

Detection of fake domain names in e-mails

Anders Gorboe



Thesis submitted for the degree of
Master in Applied Computer and Information
Technology - ACIT
(Cloud-based Services and Operations)
30 credits

Department of Computer Science
Faculty of Technology, Art and Design

Oslo Metropolitan University — OsloMet

Spring 2022

Detection of fake domain names in e-mails

Anders Gorboe

© 2022 Anders Gorboe

Detection of fake domain names in e-mails

<http://www.oslomet.no/>

Printed: Oslo Metropolitan University — OsloMet

Abstract

Phishing is social engineering attack that inflicts damages of several billion dollars each year. Phishing has increased yearly in frequency and in complexity bringing new and more clever schemes for hackers to deceive their victims. This thesis aims to assist and help fight this continuously growing concern. A common way of phishing is to impersonate other people or companies. This can be done by creating fake domain names that look identical but are not the same as a legitimate entity. In response to this a prototype application has been developed to see how effective we can stop these kinds of attacks, and spot fake domains before they can do any harm. This prototype application will compare new domains against previous already ensured domains to see if the new domain is trying to disguise itself as one of these. To test the application DnsTwist has been used to find malicious domain names.

Acknowledgments

I would like to thank both of my supervisors Anis Yazidi and Hårek Haugerud for guiding me through this past semester. Their feedback and ideas for the developed product and the thesis itself has been invaluable.

Contents

Abstract	i
Acknowledgments	iii
1 Introduction	1
1.1 Problem Statement	3
2 Background and Related Work	5
2.1 Phishing	5
2.1.1 The phishing lifecycle	6
2.1.2 Ways to disguise URL's and domain names	7
2.1.3 Homograph attacks	9
2.1.4 Malware attacks	9
2.1.5 Source Spoofing	10
2.2 Related works	11
2.3 Phishing Trends	13
3 Approach	15
3.1 Levenshtein distance	15
3.2 Common alterations	18
3.3 The process	19
3.4 Technologies used	21
4 Results	23
4.1 The program	23
4.2 Testing	27
4.2.1 DnsTwist	27
4.3 More complicated problems	30
5 Discussion	33
5.1 Provides value	33
5.2 Future works/Limitations	38
5.2.1 False-positives	38
5.2.2 Warnings and alerts	38
5.2.3 First is trusted	38
5.2.4 Mutations	40
5.2.5 Further development	42
5.3 Performance	42

6	Conclusion	45
7	Appendix	51
7.1	LevenstheinDistance.js	51
7.2	Domain Regex	52
7.3	Source code	52

Chapter 1

Introduction

Phishing is an old but steadily growing concern when it comes to staying safe online. Old in the way that it has been around since the beginning of the internet, and maybe even longer, but associated with a different name. Most likely we have all been subjected to some sort of attempted phishing or hustle during our life. I had my first encounter at the age of six where I was tricked into giving away all my coins in my favorite video game. Luckily losing imaginary money is not the worst repercussion, and on the bright side it thought me a vital life lesson. Cybersecurity attacks have grown over the years both in quantity and quality, with technology evolving ever so rapidly, making it impossible to predict how and what future exploits might look like. Fighting cybercrime is an eternal battle that swings back and forth between the forces of good and evil, the bad guys will use a new exploit and the good guys will try to respond quickly to fix it. Luckily we have come a long way since the dawn of internet and many protocols and infrastructures have been implemented to make it more difficult for hackers to access things they shouldn't. The improved base security means the cybercriminals had to find others ways of attacking, mainly through social engineering schemes also known as phishing. Phishing is when someone tries to steal sensitive information, access or assets, by preying on our biggest cybersecurity flaw throughout the times; people.

The evolution of technology has led to some people being very skilled, while many are falling behind and found lacking in internet knowledge. Don't let that fool you, even if you are quite an experienced internet user you can still fall for their trickery. With careful planning, reconnaissance and a flawless delivery message, they could fool just about anyone, and depending on the target it could cause massive amounts of damage.

Companies and governments are spending more and more money on improving their cybersecurity, ranging from tools and protocols to educating their employees, but the attacks keep coming, they keep being successful, and they keep causing damage. The latest report from Anti-Phishing Working Group (APWG) is showing record high cybersecurity attacks, as of February 2022 the amount of phishing attacks has tripled since 2020 with between 204 000 - 282 000 reported phishing attacks per month [11][12].

This increase in attacks means more money is lost to phishers. A report from FBI shows reported losses exceeding 4.1 billion US dollars in the United States alone [10]. While reports from McAfee estimates a worldwide cost of 600 billion USD in 2018 and reaching 1 trillion USD in 2020 where 945 billion USD is lost from cybersecurity incidents and 145 billion USD used to improve cybersecurity infrastructures [23][37] as seen Figure 1.1.

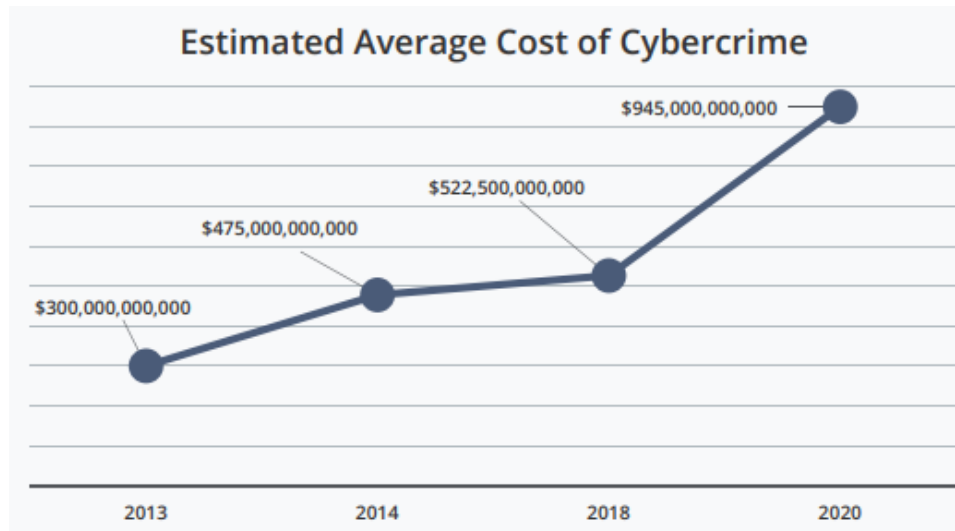


Figure 1.1: This figure shows the cost of cybercrime from 2013 to 2020. This figure is copied from McAfee’s most recent cybersecurity report [37]

Recently there has also been an increase of more sophisticated and organized attacks targeting high value targets. We are not just seeing the usual spam messages and emails from pretending Nigerian princes, but rather quite believable stories backed by fabricated evidence and sleight of hand. They have many weapons in their arsenal one of these involves fake email addresses where some part of the original domain name has been slightly altered to mimic a domain that is known to the receiver. An important part of what they do when attacking a company is reconnaissance, they figure out the possible ways of infiltrating a company by looking at their relations. From this, they can create a message that could be believable for this company, and send it with a email address that might look like it is familiar to the company, but with slight alterations. Microsoft and Google warns their users about mismatching email domains as well as other tricks phishers use, to raise awareness [37][24][11]. But is raising awareness truly enough, we cannot continue to be afraid of opening our email or our text messages. It is imperative we acknowledge that cybercriminals are experts at what they do, and to understand we need help if we are to hold them of.

1.1 Problem Statement

Phishing is an enormous problem, one that is impossible to completely fix on your own. Traversing the internet, exploring and simply using what most of us rely on a daily basis has become a risky endeavour. This thesis aims to provide an introduction to phishing and its spread as well as addressing one aspect of phishing in more detail. We are looking into more targeted attacks via email where company domains are being used with small alterations to create lookalike domains to trick users. We believe it is possible to limit the amount of successful phishing attacks from domain alterations, by storing what domains you usually communicate with and comparing these with the domain of the new emails coming in to your inbox. This will mean a tailored defence that also grows as you expand your network. For this a prototype will be created, and tests to see how effective we could spot phishy domains while trying to limit the amount of false-positives. In order to test this program we intend to use DnsTwist as a provider of fake domain names.

Chapter 2

Background and Related Work

In this chapter we will take a deeper dive into what phishing is, how it works, how people can trick us and some of the many tools they use to do so. We will examine the current state of phishing and how it has come to spread at such a rapid rate. Lastly we will look into other research done on the topic as well as other tools and methods that are closely related to this paper and phishing.

2.1 Phishing

Phishing is when someone is trying to disguise or impersonate someone else in order to extract sensitive information from a person or a company. Phishing attacks can happen over the phone, email, forums, messaging apps, or pretty much any channel of communication, including real life in-person [27]. There are many names and categories for phishing attacks, a blog post from SecurityScorecard mentions spear phishing, whaling, email phishing, pharming, pop-up phishing, clone phishing, vishing, smishing, angler phishing and many more [29].

Email phishing as the name suggests is when the attacker use e-mail as the communication channel to send malicious content either in the form of an infected attachment, or an URL taking you to a bad webpage. Email is often used for sending out spam messages that reach many targets in hopes that even just a few take the bait. Typical messages can be: "Congratulations you just won 10 000\$ click here to claim", or "Your package has been delayed, click here to track it". Email is a very popular communication channel for phishers to use, as you can reach many people in a short amount of time. In addition to this, email allows the use of HTML and JavaScript, which opens a world of different ways to trick someone else.

Spearphishing is when a phisher targets one specific person and create a scenario/scam just to trick that one person. These attacks can often be quite sophisticated and believable, and depending on the target these attacks can cost people and companies a lot. A spear phishing attack can be delivered in many ways from using email, SMS, calling or in-person. When the spear phishing attack is directed at someone important, like a CEO of a company, then it is called whaling.

2.1.1 The phishing lifecycle

A key aspect of preventing phishing is understanding its lifecycle, which can be quite difficult as there are many different technologies, methods and directions a phisher can take to fool a target. Let us bring it back to the basics; any cyber attack follows the steps in the 'cyber kill chain' [8].

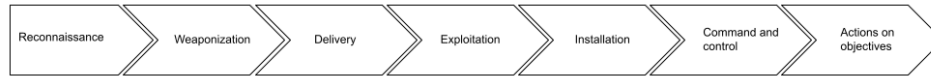


Figure 2.1: Figure of the cyber kill chain. The cyber kill chain contain 7 steps Reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives. Based on the illustration from lockheadmartin.com [8].

First of all some sort of reconnaissance must be done for the hackers to have your contact information. In many cases sending email is part of the delivery section where you give a malicious URL or attachment, but sending email can also be a way to do reconnaissance on your target. Talking to people can help you find information that is not accessible online. When they have gathered enough information they are ready to strike, but first must create a weapon to do so. Creating a malicious program or a webpage designed to steal your information are some ways of doing it, but creating a good scheme to trick someone could also serve as a weapon. The next step is delivery, get you to use this program or webpage, this can be sent using any form of communication but let us focus on email. Many email scams in particular use services that are known to you, like amazon or PayPal in conjunction with a message intended to stress you and make you act instantly and press this URL [34]. Stressing you is important for the scam as they want to shut down your defence mechanisms, your ability to think critically and be suspicious. In Figure 2.2 you can see by searching the service in your browser instead of pressing the link directly in the email you can protect yourself from many attacks. This is how you can work around the problem, it does however not solve the issue of phishing. If you press the URL in the mail you will be directed to a page that might look exactly like amazon or PayPal prompting a login. Upon login your password and username is sent to the phisher and a confirmation is sent back to you saying everything is in order.

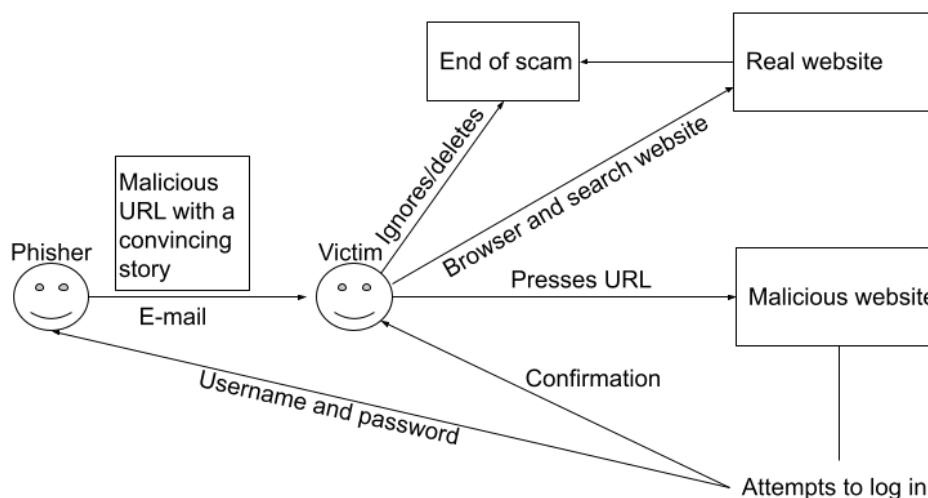


Figure 2.2: This is a diagram displaying a phishing lifecycle. It starts with an email sent to the victim containing a malicious url. If the victim ignores/deletes the email the scam i ended, but if the victim presses the url he/she is taken to a malicious webpage. On attempted login a confirmation is sent to the victim, and the username and password is sent back to the phisher.

This last part is important as they want access to your account without you realizing, giving them time to further exploit and figure out how they can get more from this. They will try other services with the password they obtained, if they gained access to your social account they might go through your pictures and find something they can hold against you. They might also try to use your account as the means to infect your friends and gain even more information passwords and accounts.

2.1.2 Ways to disguise URL's and domain names

An important aspect of phishing is the use of URLs and the many ways phishers can trick people into pressing URLs that take them to malicious webpages. The ways to disguise URLs can be divided into 4 main categories mangle, mislead, camouflage and obfuscate [4].

Mangle is a technique where the phishers take a completely legitimate site that most people would know about, like www.amazon.com and tweak some of the characters like changing the o with a 0 so we get www.amaz0n.com instead. This second URL is looking fairly similar to the first URL, but it will take you to a different webpage. In addition to this an even more deceptive approach would be to use non-ASCII characters also known as homograph attacks. [17][4]

Mislead is when phishers take a legitimate webpage, like www.oslomet.no and put the domain name oslomet in either the subdomain or in a path. For instance the URL www.oslomet.uio.no you can clearly see the oslomet part, but it's in the subdomain section of the link, while uio is the domain.

Camouflage is quite similar to the mangle category, but instead of chan-

ging letters you are adding characters that looks natural in that spot. For example with the `www.oslomet.no` URL could be camouflaged by doing `www.oslo-met.no`, if you don't know `oslomet` is not supposed to have a hyphen in the middle it will seem quite legitimate.

Obfuscate uses evasive techniques to hide the destination of a link, this is often done with the help of redirected or shortened links [4]. This is quite simple to do and there are several applications that can shorten your URL with the click of a button, one example is `bitly.com`.

In addition to this when working with email, as they allow HTML and JavaScript, there is a separation between what you see and the code behind. Using an HTML tag like:

`"www.goodwebsite.com"` it will display `www.goodwebsite.com` in your mail, but it will take you to the page `www.badwebsite.com`. In Figure 2.3 you can see an example of this attack using youtube and google. In this case using the gmail mail client we are receiving no warning about the real destination of the url.

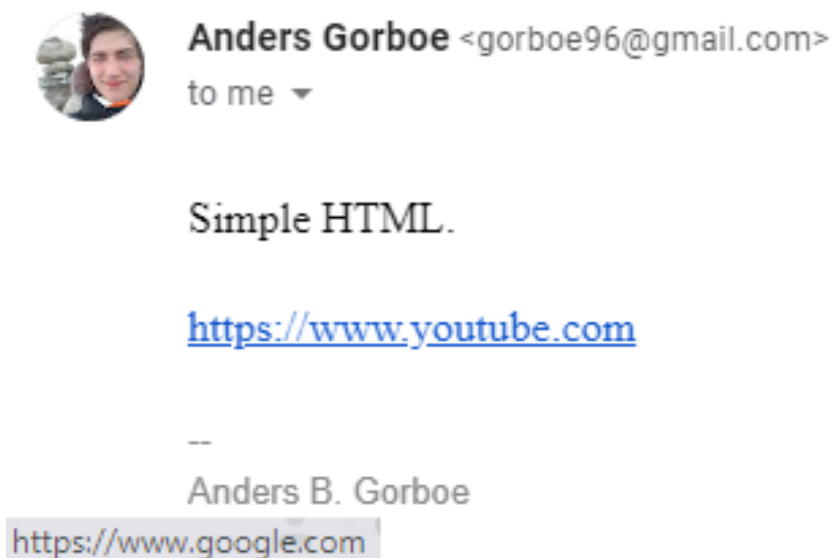


Figure 2.3: This image displays the use of a simple HTML `<a>` tag in email to display the URL `www.youtube.com`. But on closer inspection when hovering the link the real destination will pop up in the bottom left section of your screen (I moved it up so it would fit in the picture frame). In this case the URL will take you to `www.google.com`

2.1.3 Homograph attacks

Belonging to the mangle category as mentioned above we have the more deceptive homograph attacks that can make two strings identical to the human eye. With the use of non-ASCII characters, they can take characters that look like ones from the standard pool and swap them out. This will result in creating words, URLs or domain names that look exactly the same for the human eye. Even if they might look the same for the human eye these attacks can easily be prevented with a program looking for the character codes as these are different. See Figure 2.4.

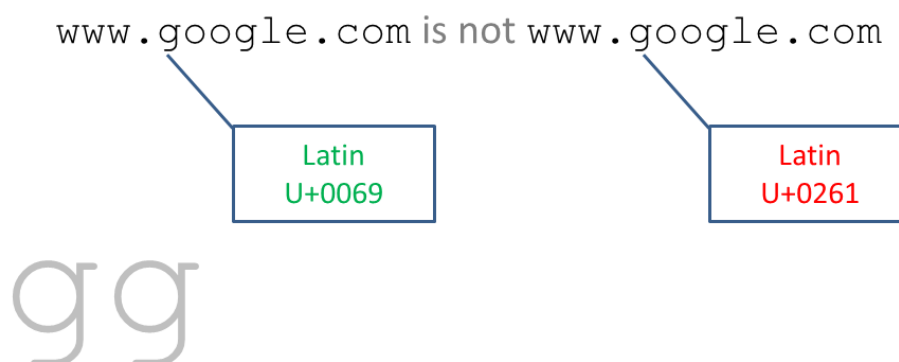


Figure 2.4: This image is showing two visually equal URLs, but the one on the right is using a g that is different from the standard ASCII table. This image is taken from <https://websec.github.io/unicode-security-guide/visual-spoofing/> [35]

2.1.4 Malware attacks

Malware attacks are also important to mention when it comes to phishing. This is when the attacker send malware in the form of an attachment or gets you to download it from a malicious webpage. There are many different types of malware some of these are key loggers, virus, worms, trojans, spyware, adware, ransomware or rootkits [3].

Virus is a piece of code that cannot "live" on its own, it needs a host also known as another program to be executed. The virus is executed whenever the host program is executed and will try to spread itself to other hosts, much like a biological virus. When enough hosts has been infected a trigger could be activated making the virus do hostile actions such as deleting, stealing or altering files, corrupting files or trying to force the computer to malfunction. Worms are similar to viruses but can live on their own and be remotely controlled. Trojans aswell are similar to both viruses and worms but the name references the "Trojan horse" that was used by the greeks to trick their way into the city of Troy. So a trojan is a piece of independent malicious software that you trust and need, and could be something as simple as a cool customizeable clock attachment for your desktop computer. Spyware as the name implies is all about spying on

your target, this could be extracting files and images, or more sensitive information like usernames and passwords. Keyloggers is a rather simple but highly effective form of spyware, it keep tracks of what keys you press and send them to the attacker. This means anything you type including usernames and passwords will be sent back to the attacker to do with as he/her wishes. Adware is not necessarily malicious, but will create a bunch of unwanted adverts when you are using the computer. The creator of the adware gains revenue from the ads that are displayed for you. Ransomware is when the attacker has either found some information they believe to be sensitive to you, or they have gained enough control to take over the system and lock you out. This resulting in a ransom to ensure your sensitive pictures does not get posted online, or for you to regain access to your own computer. Finally rootkits is there to hide the presence of malicious content on your computer.

2.1.5 Source Spoofing

Email stands for electronic mail and the core principle is the same as sending a regular mail with a piece of paper in an envelope. When working with email you have an address 'yourname@somedomain.tld' similar to real life where you have an address that might be 'Somestreet 31B' and this is how we know where to send the email/mail. You can write something on a piece of paper claiming to be someone, pack it up and walk over to someones mailbox and deliver it, and you could do the same with email. This is known as spoofing, or source spoofing and it allows you to send an email 100% identical to someone else's email address. The easy way to accomplish this is using the 'sendemail' package for kali linux. When people realized that anyone could sign an email with any name, several frameworks were created to put an end to it, SPF, DKIM and DMARC [20][1][21][15]. It depends on what email provider you use, most people use either gmail, yahoo or hotmail, which supports all three protocols, but there are providers that only support SPF and DKIM and there are some that doesn't support any of the protocols [15].

2.2 Related works

Phishing is not a new problem, it has been around as long as the internet itself has been around. As phishing became a problem we found ways of trying to deal with it from educating people to use the internet to developing tools that could help stop these attacks. Some of these tools are Faheem Slack bot [4], SpoofGuard browser plugin along with several other browser plugins to prevent spoofing [7][36], Thunderbird email client [33], the educational applications NoPhish and Anti-Phishing phil [6][31] and hundred different types of spam filters including those already implemented in outlook and gmail today.

Recent years a lot of research papers with machine learning approaches has been published to help fight phishing. These are focused toward textual analysis to detect phishing by comparing to common features of phishing emails [2][9][14][19]. There are also more ML papers aimed at fighting lateral spearphishing attacks, which are attacks from compromised emails. [13][16][32]. The only negative side with these types of approaches is that you need large amounts of data for it to work efficiently. See Figure 2.5. In the paper 'That Ain't You: Blocking Spearphishing Through Behavioral Modelling' they analyse the writing habits of each person you are sending mails to [32]. This means someone you email several times a day will have a better model than those you send less emails to. In addition to this behavior is something that the phishers can mimic by looking at the previous conversations, which could bypass the defence. There is also a cloud

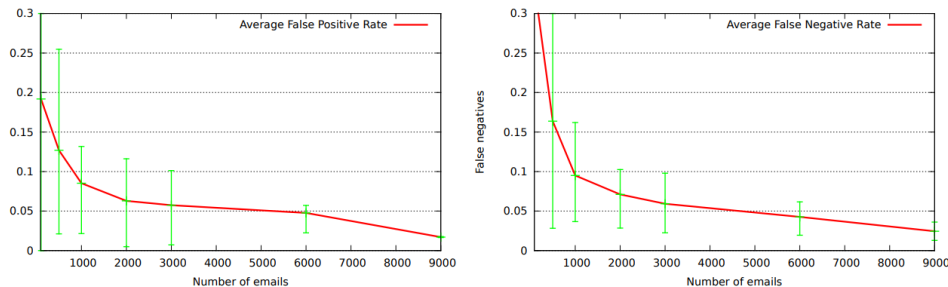


Figure 2.5: Figure from the paper 'That Ain't You: Blocking Spearphishing Through Behavioral Modelling' showing the false-positive rates and the false-negatives [32].

solution to fight lateral spear phishing, that looks promising [18]. Another great way to protect against lateral spear phishing is using two-factor authentication which both Google and Microsoft provide for their services. This works the same way as a vaccine, the more people that use two-factor authentication, the less likely of infection and account takeovers and lateral spearphishing. If everyone could ensure their account is safe, lateral spearphishing would cease to exist.

Thunderbird is a free opensource project and currently offers ways of scanning attachments in emails and checks for URL redirection tricks. It does lack protection against URL spoofing and it does not check against com-

mon domains or have any ways of inserting trusted domains. But this project is currently under development and there might be more features towards security later in the run.

Faheem slack bot is a very interesting tool that offers a thorough explanation of a URL when posted in a slack text-channel. It separates the URL into parts and provide information to the user that can help determine if the link is legit or not [4]. In addition to giving out general information about a bad protocol or a suspect TLD it also contain anti-spoof filters that checks up against the most used webpages. This ability to analyse URLs is something that should be implemented in any channel of communication.

When it comes to anti-spam filters they are amazing at blocking out the majority of messages that flood into our email addresses every day, but they are not so great at dealing with more targeted attacks. Google and Microsoft browsers are constantly blacklisting webpages that are connected with phishing attacks in order to protect their users. Numbers show that google have blacklisted about 90% of all active phishing webpages, while microsoft lies around 67% [30]. It is hard to tell how many victims are exposed before a webpage is blacklisted, and the concept of blacklisting might stop current on-going scams, but it does not stop the fresh new scams.

2.3 Phishing Trends

Briefly mentioned in the introduction of this paper, we saw according to the APWG that cybersecurity attacks has tripled since 2020 [12][11]. Phishing is popular and the numbers keep increasing at a rapid rate each year, why is this the case?

The first reason is that it must be lucrative to do these attacks, and looking at the numbers lost on a global scale that seems to be the case. Reports from McAfee claims cybercriminals can make anything from several hundred thousands to even millions of dollars each year [23]. This is more than 10 times the average salary in Norway.

The second reason is the low risk of doing cybercrime. Cybercriminals are operating from all over the globe making it hard for local law enforcement's to do much about people getting scammed. Most countries have dedicated cybersecurity departments now, but even they struggle to enforce punishment due to the many ways a scammer can hide themselves. Things like VPN and crypto currencies like bitcoin helps scammer stay of the grid and untouchable from authorities [23].

The third reason is that it is easy to do and get started with, the tools you need can be purchased on the black market aswell as files containing millions of personal user data such as phone numbers and emails.

The fourth reason is that cybercriminals are organizing and working together in teams to create more powerful attacks. Reports from McAfee shows more and more cybercrime centers are created, these are regular office buildings where people go to work, only that their work is stealing using technology [23][37]. This means they have dedicated companies that collect user information, dedicated companies that create malicious software, and dedicated companies that perform phishing attacks, they have created their own supply chain. One of the many important parts of phishing is how they gather your contact information. We live in a digital world and a lot of contact information is publicly available either on google, pages like 1881.no, social services like facebook or on your work webpage. But in most cases they are not collected by a person, but webcrawlers created to gather email addresses, phone numbers and other information that might be useful. Data about you is collected and sold in bundles to several groups working with phishing scams. In the end, once your data has been collected by a malicious group you will always be at risk for as long as you use that email address or that phone number. A very interesting project is "Project HoneyPot" that tracks and traps these e-mail harvesters [26][25][22]. In the paper "Understanding How Spammers Steal Your E-Mail Address" they say: "The best way to stop spam is to keep spammers from getting your e-mail". This statement is impossible to disagree with, because if the scammer has no way of contacting you they cannot trick you or send you malicious content. But is it possible to prevent scammers entirely from harvesting your contact information; i think not.

Chapter 3

Approach

In this chapter we will look into how we find and deal with mangled domain name attacks. We will go through the algorithms we use to find these fake domains, and try to display the entire process and inner workings of this system.

3.1 Levenshtein distance

One way of comparing two words similarities is with the Levenshtein distance algorithm. The algorithm calculate the distance between two strings using a mix of the 3 operations, 'replace', 'delete' or 'insert'. Each of these operations adds one to the distance between the words in question. An example of the insert operation could be to find the distance between the words 'some' and 'someone', which would be 3 as you add 'o', 'n' and 'e' to the word. Similarly you could go the other way with the delete operation and remove characters. An example of the replace operation could be 'kitten' to 'mitten' that would give a distance of one, as you simply change the 'k' to 'm'. To use the algorithm you place two words into a matrix like Table 3.1, in this example we use the words 'Manipulate' and 'Mediate'. To start solving this matrix we need to investigate the four cells in the top left corner marked red, and continue as you would read a book; left to right, top to bottom. The four cells marked red is our first sub-problem, and the result is used to solve the next sub-problem and so on. We look at the three numbered cells 1 0 1 bordering the 'x' and taking the lowest of the three cells and adding your operation, if there is one. In this case M = M meaning we only care about the cell in the -1 | -1 position from the x and take that for our x, meaning the first x = 0. Similarly for the next step we would solve for the x to the right and we have the cells 1 2 and our newly discovered 0. The 0 cell is the lowest but in this case we are doing a replace operation between A and E making x = 1. See Table 3.2. We continue this pattern and solve every x left to right until we filled out the entire table like in Table 3.3. Here we also see marked in the very last cell 5, which is the final distance between the two words 'Manipulate' and 'Mediate'. To see this implementation in code you can see Appendix 7.1.

		M	A	N	I	P	U	L	A	T	E
	0	1	2	3	4	5	6	7	8	9	10
M	1	x	x	x	x	x	x	x	x	x	x
E	2	x	x	x	x	x	x	x	x	x	x
D	3	x	x	x	x	x	x	x	x	x	x
I	4	x	x	x	x	x	x	x	x	x	x
A	5	x	x	x	x	x	x	x	x	x	x
T	6	x	x	x	x	x	x	x	x	x	x
E	7	x	x	x	x	x	x	x	x	x	x

Table 3.1: This shows an unsolved table with the words 'Manipulate' and 'Mediate'.

		M	A	N	I	P	U	L	A	T	E
	0	1	2	3	4	5	6	7	8	9	10
M	1	0	x	x	x	x	x	x	x	x	x
E	2	x	x	x	x	x	x	x	x	x	x
D	3	x	x	x	x	x	x	x	x	x	x
I	4	x	x	x	x	x	x	x	x	x	x
A	5	x	x	x	x	x	x	x	x	x	x
T	6	x	x	x	x	x	x	x	x	x	x
E	7	x	x	x	x	x	x	x	x	x	x

Table 3.2: This shows an unsolved table where we solve the second sub-problem of the matrix.

		M	A	N	I	P	U	L	A	T	E
	0	1	2	3	4	5	6	7	8	9	10
M	1	0	1	2	3	4	5	6	7	8	9
E	2	1	1	2	3	4	5	6	7	8	8
D	3	2	2	2	3	4	5	6	7	8	9
I	4	3	3	3	2	3	4	5	6	7	8
A	5	4	3	4	3	3	4	5	5	6	7
T	6	5	4	4	4	4	4	5	6	5	6
E	7	6	5	5	5	5	5	5	6	6	5

Table 3.3: This shows the final solved table with the words 'Manipulate' and 'Mediate'. The length marked in the last column as 5.

3.2 Common alterations

Levensthein distance is a quite useful algorithm as when the distance is low, we know the two words we are comparing are closely related, and we know when the distance is high that they are far apart and we don't need to worry. However that is not necessarily the case, as two words might be far apart in the algorithms eyes requiring many operations, but could still look quite similar for the human eye. Common alterations such as changing m with r-n, big i with small L, adding misspellings to the word or even using numbers or non-ASCII characters that make the two words still look the same, while the algorithm thinks these are far apart. To counteract this we can try to identify these swaps and finding the effective distance. The effective distance is intended to be a measurement on how equal humans find the two words, where 0 is completely equal. An example of this could be the word 'Hemmingway' compared with the word 'Herrningway' r-n-r-n instead of the two M's. Levensthein would give us a distance of 4 as the r-n would replace the m by one insertion and replacement operations, and we would have to do this for both M's. However the effective distance would be 0, as the two words look identical.

To do this we check for target characters like M, N, RN, I, J, L in the trusted domain and check for potential swaps in the new domain at that index. As we see in Table 3.4 the first thing we would investigate is the first M and we see if the index 2 of the new domain is an N, if that is not the case we check if index 2 is an R AND index 3 is an N, which in this case is true. Now we have a problem for the next M as it will be compared to index 3 N, while we want it to be compared to the R-N of index 4 and 5. To successfully do this we use an adjuster whenever we find a case of M to R-N that would make sure we are comparing the correct characters. The adjuster will add one after the first case of m to r-n and compare the next m to the r instead of n index 3. After this it would add another to the adjuster making it 2, and this ensures we are continuing to compare I with I, N with N and so on.

0	1	2	3	4	5	6	7	8	9	10	11
H	E	M	M	I	N	G	W	A	Y		
H	E	R	N	R	N	I	N	G	W	A	Y

Table 3.4: A table displaying the two words hemmingway and herrningway and how the algorithm identifies the M and compares it to the R-N sequence of the new entry.

3.3 The process

There is an input field that can be used to simulate the process of sending an email to the system. This email will have to be in the standard format of 'sender@domain.tld' for the system to process it. When we receive a new email the domain will be extracted from the email address, the domain is the part after the @ and before the top-level-domain (tld). To extract the domain we use a regex that looks for a part before the @ sign, and a tld but only extracting the domain name that is in between. This regex can be found in the Appendix 7.2. It will then be used to check if this domain exists in our trusted domain list, if it is there nothing more will happen. In the case that it is not in our trusted list we will investigate and see if this could be an attempt at fraud. First we use the levenshtein algorithm between this new domain and all of the domains in our trusted list. The trusted list are previous domains that have gone through this process successfully. We then check all the commonly exploitable characters for each existing domain against characters in this new domain and calculate the effective distance by subtracting the result we got from levenshtein distance for each match of potentially exploited character in the new domain. There is also a check against homoglyph characters, this being characters that are not ASCII-characters. If a non-ASCII character is found there will also be a warning for the user. See Figure 3.1. When it comes to the warnings given to the user if the levenshtein distance is two or less a warning will be given about this. If the effective distance is one or zero a warning will be given, as well as informing the user where an alteration might have happened and what characters are in question. Meaning if we see an M turning into RN this will be highlighted clearly for the user. Lastly a more general warning is given if there are non-ascii characters in the domain, aswell as showing what characters it applies to. This is quite important as there could be legitimate non-ascii characters like 'æøå' but also deceiving characters like unicode character 'g' (u+0261) that might look like the normal 'g', but is in fact not the same character.

The domains that successfully go through the system are added to the trusted list as mentioned earlier. Currently this is not actually a safe list, consider it more as a soft-whitelist. Domains added have indeed passed many tests, but these tests are based on the current selection of domains that are trusted. Meaning if lets say the domain 'microsoft' is not in the trusted list, a harmful domain 'micr0soft' where the 'o' is swapped with a '0' will be considered trusted and added to the trusted list. One attempt to counteract this flaw was to have an initial list of many trusted domains, but you cannot possibly add them all. This means if a fake domain reaches your inbox before the real one, the system provides a false sense of safety. With this weakness in mind the system still provides great value in the fact that it can protect against already trusted domains. These are domains that you as a person/company already have trust in, and if a mimic of a trusted domain were to say: "we have a great deal now for this and this item. This is the new account number", you are more likely to be tricked than if a new

company however also a scam, were to say the same.

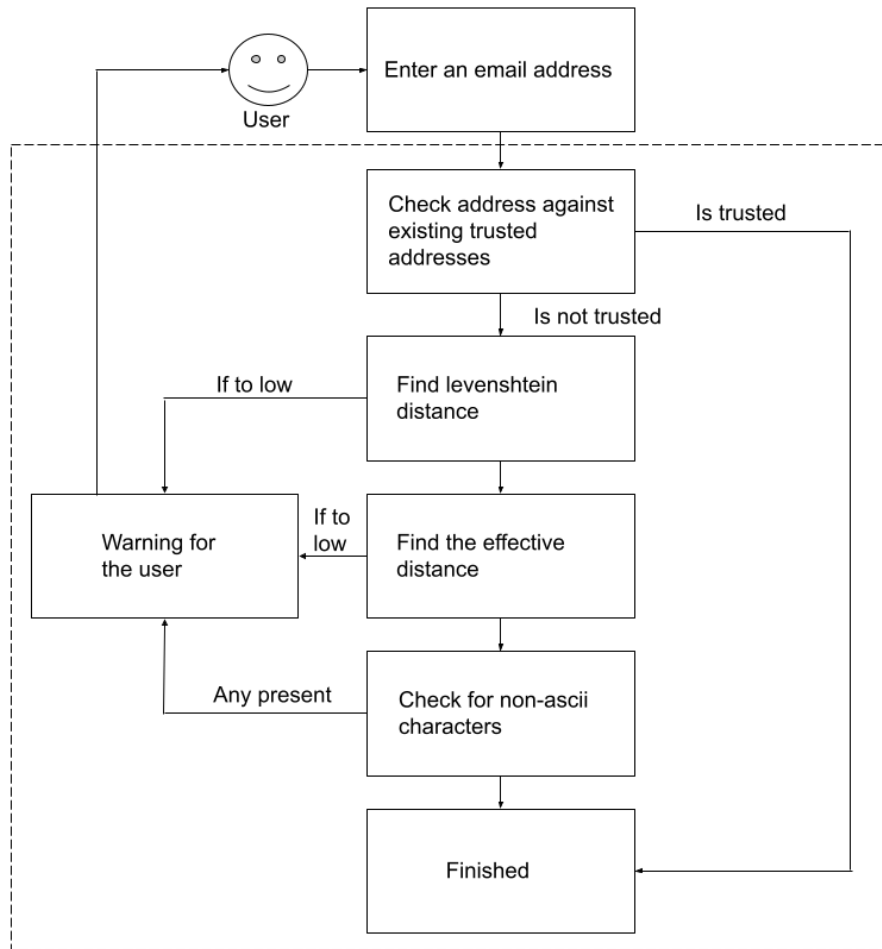


Figure 3.1: This figure shows the flow of the system and what happens when a user enters an email address to be investigated. First it will be checked against existing trusted addresses, if it is not already there levenshtein distance will be performed between this new address and every existing address. Then we will find the effective distance between the new address and every existing address. Lastly check for non-ASCII characters in the new address.

3.4 Technologies used

For this prototype application the programming language Vue.js is used, the IntelliJ IDEA is used to write the code and the GitHub desktop application is used for version control of the code. Vue is a javascript framework, and is used to create single page applications (SPA). Usually Vue is used to create the frontend of an application, but in this case it serves as both the frontend and the backend. Vue is a Model-View-ViewModel (MVVM) framework, which means there is a separation between the markup (HTML), code (JS) and style (CSS). See Figure 3.2. The application also uses the two plugins vuex and vuetify. Vuex is used for handling the data in the application. Whenever data is altered, vuex will automatically update any components relying on that data, automatically "re-rendering" the page. Vuetify is a library with beautiful components like buttons, text-fields and icons to help you quickly create your frontend UI. This is the main reason for picking this technology to develop the application, things can be created quickly and it looks nice, and considering this is just a prototype it is a great fit. It is also something that i was previously familiar with, which in the end allows more time working on the product rather than learning a new tool.

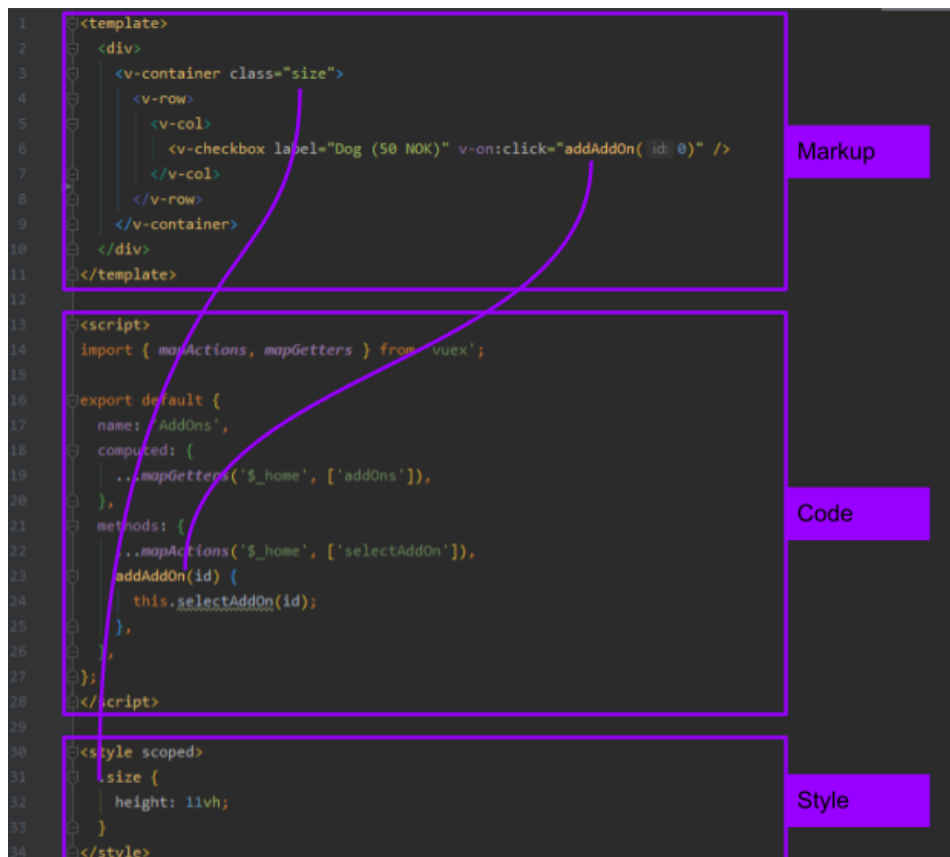


Figure 3.2: This image shows the separation of code, markup and style in a vue file and how they can be connected.

Most technologies would be adequate to develop a prototype like this one, python, java, c++, c# are just some, it really comes down to preference. But to create a deployable ready to use piece of software there are more things to consider. One thing that was considered was making this a plugin for either gmail or outlook, seeing as they have APIs necessary to do so aswell as these are the two clients most people are using. Integrating with existing technologies can in many ways be easier than making something comepletly from scratch.

Chapter 4

Results

In this chapter we will go through how the final system looks and works, and work through a series of scenarios and test for how this system could fend off potential attacks.

4.1 The program

This prototype solution has an input field that can be used to enter email addresses. When entering an address and pressing the send mail button the address will pop up in the inbox section of the program, simulating you receiving an email from that mail address. In figure 4.1 you can see input field.

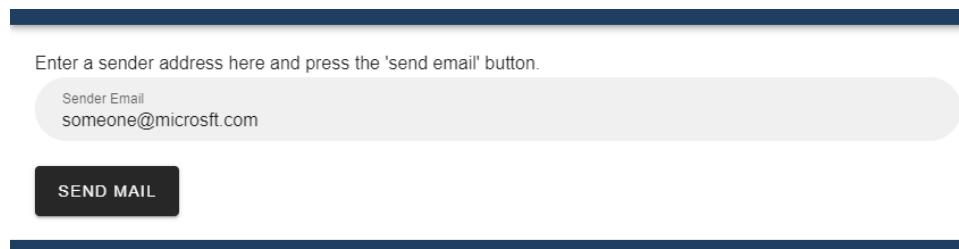


Figure 4.1: This image displays how you could enter an email address and send a pretend email to the system.

After going through a series of tests (discussed in Chapter 3) the mail will be showed in the inbox section with a green thumb signifying everything seems to be in order, a yellow warning advising caution or a red alert sign claiming this is almost certainly an attempted fraud. See figure 4.2 for the inbox and the different warnings that can be displayed.

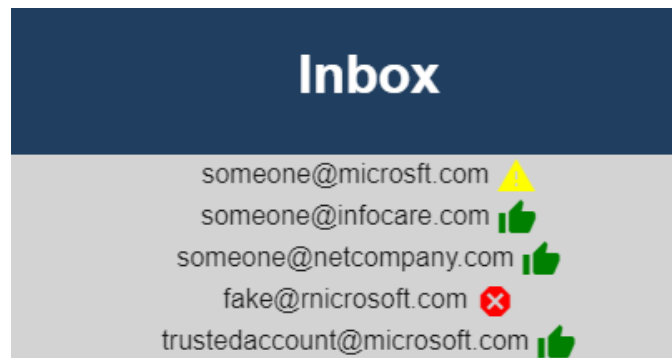


Figure 4.2: This image displays how the initial warnings might look like in a inbox with several mails. You can also see here two attempts at trying to impersonate with the domain Microsoft.

The email domains that got the green thumb is then added to the domain registry, so that it can be used to prevent scams using this brand name in the future. Domains that are not marked as trusted will not be added to the domain list but it was indented to create a feature to manually add domains to the trusted list when it failed and got a warning. This feature was thought of because there is always a chance of receiving false-positives. From the start you can quickly identify which mails that are safe to open, and which you should approach with more caution. You can also click on the emails in the inbox to view what the warning/alert messages are. This is important as you get to show the user why this message triggered a warning pinpointing the exact location, or it could help the user see that this is actually a false-positive. A warning could look like Figure 4.3 where it states what domain it is resembling and how closely it resembles by including effective distance.

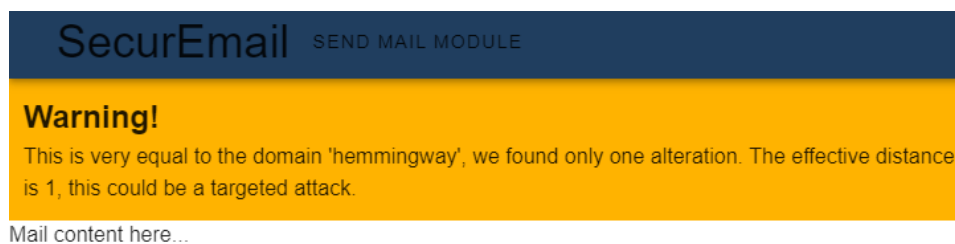


Figure 4.3: This image displays a basic warning message from a new email to the system. This warning was triggered from the domain name of the email address being to similar but not the same of an address already in the system.

You could also receive an alert which is not common to happen by accident, as the domain you received an email from has to be visually equal to a domain you know and consider safe. In most cases this will be an attempted fraud, where one or more exploitable characters have been swapped out. See Figure 4.4

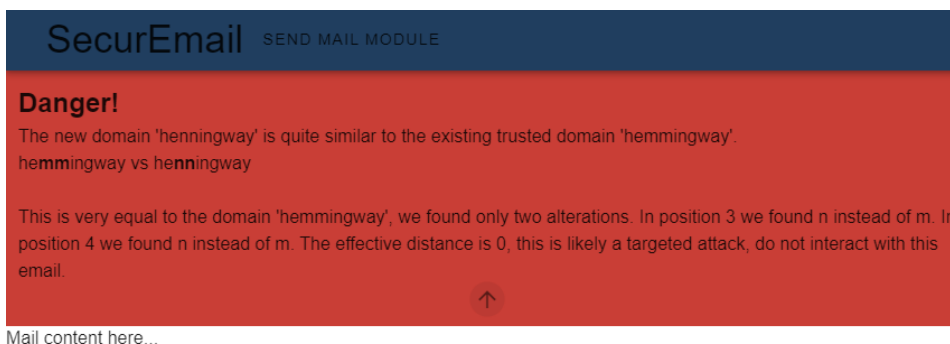


Figure 4.4: This displays the alert you receive when the program suspects attempted fraud. In this case it highlights the domains 'hemmingway' and 'henningway' and that the m's are swapped with n's is highlighted with bold text. It also provides more informations that you can either expand or compress by pressing the arrow button.

Finally the last component on the main page is the domain registry component. This is not something important for the program itself, but was helpful during testing. This component displays all the trusted domains currently in the system, which provides a valuable overview when executing tests towards the system. You can see Figure 4.5 for the full view of the main page.

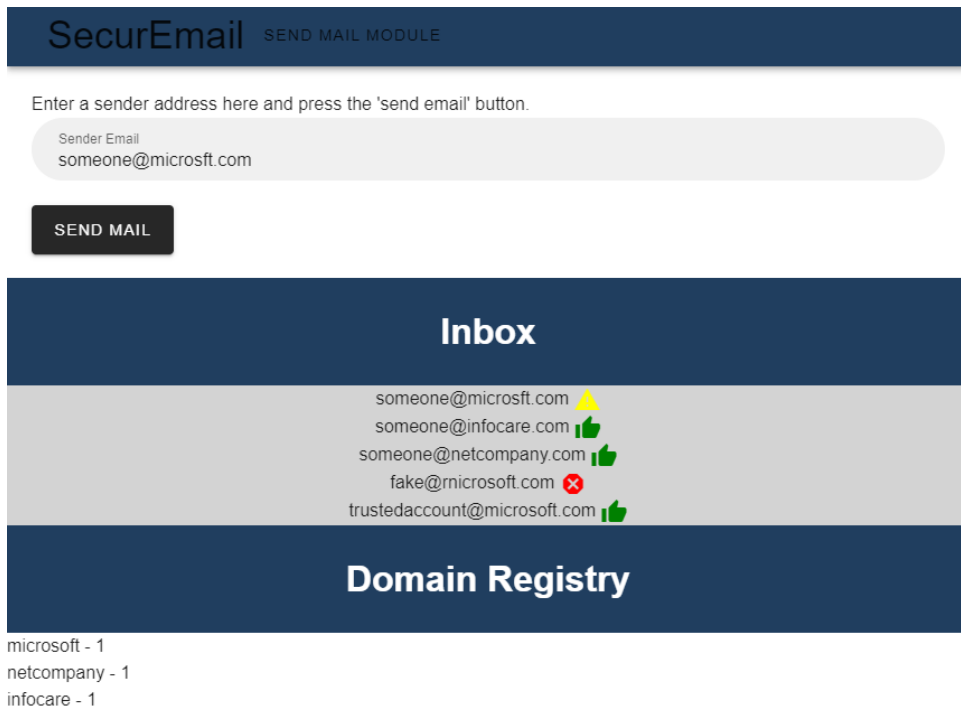


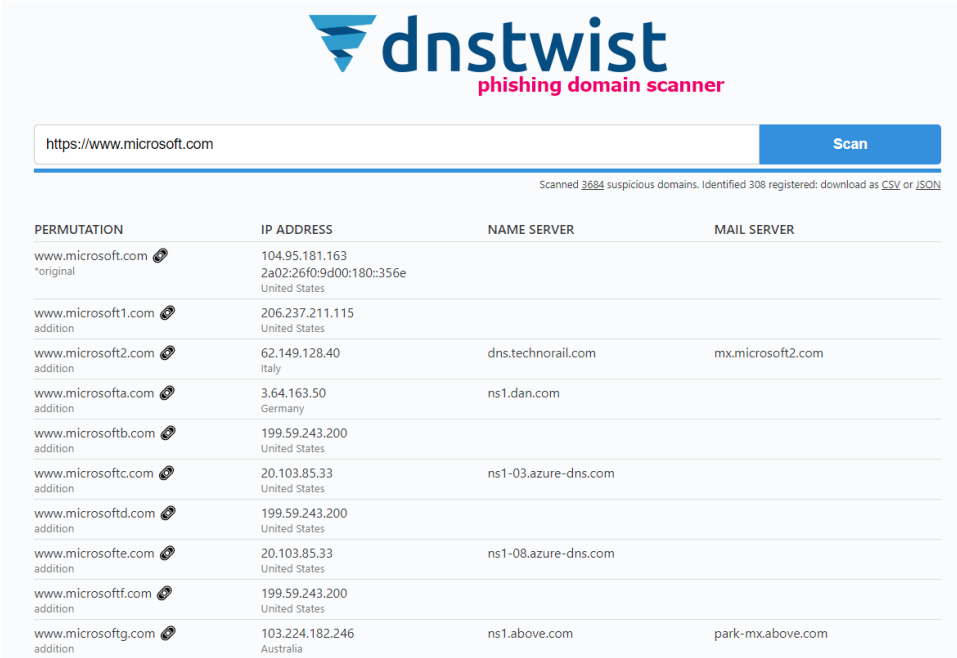
Figure 4.5: This displays the main page of the program. It has a section for adding mails to the system, and inbox to display the current mails, and a domain registry to display the currently trusted domains in the system.

4.2 Testing

For the testing of this system it is divided into two sections, one where DnsTwist is used to find potential scam domains and another part where we try to break the current system with more sophisticated and planned attacks. The second section is more about displaying what types of attacks are blocked, what weaknesses the system currently has, and possibly how we can exploit this and trick the system.

4.2.1 DnsTwist

DnsTwist is used for identifying suspicious webpages that resembles the one you are trying to search for. The quick way to test the program is to use the webpage <https://dnstwist.it>, you can enter a webpage and press scan to find others similar to it. See Figure 4.6. This will find domain names that are similar to the original domain, which can then be put into our system to test how well it detects these differences. It might not be the intended use for DnsTwist, but it provides value to have a third party create the tests as you might find more flaws, compared to you doing all the permutations yourself. The reason for using DnsTwist over a keyword typo generator is that DnsTwist finds real fraud webpages, real existing attempts at fooling people. While a typo generator will just generate permutations based on some selections like skipping letters, adding letters, swapping letters etc. There is no thought behind what characters to swap, how many and how the word looks compared to the original word.



PERMUTATION	IP ADDRESS	NAME SERVER	MAIL SERVER
www.microsoft.com *original	104.95.181.163 2a02:26f0:9d00:180::356e United States		
www.microsoft1.com addition	206.237.211.115 United States		
www.microsoft2.com addition	62.149.128.40 Italy	dns.technorail.com	mx.microsoft2.com
www.microsofta.com addition	3.64.163.50 Germany	ns1.dan.com	
www.microsoftb.com addition	199.59.243.200 United States		
www.microsoftc.com addition	20.103.85.33 United States	ns1-03.azure-dns.com	
www.microsoftd.com addition	199.59.243.200 United States		
www.microsofte.com addition	20.103.85.33 United States	ns1-08.azure-dns.com	
www.microsoftf.com addition	199.59.243.200 United States		
www.microsoftg.com addition	103.224.182.246 Australia	ns1.above.com	park-mx.above.com

Figure 4.6: This image displays DnsTwist and how you can search a webpage to find other webpages with similar names. These domains are then extracted and put in to test this system to see if it can detect them.

The tests with dnstwist were done by first identifying a legitimate domain we wished to test against. Entering that domain to our own program and to dnstwist to receive many fake versions of that domain. Then entering these domains manually to our program to see how it would be handled. It is important to keep in mind dnstwist would provide thousands of permutations for each domain that was tested, and not all of these were tested. DnsTwist categorizes the scam domains for us, some of these categories can be addition, bitsquatting, dictionary, homoglyph, insertion, omission, repetition and so on. To make the testing cover as much grounds as possible I made sure to use some fake domains from each permutation category. This excluding categories like tld-swapping and subdomains as this is not something this project covers yet, but should definitely be a thought for the future. In the first scenario we tested the domain hemmingway as our trusted domain and marked with a green thumbs-up in Figure 4.7. The system here flags all the domains provided by dnstwist with a warning, and even marks one with red, claiming that to quite likely be a targeted attack. The yellow might be attacks, or might not be, they are similar to the domain hemmingway, in this case caution and reading the domain names over is advised. There could be a case where a person is dealing with the legitimate company hemmingway, and also a legitimate company called heimingway, however such resemblance is highly unlikely. However a warning will help prompt the user reading the domain name to act more carefully and make a more educated decision on whether or not to trust the email.

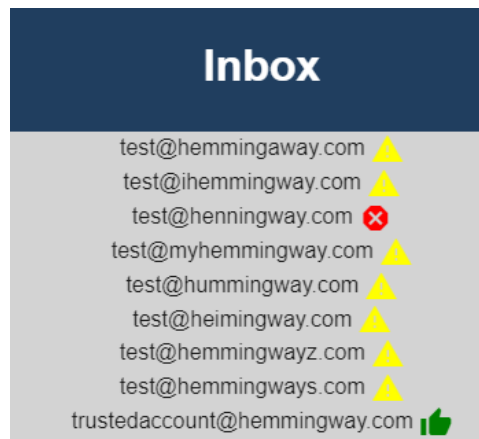


Figure 4.7: Image of testing the domain Hemmingway versus mutations created by DnsTwist.

Using DnsTwist a range of different domains and mutations were tested, some of which can be seen in Figure 4.8, the legitimate domains microsoft, facebook and volkswagen were tested. These mutations include several different styles of attack like inserting characters, removing characters, using homoglyphic characters, replacing and swapping. The system seems to catch most of the mutations received from DnsTwist with at least a warning despite all the different styles of attacks that were applied.

There was one entry that fooled the test but it might not fool the person 'facebookcom'. It is a good attempt towards webpages as they usually have paths following after to help disguise the real tld.

Inbox	Inbox	Inbox
test@microsoft.com ⚠	test@facebookcom.com 👍	test@volks-wagen.com ⚠
test@mikrosoft.com ⚠	test@faebook.com ⚠	test@volkswaghen.com ⚠
test@microsoft1.com ⚠	test@facebo0k.com ⚠	test@volkswagem.com ❌
test@microsmft.com ⚠	test@faeebook.com ⚠	test@volkswagene.com ⚠
test@microsoftf.com ⚠	test@facebook.com ⚠	test@volkswagen.com ⚠
trustedaccount@microsoft.com 👍	test@facebook.com 👍	test@volkswagen.com 👍

Figure 4.8: This image contains the result of the tests done with the help of DnsTwist towards the system. The domains 'microsoft', 'facebook' and 'volkswagen' were tested.

4.3 More complicated problems

This section is more about displaying what is caught and displaying how to not get caught by the program. DnsTwist was great to test the program and see how it would respond to the different mutations, however a common thing about all the fake domains from DnsTwist; they had too few changes. Many of the words looked fairly equal to the original domain, but got caught because levensthein only found one or two distance difference between the words. To break the system we need to create a higher distance between the words, while somehow keeping them visually equal. As mentioned earlier in this paper the character 'm' is a great tool for increasing length when swapped with the 'r-n' sequence. But this is something that is monitored as you can see in Figure 4.9 where 'hemmingway' is compared to 'hernningway'. The distance is four, but the program identifies both of the 'm' to 'r-n' swaps and finds the effective distance to be zero.

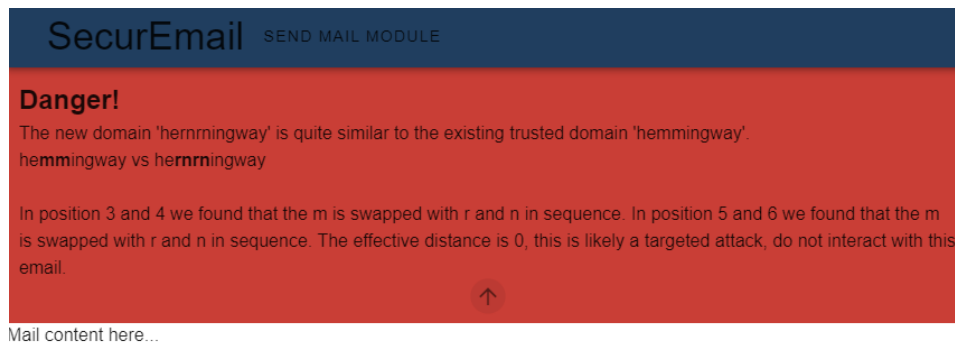


Figure 4.9: This shows an image of the alert when comparing 'hemmingway' to 'hernningway'. The system has found both 'm' to 'rn' swaps, and stops this attack even with 4 as the distance.

To avoid any alerts and warnings we need our distance to be equal or less than 2, while the effective distance is equal or less than 1. Currently our distance is 4, and our effective distance is 0, but there is another trick we could use to increase the effective distance; adding misspellings. Misspellings can be a great tool, and especially misspelling that are located closer to the end of the words as the human mind often makes assumptions and not read the word character by character [28][5]. By swapping two characters positions we would accomplish this no longer being detected by the program, while still looking fairly similar: 'hernrninw~~g~~ay' or 'micorsoft' are two examples of this. Another thing that could also bypass the program is by adding characters before a swap between 'm' to 'r-n', as this would disrupt the algorithm looking for these common character alterations. If we were to take the word 'wallmart' and add an extra l while using the 'r-n' swap for the m, we would get 'walllrnart', which looks fairly similar to the original. Table 4.1 displays that the 'm' in the original wallmart will then be compared with the 'l' of the fake wallmart.

W	A	L	L	M	A	R	T		
W	A	L	L	L	R	N	A	R	T

Table 4.1: A table showing 'wallmart' and 'wallmart' to display how the algorithm checking for 'm' to 'rn' swaps can be tricked by adding a character somewhere before.

By adding this 'l', the characters after are shifted one spot, meaning we now compare 'm' to 'l' and wont find that the 'm' has in fact been swapped with 'r-n'. However not every company name has an m that can be exploited, and without the m it can be difficult to bypass the system. Here once again the best results were by triggering our inner dyslexia by shuffling the characters around. This also means that the same characters are being used, resulting in exact same length as the original domain. Figure 4.10 shows a successful attempt that tricks the system while maintaining rather camouflaged. It must be said that swapping characters around to fool your victim only works when the domain is rather long. On shorter domains there was no way of tricking the system and maintaining a good camouflage, at least not any that I could find.

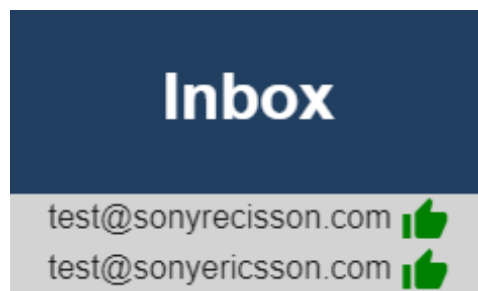


Figure 4.10: Showing a rather successful attempt at fraud with the domain 'sonyericsson' and the fake 'sonyrecisson'. Here two character swaps has been made to make the distance 4, while maintaining a good appearance.

Chapter 5

Discussion

Sophisticated cybersecurity attacks today often include thorough reconnaissance and planning from the attackers, which leads to more tailored attacks towards high value targets. One trick attackers might use is to disguise their domain name and impersonating someone else. This prototype tries to work against these types of attacks and in this chapter we will discuss the final product, the results what the system does well, its weaknesses and what needs to be considered in the future.

5.1 Provides value

Phishing is an increasing problem and everything is pointing towards more of it in the future [12][11][23][37]. Most of the attacks are sent with e-mail, but increasing attacks are happening over SMS and other messenger applications like Facebook's messenger. Most of the attacks are not tailored to attack just you, but rather a large amount of people at the hopes that just a handful fall into their trap. From an early age we are taught to thread carefully on the internet, not to trust that something is free and to never press URLs from people you don't know. So why are these seemingly poorly made spam attacks increasing? Technology is enabling this process to happen basically automatically, the phishers create a new scam and distribute it through a new email to many victims at the same time. Another interesting thing is to look at the demographics, especially age, fraud reports from the FBI 2020 show that the most victims of internet scam are elderly, despite a much larger percent of internet users being younger [10]. Could these poorly made scams still be around because of the elderly generations lacking internet education. Over the past years google gmail and microsoft outlook has been able to reduce successful attacks with spam filters. What happens when these types of attacks stop paying the bills for the phishers? They will evolve, and have done so already as there is more money to be made when infiltrating deeper and targeting higher value targets. One weakness that has proven in the past to be quite successful is simply sending your scam from hijacked accounts to the friend list of that person. This is effective because you think the URL you press is safe, because you received it from a friend. If you use any social media accounts like Facebook

or Instagram you might have been subjected to these types of attacks yourself in the past. This type of attack requires you to first hijack someones account, which is not always easy, but something that is easy is pretending to be someone else.

Pretending to be someone else brings us to our topic, and is something that has been made a whole lot easier with the evolution of technology. Pretending to be someone else in real life is difficult, you would have to change the way you look, your voice and probably many more indicators not mentioned here. Pretending to be someone else online is rather easy because you do not need to show yourself, you can hide behind your computer and make wild claims of who you are. When it comes to claiming to be someone using e-mail we have luckily come some way since the early days. Thanks to protocols like SPF, DKIM and DMARC you can no longer do the equivalent of writing a letter claiming to be president Obama and deliver it [20][1][21][15]. These protocols ensures that someone else cannot just use your exact e-mail address for their own e-mail. But something that still remains a problem is creating e-mail addresses that have close resemblance to the original. To counteract this many email clients today have now included a feature that tells you if you have not received email from that person before, or if they are outside your domain. See Figure 5.1. This is truly a great addition that can stop many impersonation attacks over e-mail. But is this kind of warning enough and does it cover all types of impersonation attacks?



Figure 5.1: Image showing a warning in the outlook client when receiving a message from a new person, who is also outside of the domain you use.

To answer the question above, no this is not enough. There are more ways of impersonating someone than just their name in their email, they could also change the domain. Most private emails will use one of the standard domains like gmail, hotmail, outlook etc. But this is not set in stone, it can be anything and companies might like the domain to be the name of their company. When it comes to the domains many especially companies depend on other companies to run their businesses, and will therefor also be in contact with other domains that are different from their own. The warning saying this is the first time you have been in contact with this person can quickly be overlooked when considering the following scenario. Your

company often deals with susy@papersupplier.com, but suddenly you get an email from mark@papersuplier.com (missing one p), you will see the warning saying this is the first time you get a mail from mark, but you will also think this is just another employee at the company papersupplier. The scammer mark has now earned your trust, and with this trust is in a good position to spread a virus, try to extract value or phish for information and further exploits.

These are the types of attacks that this program is effective at handling. Small alternations in the domain making it look similar to the original domain. You wont get a message that this is just a new contact, but rather this is a person trying to disguise themselves as a domain you know and trust. Not only that, you will also get warning and highlighting that will pinpoint you to exactly where the differences are, allowing the user to take a more educated decision if this is an attempted fraud or not. Showing where the problem lies is important as it is not always the case that the domain is faulty even if the program reacts to it.

It is important to note that most attacks are not based around a disguised domain name, but rather use one of the well known @gmail or @outlook domain names. It is also important to emphasise this thesis is not meant to fix the problem of phishing, but rather addressing one aspect of it and making things more difficult for phishers in the future. There are countless technologies in play, and currently being brainstormed that together help fight phishing. This can be firewalls, browser blacklisting, spam filters as well as newer approaches like machine learning, cloud platforms, protocols like SPF, DKIM and DMARC, using bots for URL breakdown and much more. Using machine learning for content based detection is looking more and more promising, however i don't believe this alone will ever be good enough. These approaches are content based they will look for keywords in the text, filtering, blacklisting, profiling, bad language, labeling etc. In addition to this machine learning approaches require large amount of data do work properly. What happens when the phisher doesn't type anything to raise the alarms but rather use the trust of a disguised domain name to forward the conversation to a voice chat, or a instant messenger application like telegram where they could speak in private. Instant messenger applications like telegram requires you to know the other party through adding them in order to speak, which is also why they don't have a spam filter or any form of anti phishing software.

From the results we witnessed the program being effective at handling fake domains provided by DnsTwist. Using DnsTwist is interesting as it provide real fraudulent webpages, which offers a somewhat more realistic testing environment. The domains from DnsTwist were mostly domains containing one or two alterations from the original domain. An important thing about these types of attacks is that it has to trick the defending program as well as the person using it in order to be successful. To trick the program enough misspellings or alterations need to be made, this will

however make it more likely the human can spot that something is wrong. Doing the tests were extremely difficult because you had to balance what could fool you, what could fool the program and what could fool both at the same time. Seeing what could fool the program was the easy part, but trying to judge in honesty what could fool me was impossible. There are so many factors that applies into a fake domain attack, the most important being you don't know you are being attacked. Whenever i would read the fake domains myself i would see they were fake, because i was looking for it, i already knew they were fake. This was especially hard in the second part of the testing where we played the attacker. During this stage it was all about trying to trick both myself and the system in order to display its weaknesses. The approach was to stick with using the known methods to disguise domains, like swapping similar characters and alternating positions between two non-prominent characters as this would give the best result in order to trick me. It is hard to say if assessing whether these domains were disguised enough or not was done adequately. There is a chance i evaluated the fake domains to harshly, and there is a chance that i was to lenient.

This is a program that defends against fake domain names, but maybe the most unique part about this program is that it learns. It is designed to protect you against people impersonating domains that you have conversation with. This means as you are using the program and building your network it includes your new relations and makes sure their names/domains cannot be used against you in the future. If you speak with @papercompany with email, that is something the phishers might figure out, but can no longer use against you. This solution compared to our current ways of blacklisting and whitelisting is a much better approach. The problems with blacklisting are obvious, you can blacklist the email of a phisher, but nothing stops him from making another. Blacklisting is something that happens after an attack, which means if the attack was successful you wouldn't know. Whitelisting can only work if you know exactly who should have access to you at all times. If a company relies on getting a constant flow of new messages, their email address cannot be locked to the invited people only. The program will not give any defence against domains you might know but have not had previous conversation with.

One could say these types of attacks can be prevented with proper user training through anti-phishing training courses [6][36]. Companies spend large amount of money on training their employees to identify scams [37][23]. Receiving proper training is important to identify phishing and staying safe online, but training alone is not enough. People cannot stay on-game with razor sharp focus at all times, in the end people make mistakes. Reading mail number 40 for the day it might be hard to spot that one character difference in the email domain name. We humans are also at a great disadvantage when processing these kinds of scams due to how our brain works. When reading a word or a sentence our brain will try to predict what the outcome is before we read it [28][5]. This means if you

do not fully concentrate on reading something character by character your brain is likely to fix whatever mistake in the domain name without letting you know it happened. Which is why we need help, having something to tell you about this difference and show you where to look makes it easy to not fall for the trap. There are people that believe training people is the way to go, and there are people that think we need to detect phishing without counting on people. Obviously being able to detect phishing automatically and without human assistance would be the best course of action, but we are not there yet. We need to count on both, technology to give pointers and a anti-phishing trained human to process this information and decide.

5.2 Future works/Limitations

5.2.1 False-positives

One of the bigger flaws with the current system is the amount of false-positives we are getting. This applies mainly whenever the character lengths drops below five. This makes sense as the same rules apply for any domain that enters the system, and domains with less characters means more likely to be similar to some other domain. Lets consider the domains 'IBM' and 'FBI', they have only three characters each and we can clearly distinguish them from one another. However if you were to enter this into the system it would be flagged as being too similar, simply because they share the middle 'B'. There needs to be different rules applied as the character length of the domain drops, maybe in the case of three characters we allow one alteration, but trigger if the effective distance is zero. More problematic is the middle ranged domains like 'Cisco' versus 'Costco' which in the current system will be a false-positive. Where exactly to draw the line can be difficult as by making the rules less strict you might reduce false-positives but might open the system up for failing to see real threats. One might think getting some false warnings every now and then is justified by catching more real threats. However giving frequent false warnings can also be quite dangerous and will have the same effect as the boy who cried wolf.

5.2.2 Warnings and alerts

The warnings for this system is split into two, warning (yellow) and alerts (red). The main reason for splitting it up into warning and alerts was to differentiate between what could potentially be a scam, and what is most certainly a scam. It was also considered using a percentage to rate the new domain more accurately, but this was not implemented. The way warnings and alerts work at the moment can be explained as adequate. When there is an alert it is in most cases for a good reason, whenever there is a warning in most cases this is also true with the occasional false-positive. But there are cases where the warnings should probably be treated as an alert instead of just a warning. Currently the system will only create an alert if the effective distance is equal to zero. This means if the domain 'microsft' that is missing an 'o' is only gonna get a warning, while in truth it is so equal that it deserves an alert. Whenever there is only a distance of one between these words of higher character number they should be given alerts instead of warnings. The higher character number also means it should be less likely that two legitimate domains will trigger one another.

5.2.3 First is trusted

A massive weakness right now is that the policy is really "first domain in is trusted". The system is designed to defend against phishing attempts that use company names that you recognize and trust against you. They

do this by creating a new domain that looks very similar to the original real domain. But if you receive an email from a fake domain '@anazom' before you receive one from the real '@amazon', the fake one will be added to the trusted list of domains, and it would flag @amazon as the fake simply because of the order you received the emails. This is obviously not ideal because if a trusted domain is not yet added to the system, there is no defence against it, the system will flag the domain as 'cleared' and will give a false sense of safety towards the user.

There are some ways to get around this, one could be to add a standard trusted list of the top 500 or so company domains as these would probably be the most targeted. That might help against more standard generic attacks that are sent out to many people, but it would not help anything against an attack carefully planned and targeted towards a single company or person that uses a domain not in this list. In addition to this the phishers would quickly figure out what these 500 domains are and simply not use them in future attacks, as they know they will be stopped. Adding every single good domain would be the best, however this is not viable for several reasons. Not only is it impossible to identify every single good domain, but new ones are created every day, also the system would not be able to handle checks against this many domains. Adding everything is also not important as each person/company only have some domains that they communicate with frequently.

Maybe we could create a component for adding these trusted domains to the client. Similar to a whitelist containing the same flaws, it is time consuming, is it something the user would take the time to do? It is also something continuous that has to be done, as new companies are made and start cooperating with you. There is no way to make a list, and remain safe for eternity. This would solve the problem above if the user takes the time to add the most important domains before using the program. But even this would still leave you vulnerable as the phishers could take advantage of what type of company you are, and figure out what you need. Lets consider a law firm, they are in need of a ridiculous amount of paper to run their operations and are sure to have deals with some sort of paper company 'bestpaper' for example. If the phishers know this and know that the current paper provider is 'bestpaper' they could impersonate another paper company claiming they can provide paper for them cheaper. This would then not be discovered as this new fake paper company is not added to the list of trusted domains.

It is clear that we cannot add a bunch of domains to the list and hope to be safe. There needs to be a way to help the user identify if new domains to the system are safe as you receive them. Whenever a new domain comes in to the system, a clear warning of this is given to the user, prompting some sort of step by step approach for the user to make sure this is a safe domain. For such an idea there are many things that could be added, just to generally help guide whether to trust this or not. Clearly separating

the characters, warn for homoglyphic characters, generating a link for a google search prompting the user to do just a tiny amount of research on this new entry. Yes, this could become very tedious, especially in the beginning when no domains are trusted and you have to do this process for every email you receive. But as you use it, and as the list of trusted domains grow the less times you have to do this process. The reward for doing this is that you know the emails you receive are from a trusted source, at one point in time you have verified them yourself and any malicious attempts made towards those domains will be recognized. Checking out a new domain or using google to identify if a URL is safe to press is not something new. It is a good way of testing validity, if your google search gives a low result or points you in the direction of something else that is similar you shouldn't trust it. Like all manual work, it takes time and things that takes time are often neglected, but maybe by making this research process easier by generating the links for the google search and guiding the user in what to look for would be enough to make it worth the effort. One of the biggest flaws with security is that not enough information is provided to the user. If we consider a URL, that could be broken down into many different parts that are all important to understand to remain safe from a malicious URL. There are however only a fraction of people that knows these parts by heart, and can navigate the good features and the bad. This can be something simple as knowing that the http protocol is not an encrypted protocol and you should not send any data on a webpage without the https protocol. Informing the user about small things like this can help them avoid getting tricked.

5.2.4 Mutations

After working with this project for a long time i found a rather cool idea that could solve some of the more complicated problems slipping through the system. Words that have a distance more or equal to 2 and effective distance more or equal to 1, but still visually look the same as the original. An example of this mentioned earlier in the paper is the domain 'wallmart' versus 'wallrnmart', having an extra 'l' and the 'rn' swap for the m. With the current system this would slip past and not be flagged, but what if we made a mutation of the new domain 'wallrnmart' by changing the 'rn' to 'm', and then running this mutation against the original trusted domain. We would then get 'wallmart' vs 'wallllmart' and find that the distance is only one, and a warning would be given. This takes the idea of common character alterations and flips it around, instead of checking every trusted domains vulnerable characters we make mutations of the new domain in question and check every mutation against the trusted list. Running levenshtein distance on every mutation would give us mutation distance to the trusted domains, so even if the original distance in the case of the 'wallmart' example is 3; a mutations of this only has a distance of 1. This idea has not been tested in any way and could lead to new unforeseen problems such as becoming more resource demanding as you are no longer just checking your trusted list, but your trusted list times the amount of muta-

tion the new domain has. There could also be a problem that this increases the amount of false-positives, as you are checking so many more combinations.

W	A	L	L	M	A	R	T		
W	A	L	L	L	R	N	A	R	T
W	A	L	L	L	M	A	R	T	

Table 5.1: This shows how we can mutate the fake domain wallmart as we identified 'r-n' which can be expressed as 'm'. By doing this change we now have a mutation wallmart which only has 1 in difference between the real walmart.

In addition to this we also have the 'sonyericsson' incident where the fake 'sonyrecissson' eluded the program without any common character alterations, but simply with character position swaps. Figure 4.10. Creating mutations for character swaps is not as easy, and would lead to far to many mutations and false-positives. One approach that could be promising is taking these mutations of common character alterations discussed in the paragraph above and counting the characters versus the domains in the trusted list. In the case of the 'sonyericsson' example above you would count occurrences of all the characters in both domains.

S	O	N	Y	E	R	I	C	S	S	O	N
S	O	N	Y	E	R	I	C				
3	2	2	1	1	1	1	1				

Table 5.2: Showing a character summary of the domain 'sonyericsson'.

S	O	N	Y	R	E	C	I	S	S	O	N
S	O	N	Y	E	R	I	C				
3	2	2	1	1	1	1	1				

Table 5.3: Showing a character summary of the fake domain 'sonyrecissson'.

As you can see in tables 5.2 and 5.3 they have the exact same amount of characters. Considering the length of these two words it is highly unlikely two domains have exactly the same amount of equal characters by accident. This type of check could be added to the program and would provide more defence against these types of character swap attacks. Similar to the other method of detecting maybe allow 1-2 character difference as it would still be fairly similar, depending on the length of the domain in question of course.

5.2.5 Further development

There has been some interesting ideas discussed in this chapter like creating mutations of incoming domains and test mutations for similarities to the trusted list, and counting the exact amount of each characters and using that to compare. There are probably more ideas out there that has not yet been thought of. The next step however is not to perfect the program right away, it is to reach the user base with what we got. This is because what we got right now already provides value as it is. What is created here is only a prototype, meant to be an easy way to test ideas and see them in practice quickly. To reach the users it is ambitious to believe they will come to you, to reach the most users you will have to meet them. Currently the most used clients for email are google gmail and microsoft outlook splitting nearly all of email users among themselves. Giants such as google and microsoft has luckily made things quite easy for developers in the form of creating APIs for everything, including their mail clients. So the next natural step would be to make some sort of plugin, that achieves what this prototype does. Mentioned earlier in the report it was considered making this a gmail plugin from the beginning. This was put a side as more flexibility and independence were prioritized in order for the idea to develop more freely. Starting this as a prototype project allowed for quick changes on the go, changes that would've been much more tedious to do if in addition relying on a third party software such as google. The ideas of this project were far from complete at the beginning of the project period, many changes has been done during this time, and doing the prototype first has in many ways enabled a better end result.

5.3 Performance

A stress test was performed on the system to see how well it would handle an increasing number of domains within the system. There are quite a few calculations happening so it is interesting to look at what sort delays we might expect in the future in a very busy system. When a new domain is added it will be compared to all the domains in the trusted list, then it will find levensthein distance against every single one, and finally it will check all the exploitable characters in every domain against the new one. A total of 100 unique domains were added and the time it took to successfully add the new domain was taken. Even at 100 domains the time remained under 1 millisecond from start to finish. This looks promising as the idea with the stress test was to check if the system could handle 100 entries, and it proved it could do way better than that. The increase in time per entry is linear meaning the time will steadily increase as more entries are added.

Some interesting notes from this stress test is that the domain 'ASML' got a warning for looking to similar to 'TSMC'. The domain 'Costco' got a warning of being to similar to 'Cisco' and the domain 'SAP' got a warning for being to equal to 'BHP'. This means we received three false-positives in 100

domains to the system which is more than it should be. It worth noting that all of these false-positives contain few characters, for instance 'SAP' and 'BHP' a human could clearly see right away does not look like each other, but the computer only find a distance of two between them.

Chapter 6

Conclusion

Phishing has been around since the dawn of internet and will stick around in the time to come. In this thesis we explored ways of defending against phishing emails by detecting fake domain names. To achieve this, a prototype application was developed to test and visualize how such an idea could work in practice. This program works by checking new entries to the system against a trusted list of domains, and looking for similarities. This trusted list would then grow as domains successfully passed the requirements. The program is effective at spotting small alterations but can be extended with several improvements in regards to detecting more advanced alteration techniques and limiting false-positives. Some of these shortcomings were discussed and should be possible to solve with further development. The program provides value as it is efficient at dealing with impersonation attacks that uses a fake domain to establish trust. What makes these attacks especially dangerous, is that we initially think we can trust them because we think it is from someone we know. We treat messages differently, a URL from a stranger we don't know is easy to disregard but for one from a friend we might press before even looking at the URL. These attacks doesn't necessarily look to extract value right away in the form of a malicious URL or attachment, but extracting information and building trust in order to do something down the line, using content based approaches is not effective against these types of attacks as they look for malicious content text, URL or attachments. Many experiments were done towards the program to test its efficiency on spotting fake domain names. Most of these experiments were done with the help of DnsTwist to provide us with fake domain names. DnsTwist provided a unique way of testing the program, as it does not simply generate domains for the test, it provides real fraudulent domain names that has been used to trick people in the past. These domains use a mixture of common alteration techniques like swapping character positions, adding misspellings, removing characters and many other tricks. These seemingly legitimate domains were however no match against this prototype program and were discovered and marked with either a warning or alert.

This solution provides us with information to make the right decisions. We

should not be expected to be able to spot misspellings and tricks designed to fool us. To be able to spot a misspelling, you need to be actively searching for it and reading who we received an email from is not something we do with a loupe. As stated in the introduction this thesis was not intended to solve phishing, but addressing one aspect and making it just a tiny bit more difficult for the phishers. Making it more difficult is the end goal, as making something completely safe on the internet is impossible.

Bibliography

- [1] "Crocker, D., Ed., Hansen, T., Ed., and M. Kucherawy, Ed." *Domain-Keys Identified Mail (DKIM) Signatures*. 2011. URL: <https://www.rfc-editor.org/info/rfc6376>.
- [2] Shivam Aggarwal, Vishal Kumar and S. D. Sudarsan. 'Identification and Detection of Phishing Emails Using Natural Language Processing Techniques'. In: *Proceedings of the 7th International Conference on Security of Information and Networks*. SIN '14. Glasgow, Scotland, UK: Association for Computing Machinery, 2014, pp. 217–222. ISBN: 9781450330336. DOI: 10.1145/2659651.2659691. URL: <https://doi.org/10.1145/2659651.2659691>.
- [3] Zainab Alkhalil et al. 'Phishing Attacks: A Recent Comprehensive Study and a New Anatomy'. In: *Frontiers in Computer Science* 3 (2021), p. 6.
- [4] Kholoud Althobaiti, Kami Vaniea and Serena Zheng. 'Faheem: Explaining URLs to people using a Slack bot'. In: Apr. 2018.
- [5] Moshe Bar. 'The proactive brain: memory for predictions'. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 364.1521 (2009), pp. 1235–1243.
- [6] Gamze Canova et al. 'NoPhish app evaluation: lab and retention study'. In: *NDSS workshop on usable security*. 2015.
- [7] Neil Chou et al. 'Client-Side Defense Against Web-Based Identity Theft'. In: Jan. 2004.
- [8] 'Cyber Kill chain'. In: URL: <https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html>.
- [9] Sevtap Duman et al. 'EmailProfiler: Spearphishing Filtering with Header and Stylometric Features of Emails'. In: *2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*. Vol. 1. 2016, pp. 408–416. DOI: 10.1109/COMPSAC.2016.105.
- [10] FBI. 'Elder Fraud Report'. In: (). URL: https://www.ic3.gov/Media/PDF/AnnualReport/2020_IC3ElderFraudReport.pdf.
- [11] Anti-Phishing Working Group. In: (). URL: https://docs.apwg.org/reports/apwg_trends_report_q4_2021.pdf?_ga=2.18223225.191419467.1646825763-783082516.1644834354&_gl=1*16ohr5r*_ga*NzgzMDgyNTE2LjE2NDQ4MzQzNTQ.*_ga_55RF0RHXSRTY0NjgyNTc2My4yLjEuMTY0NjgyNTc3NS4w.

- [12] Anti-Phishing Working Group. In: (). URL: https://docs.apwg.org/reports/apwg_trends_report_q4_2020.pdf?_ga=2.59653869.191419467.1646825763-783082516.1644834354&_gl=1*e9sl22*_ga*NzgzMDgyNTE2LjE2NDQ4MzQzNTQ.*_ga_55RF0RHXSRTY0Njg3MTY2Mi4zLjEuMTY0Njg3MTczMC4w.
- [13] Grant Ho et al. 'Detecting and Characterizing Lateral Phishing at Scale'. In: *28th USENIX Security Symposium (USENIX Security 19)*. Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1273–1290. ISBN: 978-1-939133-06-9. URL: <https://www.usenix.org/conference/usenixsecurity19/presentation/ho>.
- [14] Grant Ho et al. 'Detecting Credential Spearphishing in Enterprise Settings'. In: *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 469–485. ISBN: 978-1-931971-40-9. URL: <https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/ho>.
- [15] Hang Hu and Gang Wang. 'End-to-End Measurements of Email Spoofing Attacks'. In: *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, Aug. 2018, pp. 1095–1112. ISBN: 978-1-939133-04-5. URL: <https://www.usenix.org/conference/usenixsecurity18/presentation/hu>.
- [16] Xuan Hu et al. 'Detecting Compromised Email Accounts from the Perspective of Graph Topology'. In: *Proceedings of the 11th International Conference on Future Internet Technologies*. CFI '16. Nanjing, China: Association for Computing Machinery, 2016, pp. 76–82. ISBN: 9781450341813. DOI: 10.1145/2935663.2935672. URL: <https://doi.org/10.1145/2935663.2935672>.
- [17] Oliver J Hunt and Ivan Krstic. 'Preventing URL confusion attacks'. In: (2017).
- [18] Md Mazharul Islam, Ehab Al-Shaer and Muhammad Abdul Basit Ur Rahim. 'Email address mutation for proactive deterrence against lateral spear-phishing attacks'. In: *International Conference on Security and Privacy in Communication Systems*. Springer. 2020, pp. 1–22.
- [19] Mahmoud Khonji, Youssef Iraqi and Andrew Jones. 'Mitigation of spear phishing attacks: A Content-based Authorship Identification framework'. In: *2011 International Conference for Internet Technology and Secured Transactions*. 2011, pp. 416–421.
- [20] Scott Kitterman. *Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1*. RFC 7208. Apr. 2014. DOI: 10.17487/RFC7208. URL: <https://www.rfc-editor.org/info/rfc7208>.
- [21] Murray Kucherawy and Elizabeth Zwicky. *Domain-based Message Authentication, Reporting, and Conformance (DMARC)*. RFC 7489. Mar. 2015. DOI: 10.17487/RFC7489. URL: <https://www.rfc-editor.org/info/rfc7489>.

- [22] Kyumin Lee, James Caverlee and Steve Webb. 'The social honeypot project: protecting online communities from spammers'. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 1139–1140.
- [23] James Lewis. 'Economic impact of cybercrime- No Slowing Down'. In: 2018. URL: <https://www.mcafee.com/enterprise/en-us/assets/reports/restricted/rp-economic-impact-cybercrime.pdf>.
- [24] Microsoft. 'Protect yourself from phishing'. In: (). URL: <https://support.microsoft.com/en-gb/windows/protect-yourself-from-phishing-0c7ea947-ba98-3bd9-7184-430e1f860a44>.
- [25] Chee Keong Ng, Lei Pan and Yang Xiang. 'Introduction to Honey-pot'. In: *Honeypot Frameworks and Their Applications: A New Framework*. Springer, 2018, pp. 1–5.
- [26] Matthew B Prince et al. 'Understanding How Spammers Steal Your E-Mail Address: An Analysis of the First Six Months of Data from Project Honey Pot.' In: *CEAS*. 2005.
- [27] Issa Qabajeh, Fadi Thabtah and Francisco Chiclana. 'A recent review of conventional vs. automated cybersecurity anti-phishing techniques'. In: *Computer Science Review* 29 (2018), pp. 44–55. ISSN: 1574-0137. DOI: <https://doi.org/10.1016/j.cosrev.2018.05.003>. URL: <https://www.sciencedirect.com/science/article/pii/S1574013717302010>.
- [28] Mathias Scharinger et al. 'Predictions interact with missing sensory evidence in semantic processing areas'. In: *Human Brain Mapping* 37.2 (2016), pp. 704–716.
- [29] SecurityScorecard. In: (2021). URL: <https://securityscorecard.com/blog/types-of-phishing-attacks-and-how-to-identify-them>.
- [30] Steve Sheng et al. 'An Empirical Analysis of Phishing Blacklists'. In: (Jan. 2009).
- [31] Steve Sheng et al. 'Anti-phishing phil: the design and evaluation of a game that teaches people not to fall for phish'. In: *Proceedings of the 3rd symposium on Usable privacy and security*. 2007, pp. 88–99.
- [32] Gianluca Stringhini and Olivier Thonnard. 'That Ain't You: Blocking Spearphishing Through Behavioral Modelling'. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Ed. by Magnus Almgren, Vincenzo Gulisano and Federico Maggi. Cham: Springer International Publishing, 2015, pp. 78–97. ISBN: 978-3-319-20550-2.
- [33] 'Thunderbird'. In: URL: <https://www.thunderbird.net/en-GB/>.
- [34] Jingguo Wang et al. 'Research Article Phishing Susceptibility: An Investigation Into the Processing of a Targeted Spear Phishing Email'. In: *IEEE Transactions on Professional Communication* 55.4 (2012), pp. 345–362. DOI: 10.1109/TPC.2012.2208392.
- [35] Chris Weber. 'Unicode Security Guide'. In: (). URL: <https://websec.github.io/unicode-security-guide/visual-spoofing/>.

- [36] Yue Zhang et al. 'Phinding phish: Evaluating anti-phishing tools'. In: (2007).
- [37] Eugenia Lostri Zhanna Malekos Smith and James A. Lewis. 'The Hidden Costs of Cybercrime'. In: 2020. URL: <https://www.mcafee.com/enterprise/en-us/assets/reports/rp-hidden-costs-of-cybercrime.pdf>.

Chapter 7

Appendix

7.1 LevenstheinDistance.js

```
export default function getLevenstheinDistance(string1, string2) {
  const matrix = [];

  // Set up base matrix
  for (let i = 0; i < string1.length + 1; i += 1) {
    const row = [];
    for (let j = 0; j < string2.length + 1; j += 1) {
      row.push(j);
    }
    row[0] = i;
    matrix.push(row);
  }

  // Check current against [-1][-1], [-1][0], [0][-1]
  for (let i = 1; i < string1.length + 1; i += 1) {
    for (let j = 1; j < string2.length + 1; j += 1) {
      if (string1[i - 1] === string2[j - 1]) {
        matrix[i][j] = matrix[i - 1][j - 1];
      } else {
        matrix[i][j] = 1 + Math.min(
          matrix[i - 1][j - 1],
          matrix[i - 1][j], matrix[i][j - 1]
        );
      }
    }
  }
  // console.log(matrix);
  return matrix[string1.length][string2.length];
}
```

7.2 Domain Regex

```
/(?:[a-zA-Z0-9]+@)([a-zA-Z0-9\W]+)(?:.[a-z]{2,3})/g
```

7.3 Source code

The repository for all the code was at the end of the project period made public and can be viewed here <https://github.com/Gorboe/securemail>