Development and implementation of a phishing email detection application

Håkon Berg

Master of Science in Information Security
60 Credits

Department of Informatics
Faculty of Mathematics and Natural Sciences

University of Oslo

Spring 2020
Abstract

This project explores the concept of email phishing and what can be done to detect it, thereby preventing it from succeeding. Email phishing can be described as a type of social engineering attack, manipulating the victim (email receiver) into doing as the attacker (email sender) wants. It occurs when an attacker hidden as a trusted entity, fools the target into clicking malicious links or attachments found within an unsolicited email. This can lead to the installation of malware or theft of sensitive, personal, or financial information and data.

The goal of this project has been to find out if phishing emails share common recognizable patterns that can separate them from legitimate emails. Then use these patterns as the base for developing an application capable of detecting phishing emails. The application had to be compatible with a real email client to get access to emails for testing. A Microsoft Office Outlook add-in was developed and implemented, able to run on Outlook Live. A demo of the application, showing its functionality, can be seen at YouTube: https://youtu.be/T9jzo4ZIyLk

The source code of the add-in application was made as an open-source repository at GitHub: https://github.com/haakbe93/email-phishing-detection-add-in

In this project, two tests were performed. The first test being an application detection test to find out if the developed add-in application could detect phishing emails. The second test being a user study to find out if the add-in application could improve the participants' email phishing awareness.

From the application detection test results, it was concluded that the application did have a 76.66% success rate of rating emails legitimate and a 92.67% success rate of rating emails phishing. From the user test results, it was concluded that the application did improve the participants' email phishing awareness with at least 20%.

As for the project's value for email security, it was concluded that it had shed light on commonly exploited email vulnerabilities. Some of these were shown to be prevented with the right tools for email phishing detection and awareness.
Acknowledgment

I would like to thank my supervisor, Nils Gruschka, for providing me with an exciting project and for guidance and help during the work done.

I would like to thank fellow students at the Department of Informatics, faculty for Information Security, for academic cooperation, helping me produce test results, and keeping the morale up. I would also like to thank all other participants who made it possible to gather test results and provided insightful opinions that improved the results of this project.
Preface

This project is the concluding master's thesis of a two-year study program in Information Security at the Department of Informatics, University of Oslo. The thesis was written during the autumn of 2019 and the spring of 2020, making it a 60-study credits master's thesis.

With my bachelor's degree in computer science from NTNU, this project allowed me to combine my interest in system development and information security. The study of the technology used, the process, and the work developing the application, has been a great learning experience.

Oslo, 15.05.2020

Håkon Berg
# Contents

Abstract ......................................................................................................................... i  
Acknowledgment ........................................................................................................ ii  
Preface ......................................................................................................................... iii  
List of Figures ............................................................................................................... viii  
List of Tables .............................................................................................................. xi  
Terms and Abbreviations .......................................................................................... xii  

1. Introduction .............................................................................................................. 1  
   1.1 Motivation ......................................................................................................... 1  
   1.2 Research questions .......................................................................................... 2  
   1.3 Definitions ....................................................................................................... 2  
   1.4 Structure of the thesis ..................................................................................... 3  

2. Background .............................................................................................................. 4  
   2.1 Email ............................................................................................................... 4  
      2.1.1 Email structure and components ............................................................... 4  
      2.1.2 Email transferring .................................................................................... 7  
      2.1.3 Email infrastructure ................................................................................ 9  
   2.2 Email threats ................................................................................................... 9  
   2.3 Email phishing ................................................................................................ 10  
      2.3.1 Definition ................................................................................................. 10  
      2.3.2 Email phishing techniques ...................................................................... 11
2.3.3 Anatomy of common phishing emails ................................................................. 13

2.4 Email security measures .................................................................................... 23
2.4.1 Sender Policy Framework (SPF) ................................................................. 23
2.4.2 DomainKeys Identified Mail (DKIM) ........................................................... 24
2.4.3 Domain-based Message Authentication, Reporting & Conformance (DMARC) .. 25
2.4.4 Two-factor authentication ............................................................................. 26
2.4.5 Spam filter ..................................................................................................... 27
2.4.6 URL and attachment protection ................................................................... 28
2.4.7 Sandboxing .................................................................................................... 28
2.4.8 Machine learning ........................................................................................... 29
2.4.9 Human awareness and training ..................................................................... 30

3. Requirements ........................................................................................................ 32
3.1 Defining requirements ....................................................................................... 32
3.1.1 Functional requirements ................................................................................ 33
3.1.2 Informative requirements ............................................................................. 37
3.1.3 Design requirements ..................................................................................... 42
3.1.4 Operational requirements ............................................................................ 44

4. Design ...................................................................................................................... 46
4.1 Visualizing the structure of the application requirements ............................... 46
4.2 Modelling the add-in ......................................................................................... 47
4.2.1 Requirement component ............................................................................... 50
4.2.2 Total evaluation component ......................................................................... 52
4.2.3 Attribute details list component ................................................................. 53
5. Implementation  54
   5.1 Selection of development software and tools ................................................. 54
   5.2 Set-up of the development environment .......................................................... 54
   5.3 Connecting and running the add-in on Outlook ............................................... 55
   5.4 Application requirements as software architecture ........................................... 60
   5.5 Accessing information in the email header and body ........................................ 61
   5.6 Optimizing the percentage score system ......................................................... 63
   5.7 The finished add-in application ........................................................................ 65
      5.7.1 Application functionalities ......................................................................... 65
      5.7.2 Microsoft Office Outlook add-in ................................................................. 66

6. Application test  68
   6.1 Test layout ....................................................................................................... 68
   6.2 Test results ...................................................................................................... 71
   6.3 Analysis and discussion of the results ............................................................. 72
   6.4 Application improvements ............................................................................... 81

7. User study  83
   7.1 Test layout ....................................................................................................... 83
   7.2 Test results ...................................................................................................... 86
   7.3 Analysis and discussion of the results ............................................................. 87
   7.4 Value for email security .................................................................................. 89

8. Conclusion  91
   8.1 Summary ......................................................................................................... 91
   8.2 Further work ................................................................................................... 92
Appendix A – Test Results 108
A.1 Application Detection Test ............................................................... 108
A.2 User Test ....................................................................................... 116

Appendix B – Digital Appendix 119
B.1 GitHub Repository ........................................................................ 119
B.2 Demo of Application ..................................................................... 119
List of Figures

1. Illustration of email components (Source: [1]) ........................................... 5
2. Microsoft Outlook Live email header (Source: [2]) ................................. 6
3. Example of sending an email by SMTP (Source [3]) ................................. 7
4. Flow of email transfer from sender to receiver (Source: [4]) ................. 8
5. Screenshot of legitimate display name and email address ....................... 14
6. Screenshot of phishing email with fake display name ............................. 14
7. Screenshots of unpersonal email recipients (phishing emails) ............... 15
8. Screenshots of personal email recipient (legitimate email) ..................... 15
9. Screenshot of URL within html element (Source: [2]) ......................... 17
10. Screenshot of URL with mismatching text (Source: [6]) ...................... 17
11. Cross Site Scripting URL (Source: [8]) .................................................. 18
12. Explanation of Non-persistent XXS attack (Source: [9]) ...................... 19
13. Malicious attachments and extensions in email (Source: [10]) ............. 20
14. Example of malicious email attachment (Source: [11]) ....................... 20
15. Screenshots of HTML email attachment (Source: [12]) ..................... 21
16. Email header authentication protocols (Source: [2]) ............................. 22
17. Illustration of SPF (Source: [13]) ......................................................... 24
18. DKIM-Signature found in the DKIM header field (Source: [14]) ......... 25
19. Checks performed by DMARC (Source: [15]) ...................................... 26
20. Google’s 2FA email attack preventing rates 2019 (Source: [16]) .......... 27
21. Screenshot of the signs normalized to whitespace .................................. 33
22. Screenshot from web console of some of the parameters a link holds ........ 35
23. Illustration of evaluation percentage score system .............................. 41
24. Colour palette for application ............................................................... 43
25. Mock-up of functional requirement sections ....................................... 46
26. Screenshot of Outlook Live add-in side panel ...................................... 47
27. First sketches of the add-in ................................................................. 48
28. Sketch of “Sender” requirement section .............................................. 49
<p>| 29 | Sketch of requirement component | 50 |
| 30 | Model of a requirement component (made with Material UI) | 50 |
| 31 | Information bubbles explaining a requirement (Material UI) | 51 |
| 32 | The different levels of percentage score bars | 51 |
| 33 | Sketch of total evaluation component | 52 |
| 34 | Model of total evaluation component (made with Material UI) | 52 |
| 35 | Sketch of attribute details list component | 53 |
| 36 | Screenshot of “Total evaluation” section in attribute details list | 53 |
| 37 | Screenshot of the manifest.xml file | 55 |
| 38 | Screenshot of adding a personal add-in in Outlook. (Norwegian) | 56 |
| 39 | Screenshot of adding the project XML-file to personal Outlook add-ins | 57 |
| 40 | Screenshot of using the “npm start” command line in VSC terminal | 57 |
| 41 | Screenshot of Node.js running locally on a computer | 58 |
| 42 | Screenshot of sideloading the add-in in Outlook with web console | 59 |
| 43 | Screenshot of application JavaScript files | 60 |
| 44 | Screenshot of application class hierarchy | 61 |
| 45 | Screenshot of JavaScript email authentication result methods | 62 |
| 46 | Screenshot of calculation of percentage score for link requirements | 64 |
| 47 | Screenshot of how the test results would be logged | 69 |
| 48 | Screenshot of schema used in user test | 85 |
| 49 | Screenshot of the application running in Microsoft Office Outlook Live | 66 |
| 50 | Screenshot of the application’s main page | 67 |
| 51 | Screenshot of small sample of the application test schema results | 71 |
| 52 | Average % for each application test schema fields | 71 |
| 53 | Diagram of legitimate email’s total evaluation % | 72 |
| 54 | Screenshot of the 5 legitimate emails with score under 50% | 73 |
| 55 | Diagram with phishing email’s total evaluation % | 74 |
| 56 | Screenshot of lowest rated phishing email | 75 |
| 57 | Screenshot of false negative phishing emails | 75 |
| 58 | Diagram comparing phishing sender % and legitimate sender % | 76 |
| 59 | Diagram comparing phishing content % and legitimate content % | 77 |
| 60 | Average percentage score for application test schema fields | 78 |
| 61 | Diagram comparing phishing links % and legitimate links % | 79 |
| 62 | Diagram comparing authentication %’s | 80 |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>63</td>
<td>Screenshot of results from the user test</td>
<td>87</td>
</tr>
<tr>
<td>64</td>
<td>User test results w/application for week 1 and week 2</td>
<td>88</td>
</tr>
<tr>
<td>65</td>
<td>Example of malicious URL redirection (Source: [139])</td>
<td>19</td>
</tr>
</tbody>
</table>
# List of Tables

1. Table of emotion provoking email phishing scenarios (Source: [5]) 16
2. Example of SPF record for the domain “test.net” 23
3. Email safety percentage interval levels 37
4. Results from user test 86
Terms and Abbreviations

ASCII – *American Standard Code for Information Interchange*, a character encoding standard for electronic communication. It represents text in computers, telecommunication equipment, and other devices [34].

APT – *Advanced Persistent Threat*, an attack in which an intruder/intruder team establishes an illicit connection to a network to mine highly sensitive data [35].

ESMTP – *Extended Simple Mail Transfer Protocol*, a protocol for sending and receiving emails over IP networks. Extends SMTP with TLS providing security [36].

DKIM – *DomainKeys Identified Mail*, an email authentication method designed to detect forged sender addresses in emails by using digital signatures [37].

DMARC – *Domain-based Message Authentication, Reporting, and Conformance*, an email authentication protocol that builds on SPF and DKIM, allowing action based on the results from the protocols [38].

DNS – *Domain Name System*, a hierarchical naming system for computers, services, or other resources connected to the internet. It draws a parallel between various information and domain names, such as translating IP addresses to more human-readable domain names [39].

HTTPS – *Hypertext Transfer Protocol Secure*, an extension of the HTTP protocol used for secure communication over a computer network. TLS encrypts the connection, which ensures security by providing confidentiality [40].

IMAP/POP3 – *Internet Message Access Protocol / Post Office Protocol version 3*, internet standard protocols used by email clients to access and manipulate email messages from an email server over a TCP/IP connection [41].
MIME – *Multipurpose Internet Mail Extension*, an extension to the internet email protocol that allows email messages to contain text in character sets other than ASCII, as well as multimedia content [24].

ML – *Machine learning* is the study of algorithms and statistical models that computer systems use to perform a specific task without any particular instructions, relying on patterns and inference instead [42].

MALWARE – *Malicious software*, a collective name of several malicious software variants, includes viruses, ransomware, and spyware [43].

MDA – *Mail Delivery Agent*, a computer software component responsible for the delivery of the email messages to a local recipient's mailbox MUA by using POP/IMAP protocols [44].

MSA – *Mail Submission Agent*, a software agent that receives an email from an MUA and cooperates with an MTA [45].

MTA – *Mail Transfer Agent*, a mail server which transfers the email message to the next mail server using SMTP [46].

MUA – *Mail User Agent*, a mail client – an application used by the user to interact with email messages [47].

MX – *Mail Exchange Record*, specifies the mail server responsible for accepting email messages on behalf of the domain. It is one of the resource records in DNS [21].

SMTP – *Simple Mail Transfer Protocol*, an internet standard communication protocol used for transmission of electronic mail. [21] [33]

SPF – *Sender Policy Framework*, an email authentication method that allows the receiving email server to check whether the email sender uses an IP address authorized by the domain they claim to come from [48].

SPOOFING – *Spoofing*, the act of disguising a communication from an unknown source as being from a known, trusted source [49].

TCP – *Transmission Control Protocol*, an internet standard protocol that provides highly reliable, host-to-host delivery of a stream of bytes between applications communicating via an IP network [50].
**TLS** – *Transport Layer Security*, a cryptographic protocol designed to provide privacy and data integrity between two or more communicating computer applications [51].

**URL** – *Uniform Resource Locator*, is a reference to a unique web resource that holds its location on a computer network and has a mechanism for retrieving it [52].

**XML** – *Extensible Markup Language*, a standard way of storing and transferring data between programs and over the internet. The structure of the XML data format is machine-readable and human-readable. [53]

**XXS** – *Non-Persistent Cross-Site Scripting*, a malicious script that activates through a link, sending a request to a webpage with a vulnerability that enables the execution of malicious scripts [54].
Chapter 1

Introduction

1.1 Motivation

Electronic mail (email) is one of the most commonly used communication applications on the internet. Initially invented in the 1970s, it still has an essential role in how we use the internet to communicate today. It has proven to be very useful on a global scale both for business and personal communication.

The email was originally not designed with security in mind. The lack of email security has been of great concern, as many have experienced in the years after it first was invented. Several criminal threats have emerged on the internet in the last decade, exploiting the lack of email security.

This infamous genre of cybercriminal threats linked to email is known as "email phishing." It is the criminal act of sending an email to an unaware recipient falsely claiming to be an established authority, hoping to trick the receiver into giving away private information.

Email phishing incidents have existed since the start of the 1990s and have like the information technology progressed to a very sophisticated level. Careful and precise imitation and masquerading of trusted sources makes it very hard for an email recipient to determine if an email is legitimate or fraudulent.

Not everyone is fooled, but when the amount of work sending out thousand or hundred thousand emails to random recipients is not that much of a difference, of course, there will be someone that is fooled. Often identity thefts happen, which most of the time result in financial losses. Phishing emails can be very unfortunate for the average email user. Even worse, some cybercriminals target corporations. As part of a more significant attack, getting access to one employers' computer through email phishing may result in cybercriminals being able to do substantial harm on a business's internal IT network and systems.
The necessity for awareness and training at recognizing the differences between legitimate and phishing emails are pressing. There is a need for simple, inexpensive, and yet effective tools that aid this awareness. There should be no need to be exposed to the risk and uncertainties of cyber threats when using such a common and straightforward communication application.

1.2 Research questions

The goal of this thesis is to find out if it is possible to detect "phishing emails," thereby preventing them from succeeding. The following research questions are defined to facilitate the goal of the thesis:

Research question 1

*What defines phishing emails, and do they share recognizable common patterns?*

Research question 2

*Based on such patterns, is it possible to develop an application capable of detecting phishing emails?*

Research question 3

*By using such an application, can email users learn to become more aware of email phishing?*

1.3 Definitions

In this thesis, the special terms "phishing email" and "email phishing" are used quite a lot. The terms stand central in understanding the topic of this thesis. Therefore, to avoid confusion, the terms are defined as follows:

**Phishing email**

A phishing email is a type of fraudulent email, which may contain threats such as social engineering, malicious URLs, or malicious attachments. It is a type of email that tries to trick the recipient into downloading malware or giving away personal and financial information.
Email phishing

Email phishing is defined as the cybercriminal act of creating and sending phishing emails. It is the term used to describe the action, state, or occurrence or phishing emails. Ex: If a person is aware of what a phishing email looks like and what it can do, the person then has email phishing awareness.

1.4 Structure of the thesis

This thesis is structured as follows:

Chapter 2: Presents theory relevant to the work done in this project. Focuses on theory related to email threats and email security and specific email phishing methodology that are relevant for the development of the application.

Chapter 3: Describes the requirements for developing the application. Includes defining requirements based on theory in chapter 2.

Chapter 4: Describes the design of the application. The chapter includes sketches and models of the application functionality and components.

Chapter 5: Describes the implementation of the application. The chapter includes how requirements and design concepts were implemented. At last, a presentation of the finished add-in application.

Chapter 6: Describes testing the application’s phishing email capabilities. The chapter includes test layout, test results, and analysis and discussion of these results. At last, improvements to the application.

Chapter 7: User study of the application that tests if it will improve users email phishing awareness. The chapter includes user test layout, test results, and analysis and discussion of these results. At last, the application’s value for email security.

Chapter 8: Conclusion of the thesis. The chapter includes a summary of the thesis and further work.
Chapter 2

Background

This chapter will present the theory relevant to the work done in this project. It includes an overview of the email. Then, email threats and the concept of email phishing. Finally, a presentation of email security measures for detecting and preventing phishing emails.

2.1 Email

Email (electronic mail) is one of the most used services on the internet. As a communication application, it is fast, inexpensive, easy to distribute, and widely available. Invented by Ray Tomlinson, email first entered limited use in the 1960s and by the mid-1970s had taken the form now recognized as email [19]. In 2019 there were 3.93 billion active email users worldwide, including both business and consumer users. The daily email traffic was 293.6 billion emails, expected to continue growing in the coming years [20].

2.1.1 Email structure and components

The mail object defines the structure of email. As seen in figure 1, the mail object consists of the envelope and the content. The content consists of the email header and the email body [21]. Information contained in the envelope is used by servers to route the email through the network. The information contained in the content (header and body) is used by email clients, readable to the user. [21]
Figure 1: Illustration of email components (Source: [1])

**Email envelope**

The envelope consists of an originator address, one or more recipient addresses, and optional SMTP protocol extension material [21]. The originator address, also called return path address, is used to identify the origin of the email. In case of rejection of an email, the return path address can bounce a message notifying the sending server. Using this functionality is not recommended anymore as it may create a side effect called "backscatter," leading to innocent people getting bounce notifications due to spoofing (SPOOFING).

Both the envelop and the email header holds a sending address. No rules are requiring the envelope addresses and the header addresses to be the same. Therefore, it is possible to forge the header addresses making it easy to spoof an email's origin.

**Email header**

RFC 5322 [22] specifies the syntax of the email header. It consists of a collection of header fields, each consisting of a field name, followed by a colon (":") , followed by a field-body (data). The field bodies hold distinct information about the email and must be composed of printable US-ASCII characters. [22]

Each email message must have a header. The mandatory fields an email header must have are; FROM, TO, and DATE. Optional fields, but very commonly used are SUBJECT and CC [23].
As seen in figure 2, there exists a lot of email header fields. Among them are fields describing the servers an email visits when routing through the network, fields describing results from authentication protocols (SPF, DKIM, DMARC, and TLS), and fields for spam filters, see section 2.4.5 spam filter.

Email body

The message body is the part of an email visible to the user. It contains what a sender wants to convey to a receiver.

As defined in RFC 5322 [22], the message body is simply lines of US-ASCII characters. Multipurpose Internet Mail Extensions (MIME) extends this limitation by allowing email systems to carry text in other character sets (8-bit) than ASCII (7-bit). It also enables sending multimedia content as attachments or inline an email. Multimedia content can be URLs, audio, video, images, and application programs [24].

The MIME standard is specified in a series of request for comments (RFC): RFC 2045 [25], RFC 2046 [26], RFC 2047 [27], RFC 2049 [28], RFC 4288 [29] and RFC 4289 [30].

The integration with SMTP email is specified in RFC 1521 [31] and RFC 1522 [32].
2.1.2 Email transferring

Simple Mail Transfer Protocol (SMTP) is the standard internet protocol used to transfer electronic mail between computers. Defined in RFC 5321 [21], it is a set of protocol commands that authenticate and directs the transfer of email from the sender, through server(s), to the receiver(s). SMTP is independent of transmission subsystems, capable of transporting emails across multiple networks. It uses Transmission Control Protocol (TCP) for connection establishment and reliable and ordered data transfer.

Following is an example of SMTP communication between C – client and S – server when establishing a connection and transferring an email to another domain.

```
S: 220 smtp2go.com ESMTP Exim
C: HELO mydomain.com
S: 250 Hello mydomain.com
C: MAIL FROM:<sender@mydomain.com>
S: 250 Ok
C: RCPT TO:<recipient@anotherdomain.com>
S: 250 Accepted
C: DATA
S: 354 Enter message, ending with "." on a line by itself
C: Subject: sample message
C: From: sender@mydomain.com
C: To: recipient@anotherdomain.com
C: 
C: Greetings,
C: Typed message (content)
C: Goodbye.
C: .
S: 250 OK
C: QUIT
S: 221 www.sample.com closing connection
```

Figure 3: Example of sending an email by SMTP (Source: [3])

The HELO/EHLO command starts the conversation identifying the sending server domain. The DATA command signals the beginning of the email content. The AUTH command is not shown in figure 3 but is used when using the EHLO command, ensuring the client authenticates itself to the mail server by using AUTH SMTP [55].

Following is a simplified illustration of how an email transfer from a sender to a receiver.
As seen in figure 4, first, a sender composes an email in a mail user agent (MUA). Examples of popular mail user agents are Microsoft Outlook and Google Gmail.

Then, submitting the email to a mail submission agent (MSA) which, transfers it to a cooperating mail transfer agent (MTA) by using extended SMTP (ESMTP). Then, MTAs, which are mail servers, move the email through the internet with the help of SMTP. An email can hop through one or more MTAs depending on the distance of the destination. The email header is stamped with a date, time, and recipient each time an email is sent or forwarded by an MTA.

The MTA uses the domain name system (DNS) to look up the mail exchanger record (MX), a DNS resource for the recipient's domain to route the email to the targeted host.

The last step is done by the message delivery agent (MDA), which saves and delivers the email to the recipient's mailbox. The protocols Post Office Protocol version 3 (POP3) or Internet Message Access Protocol (IMAP) delivers incoming emails from the MDA to the MUA. [21] [41]
2.1.3 Email infrastructure

Traditionally email servers have been on-premise. Being expensive and needing administration but locally protected. Many of the big email client providers (Microsoft, Google, Yahoo) have migrated to the Cloud. This has produced a new email environment with the latest cybersecurity concerns and opportunities regarding centralizing all emails in one place. Contradicting the phrase "Don't put all your eggs in one basket." but providing businesses with a cheaper and accessible alternative. [56] [57]

2.2 Email threats

The email threat landscape is continuously evolving. Every year, cybercriminals develop new ways of deceiving and attacking their victims through email. The context, scenarios, and types of emails differ, but the main threats stay the same. The three main email threats are malicious attachments, malicious URLs, and social engineering.

These threats can all result in data loss, stolen information, disruption of business, and monetary loss. Most phishing emails involve one or more of these three threats.

Malicious attachments

Malicious email attachments are known to contain malicious software (Malware), which can install viruses, trojan horses, spyware, bots, set up ransomware attacks, infect Office files through macros, or launch advanced persistent threats (APT). Malware is designed to launch when an email attachment is opened. It can be disguised as documents, PDFs, voicemails, e-faxes, images, and other types of files that would seem to be trustworthy or exciting. [60]

Symantec reported in 2019 a malicious email rate of 1 in 412, where 48% of all malicious attachments where Office files such as word and excel files [61]. Verizon's Data Breach Investigation Report 2019, states that 94 % of all malware was delivered through email [62].

Malicious URLs

A malicious Uniformed Resource Locator (URL) is a clickable link embedded within the body or attachment of an email. It is created with the sole purpose of compromising the recipient of the email. Malicious URLs are often disguised in images, buttons, or text that do
not match the intended use. Symantec reported in 2019 that in their collected data from 2018, a malicious URL was found in every 170 email URL [61].

Clicking on a malicious URL can download and execute malicious scripts or install malware. It can also be a web address that takes the target to a fake website. This to persuade them to unintentionally giving away sensitive information, such as usernames and passwords, or expose them to an insecure location capable of installing malware on their computer [63].

Social engineering

Social engineering used in emails involves a form of psychological manipulation, fooling otherwise unsuspecting email recipients. Such manipulation tries to invoke urgency, fear, or interest in the victim through text in an email. Social engineering may lead the victim to click malicious links, open malicious files, or perform actions such as giving away sensitive information or transferring money to an illegitimate source. It can be tough to prevent such threats as it exploits human errors [64].

2.3 Email phishing

Email is rapidly becoming available around the world. Unfortunately, like with all technology, nothing is done perfectly the first time. This means shortcomings and faults are a necessary part of technological evolution, continually expanding and improving.

Some people, known as cybercriminals, have found ways to exploit shortcomings and faults found within the email, exploiting core email protocols, email functionalities, and weaknesses in human to machine interactions. This is known as email phishing, a cybercriminal attack vector that has increased dramatically in the number of incidents in the last years.

2.3.1 Definition

Email phishing can be described as a type of social engineering attack, manipulating the victim (email receiver) into doing as the attacker (email sender) wants. It occurs when an attacker masqueraded as a trusted entity, fools the target into performing actions based on the content of an unsolicited email. This can be clicking malicious links or attachments found
within the unsolicited email. This can lead to the installation of malware or theft of sensitive, personal, or financial information and data. [58]

The term "phishing" is a spin on the word fishing because criminals are dangling a fake' lure' (a legitimate-looking email) hoping targets will' bite' by doing what the criminals have requested [59].

Email phishing is most known for trying to steal personal and financial information, but it is also used to compromise computers and IT networks on a personal, business, and national level. It serves as a gateway and early phase of cybercriminal attacks, leading to more complex and dangerous situations.

2.3.2 Email phishing techniques

Email phishing techniques build on the three main email threats, see section 2.2. The differences between the email phishing techniques are the context and scenarios the threats are used in.

There are different types of techniques based on who the target is and how the target is approached. The target can be of quantitative or qualitative nature based on the number of emails sent and the work put into them. The qualitative way of choosing the target requires more preciseness in detail and information gathering. It requires more work but has a higher success rate of deceiving the target. Examples of this are spear phishing and business email compromise.

Sending emails to unknown persons in large numbers is the quantitative way of choosing the target. These types of emails are often generic and easily duplicated. They contain less personal information, can be irrelevant for the receiver, and are written poorly with multiple grammatical errors. A smaller percentage of targets fall for this type of attack, but because of the large number of people receiving these emails, chances of fooling someone increases. Examples of this are common email phishing and spam emails.

**Common email phishing**

Common email phishing is the most widespread email phishing technique. These types of emails are sent in high volumes and are known to be generic and non-personal. They succeed
due to imitating well-known and trusted sources, such as service and technology brands. They involve social engineering, trying to provoke fear and urgency to trick the target into doing what the attacker wants. This usually means clicking links leading to fake websites where the targets are asked to give away personal or financial information such as usernames, passwords, and credit card numbers. [65]

It is easier to spot these types of phishing emails as suspicious signs are more common. See section 2.3.3 for detail about common phishing emails.

**Spear phishing**

Spear phishing is a more advanced form of common email phishing, aiming for individual targets. It involves deceiving the target into clicking malicious links or opening malicious attachments disguised as important documents/files.

The success of a spear phishing relies heavily on a cybercriminal's ability to gather information about the target. The more accurate and restricted the information is, the more likely the target is to believe the attacker. Information about personal life, memberships, accounts, family members, colleagues, a company's internal structure, processes, and software are information sources the attacker aims to use. Even though these information sources seem private, some can be obtained through social media such as LinkedIn or Facebook. Most companies also keep a web presence offering detailed and relevant information. [66]

**Business email compromise (BEC)**

Business email compromise (BEC) is a type of spear phishing aiming for a specific group of individuals such as C-level executives and employees in finance. In a BEC email phishing attack, the cybercriminal utilizes social engineering, trying to convince the target to give away highly sensitive data or perform financial transactions such as vendor payments, direct deposit, or wire transfer.

Private and important information is often known and presented by the attacker, also with few to none errors or suspicious signs in the emails. They are harder to detect as it aims to make the target perform actions outside the email.

To enhance trustworthiness, the attacker either compromises a legitimate business email account and imitates the owner's identity or creates a spoofed identity almost identical to one on the corporate network [67].
In September 2019, The FBI reported that BEC attacks had cost victims over 26 billion US dollars from 2016 to 2019 [68].

**Spam emails**

Spam is junk emails sent in bulk. They typically contain unsolicited messages such as random commercials, adult content, or promises of financial gains. According to Talos Intelligence, in February 2020, the average daily volume of spam emails was 288.61 billion, which was 84.8% of all daily emails sent. This means that for every legitimate email, almost six spam emails were sent [69]. The number of replied and interacted spam emails is far less but is still a significant threat due to the substantial amount of email traffic.

By utilizing botnets (compromised computers), open mail relays, and open proxy servers, cybercriminals can send emails from multiple trusted sources, making it harder to stop spam [70].

In most cases, spam is an advertisement by nature, but 1 in 3,000 emails contains links leading to malicious websites or malicious files that install malware [137]. The Center for Internet Security (CIS) reported in 2018 that spam emails were the top source of malware infection [71].

### 2.3.3 Anatomy of common phishing emails

To better be able to determine if an email is phishing or not, it can be split into distinct parts that can serve as a checklist for specific patterns.

The following are patterns that constitute the "anatomy" of common phishing emails. These patterns are based on information gathered from reports made by Accenture, Cisco, Verizon, Barracuda, PhishMe and Symantec [72] [73] [74] [75] [62] [76] [61].

**Fake display name**

Email display names are used in emails as identification. It is usually the name of the sender or name of the company they are representing. In legitimate emails, the domain name (the part of the email address after "@") is usually the name of the company the sender represents. It is also very typical to find parts of the sender's name or initials in the email address. This means
that there would be a similarity between senders display name and senders email address, as seen in figure 5.

Figure 5: Screenshot of legitimate display name and email address

In phishing emails, the sender email address does not align or have little similarities with the display name. Often being something completely different than the company or person it claims to be.

Figure 6: Screenshot of a phishing email with a fake display name

Cybercriminals usually use subdomain spoofing to make it harder to spot a phishing email address. This means that they add subdomains to their domain, which imitates known company names. It is always the rightest part of an email address that is the true domain of the sender. Ex; "service@apple.notlegit.com". The company name Apple is included as a subdomain, but the right domain is notlegit.com, meaning it is not an Apple domain email address.

Unpersonal recipient

When companies or other authorities contact their customers or recipients through email, they address them with their name, username, or other personal forms of identification. Emails that do not address the recipient in a personal way, but rather in a generic and non-personal way, is a sign that the email was sent out in bulk. Emails asking for personal information without referencing any information about the receiver or facts that only the two parts share is most likely phishing emails. The more information or interaction an email asks for; the more ensuring personal details should the email contain.

Dear Customer

Your Apple ID was locked due to security reasons.
We have detected a sign-in from an unknown device and an unusual activity from your account.

(a) Example of non-personal customer recipient #1
Hello,

(b) Example of non-personal customer recipient #2

**NETFLIX**

Automatic payment.

Hi Customer,

(c) Example of non-personal customer recipient #3

Figure 7: Screenshots of non-personal email recipients (phishing emails)

(a) Example of personal customer recipient #1

(b) Example of personal customer recipient #2

Figure 8: Screenshots of personal email recipient (legitimate email)
Grammatical errors

Every big company takes marketing, information sharing, and customer relations very seriously, resulting in spending vast amounts of time on email. Therefore, it is unlikely to find misspellings and grammatical errors in emails sent from legitimate companies. Emails that are not well crafted and seems to be made in a hurry with lots of mistakes are indicators of being a phishing email. Cybercriminals commonly use language translating tools to craft phishing emails for native speakers. This may also be one of the reasons for a higher number of grammatical errors.

Emotion provoking content

Phishing emails often try to play on their victim’s emotions. Table 1 contains examples of emails playing on emotions connected to opportunities, fear, urgency, and curiosity.

Emails asking for immediate action has a sense of urgency or tries to pique your curiosity is typical for phishing email attempts. Trusted companies or authorities have no intention to scare or stress their customers via email. They usually practice far better customer service.

Table 1: Table of emotion-provoking email phishing scenarios (Source: [5])
Request for personal or sensitive information

Phishing emails usually ask for sensitive, financial, or personal information such as credit card numbers, usernames, and passwords. No company would like its customers to provide such information through an email. They are more likely to ask them to visit their webpage, log in, and then ask about sensitive information.

Emails that ask for personal information without providing any proof for the relation between the sender and receiver are most likely spam or phishing attempts.

Suspicious URLs

URLs can be hidden within a text link or HTML element, as seen in figure 9. The button displays the text "Go to Apple ID Account," but if clicked will take you to a fake Apple website. Hovering the button shows the URL, which has no similarity with Apple's legitimate website: https://www.apple.com/

![Go to Apple ID Account](image)

Figure 9: Screenshot of URL within an HTML element (Source: [2])

"Log in your account now" seems like a fast and straightforward way to access your PayPal account. Hovering the link displays the true destination of the URL, which has little to do with PayPal.

![Log in your account now](image)

Figure 10: Screenshot of URL with mismatching text (Source: [6])

It is recommended not to click on links that do not support Hypertext Transfer Protocol Secure (HTTPS), (uses HTTP instead). Data transmitted from your computer to a website’s server over regular HTTP connections do not provide secure transaction (integrity and confidentiality) and authorization, which poses a risk for the security of the data you transmit.
Without it, attackers can exploit unprotected resources that travel between websites and users' computers. Images, cookies, scripts, and HTML are all exploitable. Intrusions can occur at any point in the network, including a user's machine, a Wi-Fi hotspot, or a compromised ISP [138].

The mentioned shortcomings are very desirable for cybercriminals. Therefore, HTTP is commonly used on malicious websites instead of HTTPS. Any URLs that contain http:// and not https:// is phishing warning sign.

Other URL spoofing techniques typically used; links may appear legitimate by being subdomains of a trusted website. Ex: www.apple.notlegit.com.

URLs looking very similar but with misspellings (homograph attack). Ex: "r + n looks very similar to m." URLs may also be shortened with the help of services such as tiny URL [7].

URLs are sent over the internet using the ASCII character set. Often URLs contain unsafe characters that must be replaced with hex encoding values, "%" followed by two hexadecimal, to represent the unreliable character in ASCII format [77]. This makes it much harder to read the URL. Figure 8 and figure 9 shows how an XXS attack is performed, and how hex encoding helps obfuscating an URL.

Non-Persistent Cross-site Scripting attacks (XXS) can be used in URLs:

```html
index.php?name=<script>window.onload = function() {var link=document.getElementsByTagName("a");link[0].href="http://not-real-xssattackexamples.com/";}</script>
```

a) XXS URL

```html
index.php?name=%3c%73%63%72%69%70%74%3e%77%69%6e%64%6f%77%2e%6f%6e%6c%6f%61%64%20%3d%28%29%7b%76%61%72%20%6c%69%6e%6b%3d%64%6f%63%75%6d%65%6e%74%65%6d%65%6e%74%73%42%79%4c%6f%67%54%73%74%61%74%65%2d%61%63%6b%65%72%2e%63%6f%6d%2f%22%3e
```

b) XXS URL with hex encoding

Figure 11: Cross Site Scripting URL (Source: [8])
Figure 12: Explanation of Non-persistent XXS attack (Source: [9])

As seen in figure 65, an URL may contain redirection to an untrusted external site. The user sees the original trusted site and does not realize that a redirection are taking place. This can be combined with hex encoding for further obfuscating. [139]


Figure 65: Example of malicious URL redirection (Source: [139])

Suspicious attachments

Email attachments are dangerous due to the unknown content they may contain. Email attachments only provides the user with information about the file name and file type, which makes it difficult to evaluate the content without opening it. A malware contained within an attachment can execute if opened.

As seen in figure 13 Cisco's email threat report from 2019 states that the most used attachment types in phishing emails are Microsoft Office files such as .doc and archive files such as .zip [73].
These types of files are also among the most used attachments in emails, making it hard to distinguish between legitimate and malicious attachments.

Figure 13: Malicious attachments and extensions in the email (Source: [10])

Figure 14: Example of malicious email attachment (Source: [11])

Figure 14 shows how a Microsoft Office word document can lead to a ransomware attack.

HTML file (HTM/HTML) attachments are also commonly used to trick the target into opening a phishing webpage. The webpage looks legitimate and will ask for personal information such as a username and password [78].
Social engineering and malicious attachments make a dangerous combination resulting in unaware targets downloading malicious content or giving away credentials. Therefore, it is advised not to open attachments included in emails that show suspicious phishing signs.

Lack of authentication

Cybercriminals will try to spoof the origin of phishing emails. This is done by changing the sender address found in the email header, the "FROM:" address, which will then be different from the originator address found in the envelope. The recipient's email client will display a misleading sender address, and the recipient will thus be deceived about the message's real origin. By doing so, they can imitate legitimate domains and hide their own identity. Emails use the authentication protocol suite SPF, DKIM, and DMARC to counter spoofing. See section 2.4.

Lack of authentication is a warning sign and should alert the user. Most email clients provide some form of anti-spam filter that utilizes the results from the protocol suite. Domain
administrators must implement authentication protocols properly so that they can effectively detect spoofing.

Compromised email accounts used in BEC and more advanced phishing attack methods have no problem passing authentication as they use legitimate domains with verified digital signatures. Figure 16 is an example of a phishing email passing authentication.

Authentication-Results: spf=pass (sender IP is 198.2.128.131) smtp.mailfrom=track.thesynergist.org; hotmail.com; dkim=pass (signature was verified) header.d=thesynergist.org;hotmail.com; dmarc=bestguesspass action=none header.from=thesynergist.org;comphpath=pass reason=109

Figure 16: Email header authentication protocols (Source: [2])

Combination of phishing signs

Multiple parts of an email showing signs of suspiciousness increase the likelihood of it being a phishing email. If an email fulfills all these signs, it is a common phishing email.

It is essential to be observant of all parts, as phishing emails may vary. Some may seem legit in different parts than others.

Summary of patterns found in the anatomy of common phishing emails:

1. Fake display name
2. Unpersonal recipient
3. Grammatical errors
4. Emotion provoking content
5. Asking for personal information
6. Suspicious URLs
7. Suspicious attachments
8. Lacking authentication
2.4 Email security measures

Several security measures have been developed to improve email security. Among them are extensions to SMTP, which can be implemented to provide authentication in transit of emails. Other measures can provide security by inspecting and filtering emails based on its content, such as spam filters, link and attachment protection, sandboxing, and machine learning. For non-technical security measures, human awareness and training have proven to be important for the detection and prevention of email phishing.

2.4.1 Sender Policy Framework (SPF)

Sender Policy Framework (SPF) is an email authentication protocol designed to detect forging sender addresses during the delivery of email. It is specified in RFC 7208 [48] and updated in RFC 7372 [79].

SPF does not validate against the "From" address found in the email header. Instead, SPF looks at the originator address found in the email envelope to validate against the domain it claims to come from.

A domain administrator can utilize SPF by publishing a public SPF TXT record to a domain's DNS, specifying who can use the domain to send emails. The record contains a list of authorized sending servers and IP addresses for the domain [80].

test.net. TXT “v=spf1 mx a:hello.test.net include: spf.google.com -all"

The different parts in the record:

<table>
<thead>
<tr>
<th>v=spf1</th>
<th>SPF Version 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>mx</td>
<td>The incoming mail servers of the domain are authorized to also send mail for test.net</td>
</tr>
<tr>
<td>a:hello.test.net</td>
<td>The machine hello.test.net is also authorized</td>
</tr>
<tr>
<td>include: spf.google.com</td>
<td>Everything that is considered legitimate by gmail.com is also legitimate for test.net</td>
</tr>
<tr>
<td>-all</td>
<td>All other machines are not authorized</td>
</tr>
</tbody>
</table>
Table 2: Example of an SPF record for the domain "test.net."

During an email delivery, the receiving email service can check the SPF record to validate the email sender on behalf of the domain. If the sender is included in the record, the email is passed on as valid. If not, the policy fails and is rejected.

![Illustration of SPF](source: [13])

SPF is only able to validate an email’s originator address and not the "From" address, meaning it is easy to spoof. Cybercriminals can use their own domain and get SPF passed and then spoof the "From" address, making it seem legitimate. Also, the SPF record only specifies to which degree a message will fail but not what to do with a failure message. Emails can reach a user's inbox, even when failing SPF [81].

2.4.2 DomainKeys Identified Mail (DKIM)

DomainKeys Identified Mail (DKIM) is an authentication protocol designed to allow senders to associate a domain name with an email message, thereby proving authentication. It is specified in RFC 6376 [82] and updated in RFC 8301 [83], RFC 8463 [84], RFC 8553 [85] and RFC 8616 [86].

DKIM uses digital signatures to ensure authenticity and integrity. It allows the receiver to check whether emails are authorized to use the domain they claim to come from. Domains that utilize DKIM have an asymmetric cryptography key pair, one public key, and one private key. The public key is published in the DNS records, which the receiving email server can use
to verify the authenticity of incoming emails. The private key is used for signing outgoing emails [87].

**Figure 18:** DKIM-Signature found in the DKIM header field (Source: [14])

Before signing an email, the sending mail server hashes parts of the email. To form the hash, DKIM will use the email body, several headers fields, including TO, FROM, and DATE. Then the hash is signed with the domain's private key. For receiving mail servers that are trying to verify emails, the same steps are done in reversed order, and the public key is used instead of the private key.

DKIM can detect which domain an email came from, but not from who within that domain. The domain uses only one cryptographic key pair, not one for each individual sender. As only parts of the email header are hashed and encrypted, the possibility of adding or changing un-encrypted email content during transit poses a weakness.

### 2.4.3 Domain-based Message Authentication, Reporting & Conformance (DMARC)

As defined in RFC 7489, Domain-based Message Authentication, Reporting & Conformance (DMARC) is an email authentication protocol that organizations operating mail servers can use to express domain-level policies and preferences for message validation, disposition, and reporting. This can improve email handling and is commonly used to protect a domain from unauthorized use known as email spoofing [88].

DMARC builds on the existing email authentication protocols SPF and DKIM, improving upon their shortcomings. It verifies that the email "From" header field matches the domain validated by SPF and the digitally signed domain by DKIM, see figure 19. It also allows domain owners to set up policies that tell what to do if an email either passes or fails SPF and
DKIM. This can be set to none, quarantine or reject. DKIM is also capable of producing reports, informing domain owners of any failing checks, making it easier to identify email phishing campaigns. [38] [89]

![Diagram of DKIM processing](image)

**Figure 19:** Checks performed by DMARC (Source: [15])

### 2.4.4 Two-factor authentication

Two-factor authentication (2FA) adds an extra verification layer when logging into email clients. 2FA relies on users having something they know, knowledge-based, such as username and password, and something they have, device-based, such as a smartphone. If an email account is compromised, 2FA will prevent the use of compromised credentials, since these alone are insufficient to gain entry. This makes it an effective email security measure.

The more people using 2FA, the greater the security effect it yields, making it harder to utilize compromised legitimate email accounts in automated bot attacks and phishing attacks.
Unfortunately, 2FA has not yet become widespread among email users. Reportedly less than 10% of Gmail users had enabled 2FA in 2018 [90]. But Dou reported an increase from 28% in 2017 to 53% in 2019 for 2FA usage on the internet [91]. The inconvenience of doing more to use email is less appealing, even though it provides much-needed security measures.

2.4.5 Spam filter

A spam filter is a commonly used protection tool against spam emails. As described in section 2.2.3, spam emails represent a massive amount of daily global email traffic, consuming large amounts of bandwidth and data storage. It is also very annoying for email users to have their inboxes filled with spam emails, exposing them to potential threats.

Spam filters inspect and filters on the available information found in the email header and email body. Examples of techniques used to filter are; content filters, header filters, whitelisting, blacklisting, rule-based filtering, and machine learning. Incoming emails are processed through these filters, quarantining emails that get identified as spam. [92] [93] [94]

A common problem among spam filters is the varying rate of false positives and false negatives. This is due to the challenging task of updating filtering lists to match the continually changing email attack vector.
Microsoft Outlook has a spam protection feature called anti-spam protection in Office 365. It uses filters such as connection filtering, content filtering, outbound spam filtering, and spoof intelligence [95].

Also, Outlook inserts a header field called "X-Microsoft-Antispam" to incoming emails, which provide additional information to considered bulk mail and phishing. This header field has values called bulk complaint level (BLC) and spam confidence level (SCL). These levels can be customized but can be hard to tune in the right way due to varying types of emails [96] [97].

2.4.6 URL and attachment protection

URL and attachment protection prevents malicious email content from reaching the user's inbox. This is done by scanning and comparing email content, classifying, and predict potential malicious email links and attachments before the users open them. Examples of methods used are checklists for typical malicious file extensions such as Microsoft Office files (macros), signatures found in known malware, using sandbox technology, historical statistical modeling, and analyzing sender behavioral data [98].

Mimecast's URL protection and Microsoft Office 365 ATP Safe Link and Safe Attachments are examples of paid URL and attachment protection tools [99] [100].

2.4.7 Sandboxing

Sandboxing adds another layer of protection by testing URLs and attachments found in emails. A sandbox is an isolated test environment. It is a secure virtual space that emulates the CPU of your production servers and desktops. It enables opening potential malicious files and links without the risk of infecting the real computer network. This makes it possible to analyze malware when executed, finding new ways to protect against them.

It is especially effective to defend against zero-day threats. The downsides to sandboxing are that it is time- and resource-intensive. Running all email traffic through a sandbox is impractical and costly. [101] Zero-day malware has been observed as capable of detecting
emulated environments. Lying dormant only to execute when in a real desktop, making sandboxing ineffective [102].

Microsoft Office 365 Advanced Threat Protection is an example of a paid cloud-based email filtering service that utilizes sandbox technology to safeguard against malicious links and attachments [103].

2.4.8 Machine learning

Machine learning (ML) is a field of computer science that has existed in theory for decades but have in recent years proven to be very useful in practice. By normalizing emails to a data format that is readable for ML algorithms, they can with enough processing power and available data, process enormous amounts of information, and learn almost any email phishing pattern possible. Not only able to recognize known patterns but also use that knowledge to find new ones [104].

The data used for processing and training is what can be found in the email content. Metadata can be used to detect email spoofing. Linguistic data can be used to spot commonly recurring linguistic patterns. Email routing and network usage data can be used to identify compromised domains and phishing email campaigns [105].

With the constant flow of new data to process and learn from, the ML algorithms evolve as the email phishing attack vector evolves. This makes it able to have an almost 100% email phishing detection rate with low false positives and false negatives.

Google announced in 2017 that ML models were helping prevent 99.9% of spam and phishing messages from reaching Gmail user's inboxes. February 2020, they started using "Deep learning" AI to prevent emails containing malware from reaching their users' inboxes. With a detection rate of 99.9%, the Gmail scanner processes 300 billion Gmail attachments every week. With over 1.5 billion active users, there is almost no limit to data available for google Gmail's ML algorithms to process and self-improve. Keep in mind, 0.1% is 300 million Gmail attachments it could not detect, still making it a substantial threat [106].

Downsides to this technology are the need for vast amounts of normalized data, storage capacity, and processing power to train the ML algorithms effectively.
2.4.9 Human awareness and training

As described in previous sections, there exist many different types of technical email security measures that can be implemented to prevent phishing emails from reaching the email inbox. Unfortunately, these are not failproof security measures. Neither are they adaptable for everyone, due to resource dependencies. Therefore, human awareness and training should be implemented to protect against phishing emails that do reach the recipient's email inbox. This to balance the different measures of email security, ensuring defense-in-depth [107].

It is essentially humans who use emails, making it important that it is a simple and understandable communication service. This must be true for email phishing awareness and training as well. The efficiency of such measures is dependent on the participants' motivation to learn, which is often driven by interest. Taking courses and learning new things that seem trivial and boring do not produce results.

In many organizations and firms, it has become common to practice email phishing awareness and training. In the hope of increasing attention and interest in learning about email phishing, several approaches have been taken:

**Security videos**
Employees are shown short videos of how email phishing works and what to look out for in emails that seem suspicious. Short videos have a lower threshold to be watched and understood rather than reading many pages about a subject [108].

**Gamification**
Gamification, which is a new learning method aimed to motivate and engage the user to carry out tasks, is more and more applied to prevent cyber threats. These types of games make the user learn about email phishing by solving problems resulting in making it seem more fun and rewarding [109].

**Phishing test**
The IT department of a company regularly selects employees that are sent simulated email phishing attacks. Failing such tests results in remedial email phishing training [110].
Human firewall

Everyone in the organization is tested, finding out what percentage of the employees who are prone to phishing attacks. Everyone is then trained on simulated phishing attacks regularly [111].
Chapter 3

Requirements

This chapter will present the requirements used to develop a phishing email detection application in Microsoft's Outlook Office add-in environment. The requirements are defined based on the theory presented in Chapter 2 - Background. This chapter includes defining functional-, informative-, design- and operational requirements.

3.1 Defining requirements

Before starting to develop a phishing email detection application, requirements had to be defined. Four main groups of requirements were defined:

1. The application must provide functional requirements capable of detecting the anatomy of a common phishing email, as presented in section 2.3.3.
2. The application must be informative. The user must receive helpful information to make better decisions in email interactions.
3. The application must have a design that is intuitive, easy to understand, and easy to navigate.
4. The application must be able to run on a real email client for easy access to testing.
3.1.1 Functional requirements

Functional requirements were developed as countermeasures to the anatomy of a common phishing email, see section 2.3.3. The functional requirements were divided into the following sections:

- Sender information
- Message content
- Links
- Authentication
- Attachments

Each section had to be split up in various requirements to deduce phishing email's common parts and patterns. Each requirement had to give a percentage score, see section 3.1.2 for more details about the percentage scores and the total evaluation system.

**Sender information**

Two requirements were created for sender information. The first requirement compared the sender's display name and the sender's email address. The comparison checked whether there are words in the display name that also can be found in the email address. The words compared had to be relevant to each other and consist of more than two letters.

Before comparing the display name with the email address, words and letters had to be normalized. Normalization had to be done by turning all text to lower case, and signs, seen in figure 21, had to be replaced with whitespace.

```
~!@#$%^&*()_+-=[]{}|\;"',<>"
```

Figure 21: Screenshot of the signs normalized to whitespace

The second requirement compared the email sender's IP address against a blacklist database. This blacklist database contains IP addresses and email domains that have been reported for sending spam and phishing emails. The blacklist database is an external API.

MxToolBox's IP Blacklists Database [112].
Requirements:

1. Compare word similarities between the sender's display name and the sender's email address.
2. Check the sender's IP address against MxToolBox's IP blacklist database.

Message content

Two requirements were created for the email message content. The first requirement had to check if the recipient's name was used in the email message. This checked for a match in the recipient's surname, last name, and full name.

The second requirement counted the number of typical phishing words found in the email. Typical phishing words are defined by their nature of provoking emotions and are related to the topics financial-, commercial-, health-, reward- and personal information (see section 2.3.3). The list of phishing words had to be iterated through and compared with words found in the email. A word match had to increase the phishing word count by one. The list had to contain both Norwegian and English phishing words.

The full list of phishing words can be found at: https://github.com/haakbe93/email-phishing-detection-add-in/blob/master/src/taskpane/components/spamWordData.json.

Requirements:

1. Check if the recipient's name is used in the email message.
2. Check numbers of typical phishing words used in the email message, both English and Norwegian words.
Links

Five requirements were created for email links (URLs). The first requirement had to check the relation between the sender's display name, the sender's email address, and distinct links found in the email, see figure 22.

```javascript
href: "https://www.linkedin.com/comm/feed/?midToken=AQHHEPwRHOy6..."  
origin: "https://www.linkedin.com"  
protocol: "https:"  
username: ""  
password: ""  
host: "www.linkedin.com"  
hostname: "www.linkedin.com"  
port: ""  
pathname: "/comm/feed/"  
search: "?midToken=AQHHEPwRHOy6zg&trak=eml-email_job_alert_single.."
```

The second requirement had to check if distinct links in the email use HTTPS. This had to be done by using the parameter "protocol" as seen in figure 22, to determine if a link uses HTTP/HTTPS.

The third requirement had to check if there was any duplication of distinct links in the email, using the information found in the "href" parameter seen in figure 22. The number of duplicated links of the total links found in the email had to be displayed to the user.

The fourth requirement had to check if the distinct links in the email contain URL encoding. Hex is a common form of URL encoding, which obfuscates the real URL, see section 2.3.3 – Suspicious URLs.

The fifth requirement had to check whether distinct "href" links include redirection to an untrusted site, see section 2.3.3 – Suspicious URLs. Redirection means checking the usage of "/?url=" in the URL path component.

Requirements:

1. Compare the sender's display name and the sender's email address with distinct links found in the email (Sender-link relation).
2. Check whether distinct links use HTTPS or HTTP.
3. Check the number of duplicated links.
4. Check whether distinct links include URL encoding.
5. Check whether distinct links include redirection.
Authentication

Three requirements were created for email authentication. The requirements checked the results of the authentication protocols SPF, DKIM, and DMARC. As seen in figure 2, the "Authentication-Results" is a part of the email header found in every Microsoft Outlook email. The authentication protocols can have the following results:

SPF:

1. None, pass, neutral, fail, softfail, permerror or temperror

DKIM:

2. None, pass or fail

DMARC:

3. None, pass, bestguesspass or fail

The only relevant results from the authentication protocols were if they fail or pass, any other results counted as a fail. The only exception being the result from DMARC "bestguesspass", which gave a pass with a lower percentage score.

Requirements:

1. Check if SPF pass or fail
2. Check if DKIM pass or fail
3. Check if DMARC pass, bestguesspass, or fail

Attachments

One requirement was created for email attachments. It checks whether an email contains attachments and displaying the name of the attached files (not counting inline attachments). Email attachments should not be opened unless you are sure that the email comes from a legitimate source. If several parts of an email are suspicious, opening attachments should be avoided.

Requirements:

1. Check if the email contains any attachments and display the attached file names.
3.1.2 Informative requirements

The application had to give users informative and structured feedback on the defined functional requirements. It was needed to make it easier for users to understand and better separate between legitimate emails and phishing emails. To fulfill the informative requirement, a percentage score system was developed that rates the functional requirements. To make the application more understandable, a "user manual" had to be implemented as well.

**Percentage score system**

A systematic and understandable rating system was developed for the functional requirements. The functional requirement sections: sender, content, links, authentication, and attachments all have multiple requirements, which each had to have a percentage score rating.

The percentage score system rates a functional requirement from 0% to 100%. A higher percentage score represents a higher likelihood of legitimate emails, and a lower percentage score represents a higher likelihood of phishing emails.

As seen in table 3, the percentage score can be between 0 and 100. It was divided into four levels of percentage score intervals. These levels give a broad estimate of email security.

<table>
<thead>
<tr>
<th>High safety</th>
<th>Medium safety</th>
<th>Low safety</th>
<th>No safety</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 – 76</td>
<td>75 – 51</td>
<td>50 – 26</td>
<td>26 – 0</td>
</tr>
</tbody>
</table>

Table 3: Email safety interval levels

Requirements:

1. A percentage score must be a minimum of 0 and a maximum of 100.
2. A percentage score rating must be in one of the four score interval levels.
**Sender information**

The sender information section had two functional requirements. The first requirement was based on comparing the sender's display name and the sender's email address. This requirement starts at a score of 0%, giving a score of 30% for each word that is matched. An extra 50% score is given if the sender's display name was found in the email domain.

The second requirement, the IP address check, provides valuable information about the email sender, but due to it being an external API, it gave no percentage score.

Requirements:

1. The display name/email address similarity check must start at 0%. Each word match must give a 30% score. Word match with domain name must give an extra 50% score.
2. IP address check gives no percentage score.

**Message content**

For the first message content requirement, if the full name of the recipient was found in the email message, a score of 100% was given. A lower percentage score was given if only parts of the recipient's name were found, such as only the surname or last name. Then only a 30% score was given per surname/last name found in the email message.

Each word matched in the phishing word list increased the word count by one. One word represents two percentages. The score starts at 100% and is subtracted by 2% for each phishing word found.

Requirements:

1. The recipient name check must start at 0%. For each time the recipient's surname or last name is found in the email message, a 30% score must be given. If the recipient's full name is present, a 100% score must be given.
2. For each phishing word match, 2% of the starting score of 100% must be subtracted.
Links

For the requirements HTTPS links and Sender-link relation, the percentage score starts at 0%. HTTPS links percentage score was added based on the following calculation:

\[ \text{((Number of links containing HTTPS) / (Total number of links))} \times 100 \]

The fewer links containing HTTPS (HTTP instead), the lower the percentage score was given.

Sender-link relation requirement percentage score was added by 40% for each word match found between the sender's display name, the sender's email address, and distinct links.

For the requirements encoded links, redirected links and duplication links the percentage score starts at 100% and subtracts a percentage score calculated as follows:

\[ \text{((Number of Encoded/Redirect/Duplicate links) / (Number of total links))} \times 100 \]

This calculation must subtract the percentage score due to a higher number of links having a negative effect.

The percentage score given to the requirements encoded links, redirection of links, and duplication of links only counts 20% in the evaluation of the link section. These requirements were rated lower compared to the requirements HTTPS and link-sender relation, which counts 100% in the evaluation of the link section. This is due to the different levels of a threat the requirements pose and the number of occurrences in both phishing and legitimate links.

Emails that do not contain links are considered a lot safer. It excludes a lot of email phishing attempts, as links are a vital component in the success of such phishing emails. Therefore, emails without links must be given a 100% score in the link section evaluation.

Requirements:

1. Sender-link relation percentage score must start at 0% and increase by 40% for each word match found between the sender's display name, the sender's email address, and distinct link.
2. HTTPS links percentage score calculation must be: 0% + ((the number of links containing HTTPS / by the total number of links) * 100).
3. Encoded links percentage score calculation must be: 100% - ((the number of links containing encoding / by the total number of links) * 100).
4. Duplicated links percentage score calculation must be: \(100\% - \left( \frac{\text{the number of duplicated links}}{\text{the total number of links}} \right) \times 100\).

5. Redirected links percentage score calculation must be: \(100\% - \left( \frac{\text{the number of redirected links}}{\text{the total number of links}} \right) \times 100\).

6. Encoded links, redirection of links, and duplication of links must only count 20\% in the link section percentage score evaluation.

7. HTTPS links and Sender-link relation must count 100\% in the link section percentage score evaluation.

8. If there are no links present in the email, a score of 100\% must be given as the evaluation of the link requirement section.

**Authentication**

Each authentication protocol is given its own percentage rating. Each has the same rating system, where a fail on authentication gives a score of 0\%, and a pass gives a score of 100\%. The only exception is the "bestguesspass" for DMARC, giving a score of 50\%.

The "bestguesspass" result indicates that no DMARC TXT record for the domain exists, but if one had existed, the DMARC check for the message would have passed. This is because the domain in the originator address (email envelope) matches the domain in the "From" address (email header) [114].

**Requirements:**

1. Pass on SPF must give 100\% score, and fail must give 0\%.
2. Pass on DKIM must give 100\% score, and fail must give 0\%.
3. Pass on DMARC must give 100\% score, bestguesspass must give 50\%, and fail must give 0\%.

**Attachments**

Emails containing attachments were given a percentage score of 0, and emails without attachments were given a 100\% score. The email attachment percentage score was not included in the total evaluation of the email. This is because too little information about attachments is available in an email. Not all phishing emails include attachments, making them affect the total evaluation score too much if included.
Requirements:

1. Emails with attachments must give 0%.
2. Emails without attachments must give 100%.

**Total evaluation**

For each section of requirements, a total evaluation was given. All section's total evaluation is then calculated into a single percentage score, which determines the likelihood of an email being legitimate or phishing.

![Diagram showing evaluation process](Diagram)

Figure 23: Illustration of evaluation percentage score system

As seen in figure 23, one section can contain many requirements, each having a percentage score based on its unique functionality. All requirement's percentage scores were added together, forming one percentage score representative for the whole section. These section percentage scores were then added together forming a total evaluation percentage score for the whole email.

The evaluation score of all sections and the total evaluation score of the whole email is an average of the number of requirements. Ex: a section containing three requirements would be added together and divided by three to find the average score for the section.

Requirements:

1. Each requirement section must have its own percentage score evaluation.
2. The whole email must have one total evaluation score based on the evaluation scores from the requirement sections, excluding the attachment section.
3. The section evaluation percentage score and the total evaluation percentage score must be an average of summed requirements percentage scores.
User manual

A user manual had to be implemented to ensure that users understand the meaning of the functional requirements. This was important so that users with a non-technical background could learn about email phishing awareness.

The implementation of the user manual had to be smart. This meant it had to fit in information without distracting the user. It had to give answers to the users in scenarios where it would be natural to question application functionality. When presented with some functionality, there had to be an easy way to be informed with minimal interaction.

Requirements:

1. For each functional requirement, an explanation must be given.
2. The explanations must be easy to access.
3. The explanations must not be distracting or misleading.

3.1.3 Design requirements

To be able to fulfill the requirements of designing an application that is intuitive, easy to understand, and easy to navigate, the application had to utilize certain development practices and design principles.

Microsoft offers design principles and best practice development for add-ins:

- Best practices for add-in development: [116].
- Add-in design principles: [117].

Office design principles recommend using Microsoft's Office Fabric UI (Fluent UI React) as JavaScript application design [118]. For this project, Material UI was used instead due to a wider selection of components and preferable designs such as styles, icons, and controls [18]. It was important to utilize design principles for reusable components, as there would be many functional requirements that could use the same component structure. Material UI had the preferred components for this as they were easy to implement and customize for own projects.

Specific colors had to be chosen for the UX design to enhance the feedback users were given. The context of email safety was enhanced by using the traffic light colors red, orange, yellow,
and green. Green is associated with being safe, orange, and yellow associated with the attention needed and red being a warning or alert of danger [119].

The design of the application had not to be annoying or distracting for the user. Using the neutral colors white, black, and grey to not distract the user from the actual email client. Also, it made the bright colors red, orange, yellow, green, and blue even more visible. The color blue was chosen for interactivity, such as clickable buttons. Making it intuitive which components hold interactable functionality [120].

![Colour palette for application](image)

**Figure 24: Colour palette for application**

**Requirements:**

1. The application must use Material UI for UX design.
2. The design must utilize traffic light colors to represent levels of email safety.
3. The design must use neutral forms and colors not to distract or annoy the user.
4. For clickable buttons and interactable components, the color blue must be used.
5. The components must be intuitive enough to understand its functionality without further explanation.
3.1.4 Operational requirements

To fulfill the requirement of testing the application in an email client, it had to be made in an environment that supports the development and integration of applications. It also had to be free and easily accessible as resources available for this project were limited.

Microsoft's email client Outlook fitted these requirements. Microsoft provides an application development environment called Office add-ins [121]. It is an environment with cross-platform support that can run on the web, Windows, Mac, and iPad. Office Add-ins extend the UI and functionality of Office applications and interact with content in Office documents [122].

Outlook.com is one of the biggest email platforms in the world. It has over 500 million users worldwide. It also has a large add-in community for developers with proper code documentation and helpful tutorials and guides [123]. By using Outlook.com, free and easy access to the development and testing of emails was given. Therefore, for this project, the web version of Outlook (Outlook.com) was chosen.

Outlook add-ins are developed in JavaScript by using the Office.js library to fetch data from emails such as headers fields and message body content [124]. This meant the application had to be written in JavaScript.

Helpful tools such as Yeoman generator was used as it provides easy setup of a React.js template project for Office add-in development. Microsoft also encourages the usage of Yeoman for add-in development [125]. This made React.js the JavaScript framework to use [126].

The sideload feature in the Microsoft development environment also enabled real-time development testing of the add-in application [127]. Node.js made it possible to run this locally on a computer [128]. Microsoft's Visual Studio Code was chosen as the IDE to write code as it is a Microsoft that is easy to learn [129].

Requirements:

1. The application had to be developed on Microsoft's Office add-ins platform to be able to run on Microsoft's email client Outlook Live (Outlook.live.com).
2. Must use free and easily accessible tools and frameworks for development due to limited resource access.

3. Must use software that is compatible with the Outlook Office add-in environment.
Chapter 4

Design

With all application requirements defined, the structure of the add-in was set. The next step was to design the application. This chapter includes the visual design of the add-in application.

4.1 Visualizing the structure of the application requirements

![Diagram showing the structure of the application requirements]

Figure 25: Mock-up of functional requirement sections % connection
As seen in figure 25, a visualized model of the functional requirement sections with their requirements, each giving a percentage score, all added together to a total evaluation percentage score. The connection between the requirement sections is illustrated as the black lines between the boxes. The connection involves the "Total evaluation" box being the main page of the application, and each subpage (requirement sections) sends their percentage score data to it.

4.2 Modelling the add-in

Figure 26: Screenshot of Outlook Live add-in side panel

The rectangle area with the red outline seen in figure 26 represents how add-ins are displayed in Outlook Live. The add-in is a side panel that can be displayed when an email is open (read mode). There was not a lot of space to work with, putting restraints on the architectural structure and possible design of the application.
Figure 27: First sketches of the add-in

As seen in figure 27, sketches were drawn based on the requirements defined and the available space for the add-in.

The application would contain three main components;

- The requirement component
- The total evaluation component
- The attribute details list component

As seen in figure 27, the right sketch illustrates the "attribute details list" component. "The total evaluation" component was placed above this list.

The sketches in figure 27 represented the main page of the application. Links in the attribute details list would take the user away from the main page to the selected requirement section page. The sections had to display their own requirements and percentage score evaluation.
As seen in figure 28, the requirement section pages had a reversed page layout to the main page. The evaluation was placed at the bottom of the page rather than the top. Each section had to have a way of taking the user back to the main page using a button.
4.2.1 Requirement component

Figure 29: Sketch of requirement component

![Sketch of requirement component](image)

**Figure 30**: Model of a requirement component (made with Material UI)

![Model of a requirement component](image)

Figure 30 is the result of including the functional-, informative-, design- and operational requirements to create the requirement component.

The requirement component model was made simple and generic, therefore enabling it to be duplicated easily. This was important when developing the application as it contains many similar requirements. The component was also used as a component template for the attribute details list, the total evaluation, and the requirement section evaluation. The calculation of the percentage score was different for most of the requirements, see section 3.1.2 Informative requirements.

**Component title**

As seen in figure 30, the text field "Title component" would be replaced with a title describing the specific requirement.

**Information bubble**
Figure 31: Information bubbles explaining a requirement (Material UI)

Based on the informative requirements for the user manual, the information bubble was created to provide information that was easy to access with minimal user interaction, given at the right place in the application. Based on the design requirements, the (?) icon is an intuitive symbol connected to questions and answers, and the blue color displays that it is interactable.

**Percentage score bar**

As seen in figure 32, based on the informative- and design requirements, a model of the percentage bar and its possible interval levels was created.

![Percentage Score Bar](image)

Figure 32: The different levels of percentage score bars

**Component information**

As seen in figure 30, the "test text" would be replaced with important information about the specific requirement.
4.2.2 Total evaluation component

Figure 33: Sketch of total evaluation component

![Sketch of total evaluation component](image)

Figure 34: Model of total evaluation component (made with Material UI)

![Model of total evaluation component](image)

As seen in figure 34, a model of the total evaluation component displays the percentage score from each requirement section. These are summed together and divided by four, giving an average total percentage score.

The levels of email safety were included in the total evaluation components. Figure 34 shows two of the email safety levels, low and high. This was added to make it easier for users to understand the results of the total evaluation.
4.2.3 Attribute details list component

Figure 35: Sketch of attribute details list component

The attribute details list component was designed to provide links to the requirement sections. Each link had to be placed inside a requirement component model with a title and description of the given requirement section.

As seen in figure 36, the "learn more" text is a button with a hoovering effect (displays a darker color when the mouse is over it). Clicking this button would take the user to a new page, displaying the represented attribute (requirement section).

The total evaluation attribute had nothing to do with the functional requirements or percentage score system. It was a page in the application dedicated to explaining how the percentage score system works and how it should be used as guidance in email interaction and awareness of phishing emails.
Chapter 5

Implementation

After defining the requirements and designing a visual presentation of the application, it was time to implement a prototype solution. For that, the presented concepts had to be turned into code that could run the actual application. This chapter includes implementation and the result of the complete application.

5.1 Selection of development software and tools

The following is a list of software, frameworks, and tools used to develop the application.

1. React.js (JavaScript Framework) [126]
2. Material-UI (React.js components) [18]
3. Office.js library (Outlook JavaScript API) [124]
4. Yeoman generator for Office Add-ins [125] [130]
5. Node.js to run a server locally [128]
6. Visual Studio Code (IDE) [129]

5.2 Set-up of the development environment

Before starting to code, the development environment had to be set-up. First, all the necessary programming tools and software were downloaded, see section 5.1 for the list of development tools and software used.

For the reasoning behind the selection of development tools and software, see Section 3.1.4 Operational requirements.
For installation guide and set-up of these tools see the Digital Appendix B.1 GitHub – Read me file online: https://github.com/haakbe93/email-phishing-protection-add-in/blob/master/README.md

5.3 Connecting and running the add-in on Outlook

It was important to be able to run the application when implementing functionality. Making it possible to instantly see new changes and outcomes of implementing code in the application. Through this iterative process of coding, then running the application, then making the needed changes, making it a lot easier to spot bugs and errors, ultimately saving a lot of time.

To make this possible, the application had to be connected to Outlook’s web interface and run locally on a computer.

XML manifest

An Extensible Mark-up Language (XML)-file would be used to connect the application to Outlook’s web interface. Generating an Office Outlook add-in template with Yeoman generator created an XML-file in the project called manifest.xml.

![manifest.xml file](image)

Figure 37: Screenshot of the manifest.xml file

The Office add-in manifest (the XML-file) defines the settings and capabilities of the add-in. It can be configured to specify things such as:
- Metadata that describes the add-in (for example, ID, version, description, display name, default locale).
- Office applications where the add-in will run.
- Permissions that the add-in requires.
- How the add-in integrates with Office, including any custom UI that the add-in creates (for example, custom tabs, ribbon buttons).
- Dimensions of the add-in (for example, dimensions for content add-ins, requested height for Outlook add-ins).

[131]
The manifest.xml with its specifications was uploaded to Outlook.com, making it available as a personal add-in. It will only be available for the Outlook community if it is uploaded to AppSource [132] or if the XML file is shared and installed locally on a user's Outlook.

Figure 38: Screenshot of adding a personal add-in in Outlook. (Norwegian)
Figure 39: Screenshot of adding the project XML-file to personal Outlook add-ins

Figure 38 and figure 39 shows how the XML-file was added to personal add-ins in Outlook. Outlook reads the XML-file, converting it into runnable code, ultimately displaying the add-in for the user. For the application to work, it had to be run on a server, which could be done with the help of Node.js.

Running Node.js

Figure 40: Screenshot of using the "npm start" command line in VSC terminal
For the application to be able to run locally on a computer, it had to be deployed through Node.js. This was done by running the command "npm start" in Visual Studio Code's (VSC) integrated terminal. NPM is the default package manager for the JavaScript runtime environment Node.js, which includes npm as the command-line client. To be able to use these commands, NPM had to be installed [128].

**Sideloading add-in**

With the XML-file uploaded to personal add-ins in Outlook and having Node.js running, it was possible to sideload the application and see how the code implementation would look like and work in the add-in.
Figure 42: Screenshot of sideloading the add-in in Outlook with the web console

Figure 42 shows how debugging and testing of development was done. Any changes made in the code in Visual Studio Code (IDE) would be updated and shown in the application. The web console would show any errors or other outcomes of changes made in the code [127].
5.4 Application requirements as software architecture

All the defined requirements had to be translated into an organized architectural structure containing code that would run without errors.

JavaScript files were created for each of the requirement sections: sender, content, links, authentication, and attachments. Each of them had to contain code based on the defined functional-, informative- and design requirements.

![Screenshot of application JavaScript files](image)

Figure 43: Screenshot of application JavaScript files

As seen in figure 43, the "CardinfoList.js" file represents the "attribute details list" component, and the "MainList.js" file represents the main page of the application containing the "total evaluation" (total percentage score) component and the "attribute details list" component.

The "TotalEvaluation.js" file represents the user manual page in the application, found in the "attribute details list" component. The file "spamWordData.json" is a JSON-file that holds the phishing words used in the "Content.js." "App.js" holds the routing of the application and code, enabling the Office.js API. "MouseOverPopover.js" holds the code for the information bubble found in each requirement component.
In figure 44, the arrows pointing downwards to another class represents the parent-child class relation. Starting at the top being the parent root class index.js, which points to the children class App.js, which further points to its child MainList.js and so on.

5.5 Accessing information in the email header and body

Information found in the email header and email body had to be accessed to be able to use the functional requirements and calculate percentage scores. The web application uses Office JavaScript API to interact with content in the Office document where the add-in is running. This means that special objects and methods in the Office.js API would need to be used to access desired information in Outlook emails.

The JavaScript API interfaces can be found at Microsoft’s developer documentation website: [133].

To access objects and members specifically for use in Outlook add-ins, there were one very important API references that had to be used:

```javascript
var message = Office.context.mailbox.item;
```

The "Office" namespace provides a shared interface that is used by add-ins in all the Office
apps. The "context" namespace represents the runtime environment of the add-in and provides access to key objects of the API. The "mailbox" namespace provides access to the Outlook add-in object model. The "item" namespace is used to access the currently selected message.

The object reference was fetched with the variable "message."

message.getAllInternetHeadersAsync

The reference "message" was used to access the method "getAllInternetHeadersAsync." The method made it possible to access all email headers fields and their content for each email that was opened in Outlook. This was important as the application needed access to the results of the authentication protocols SPF, DKIM, and DMARC, which could be found in the email header. The method was also used to access information about the sender's domain and IP address. The method would not list email header fields, but rather the whole email header as a text string. Therefore, special search methods had to be used to find the authentication results.

```javascript
//SPF
var SPF = "spf=";
var splitt_SPF = dir.split(SPF).pop();
var t_1 = splitt_SPF.match(/none|pass|neutral|fail|softfail|permerror|temperror/g);
var val_1 = t_1[0];
//DKIM
var DKIM = "dkim=";
var splitt_DKIM = dir.split(DKIM).pop();
var t_2 = splitt_DKIM.match(/none|pass|fail/g);
var val_2 = t_2[0];
//DMARC
var DMARC = "dmarc-"
var splitt_DMARC = dir.split(DMARC).pop();
var t_3 = splitt_DMARC.match(/none|pass|bestguesspass|fail/g);
var val_3 = t_3[0];
```

Figure 45: Screenshot of JavaScript email authentication result methods

message.body.getAsync(Office.CoercionType.Text, function callback(result) {

The "body" namespace represents the content of the email message. It could be accessed by referencing message.body. This reference allowed for the use of the "getAsync" method, which would return the email body message. Inside the body message could elements such as URL be found. Therefore, the method was used to fetch all URLs in Outlook emails.

message.body.getAsync("html", function callback(asyncResult) {
As seen below, the three references were used to get the sender's email address, display name, and the email account owner's display name. These were used in the sender and content requirements sections. "userProfile.disaplayName" fetches the username of the logged-in email account. This was used in a method to check that yourself was not the sender of the email.

\begin{verbatim}
message.sender.displayName
message.from.emailAddress
message.userProfile.displayName
\end{verbatim}

5.6 Optimizing the percentage score system

It was important that the percentage score for each functional requirement would be balanced compared to each other. There were some requirements that had a greater impact on the safety of an email. Requirements for authentication results, use of https in links, and sender's display name and email address similarities, were among the most impactful. Therefore, they had to be given a higher percentage score. The score given had to be balanced compared to other requirements. Ex: If a lower rated requirement would give a percentage score of +30% for a match, a higher rated requirement had to give a percentage score of +50% for a match. This score difference relation had to be the same for all functional requirements.

As requirements were developed and implemented, testing of the application was done. The testing was done on both phishing and legitimate emails. This feedback loop of test information helped tweak the percentage scores. Ex: one phishing email could be triggered on certain requirements making it seem important to have a high percentage score, but the next day testing could show that new phishing emails triggered on other requirements, making the previously percentage score tweaks inaccurate. Therefore, it became an iterative process until the very end of the application development. This to give the best possible percentage score balance based on the emails that were tested.
Figure 46: Screenshot of calculation of percentage score for link requirements

The percentage score algorithms for the link section as they were presented in the informative requirements, can be seen in figure 46. See section 3.1.2 Informative requirements.
5.7 The finished add-in application

The result of the development done in this project can be divided into the following sections:

1. Application functionalities
2. Microsoft Office Outlook add-in

5.7.1 Application functionalities

The GitHub repository contains components giving the Outlook add-in the following functionalities:

1. Routing between application pages
2. GUI for accessing functionalities
3. User manual
4. Percentage score system for rating section's functionalities
5. Sender section:
   a. Fetching sender's display name and email address
   b. Comparing similarities between them
   c. Fetching sender's IP address
6. Content section:
   a. Fetching profile name of email user
   b. Comparing email body text to the phishing word list
7. Link section:
   a. Fetching links found in the email
   b. Checking encoding of links
   c. Checking links usage of HTTP/HTTPS
   d. Checking links usage of redirection
   e. Checking duplication of links
8. Authentication section:
   a. Fetching email authentication protocol results
   b. Checking results of SPF
   c. Checking results of DKIM
   d. Checking results of DMARC
9. Attachment section:
   a. Fetching email attachments
   b. Counting and displaying file names

All developed files can be found in the project's GitHub repository:

https://github.com/haakbe93/email-phishing-detection-add-in

5.7.2 Microsoft Office Outlook add-in

The application developed in this project was implemented in Microsoft Office Outlook Live. To be able to run on Outlook Live, the application had to be developed in Microsoft's Office Outlook add-ins environment as described in section 3.1.4 Operational requirements.

Figure 49: Screenshot of the application running in Microsoft Office Outlook Live
Figure 50: Screenshot of the application's main page

As seen in figure 50, the final version of the application's main page, showing how a phishing email is rated and how a legitimate email is rated.

The result of developing the application in this project can be seen in a demo video, which shows how the application works. This video can be found in the Digital Appendix B.2 Demo of Application, which is published online at:

https://youtu.be/T9jzo4Z1yLk
Chapter 6

Application test

This chapter will present testing of the application's capabilities to detect phishing emails based on the functionalities implemented.

Research questions 2 of this thesis, listed in section 1.2, was:

"Based on such patterns, is it possible to develop an application capable of detecting phishing emails?"

The chapter includes results from the test and analysis and discussion of the results, answering the research question.

6.1 Test layout

An application detection test was performed to measure the application's email phishing detection capabilities. This involved being able to separate between legitimate and phishing emails. The application was tested on 300 emails.

The test was split up in two sections:

- Legitimate emails: test the application on 150 emails that are known to be legitimate.
- Phishing emails: test the application on 150 emails that are known to be phishing.

Setup

All emails tested were gathered from Outlook.com. Emails were sampled from the inboxes of six persons. The number of phishing emails collected from each person's inbox varied due to the nature of receiving phishing emails being random. Only emails that managed to get through Office Outlook spam filter were tested. In many cases, spam and phishing emails are
quarantined to the spam inbox, but in some cases, they don't. One of the reasons for this is due to different settings on the spam filter. The settings for the participant's spam filters were all set to the same default settings in this test.

Each participant logged in to their email account on the same computer and ran the add-in application locally. To run the add-in application locally on a computer, see section 5.3 Connecting and running the add-in on Outlook.

For the two sections of 150 emails, it was known beforehand of the test if the emails were phishing or legitimate. This was known based on thoroughly analyzing the emails to determine their nature.

For each email, it's percentage score for the sections; sender, content, links, and authentication were collected. Data from the attachment section of the application was not collected in this test due to it not affecting the total evaluation percentage score. It was tested in the user study instead, see chapter 7.

<table>
<thead>
<tr>
<th>Email #</th>
<th>Sender %</th>
<th>Content %</th>
<th>Links %</th>
<th>Authentication %</th>
<th>Total evaluation %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Email #</th>
<th>Sender %</th>
<th>Content %</th>
<th>Links %</th>
<th>Authentication %</th>
<th>Total evaluation %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 47: Screenshot of how the test results would be logged

As seen in figure 47, an excel document was created to log all results from the test, displaying how accurate the application could separate legitimate and phishing emails based on the percentage score evaluation it gave.
The email safety levels: no safety, low safety, medium safety, and high safety, where used in the test to evaluate the emails. See section 3.1.2 Informative requirements for details about these levels.

Tolerance limits were set for the test. For legitimate emails, a tolerance limit was set to be above a 70% total average score. For phishing emails, it had to be below a 50% total average score. This meant that legitimate emails had to be in the upper "medium safety" level to be considered legitimate. Phishing emails had to be in the middle "low safety" level to be considered phishing.

The tolerance limits were set based on the application requirement’s percentage score calculations, see section 3.1.2 Informative requirements. The most impactful requirements would either give a high percentage score ratings for passing requirements or low percentage score ratings for failing requirements. See section 5.6 Optimizing the percentage score system. This meant that broad limits of 50% and 70% would fit well. The tolerance limits were set so that it would be easier to rate emails as phishing and harder to rate emails as legitimate. It was desired to have a higher false-positive rate for phishing emails rather than having any phishing emails labeled as legitimate.

**Test reliability and validity**

Reliability refers to the degree to which an instrument yields consistent results [136]. The reliability of the application detection test was dependent on the static functionality of the application. This meant the test would give the same results if the same emails were to be retested.

Validity refers to the degree to which an instrument accurately measures what it intends to measure [136]. The validation of the test was dependent on the emails used. The test only included 300 emails, which cannot be representative for the billions of emails that are sent through Outlook Live. There exists lots of different email scenarios and context not included in the test that could have impacted it differently. In general, a larger test dataset would give a better average score. The results of smaller datasets tend to be more sensitive to outer points.
6.2 Test results

The data collected in the application phishing detection test was taken from 300 emails, 150 legitimate emails, and 150 phishing emails. The data log form consisted of five fields, which each had a percentage score from 0 to 100, as seen in figure 51. The percentage scores represented each of the requirements sections: sender, content, links, authentication. See section 3.1.1 for details about the requirements sections. The total evaluation score of these sections (the average score) can be seen in the right-most field.

The data collected were logged using the schema presented in section 6.1 Test layout. The data were given colors based on what email safety level they would fit. See section 3.1.2 for details about percentage scores and email safety levels.

<table>
<thead>
<tr>
<th>Legitimate emails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Email #</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Phishing emails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Email #</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Figure 51: Screenshot of a small sample of the application test schema results

A full log of data collected in the application detection test can be found in Appendix A.1 Application Detection Test.

<table>
<thead>
<tr>
<th>Average percent score of 150 legitimate emails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Emails</strong></td>
</tr>
<tr>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average percent score of 150 phishing emails</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Emails</strong></td>
</tr>
<tr>
<td>150</td>
</tr>
</tbody>
</table>
The average percentage score was calculated as \((\text{the sum of a column}) / 150\).

As seen in figure 52, the total average score for each of the columns in the test data schema. The average percentage score for the total evaluation field fell within the tolerance limit that was set in section 6.1, which was above 70% score for legitimate emails (result: 76%) and below 50% for phishing emails (result: 28%). This was the average percentage score of all the emails in the application test. It did not mean that every email met the tolerance limit. See section 6.3 for further analysis and discussion of the application test results.

### 6.3 Analysis and discussion of the results

Following is an analysis and discussion of the results from the application detection test. All models presented in this section were produced from the data collected in the application detection test.

**Total evaluation % for legitimate emails**

The following diagram contains all logged data from the application detection test schema field "total evaluation" for legitimate emails.

![Diagram of legitimate email's total evaluation %](image)

Figure 53: Diagram of legitimate email's total evaluation %
In figure 53, the interval of the possible score was set between 0% to 100%. There were no legitimate emails that scored lower than 40%.

As seen in figure 53, the 150 legitimate emails total percentage scores had some variations. Some of the emails did not fulfill the tolerance limit that was set to 70%, see section 6.1 Test layout. A total of 35 out of the 150 emails had a lower total evaluation score than 70%. That meant 23.33% of the legitimate emails used in the application detection test did not meet the tolerance limit. But that also meant that 76.66% of the emails were successfully classified as legitimate.

Out of the 35 emails that did not meet the tolerance limit of 70%, a total of 5 emails had a percentage score lower than the tolerance limit set for phishing emails, which was 50%. This meant that these five legitimate emails could be considered being phishing emails. 5 out of 150 emails equals 3.33%, meaning the application had a false negative rate of 3.33% for legitimate emails.

Figure 54: Screenshot of the five legitimate emails with a score under 50%

In figure 54, four of the five false-negative legitimate emails had a zero percent score in one of the sections, drastically reducing the total percentage score. False-negative legitimate emails were spotted due to the context of the emails. They were known to come from trusted sources but did not meet the application requirements.

As seen in figure 53, there were no emails that got a total evaluation score of 100%. The highest score was 99%, and the lowest was 41%. Out of the 150 legitimate emails, 83 of them had a total evaluation score between 73% and 87%. Knowing these numbers and seeing the diagram in figure 53, indicated that the application's percentage score system had too specific criteria for emails to receive high percentage scores.
**Total evaluation % for phishing emails**

The following diagram contains all logged data from the application detection test schema field "total evaluation" for phishing emails.

![Diagram with phishing email's total evaluation %](image)

**Figure 55: Diagram with phishing email’s total evaluation %**

In figure 55, the interval of the possible score was set between 0% to 100%. There were no phishing emails that scored higher than 85%.

The phishing email total evaluation percentage scores were more clustered compared to the total evaluation percentage score for legitimate emails. As seen in figure 55, most of the phishing emails can be found to the left, below the phishing email tolerance limit of 50%. Only 11 emails were rated above the 50% phishing tolerance limit, which was only 7.33% of the 150 phishing emails. Meaning the application had a 92.67% success rate of rating phishing emails. This was a lot better than the 76.66% success rate for legitimate emails.

120 of the 150 phishing emails had a total evaluation percentage score between 14 and 32. The highest being 83 and the lowest being 11. This was another indication of the percentage score system being strict when giving a score. The application gave a lower percentage score overall, both for phishing and legitimate emails. This made it harder for phishing emails to pass as legitimate emails. But it was considered better to have higher numbers of false-positive phishing emails, than having any phishing emails pass as legitimate emails.
No phishing emails received a 0% total evaluation score. As seen in figure 56, the content and link section were the reasons that the phishing email did not receive a lower score than 11%. None of the 150 phishing emails received 0% in the content and link section, indicating that these sections would keep the total evaluation score above 11% for all emails. The lowest email safety level was between 0% and 25%, meaning that many phishing emails were still labeled with the lowest email safety possible.

Figure 56: Screenshot of lowest rated phishing email

The application was stricter at giving a higher total evaluation score overall but never lower than 11%. Having a system rating emails not too high and not too low, would indicate that emails would be rated in the middle, an average, which could make it harder to separate phishing emails from legitimate emails. This meant that the email safety interval levels played a more critical role than initially thought. The email safety intervals levels ensured good enough separation of the emails when the percentage scores would have less significance. See section 3.1.2 Informative requirements for details about the email safety interval levels.

Figure 57: Screenshot of false-positive legitimate emails

There were some false-positive legitimate emails. Figure 57 shows the percentage scores of the five phishing emails that tricked the application detection test as being legitimate emails. This meant the application had a false-positive legitimate email rate of 3.33%. They were only considered phishing emails due to the context of the emails, which the application could not detect. Ex: some of the emails contained information about picking up a package from the post office that the recipient of the email never ordered.

This showed that it was possible to trick the application into rating phishing emails as legitimate emails, which was a sign of it not being flawless. The reason for this was the
functional requirements that were developed were not failproof. Cybercriminals are known to compromise legitimate email accounts and use them in further phishing campaigns. This allows them to pass authentication protocols and use valid display names and email addresses. The use of HTTPS has also become more prevalent in phishing emails, making it harder to detect malicious URLs.

**The sections impact on the total evaluation %**

The sections: sender, content, links, and authentication are what make up the total evaluation percentage score for both legitimate and phishing emails. It consists of the sum of their percentage score added together and divide by four. The different section's percentage score was analyzed to understand the outcome of the total evaluation percentage score.

Figure 58: Diagram comparing phishing sender % and legitimate sender %

As seen in figure 58, the percentage score given to the sender section of both phishing emails and legitimate emails. Phishing emails would have mostly 0% and legitimate emails 100%, with some occurrences on 30%, 60%, and 90%.

Some phishing emails managed to get 100% in the sender section. As mentioned earlier, one reason for this could be phishing emails using compromised email accounts. Under the application test, it was found out that the functionality that compared the sender's display name and the sender's email address could not read non-ASCII signs, such as emojis. This led
to comparing words with empty space, making it match with anything, resulting in a percentage score of 100.

As seen in figure 58, some legitimate emails got 0%. Under the application test, it was found out that the application does not check for the use of the phonetic alphabet letters "æøå." Some of the emails used in the test were received from Norwegian persons with names containing these letters. Their sender's display name would contain letters that their email address would not, resulting in no match, giving a percentage score of 0.

Other than that, the sender section seemed to be efficient enough to separate the emails used in the application detection test.

![Diagram comparing phishing content % and legitimate content %](image)

**Figure 59: Diagram comparing phishing content % and legitimate content %**

As seen in figure 59, the orange diagram lines represent the percentage scores for the 150 legitimate email's content section. The blue lines represent the percentage score for the 150 phishing email's content section.

Legitimate emails scored higher than phishing emails, but most of the emails were in the same percentage score interval between 34% and 64%. This can be seen in figure 59, as the graph spikes in the middle of the diagram. 22 was the highest number of emails with the same percentage score, which was at 50%.
The main reason for the content section score similarities for phishing and legitimate emails was that the recipient's name was missing in the emails. Lack of recipient's names proved to be more common for legitimate emails than initially thought (at least in the 300 emails tested).

The content evaluation score could never be zero, due to the phishing word list requirement (at least not recorded in the 300 emails tested). There were never enough word matches to give a 0% score. This was one of the reasons the application would never rate any emails with a total evaluation of 0%.

These factors led to the content section percentage score having little impact on the total evaluation as it gave almost the same percentage score for phishing emails and legitimate emails. It proved to be a functionality that did little to separate between phishing emails and legitimate emails.

<table>
<thead>
<tr>
<th>Average percent score of 150 legitimate emails</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emails</td>
</tr>
<tr>
<td>150</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Average percent score of 150 phishing emails</th>
</tr>
</thead>
<tbody>
<tr>
<td>Emails</td>
</tr>
<tr>
<td>150</td>
</tr>
</tbody>
</table>

Figure 60: Average percentage score for application test schema fields

As seen in figure 60, the gap between the average percentage score for the legitimate email's content section and the phishing email's content section was much smaller than for the other sections. This further illustrates that the content section had less of an impact on separating phishing emails and legitimate emails.
Figure 61: Diagram comparing phishing links % and legitimate links %

Unlike the content section, the link section had a much more evident division of phishing emails and legitimate emails. In figure 61, 94 of the 150 phishing emails got a percentage score between 15 and 23. Phishing emails could mostly be found on the lower part on the percentage score range, but for legitimate emails, it was more scattered between 46% and 100%. 49 of the 150 legitimate emails had a score below 70%. A reason for this was that some legitimate emails contained links to other pages such as social media that the application's functionality compared inefficiently with the sender's email address or display name, resulting in a lower percentage score rating.

The highest amount of legitimate emails found at a percentage score were at 100%. Therefore, the link section still made a good indicator for separating legitimate emails from phishing emails.

Nine phishing emails were given 100% because they did not contain any links. The logic behind this was that the threat landscape of URLs was eliminated, making only attachments and social engineering the possible threats. Only five of the legitimate emails contained no links, making the 100% score still valid.

Four legitimate emails were given a 15% score. One of the reasons for this was that they contained links using HTTP, not HTTPS. Some legitimate websites are outdated, still using HTTP, which is concerning.
Like the content section, the link section would never give an evaluation score of 0%. This was because the link requirements: encoding, duplication, or redirection would always give a percentage score higher than 0. Together with the content section, this made it impossible to give an email a total evaluation score of 0%.

![Diagram comparing authentication %'s](https://via.placeholder.com/150)

**Figure 62: Diagram comparing authentication %'s**

For the authentication section, there were fewer outcomes. As seen in figure 62, most phishing emails could be found at 0% due to failing authentication. Some phishing emails could be found at 33% and 50% due to passing either one or two of the three authentication protocols. The ones with 83% or 100% could be phishing emails using compromised email accounts.

For legitimate emails, most got 100%, but there were quite a few with 67% and 83%. The reason for this is that not every domain has enabled SPF, DKIM, and DMARC as it is something a domain administrator must setup. This makes email authentication protocols less consistent than desired. Authentication was still an essential indicator of separating phishing emails from legitimate emails.
6.4 Application improvements

The application functionalities had varying levels of impact on email phishing detection. Some features performed better than others, revealing what had an effect and what did not. For the functionalities that did not have a desired level of impact, improvement should be implemented. These improvements can be divided into the following categories:

**Improve data normalization**

All input from display names and email addresses should be checked for non-ASCII signs. It should be ensured that such signs do not create valid word matches.

If the phonetic letters "æøå" are present in the sender's display name or email address, they would need to be replaced by the letters a/ae, o, and a/aa. This must be expanded to other unusual letters as well.

Other options would be to include better algorithms for string similarities such as Levenshtein distance [134] or Jaro-Winkler distance [135]. These could give a more exact comparison.

**Improve analysis of email message content**

The phishing words checklist had little impact on detecting phishing emails, as it produced almost the same results for both phishing and legitimate emails. To improve this functionality, an expanded list of phishing words, phrases, and sentences should be implemented.

For a dynamic solution, the implementation of machine learning for text classification of email messages could give better results. This requires more resources but could be much more reliable. Implementing TensorFlow.js, an end-to-end open-source machine learning platform for JavaScript [105], could be a solution. With enough data, both phishing emails and legitimate emails, it would be possible to train the machine learning algorithms to classify the text found in emails.

**Improve the percentage score system**

Tuning the percentage scores given to the different application sections could improve the accuracy of rating phishing emails and legitimate emails. The sections sender, links, and authentication proved to be the most important sections for separating phishing emails from legitimate emails. The percentage score given to these sections should impact the total evaluation score more.
For the section "message content," the phishing word checklist requirement should have a more significant impact on the section's percentage score. Meaning each phishing word match should subtract 4% instead of 2% of the requirement's starting score of 100%. See section 3.1.2 Informative requirement.

For the link section, the functional requirements encoding, duplication, and redirection percentage scores should be tuned in a way that allows emails to receive a total evaluation percentage score lower than 11%.
Chapter 7

User study

This chapter will present a user test to see if the application would be helpful for people when interacting with emails and if it would improve their awareness of email phishing.

Research questions 3 of this thesis, listed in section 1.2, was:

“By using such an application, can email users learn to become more aware of email phishing?”

The chapter includes results from the user test and analysis and discussion of the results, answering the research question.

7.1 Test layout

A user test was conducted to measure the application’s effect on email users' awareness of email phishing.

Test subjects

The user test was done on individuals that had little to no knowledge about email phishing but uses email daily. This was important as it was thought that persons with minimal experience would show more significant improvement in awareness of email phishing, making it easier to measure a difference.

The test group consisted of five persons. Ranging from the age of 23 to 57, none of them with any IT or computer technical background. There were one female and four male participants.
Setup
The test consisted of two rounds, which each consisted of three phases. Each phase involved ten emails, which the test persons had to look at and decide if were either phishing or legitimate emails.

- Phase 1: evaluate ten emails without any tools.
- Phase 2: evaluate ten emails with the help of the application.
- Phase 3: evaluate ten emails without any tools.

First, the test persons would go through phase one, consisting of ten emails. Out of the ten emails, half of them would be phishing, and the other half would be legitimate emails. The test persons would need to look through the ten emails and decide on which of them were phishing or legitimate. For the second phase, also consisting of ten emails, the test persons would use the application to help them decide which emails were phishing or legitimate. For the third phase, the test persons had to determine which emails were legitimate or phishing without the application. This to see if using the application would have any effect on their decision making and if they were capable of learning anything from using it in the first place.

The user test was performed in two rounds on the same individuals with a one-week interval. The reason for this was to see if they would learn and remember something from the first round that could help them perform better in the second test round one week later.

The emails that were used in the test were gathered from Outlook. A total of 60 emails were used in the test, consisting of 30 phishing emails and 30 legitimate emails. The test would be done through Outlook.com. The application would be run locally on one computer, which would let the test persons use it in Outlook.

The test conductor would observe the test persons, making notes on what they said and did during the test. The participants stood on their own to find out how the application works. It was allowed for questions to be answered, but the goal was that the application would give the participants enough information to separate between legitimate emails and phishing emails.
The schema seen in figure 48, was used to document the results of the user test. For the whole test, six schemas were used, one for each of the three phases, in two rounds (one-week interval). See appendix A.2 User test for the full user test.

The user test also included questions about the participant's knowledge of email phishing. These were asked before the test started. After the test was finished, the participants were asked to comment on the test and the application. They were also asked questions about what they had learned.

**Test reliability and validity**

The reliability of the user test was dependent on whether the participants would improve their email phishing awareness from the first to the second time they took the test. The whole point of the test was to try to measure this improvement. This meant that if the test were conducted on new participants with similar email phishing knowledge, they would show a similar improvement in email phishing awareness.

The test was thought to be valid due to the test environment. The participants had to read emails like they usually would do. Then based on their evaluation, they were to answer if they thought the emails were phishing or legitimate. A factor that may have separated the test from a real situation was the fact that the participants were being observed and knew they would be exposed to phishing emails. Therefore, they may have been more observant and alert than usual.

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
<td>L</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>P</td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
<td>L</td>
</tr>
</tbody>
</table>

Figure 48: Screenshot of the schema used in the user test
7.2 Test results

The results in the user test were collected from five participants. The data collected shows how good the participants were at categorizing 60 emails as either phishing or legitimate. Each participant categorized 40 emails without the help of the application and 20 emails with the help of the application.

<table>
<thead>
<tr>
<th>Test person #</th>
<th>Wrong</th>
<th>Right</th>
<th>Legitimate</th>
<th>Phishing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>25</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>25</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>23</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>22</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>27</td>
<td>16</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 4: Results from user test

The number of right and wrong guesses of labeling emails phishing or legitimate can be seen in table 4. All participants showed at least a 20% improvement in labeling emails during the test. Further discussion and analysis of the user test results can be found in section 7.3. Questions and comments regarding the user test can also be found in section 7.3.

A log of data collected in the user test can be found in Appendix A.2 User Test.
7.3 Analysis and discussion of the results

The user test was created to see if the application would have any impact on the participants' email phishing awareness.

<table>
<thead>
<tr>
<th>Test person #</th>
<th>Wrong</th>
<th>Right</th>
<th>Legitimate</th>
<th>Phishing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>25</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>25</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>7</td>
<td>23</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>22</td>
<td>11</td>
<td>19</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>27</td>
<td>16</td>
<td>14</td>
</tr>
</tbody>
</table>

Week 2

<table>
<thead>
<tr>
<th>Test person</th>
<th>Wrong</th>
<th>Right</th>
<th>Legitimate</th>
<th>Phishing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>29</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>26</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>26</td>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>28</td>
<td>17</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>28</td>
<td>15</td>
<td>15</td>
</tr>
</tbody>
</table>

Figure 63: Screenshot of results from the user test

As seen in figure 63, the participants answer from the user test. Comparing the results from week 1 to week 2 showed a decrease in the number of wrong answers. The improvement was at least 20% for every participant. This showed that the test had a positive effect on the participant's email phishing awareness. Participant number four improved the most by going from eight wrong answers in week 1 to only two wrong answers in week 2.

The context was essential in the email evaluation. Ex: a person would know that something is wrong if they receive an email about picking up a package from the post office that they didn't order. Therefore, trying to prevent confusion, emails that were used in the user test were chosen to have a non-personal and neutral context.
Figure 64: User test results w/application for week 1 and week 2

In figure 64, the left table was from week 1, and the right table from week 2. It was interesting to see that with the application, some participants performed worse in week 2, especially when all of them had performed flawlessly in week 1. The participants answered wrong due to not trusting the application. The reason behind this was that the content of the emails seemed too legitimate or suspicious to trust the application. This showed that an email's context had an impact on how a person would react, even when using the application.

The attachment section in the application was not used in the application detection test due to it being hard to measure its effect. Therefore, it was tested in the user test instead. Three of the emails in the user test contained attachments, two legitimate attachments, and one malicious attachment. All participants managed to label these emails correctly. The participants did not want to open the attachments, especially the attachment in the phishing email. It was not clear whether email attachments had any real impact on the results due to the low number of emails containing attachments. But it seemed like the participants were more hesitant to open attachments included in suspicious-looking emails.

Questions and comments

The participants in the user test were asked questions before and after the test was conducted. See Appendix A.2 User test for the questions asked. All participants had little to no knowledge of email phishing, which was why they were chosen to take part in the test. Three of the five participants answered that they had never heard of the concept of email phishing before. Two of them responded that they did not know the possible consequences of interacting with such emails.

All five participants made comments about liking the application's layout and design. They all said that they understood how to use the application. It was simple, which made it less distracting and easy to learn.
The functionalities of the technical security measures were not that clear for all participants. Especially the authentication protocols were unfamiliar and harder to understand for some of them. This was understandable as the little explanation of the terms in the application was not enough to fully grasp the functionality of the protocols. These protocols are typically hidden from email users, meaning it would be less chance of someone knowing about them. Other technical terms like HTTPS were also unfamiliar to the participants, but they were capable of learning how to check for it in the links.

After the second round of the user test, all five participants commented that they felt they had learned something new and essential about email security and usage.

7.4 Value for email security

As presented in section 2.4, there exist many solutions to detect and prevent email phishing. Two-factor authentication, spam filters, sandboxing, URL and attachment protection, machine learning, and human awareness being some of the most popular. Each of them provides different pros and cons regarding email security. Depending on available resources and the situation, each solution has different fits.

It can be hard to measure the value this project provides for email security. But compared to the paid professional alternatives, the following can be said that this project has produced of value:

**Highlighting email shortcomings**

The project has highlighted some of the most common email security vulnerabilities that cybercriminals try to exploit when crafting phishing emails. It has shown that there are specific patterns that can be found in emails that can be used as countermeasures to eliminate a significant amount of email phishing attempts.

**Open-source project**

The application was created as an open-source project, free of charge, available to anyone that would be interested in learning about email phishing or build further on this project. Publishing channels such as YouTube and GitHub have been used to share the knowledge and source code. See Digital Appendix for the online published resources. This way, the project
would potentially directly or indirectly be making an impact on the prevention and detection of email phishing.

**Email phishing awareness**

As explained in section 2.3.9, Human awareness and training, the importance of making people aware of email phishing is an essential factor in hindering it succeeding. This project has shown that it is possible to make people increase their email phishing awareness and learn to interact with emails more securely. What measures that should be used, such as gamifying the learning experience, practicing rewards and punishments, or as this project providing a support tool, is not of discussion in this report. But as the project has shown, a support tool is a valid alternative for email phishing training and awareness.
Chapter 8

Conclusion

8.1 Summary

In this thesis, email phishing prevention was studied. This study was done with a focus on researching what phishing emails consist of and what common patterns could be found within them. The patterns found led to developing and implementing a Microsoft Office Outlook add-in application capable of detecting phishing emails. The application would inform the user to make safer email interactions, ultimately preventing email phishing attempts.

Most of the work done in this project revolved around developing the add-in application. This involved defining application requirements, designing application structure and layout, implementation, and integrating the solution in Microsoft Outlook Live. The result of this work was a client-sided add-in application that has been made open-source on GitHub.

To answer research question 2 and 3, see section 1.2, two tests were conducted. The first test being an application detection test, which tried to measure the application's capabilities of separating legitimate emails from phishing emails. The second test being a user test that attempted to measure the application's capabilities of improving the participants' email phishing awareness.

From the application detection test results, it was concluded that the application did have a 76.66% success rate of rating emails legitimate and a 92.67% success rate of rating emails phishing. From the user test results, it was concluded that the application did improve the participants' email phishing awareness by at least 20%. This measure was based on how many emails the participants labeled right or wrong as to either phishing or legitimate.

This application was an alpha version, which meant there was room for improvement. These improvements could increase the success rate of detection and make the application perform in a broader range of email contexts and scenarios.
As for the project's value for email security, it was concluded that it had shed light on commonly exploited email vulnerabilities. Some of these exploits were shown to be prevented with the right tools for email phishing detection and awareness.

8.2 Further work

In section 6.4, improvements in application functionalities were discussed. These improvements would adjust existing functionality so that the application would increase its email phishing detection rate.

Further work of the project revolves around developing and implementing new application features and functionalities that would take the application to the next level. These are as follows:

1. Develop and implement Tenserflow.js machine learning for text classification of email body messages.
2. Further integration with Office Outlook Live to access more personal user data such as email contact lists.
3. Convert application from client-side to server-side (publish the add-in on Microsoft AppSource). This involves migration to the cloud (Microsoft Azure).
4. Implement storage of user data to enable functionality such as:
   a. Collect email data used in machine learning.
   b. Application customization and user preferences.
   c. Store user information, such as the email contact list.
Bibliography


[34] TechTerms. ASCII. Available at: https://techterms.com/definition/ascii (Accessed: 04-09-2019).


[106] Forbes, Davey Winder (2020). *Google AI scans 300 Billion Gmail attachments every week*. Available at: https://www.forbes.com/sites/daveywinder/2020/02/28/google-
confirms-new-ai-tool-scans-300-billion-gmail-attachments-every-week/#4a4d0813edd1 https://techjury.net/stats-about/gmail-statistics/#gref

[107] Forcepoint (2020). What is defense in depth? Available at:


[110] Accenture (2020). Accenture Social Engineering Awareness Program. Available at:

[111] HumanFirewall (2020). How does human firewall works? Available at:

[112] Mxtoolbox (2020). IP blacklist database. Available at:

[113] Microsoft Docs (2020). Anti-spoofing protection. Available at:

[114] Microsoft Docs (2020). Anti-spam message headers. Available at:

[115] Mkyong (2009). Difference between inline and attachment in email. Available at:


Appendix A – Test Results

A.1 Application Detection Test

<table>
<thead>
<tr>
<th>Email #</th>
<th>Sender %</th>
<th>Content %</th>
<th>Links %</th>
<th>Authentication %</th>
<th>Total evaluation %</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>100</td>
<td>33</td>
<td>92</td>
<td>100</td>
<td>81</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>43</td>
<td>100</td>
<td>100</td>
<td>86</td>
</tr>
<tr>
<td>3</td>
<td>100</td>
<td>87</td>
<td>90</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>41</td>
<td>100</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>49</td>
<td>100</td>
<td>100</td>
<td>87</td>
</tr>
<tr>
<td>6</td>
<td>100</td>
<td>87</td>
<td>90</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>7</td>
<td>100</td>
<td>48</td>
<td>89</td>
<td>83</td>
<td>80</td>
</tr>
<tr>
<td>8</td>
<td>100</td>
<td>95</td>
<td>100</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>9</td>
<td>100</td>
<td>47</td>
<td>92</td>
<td>100</td>
<td>85</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>59</td>
<td>95</td>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>42</td>
<td>92</td>
<td>100</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>100</td>
<td>87</td>
<td>90</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>13</td>
<td>100</td>
<td>48</td>
<td>100</td>
<td>67</td>
<td>79</td>
</tr>
<tr>
<td>14</td>
<td>100</td>
<td>59</td>
<td>62</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>58</td>
<td>62</td>
<td>67</td>
<td>72</td>
</tr>
<tr>
<td>16</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>67</td>
<td>79</td>
</tr>
<tr>
<td>17</td>
<td>100</td>
<td>43</td>
<td>100</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>18</td>
<td>100</td>
<td>95</td>
<td>60</td>
<td>67</td>
<td>81</td>
</tr>
<tr>
<td>19</td>
<td>100</td>
<td>57</td>
<td>98</td>
<td>83</td>
<td>85</td>
</tr>
<tr>
<td>20</td>
<td>100</td>
<td>48</td>
<td>89</td>
<td>83</td>
<td>80</td>
</tr>
<tr>
<td>21</td>
<td>100</td>
<td>42</td>
<td>100</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>22</td>
<td>100</td>
<td>44</td>
<td>98</td>
<td>67</td>
<td>77</td>
</tr>
<tr>
<td>23</td>
<td>100</td>
<td>39</td>
<td>100</td>
<td>50</td>
<td>72</td>
</tr>
<tr>
<td>24</td>
<td>100</td>
<td>44</td>
<td>100</td>
<td>50</td>
<td>74</td>
</tr>
<tr>
<td>25</td>
<td>100</td>
<td>48</td>
<td>89</td>
<td>83</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>26</td>
<td>100</td>
<td>45</td>
<td>96</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>27</td>
<td>100</td>
<td>87</td>
<td>90</td>
<td>100</td>
<td>94</td>
</tr>
<tr>
<td>28</td>
<td>60</td>
<td>49</td>
<td>58</td>
<td>67</td>
<td>59</td>
</tr>
<tr>
<td>29</td>
<td>100</td>
<td>81</td>
<td>82</td>
<td>100</td>
<td>91</td>
</tr>
<tr>
<td>30</td>
<td>100</td>
<td>60</td>
<td>100</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>31</td>
<td>100</td>
<td>59</td>
<td>94</td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td>32</td>
<td>60</td>
<td>60</td>
<td>55</td>
<td>67</td>
<td>61</td>
</tr>
<tr>
<td>33</td>
<td>100</td>
<td>60</td>
<td>62</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>34</td>
<td>100</td>
<td>42</td>
<td>62</td>
<td>100</td>
<td>76</td>
</tr>
<tr>
<td>35</td>
<td>100</td>
<td>69</td>
<td>91</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>57</td>
<td>61</td>
<td>100</td>
<td>55</td>
</tr>
<tr>
<td>37</td>
<td>100</td>
<td>59</td>
<td>94</td>
<td>100</td>
<td>88</td>
</tr>
<tr>
<td>38</td>
<td>100</td>
<td>60</td>
<td>94</td>
<td>100</td>
<td>89</td>
</tr>
<tr>
<td>39</td>
<td>100</td>
<td>48</td>
<td>100</td>
<td>50</td>
<td>75</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>60</td>
<td>61</td>
<td>100</td>
<td>55</td>
</tr>
<tr>
<td>41</td>
<td>100</td>
<td>59</td>
<td>62</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
<td>48</td>
<td>69</td>
<td>50</td>
<td>42</td>
</tr>
<tr>
<td>43</td>
<td>100</td>
<td>69</td>
<td>91</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>44</td>
<td>100</td>
<td>60</td>
<td>62</td>
<td>83</td>
<td>76</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
<td>63</td>
<td>61</td>
<td>100</td>
<td>56</td>
</tr>
<tr>
<td>46</td>
<td>100</td>
<td>53</td>
<td>95</td>
<td>100</td>
<td>87</td>
</tr>
<tr>
<td>47</td>
<td>100</td>
<td>60</td>
<td>94</td>
<td>100</td>
<td>89</td>
</tr>
<tr>
<td>48</td>
<td>100</td>
<td>45</td>
<td>58</td>
<td>67</td>
<td>68</td>
</tr>
<tr>
<td>49</td>
<td>0</td>
<td>79</td>
<td>100</td>
<td>67</td>
<td>62</td>
</tr>
<tr>
<td>50</td>
<td>100</td>
<td>69</td>
<td>91</td>
<td>100</td>
<td>90</td>
</tr>
<tr>
<td>51</td>
<td>100</td>
<td>60</td>
<td>98</td>
<td>33</td>
<td>73</td>
</tr>
<tr>
<td>52</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>67</td>
<td>67</td>
</tr>
<tr>
<td>53</td>
<td>30</td>
<td>96</td>
<td>70</td>
<td>50</td>
<td>62</td>
</tr>
<tr>
<td>54</td>
<td>100</td>
<td>44</td>
<td>100</td>
<td>50</td>
<td>74</td>
</tr>
<tr>
<td>55</td>
<td>100</td>
<td>48</td>
<td>92</td>
<td>50</td>
<td>73</td>
</tr>
<tr>
<td>56</td>
<td>60</td>
<td>42</td>
<td>52</td>
<td>67</td>
<td>55</td>
</tr>
<tr>
<td>57</td>
<td>0</td>
<td>93</td>
<td>92</td>
<td>67</td>
<td>63</td>
</tr>
<tr>
<td>58</td>
<td>100</td>
<td>37</td>
<td>47</td>
<td>100</td>
<td>71</td>
</tr>
<tr>
<td>59</td>
<td>100</td>
<td>43</td>
<td>100</td>
<td>100</td>
<td>87</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
<td>58</td>
<td>77</td>
<td>50</td>
<td>46</td>
</tr>
<tr>
<td>61</td>
<td>100</td>
<td>48</td>
<td>100</td>
<td>100</td>
<td>87</td>
</tr>
<tr>
<td>62</td>
<td>100</td>
<td>50</td>
<td>100</td>
<td>67</td>
<td>79</td>
</tr>
<tr>
<td>63</td>
<td>50</td>
<td>49</td>
<td>31</td>
<td>100</td>
<td>68</td>
</tr>
<tr>
<td>64</td>
<td>100</td>
<td>47</td>
<td>85</td>
<td>83</td>
<td>79</td>
</tr>
<tr>
<td>65</td>
<td>30</td>
<td>82</td>
<td>56</td>
<td>67</td>
<td>59</td>
</tr>
<tr>
<td>66</td>
<td>90</td>
<td>92</td>
<td>80</td>
<td>100</td>
<td>91</td>
</tr>
<tr>
<td>67</td>
<td>30</td>
<td>58</td>
<td>81</td>
<td>83</td>
<td>63</td>
</tr>
<tr>
<td>68</td>
<td>0</td>
<td>89</td>
<td>63</td>
<td>100</td>
<td>63</td>
</tr>
<tr>
<td>Email #</td>
<td>Sender %</td>
<td>Content %</td>
<td>Links %</td>
<td>Authentication %</td>
<td>Total evaluation %</td>
</tr>
<tr>
<td>---------</td>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>1</td>
<td>80</td>
<td>43</td>
<td>23</td>
<td>0</td>
<td>37</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>39</td>
<td>23</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>40</td>
<td>15</td>
<td>33</td>
<td>22</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>46</td>
<td>23</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>58</td>
<td>17</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>46</td>
<td>23</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>46</td>
<td>23</td>
<td>0</td>
<td>17</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td>47</td>
<td>23</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>47</td>
<td>23</td>
<td>33</td>
<td>26</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>47</td>
<td>23</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>11</td>
<td>100</td>
<td>49</td>
<td>100</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>58</td>
<td>17</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>46</td>
<td>19</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>49</td>
<td>23</td>
<td>50</td>
<td>31</td>
</tr>
<tr>
<td>15</td>
<td>100</td>
<td>49</td>
<td>100</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>42</td>
<td>35</td>
<td>0</td>
<td>19</td>
</tr>
<tr>
<td>17</td>
<td>80</td>
<td>43</td>
<td>12</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>56</td>
<td>18</td>
<td>50</td>
<td>31</td>
</tr>
<tr>
<td>19</td>
<td>80</td>
<td>42</td>
<td>18</td>
<td>0</td>
<td>35</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>39</td>
<td>23</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>21</td>
<td>80</td>
<td>43</td>
<td>12</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>40</td>
<td>17</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
<td>48</td>
<td>23</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
<td>44</td>
<td>18</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>54</td>
<td>18</td>
<td>50</td>
<td>31</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>50</td>
<td>23</td>
<td>33</td>
<td>27</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>37</td>
<td>23</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>44</td>
<td>23</td>
<td>50</td>
<td>29</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>45</td>
<td>23</td>
<td>50</td>
<td>30</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>65</td>
<td>23</td>
<td>0</td>
<td>23</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>39</td>
<td>17</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
<td>60</td>
<td>23</td>
<td>0</td>
<td>21</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>37</td>
<td>20</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
<td>34</td>
<td>100</td>
<td>0</td>
<td>34</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>58</td>
<td>19</td>
<td>50</td>
<td>32</td>
</tr>
<tr>
<td>36</td>
<td>100</td>
<td>49</td>
<td>100</td>
<td>83</td>
<td>83</td>
</tr>
<tr>
<td>37</td>
<td>0</td>
<td>50</td>
<td>38</td>
<td>83</td>
<td>43</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
<td>47</td>
<td>23</td>
<td>33</td>
<td>26</td>
</tr>
<tr>
<td>39</td>
<td>0</td>
<td>29</td>
<td>100</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>44</td>
<td>18</td>
<td>50</td>
<td>28</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
<td>42</td>
<td>19</td>
<td>0</td>
<td>15</td>
</tr>
</tbody>
</table>
A.2 User Test

Questions:
Do you know what email phishing is?
Do you know what it can do?
Did your email phishing awareness become better after using this application?
Comments on what could have been done different; application improvements, tests,

Green colour means right answer, red colour means wrong answer.

**Week 1**

Phase 1 (without application):

TP = Test person (with number)
P = phishing email
L = legitimate email

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>L</td>
<td>P</td>
<td>P</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>8</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>9</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>P</td>
<td>P</td>
<td>L</td>
<td>10</td>
<td>L</td>
</tr>
</tbody>
</table>

Phase 2 (with application):

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>1</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>2</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>3</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>6</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>8</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>9</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>10</td>
<td>P</td>
</tr>
</tbody>
</table>
Phase 3 (without application):

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>3</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>P</td>
<td>P</td>
<td>L</td>
<td>4</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>8</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>L</td>
<td>9</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>10</td>
<td>P</td>
</tr>
</tbody>
</table>

Week 2

Phase 1 (without application):

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>L</td>
<td>1</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>2</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>3</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>6</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>L</td>
<td>8</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>9</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>10</td>
<td>P</td>
</tr>
</tbody>
</table>

Phase 2 (with application):

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>L</td>
<td>P</td>
<td>3</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>4</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>P</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>8</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>9</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>10</td>
<td>L</td>
</tr>
</tbody>
</table>
### Phase 3 (without application):

<table>
<thead>
<tr>
<th>TP.1</th>
<th>TP.2</th>
<th>TP.3</th>
<th>TP.4</th>
<th>TP.5</th>
<th>Email #</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>1</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>2</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>3</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>4</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>5</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>6</td>
<td>P</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>7</td>
<td>P</td>
</tr>
<tr>
<td>L</td>
<td>P</td>
<td>L</td>
<td>L</td>
<td>P</td>
<td>8</td>
<td>L</td>
</tr>
<tr>
<td>L</td>
<td>L</td>
<td>P</td>
<td>L</td>
<td>L</td>
<td>9</td>
<td>L</td>
</tr>
<tr>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>P</td>
<td>10</td>
<td>P</td>
</tr>
</tbody>
</table>
Appendix B – Digital Appendix

B.1 GitHub Repository

All files used for the development of the application can be seen at GitHub:

https://github.com/haakbe93/email-phishing-detection-add-in

B.2 Demo of Application

Demo video of the application functionalities can be seen online at YouTube:

https://youtu.be/T9jzo4Z1yLk