousness of the macros.
The test data and results is listed in Table 6.4.

File	Expected output	Test result
01	Macro containment	Correct
	Macro behaviour	Correct
05	Macro containment	Correct
	Macro behaviour	Correct

Table 6.4: Test results: Macro analysis

6.1.4 Object analysis

This part of the framework has been tested solely with malicious PDF-documents, which I have developed myself. Due to this, the part has not been tested as exhaustively as the macro analysis-part.
The two analysis tools, dedicated to PDF-analysis has been able to detect all malicious objects in the tested PDF-documents. AnalyzePDF makes an assessment of the maliciousness of the file, on a scale of low-medium-high. This assessment is forwarded to the user, with the rest of the analyses. The test data and results is listed in Table 6.5.

File	Expected output	Test result
03	Object containment	Correct
	Object behaviour	Correct
	Level of suspiciousness	Correct

Table 6.5: Test results: Object analysis

6.1.5 Known malicious activity

This part of the framework has been tested with a wide range of both malicious and non-malicious files. VirusTotal returns the number of anti-virus engines that classifies the file as malicious. This result is parsed directly on to the user. The analysis time of VirusTotal is highly depending on whether the file has been analysed before or not, however it classifies the malicious files correct when testing.
ClamAV gives a short analysis, which determines whether it classifies the file malicious or not. The analysis is supplemented with a longer analysis of what makes the file malicious. ClamAV catches most malicious file when testing. The challenge with ClamAV is the database update which has to be done frequently to ensure freshness of the analysis. The test data and results is listed in Table 6.6.

6.1.6 Behaviour analysis

The testing of Cuckoo has been done with a range of Microsoft Office files and PDF files. The analysis works, however the output from Cuckoo is rather exhaustive, and it is a challenge to sort of relevant information to the user. The result of the test is listed in Table 6.7

File	Expected output	Test result
01	VirusTotal detection rate	36/56
	ClamAV assessment	Malicious
02	VirusTotal detection rate	No data
	ClamAV assessment	Malicious
03	VirusTotal detection rate	No data
	ClamAV assessment	Malicious
04	VirusTotal detection rate	0/56
	ClamAV assessment	Clean
05	VirusTotal detection rate	36/54
	ClamAV assessment	Malicious

Table 6.6: Test results: VirusTotal and ClamAV

File	Expected output	Test result
01	Networkdetection	Detected
	Harddrive changes	Detected
02	Networkdetection	Non detected
	Harddrive changes	Detected
03	Networkdetection	Non detected
	Harddrive changes	Detected
04	Networkdetection	Non detected
	Harddrive changes	Non detected
05	Networkdetection	Non detected
	Harddrive changes	Detected

Table 6.7: Test results: Behaviour analysis

6.2 Evaluation of link analysis

The first part of the link analysis is to determine the IP address behind the domain of the link. We are using the `traceroute`-command to do so. To confirm the IP address from traceroute, IPinfo is collecting the IP address as well, and these are compared to ensure that the rest of the analysis will handle the correct IP address. Two links has been used during the formal testing of the linkanalysis. The links are found in Table 6.8. Since the both links has been used throughout the analysis, we have collected the testresults in one table, namely Table 6.9.

No.	Host	Malicious content
01	www.bla.dk	Non – link to advertising related webpage
02	www.seolondon-careers.com	Scamware distribution site

Table 6.8: Links used for testing. Full URLs are listed in Appendix C

.

6.2.1 Registrant

The output of the `whois`-command is parsed into the report, to inform the user of the registrant of the IP address. Since the `whois`-command is an integrated part of the Linux-distribution, we trust the genuineness of the information gathered. In addition of the registrant we parse the information about the registration date and expiration date on to the user.

6.2.2 Geographical location

IPinfo retracts the geographical location of the IP address, hence it is part of the information parsed by this tool. In the testing phase, we supplement the information from IPinfo with the `geoip-lookup`-command, to ensure the information from IPinfo is correct.

6.2.3 Known malicious activity

IPinfo retracts information about known malicious activity from:

- Google Safebrowsing
- VirusTotal
- Netdemon
- URLvoid

This infomation is collected and parsed on to the user. The tool is tested with a range of known malicious domains and a range of known safe domains, and makes a good distinguishing trustworthy and untrustworthy domains. However it is not 100% perfect, and some malicious domains is not captured by the tool. There is a number of databases that would be obvious to include, but at the time of writing we will stick to the four, used by IPinfo.

6.2.4 Content

The content analysis is tested in two ways:

Redirection The redirecting analysis, will download the content of the webpage, and look for redirecting patterns. This part of the analysis is tested on webpages known to redirect to other pages, and it seems to work.

Malicious content Thug is tested on a couple of websites, known to contain malicious activity. It retracts all relevant objects of the website and marks the malicious ones. We notify the user, if at least one object has been marked malicious.

No.	Test	Expected result	Test result
01	Registrant	Data of registration	Correct
	Geo location	Data of geo location	Correct
	Known malicious activity	Data of malicious activity	Assessed malicious
	Redirections	One redirection	Correct
	Content	No malicious content	Correct
02	Registrant	Data of registration	Correct
	Geo location	Data of geo location	Correct
	Known malicious activity	Data of malicious activity	Assessed malicious
	Redirections	Zero redirections	Correct
	Content	Some malicious content	Correct

Table 6.9: Test result of link analysis

.

6.3 Final evaluation

Each of the tools in the framework is working as intended, and the final report gives a overview of the content analysed. The report is sent in .txt-format, which is rather simple, yet good solution. In a future edition of the framework, it would be preferable to sent a PDF-file to the user instead. I couple of the output of the tools can be parsed in a more elegant way, however the output right now is workable. The report is evaluated by presenting five reports to a group of potential users. The test group will not know whether the file linked to each report is malicious or not. For each report they are asked to assess if they would trust the file or not. The result of the evaluation is listed in Table 6.10.

No.	Correct classification	Testresults (% of right answers from test group)
01	Malicious	66%
02	Malicious	100%
03	Malicious	66%
04	Non-malicious	100%
05	Malicious	66%

Table 6.10: Evaluation of user report

6.4 Summary of *Evaluation*

The section has for each of the tools in the framework – both file and link analysis tools – made an evaluation of the testing. All tools works as intended, however the parsing of the output from some tools – especially Cuckoo Sandbox – can be optimised from the current version.
The testing and evaluation of the framework has concluded that it works as intended.

Chapter 7

Conclusion

In this project, we have handled one of the challenges regarding spam mails. Namely the problem regarding spam mails, which the spam filter of the email client has recognised as genuine, but which are in fact malicious.
The challenges when developing spam filters are, that the user expect the spam filter to let all genuine emails through. However when lowering the rate of genuine emails marked as malicious in the spam filter (false positives), we coherent increase the rate of malicious emails marked as genuine (false negatives).

This gives the user some responsibility for making a second-line assessment of the email – to ensure that no malicious emails are opened. The need for user awareness has been discussed, however no matter how aware the users are, the common user has limited methods to make an exhaustive analysis of the suspicious content of a email. The contribution of this project was to design and implement such an analysis tool, which will help the user to be able the determine if the content of the suspected email is to be trusted or not.

Based on the analysis in Chapter 3, we have determined that the tool will have to be able to make an exhaustive analysis of any attached file the user may receive, and/or a forensic analysis of any link in the email. The result of the analysis should be returned to the user. The result will not make a black and

white decision whether the content is good or bad, instead the user should compare the result to his own expectations of the content, and determine if the analysis result and his expectations is equivalent enough to trust the content.

The analysis tool is implemented as a framework on a stand-alone email server. The user is able the forward a suspicious email to the server, which automatic processes the email and any attachment, and return the result to the user in a email.
The framework is written in Python, and incorporates a range of analysis tools, some developed exclusively for the tool, and some developed by other. The framework parses the result from the tools, and merge them all together into the report to the user.

The evaluation of the framework shows that the majority of the embedded tools, gives correct and useful output which is parsed on to the user. However we encounter a challenge in the current version of the framework: The behavioural part of the file analysis uses Cuckoo Sandbox. The challenge of Cuckoo is that it is hard to automate, when we are trying to expand the file type supported as much as possible. This means, that right now the output from Cuckoo is very limited.
Another challenge is the report of the analysis. When evaluation the report, we gave a group of test persons, the results of five different analyses. The evaluation showed some difficulty in reading the reports. One of the problems was, that the test group was not presented with the analysed files, only the analysis. This means, that they did not know what to expect from the analysis. Due to this, the report evaluation is not complete, and will require that the test group receives some malicious emails themselves, and forwards it to the server. The report is – in the current version of the framework – parsed and sent as .txt-files. This has limitation and it would be preferable to implement a more sophisticated way of presenting the analysis results (e.g. in LaTeX).

To summarise, the framework works as intended. The requirements for the tools was listed in Table 1.1. Requirements 01 (Automated analysis) and 02 (User friendliness) is fulfilled, with the exception of the limitations of Cuckoo Sandbox stated above. Requirement 03 (Reporting) is partly fulfilled and will require the implementation of a sophisticated report generation in the framework to be fulfilled completely.

7.1 Future work

This section will review a list of possible future extensions of the framework:

LaTeX-reporting As stated above, the report generation should be upgraded. One of the possibilities is to generate the report with LaTeX. This would professionalise the framework, and make the report more readable. When installing the framework on-site, the report generation should be discussed with the users at the site, such that the report could be generated to their need.

Relevant sandbox The current implementation of the sandbox is to broad to work as intended. If installing the framework at a organisation, it would be obvious to install a sandbox environment, equivalent to the real environment used at the organisation. This would give a more specialised analysis.

Automated output data selection In the current version of the file analysis in the framework, all parts of the analysis has only implemented a limited amount of tools, to ensure that we don't get to much repeated data. The means that the result of the analysis relies on the result of the limited number of tools. In a future edition of the framework, a more sophisticated sorting method could be implemented, such that if two tools produced the same analysis, the report would not contain the data twice. This would allow us to implement more tools, without worrying about repeated data in the analysis result.

Mobile platform malware A SMS-receiving service to analyse Android or iOS malware, alternatively an app. As the amount of malware for the mobile platform is increasing, it would be obvious to implement a version of the framework that can handle this sort of malware. The solution could be application-based or be implemented as a SMS receiving service.

Appendix A

How to run the server

The testing environment has been handed in, in addition to this report. It is uploaded to Google Drive, and can be downloaded from:
`https://drive.google.com/file/d/0B1-QTBwJJp8IZXYwVU5nQVV4eW8/view`

The environment consists of:

DNS server The DNS server acts as router in the virtual network.

Mailclient The mailserver where the framework is located

Client The machine from which emails can be forwarded to the server.

The three virtual machines are handed in as a single .ova-file which should be imported by a Virtual Guest Machine-manager (I have used Oracle VM VirtualBox[1]).
The two client machines rely on the DNS server to be configured correctly. When opened, login credentials for all machines are Username: daniel / Password: daniel
To test the service, run the script `test_example` from the client. After a 30 seconds open `mutt` on the client and the analysis reports should appear.

[1] https://www.virtualbox.org/

Appendix B

Example report

```
Following file types is detected in the file:
--100.0% (.PDF) Adobe Portable Document Format (5000/1)

Static filetype comparison APPROVED:
The filetypes detecting in the file is equivalent to the file-extension.


Following meta-data is found in file.
Please compare to your expectations:
File Name                       : BestComputers.pdf
File Size                       : 6.4 kB
File Modification Date/Time     : 2017:02:28 16:57:10+01:00
File Access Date/Time           : 2017:02:28 16:59:07+01:00
File Inode Change Date/Time     : 2017:02:28 16:57:10+01:00
File Permissions                : rw-------
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : No
Analyzing PDF for suspecious objects..
MEDIUM probability of being malicious
Contains Javascript
Contains suspecious elements:
```

```
-OpenAction (1)
-JS (1)
-JavaScript (1)
Contains known exploitation method: CVE-2008-2992

Received and scanned on VirusTotal.com: 2017-02-04 04:33:40
Detections:
36/54 Positives/Total

Recognised as a malicious file by anti-virus engine.
```

Appendix C

Testing URLs

The following list is the full URL, for the links used for evaluation of the link analysis part, c.f. Section 6.2

01 `http://tj-dxxy.com/qiyueadmin/skymoneyEditor/sysimage/tree/ocmtcym/`

02 `http://www.seolondon-careers.com/cset/sikker-TDC`

Bibliography

[AB17] Mamoun Alazab and Roderic Broadhurst. An analysis of the nature of spam as cybercrime. In *Cyber-Physical Security*, pages 251–266. Springer, 2017.

[AC09] Ahmed Abbasi and Hsinchun Chen. A comparison of tools for detecting fake websites. *Computer*, 42(10), 2009.

[Apa10] Apache. Apache tika - a content analysis toolkit. 2010.

[Azi13] Ashar Aziz. The evolution of cyber attacks and next generation threat protection. In *RSA Conference*, 2013.

[BCASMV93] Tim Bienz, Richard Cohn, and Calif.) Adobe Systems (Mountain View. *Portable document format reference manual*. Citeseer, 1993.

[Boy16] Magnus Boye. Dr sendte falsk phishingmail til 3.000 ansatte: 1.406 gik i fælden, June 2016. [Online; retrieved 20-January-2017; `https://www.version2.dk/artikel/dr-sendte-falsk-phishingmail-til-3000-ansatte-1406-gik-i-faelden-834`

[CAM+08] Xu Chen, Jon Andersen, Z Morley Mao, Michael Bailey, and Jose Nazario. Towards an understanding of anti-virtualization and anti-debugging behavior in modern malware. In *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, pages 177–186. IEEE, 2008.

[CDH14] Ping Chen, Lieven Desmet, and Christophe Huygens. A study on advanced persistent threats. In *IFIP International Confer-*

ence on Communications and Multimedia Security, pages 63–72. Springer, 2014.

[Con11] Lucian Constantin. Drive-by download attack on facebook used malicious ads, October 2011. [Online; retrieved 31-January-2017; `http://www.pcworld.idg.com.au/article/403127/drive-by_download_attack_facebook_used_malicious_ads/`].

[Del12] Angelo Dell'Aera. Thug: a new low-interaction honeyclient, 2012.

[DHG09] John D'Arcy, Anat Hovav, and Dennis Galletta. User awareness of security countermeasures and its impact on information systems misuse: a deterrence approach. *Information Systems Research*, 20(1):79–98, 2009.

[Dig17] Digitaliseringsstyrelsen. Nemid – self-service on the internet. *Online: https://www.nemid.nu/dk-en/about_nemid/*, 2017.

[Dub17] Dubex. Danske tømrere misbrugt i ransomware-kampagne, February 2017. [Online; retrieved 19-February-2017; `https://www.dubex.dk/update/danske-toemrere-misbrugt-i-ransomware-kampagne/`].

[EKK09] Manuel Egele, Engin Kirda, and Christopher Kruegel. Mitigating drive-by download attacks: Challenges and open problems. In *iNetSec 2009–Open Research Problems in Network Security*, pages 52–62. Springer, 2009.

[Ens17] Chris Ensey. Ransomware has evolved, and its name is doxware. In *DARKReading*. nformationWeek Business Technology Network, 2017.

[Esp11] Jose Esparza. peepdf-pdf analysis and creation/modification tool. *Online: https://github.com/jesparza/peepdf/wiki*, 2011.

[Fal03] Deborah Fallows. *Spam: How it is hurting email and degrading life on the Internet*. Pew Internet & American Life Project Washington, DC, 2003.

[GTBS12] Claudio Guarnieri, Allessandro Tanasi, Jurriaan Bremer, and Mark Schloesser. The cuckoo sandbox. 2012.

[Had11] Christopher Hadnagy. *Social Engineering: The Art of Human Hacking*. Wiley Publishing, Inc., 2011.

[Har13] Phil Harvey. Exiftool: Read, write and edit meta information. *Software package available at http://www.sno.phy.queensu.ca/~phil/exiftool*, 2013.

[Hon12] Jason Hong. The state of phishing attacks. *Communications of the ACM*, 55(1):74–81, 2012.

[Ill12] Hidden Illusions. Ipinfo - searches various online resources to try and get as much info about an ip/domain as possible. *Online: https://github.com/hiddenillusion/IPinfo/blob/master/Readme.md*, 2012.

[Ill13] Hidden Illusions. Analyzepdf - bringing the dirt up to the surface. *Online: https://hiddenillusion.github.io/2013/12/03/analyzepdf-bringing-dirt-up-to-surface/*, 2013.

[Kei13] Gregg Keizer. Oracle will continue to bundle 'crapware' with java. January 2013. [Online; retrieved 31-January-2017; `http://www.computerworld.com/article/2494794/malware-vulnerabilities/oracle-will-continue-to-bundle--crapware--with-java.html`].

[Kin13] Darien Kindlund. Holyday watering hole attack proves difficult to detect and defend against. *ISSA J*, 11:10–12, 2013.

[KM04] Jeremy Z Kolter and Marcus A Maloof. Learning to detect malicious executables in the wild. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 470–478. ACM, 2004.

[Koj04] Tomasz Kojm. Clamav, 2004.

[Lag13] Philippe Lagadec. Oletools - python tools to analyze ole and ms office files. *Software package available at http://www.decalage.info/python/oletools*, 2013.

[Lit17] Litmus. Email client market share. January 2017. [Online; retrieved 31-January-2017; `http://emailclientmarketshare.com/`].

[LSS+07] Wei-Jen Li, Salvatore Stolfo, Angelos Stavrou, Elli Androulaki, and Angelos D Keromytis. A study of malcode-bearing documents. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 231–250. Springer, 2007.

[MC09] Tyler Moore and Richard Clayton. Evil searching: Compromise and recompromise of internet hosts for phishing. In *International Conference on Financial Cryptography and Data Security*, pages 256–272. Springer, 2009.

[NM14a] Hiran V Nath and Babu M Mehtre. Static malware analysis using machine learning methods. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 440–450. Springer, 2014.

[NM14b] Hiran V Nath and Babu M Mehtre. Static malware analysis using machine learning methods. In *International Conference on Security in Computer Networks and Distributed Systems*, pages 440–450. Springer, 2014.

[OBr16] Dick OBrien. Dridex - tidal waves of spam pushing dangerous financial trojan. *Symantec Security Response, Tech. Rep*, 2016.

[Par12] Bimal Parmar. Protecting against spear-phishing. *Computer Fraud & Security*, 2012(1):8–11, 2012.

[PKSH16] Martin Potthast, Sebastian Köpsel, Benno Stein, and Matthias Hagen. Clickbait detection. In *European Conference on Information Retrieval*, pages 810–817. Springer, 2016.

[Pon03] Marco Pontello. Trid-file identifier. 2003.

[RH11] Sara Radicati and Quoc Hoang. Email statistics report, 2011-2015. *Retrieved May*, 25:2011, 2011.

[Rob] Alexander Robertson. File compression techniques.

[SBW01] Martina Angela Sasse, Sacha Brostoff, and Dirk Weirich. Transforming the 'weakest link'—a human/computer interaction approach to usable and effective security. *BT technology journal*, 19(3):122–131, 2001.

[SJ05] Daniel J Sanok Jr. An analysis of how antivirus methodologies are utilized in protecting computers from malicious code. In *Proceedings of the 2nd annual conference on Information security curriculum development*, pages 142–144. ACM, 2005.

[Ste] D Stevens. Didier stevens pdf-parser. py. *Online: blog. didierstevens. com/programs/pdf-tools/*.

[Tot12] Virus Total. Virustotal-free online virus, malware and url scanner. *Online: https://www.virustotal.com/en*, 2012.

[TSPM11] Zacharias Tzermias, Giorgos Sykiotakis, Michalis Polychronakis, and Evangelos P Markatos. Combining static and dynamic analysis for the detection of malicious documents. In *Proceedings of the Fourth European Workshop on System Security*, page 4. ACM, 2011.

[Wan06] Wallce Wang. *Steal this Computer Book 4.0*. No Starch Press, 2006.

[YIM+09] Katsunari Yoshioka, Daisuke Inoue, ETO Masashi, Yuji Hoshizawa, Hiroki Nogawa, and Koji Nakao. Malware sandbox analysis for secure observation of vulnerability exploitation. *IEICE transactions on information and systems*, 92(5):955–966, 2009.