PHISH ALERT

Katie Doyle

C00240662

Christopher Staff

Institute of Technology, Carlow

26 November 2021

Abstract

This paper researches in detail several technology methods to be used in a system to detect the presence of a Phishing Uniform Resource Locator, or URL*, contained in an Electronic Mail message. An implementation of a system with similar characteristics has been successfully done in the past, with that system producing some limitations or gaps in a production environment, allowing me to respond to these limitations with my system. This paper focuses on researching innovative methodology's alongside mainstream, while keeping security at the forefront of the scope. Extensive research subsequently lead to the discovery of several limitations of implementing a Machine Learning system, confirming the dominant underlying implementation technology for this set of deliverables to be Conventional Programming. The Programming language selected is Java, which will aid the implementation with the use of built-in libraries, allowing for mail manipulation. The Email Client proven to be transparent in relation to User Interface Add-Ons or Extensions is Gmail. The chosen dataset for this implementation is an online repository of Phishing Emails, called Phish Tank.

Table of Contents

Table of Figures

Introduction

As the use of Electronic Mail (Email*) has remained as popular as ever, with more than half of the world's population actively using this electronic way of communication (Pitch Funnel, 2021), it is no surprise that cybercriminals have adopted Electronic Mail as a way of delivering sophisticated attacks to victims, known as Phishing attacks. In Ireland alone, Phishing is described as being the highest Cybersecurity threat at 71%, higher than both Human Error at 56%, and Ransomware at 47% (The EMEA, 2021). Many of the Phishing attacks delivered via Electronic Mail are proven successful as they depict to the recipient that immediate action is required or else something bad will occur, hoping that a swift action will take place out of fear, rather than looking at the email contents closely or having time to think of an appropriate response. The contents of these emails vary, but many successful attacks may mention topics such as unpaid bills, account cancellation, password resets and many more (Tech Radar, 2020).

Phishing attacks have become so popular through Email that a specialized area of the Cybersecurity industry has been dedicated to Email Security. Email Security is a term used for describing different procedures and techniques for protecting email accounts, content, and communication against unauthorized access, loss or compromise (Proofpoint, 2021).

Some Phishing emails may contain hyperlinks or URLs to direct a recipient to an illegitimate website for ill-intent. Below, it can be seen how the number of Phishing websites has overtaken the number of malware sites over the years.

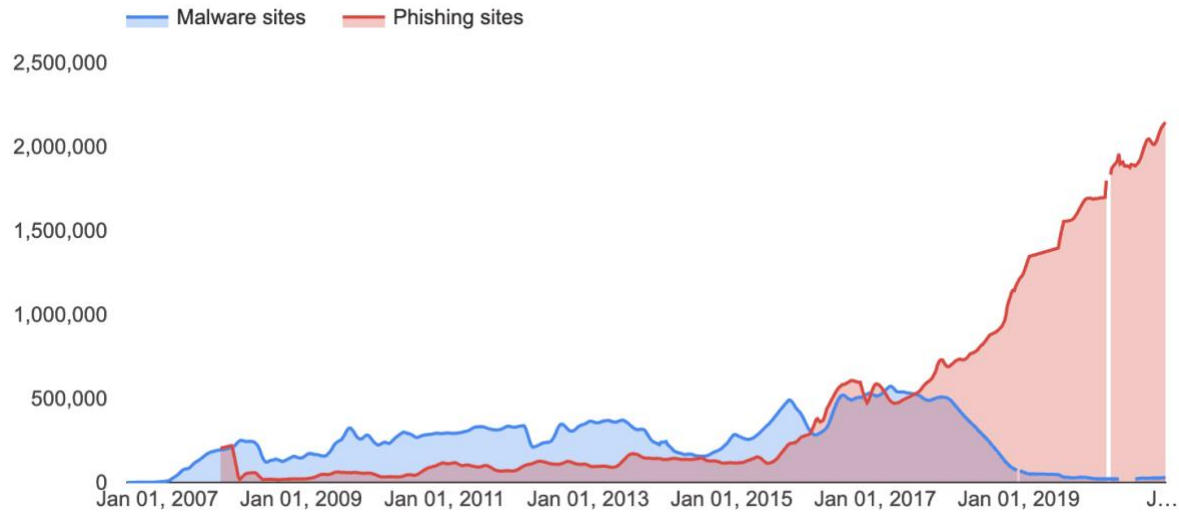

Figure 1: The number of websites deemed unsafe between January 2016 and January 2021

(Google Safe Browsing, 2021)

These Phishing attacks that are taking place today are sophisticated, targeted and increasingly difficult to spot (Meta Compliance, 2021). To an untrained eye, it is no surprise that Phishing delivered through Electronic Mail hold a high success rate. One simple way to detect Phishing is through looking at the information readily available alongside a received email – email headers.

The image below (Figure 2) depicts how email header analysis can indicate an email received is Phishing by checking to see if the ‘Header From’ field of an email which is actively visible to the recipient matches the ‘Mail From’ or ‘Sender From’ address which can be viewed by the recipient, but the recipient must seek message source information. In Figure 2, it can be seen that the ‘Header From’ and ‘Mail From’ addresses do not match, and taking in to account the subject line included, I would classify this email as Phishing before even looking at the message body. Figure 3 is another example of the conditions mentioned above. Email header analysis is discussed below in greater detail.

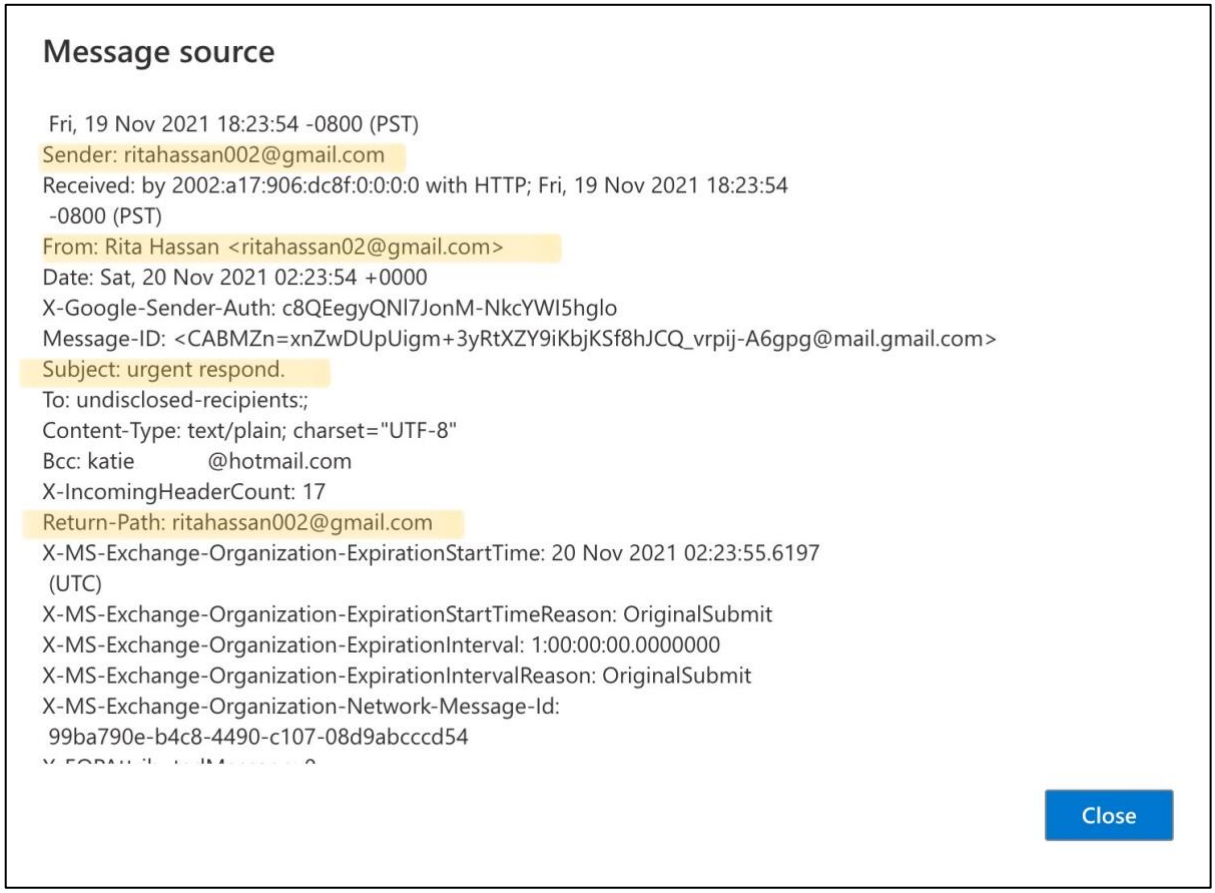

Figure 2: Email Header Analysis Example 1

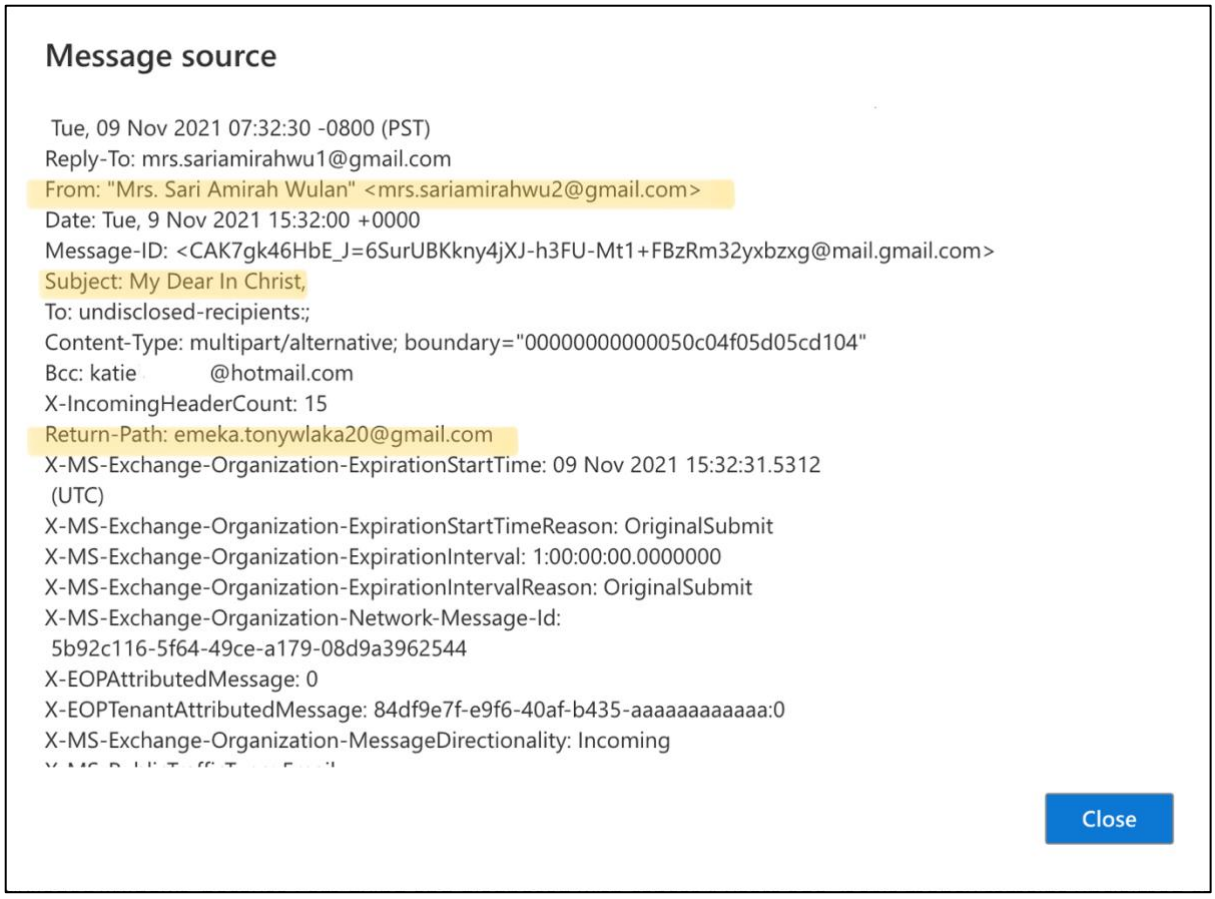

Figure 3: Email Header Analysis Example 2

Currently, there are several Phishing detection techniques available to compact these attacks. These include black-list, white-list, content-based, URL-based, visual-similarity and machine learning techniques (Arxiv, 2021). All of these techniques provide different accuracy levels in terms of Phishing detection, leaving an entry barrier for future detection systems of a higher accuracy.

As such, the need for a high accuracy application to detect if Electronic Mail messages are likely genuine or malicious has never been greater. Uniform Resource Locators, or URL's, are clickable links which often in appear in Emails, directing an unsuspecting recipient to a malicious or fraudulent website (Savvy Security, 2021). These URL's will need to be the focus point of a successful implementation. An implementation of these deliverables has been conducted before, basing the underlying technology on an ensemble of techniques, including the use of Machine Learning algorithms such as Bagging, AdaBoost, Random Forest and Gradient boosting. This approach produced high variations in accuracy ratings between each algorithm used, although all algorithms were trained from the same dataset. The highest accuracy documented is 96.15% and lowest being 54.31%. The dataset used contained 11,000 malicious URL's (National College of Ireland, 2020). One of the limitations to this approach is the dataset used. It contained only 11,000 URL's, of which all were Phishing. This means that outside of this dataset, the system would not withstand much Phishing detection. I intend to respond to this limitation with a system that will be able to withstand all types of Phishing, with an algorithm designed on common characteristics all Phishing emails contain, to ensure accurate detection in an unpredictable, production environment. My research work is based on selecting several technologies, after extensive

research, to use in order to implement a system which will address the limitation in the previous system implementation, mentioned above. This paper argues that adopting the use of a Conventionally Programmed system to accurately identify Phishing Emails is more beneficial in terms of accuracy and security than a Machine Learning system.

This paper first discusses several examples of technology research, then goes on to conclude on a set of technologies to be used in the implementation of a Phishing detection system.

Methods

In order to respond to the limitation mentioned above relating to a previous implementation of similar deliverables, I began by researching with the help of the Google Scholar[1] search engine. This gave me confirmation that there has been many variations of this project implemented successfully before, and gave me insight into the different types of technologies to research further. Below is an exhaustive list of the different technologies I researched in detail to narrow down my endless choices.

Machine Learning

Due to the nature of the deliverables of this project, I had a variety of different technologies to choose from for implementation. When I began the research phase, I noticed that this project had been executed successfully in the past, with the use of Machine Learning. This lead me to begin researching Machine Learning to see if I would use this type of system for my project implementation.

Machine Learning can be defined as an application of Artificial Intelligence (AI*) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Expert AI, 2020). Artificial Intelligence is an umbrella term that refers to systems that mimic the problem-solving and decision-making capabilities of the human mind (IBM, 2020). We all interact with Machine Learning daily when undertaking tasks such as Banking, Online Shopping, and Social Media use (Oracle, 2021). A Machine Learning approach can be taken to implement my deliverables using an ensemble of Machine Learning algorithms that analyse different elements of messages. Figure 4 below depicts the process of email phishing detection technology using Machine Learning. As Figure 4 depicts, using two algorithms provides a solution that can automatically detect and block Phishing (Info Security, 2020).

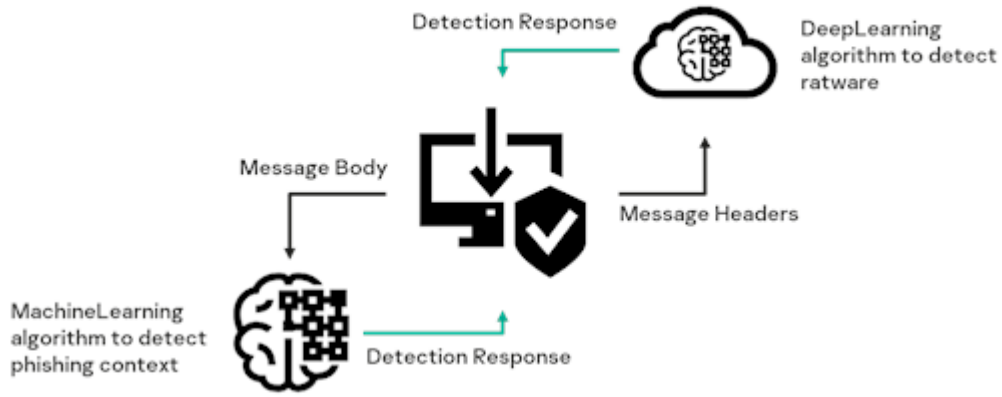

Figure 4: How it Works: Machine Learning Against Email Phishing

(Info Security, 2020)

Conventional Programming

[1] Google Scholar: https://scholar.google.com/

Throughout my research phase I have been unsuccessful in locating similar projects executed using Conventional Programming. This lead me to begin looking at Conventional Programming as a way of implementing this project as it doesn't appear to be as popular as a Machine Learning approach.

Conventional or Traditional Programming can be defined as a manual process where a person creates a program and without anyone Programming the logic like Machine Learning, the person creating the program must manually code or formulate rules. This is as opposed to Machine Learning, where the algorithm automatically formulates the rules from the dataset supplied, as mentioned above (Kaggle, 2020). Some Conventional Programming languages offer mail based libraries in order to interact with electronic messages using basic CRUD* (Create, Read, Update, Delete) operations.

Programming Languages

As Conventional Programming is now being considered for implementation, I began to research various high-level Programming languages to assist with this implementation.

My options were Java or Python; which are undoubtedly the two most popular Programming languages (Snap Logic, 2021). Java can be defined as a general-purpose, class-based, object-oriented Programming language. As Java is fast, secure and reliable it is no surprise it is commonly used for developing Java applications in laptops, data centres, gaming consoles and mobile phones (Guru 99, 2021). Java is known to have less identified vulnerabilities than other popular languages, including Python, making it a secure programming language (Info World, 2020). Python on the other-hand, can be defined as an interpreted, object-oriented Programming language with dynamic semantics. Python is attractive for rapid application development due to its high level built-in data structures (Python, 2021). I limited my choices to these two programming languages due to their powerful cross-platform support and extensive libraries.

Hardware Requirements and Operating Systems

As part of my research phase, I had to see early-on if any hardware requirements or Operating Systems were needed to undertake the project I was assigned.

For Java or Python Software Development, personal preference is the biggest factor for selecting an Operating System, one particular Operating System is not compulsory. That being said, the most commonly preferred Operating Systems for Java and Python development is MacOS, Linux and Windows (Career Karma, 2020).

Microsoft Windows[2], commonly referred to as Windows and Windows OS, is a computer Operating System (OS) developed by Microsoft to run on personal computers (PCs) (Britannica, 2021).

MacOS[3] is an Operating System developed by Apple, to run on the company's Macintosh line of personal computers (PCs) (Britannica, 2021).

[2] Microsoft Windows: https://www.microsoft.com/en-ie/windows
[3] MacOS: https://www.apple.com/ie/macos/monterey/

Linux[4] is an Operating System, which is open source and freely available which can be installed on a range of devices from desktops, to mobile phones, to supercomputers (SUSE, 2021).

After extensive research to see if there are any Hardware Requirements for Java or Python Development, I found that both languages require the following:

- Java
    1. RAM: At least 128 MB
    2. Disk Space: 124 MB for JRE, 2 MB for Java Update
    3. Processor: Minimum Pentium 2 266 MHz processor (Refresh Java, 2021)
- Python
    1. RAM: 4GB
    2. Disk Space: 5GB
    3. Processor: x86 64-bit CPU* (Intel / AMD* architecture)

These findings indicate that most computers purchased in recent years will satisfy these requirements easily (Enthought, 2020).

Email Clients

Due to the nature of my project specification, one of my deliverables is to successfully create a browser extension or add-on which will house my system for detecting potentially malicious emails. To do this, I researched both the Email Clients Gmail and Outlook to see if there was technological reason as to why one could not be used over the other.

Aside from being competitors, Gmail and Outlook are very different. Gmail is a web application developed by Google[5], with desktop access being offered to clients through the Gmail API*. Outlook is an email application for various Operating Systems, with Outlook on the web being offered as part of Microsoft's consumer-level email service[6]. This means the Outlook application is free but the web counter-part being part of the Microsoft 365 subscription. In terms of Security, Gmail offers 2-step verification and emails sent are automatically encrypted using Transport Layer Security (TLS*) and Google will alert a user if a virus is detected in an attachment or if an email is suspicious. Outlook scans attachments for viruses and malware using advanced detection techniques and will also check to see if links contained in emails are related to Phishing scams or malware. The big difference here is Gmail offers all these advanced security features for free. One aspect I needed to research in depth was how these Email Clients interact with extensions or add-ons. Outlook extensions are limited to the desktop version of Outlook, so they are not available on the web. Gmail extensions are installed through the Chrome browser, which means not only are extensions available on the web interface, they also sync to other computers you are signed into Chrome on (Spike, 2021).

[4] Linux OS: https://www.linux.org/

[5] Gmail, the web application developed by Google: https://www.google.com/account/about/?hl=en-US

[6] Microsoft Consumer-Level Service: https://www.microsoft.com/en-ie/microsoft-365/buy/compare-all-microsoft-365-products

Java Mail API

After researching potential programming languages, I then began to look at the libraries available as part of these languages, to see if any would allow me to access basic CRUD (Create, Read, Update, and Delete) operations in relation to Electronic Messages.

The Java Mail API[7] is used to read, write and exchange emails to or from a Java Program. The API comes with different interfaces, classes, and methods to make email functionalities possible. Although the protocols Simple Mail Transfer Protocol (SMTP*), Post Office Protocol (POP*), Internet Message Access Protocol (IMAP*), and Multiple Internet Mail Extension (MIME*) are available in the Java Mail API, the one needed to satisfy my project specifications is POP. There is only one minor disadvantage which arises when using this API, that being memory efficiency. Not very significant but still worthy to note that the code might take a little longer to compile using the API (Go Coding, 2020).

Python Libraries for Sending and Receiving Mail

After researching potential programming languages, I then began to look at the libraries available as part of these languages, to see if any would allow me to access basic CRUD (Create, Read, Update, and Delete) operations in relation to Electronic Messages.

Python comes with built-in libraries for the protocols Internet Message Access Protocol (IMAP), Post Office Protocol (POP), and Simple Mail Transfer Protocol (SMTP). This will allow a Python application to send, read and manage email. As mentioned above, the only protocol I would need to focus on for my deliverables is Post Office Protocol (POP), which will allows mail to be received (Humberto Rocha, 2018).

Java Naming Directory Interface (JNDI)

After researching potential programming languages, I then began to look at the libraries available as part of these languages.

This is a Domain Name System (DNS*) provider which allows Java Naming Directory Interface[8] (JNDI*) applications to access information stored in the Internet Domain Name System. This can be useful for a number of reasons, the main one being its ability to check the age of a domain. Checking the age of a domain is useful for detecting potential Phishing attempts as the younger a domain is, the more likely its Phishing related (Oracle, 2021).

DNSPython Module

After researching potential programming languages, I then began to look at the libraries available as part of these languages. This is a Domain Name System (DNS) provider module within Python which allows DNSPython[9] applications to access information stored in the Internet Domain Name System. This can be useful for a number of reasons, the main one being its ability to check the age of a domain. Checking the age of a domain is useful for

[7] Java Mail API: https://javaee.github.io/javamail/
[8] Java Naming Directory Interface: https://docs.oracle.com/javase/jndi/tutorial/getStarted/overview/index.html
[9] DNSPython: https://www.dnspython.org/

detecting potential Phishing attempts as the younger a domain is, the more likely its Phishing related (Tutorials Point, 2021).

Phish Tank

No matter which technologies I choose to use, I will need a repository of Phishing emails to use as my dataset.

Phish Tank[10] is a free, public website where anyone submit, verify, track and share Phishing data. Aside from being a web application, Phish Tank also offers an API to be used for application developers (Phish Tank, 2021). The Phish Tank API is free to use once the developer is registered to the website.

Intelligent phishing URL detection

To keep a variety of techniques open as options for this project, I came across an extract from a book called *A Machine-Learning Approach to Phishing Detection and Defense, 2015*. This extract from the book published online goes into detail about methods for executing similar deliverables, and methods to which you can identify Phishing easily. Even if I don't implement a Machine Learning system like the one mentioned in the book, I could use some of the methods contained in the book to identify Phishing emails to improve system accuracy.

Safe Browsing Application Programming Interfaces (APIs)

As part of a safe browsing initiative, Google have developed a Safe Browsing Lookup API[11] which allows client applications to send a HTTP* POST request to the Safe Browsing servers to check to see if URLs are included on any of the Safe Browsing lists. These lists contain a repository of unsafe web resources. If a URL is present on one or more of these lists, the information is returned accordingly. An example of one of the Safe Browsing lists is Malware/Windows/URL (Google, 2021).

Outlook Task Pane Add-In

To satisfy one of my deliverables, which is to develop an add-on or extension for an email client, I researched in detail how this process occurs with Outlook. Add-Ins in Outlook are programs or utilities that help the end user automate tasks when viewing or creating messages. A task pane Add-In can be created using Yeoman generator for Office Add-Ins[12] or Visual Studio Code[13]. These Add-Ins are featured on the control bar in Outlook, contained within the 'Show Task Pane' option. The image below (Figure 5) shows in detail where the Add-Ins are installed in the Outlook User Interface (Microsoft, 2021).

[10] Phish Tank: https://phishtank.org/
[11] Google Safe Browsing Lookup API: https://developers.google.com/safe-browsing/v4/lookup-api?csw=1#Overview
[12] Yeoman generator for Office Add-Ins: https://github.com/OfficeDev/generator-office
[13] Visual Studio Code: https://code.visualstudio.com/

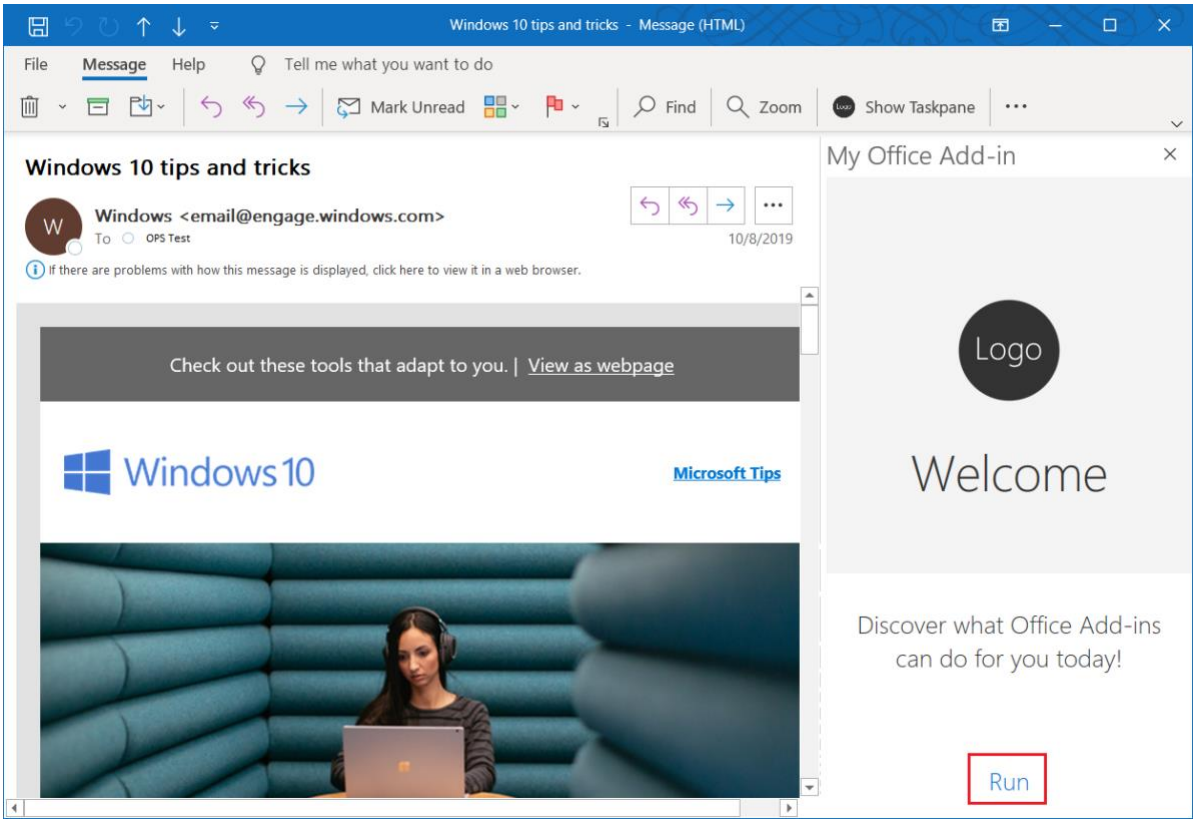

Figure 5: Build your first Outlook add-in

(Microsoft, 2021)

Google Workspace Add-Ons

To satisfy one of my deliverables, which is to develop an add-on or extension for an email client, I researched in detail how this process occurs with Gmail.

Google Workspace offers a way of automating tasks to improve end user usability with Add-Ons. When a user reads or composes a message, an Add-On can present an interactive, customized User Interface (UI*) which allows the user to act on the message in various ways. Google Workspace Add-Ons are built using Apps Script[14], and their interfaces defined using Apps Script Card Service[15]. The image below (Figure 6) shows in detail where the Add-Ins are installed in the Gmail User Interface (Google, 2021).

[14] Google Apps Script: https://developers.google.com/apps-script
[15] Google Apps Script Card Service: https://developers.google.com/apps-script/reference/card-service

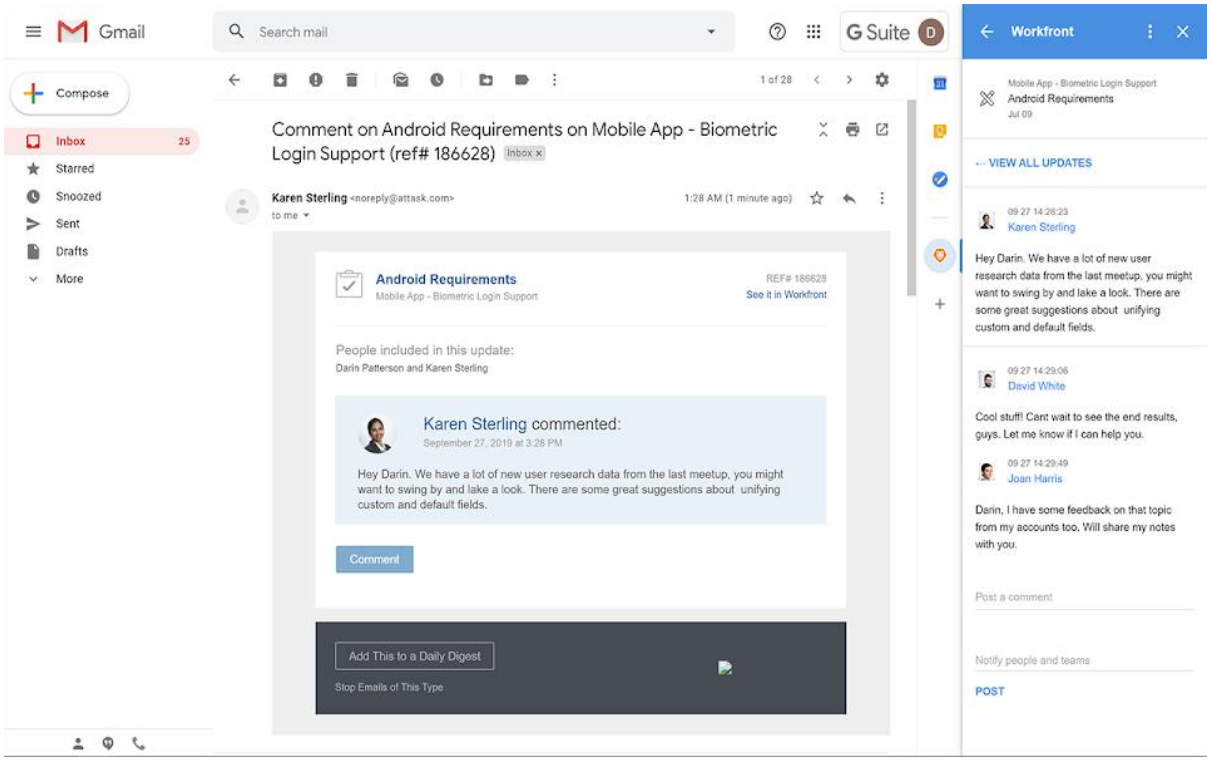

Figure 6: Google Workspace Add-ons

(Google, 2021)

Stanford Named Entity Recognizer

In order to potentially help with identifying the legitimacy of an organization named in an Electronic Message, I researched the Stanford Named Entity Recognizer (NER*) software[16].

Stanford Named Entity Recognizer is a java implementation of a Named Entity Recognizer. Named Entity Recognition labels sequences of words in a text which are the names of things, such as person and company names or gene and protein names. The software comes with feature extractors for Named Entity Recognition, and many options for defining feature extractors. Included with the software download, are good Named Entity Recognizers for English, especially the three class (Person, Organization, Location) (The Stanford NLP Group, 2021).

Email Header Analysis

In order to Conventionally Program an application to detect a potentially Phishing email, I needed to research some common detection methods.

One of the common Phishing detection techniques is Email Header Analysis. An email header contains information about the email, its sender and route to the recipient, date of creation, ID* number, encoding type, mailing address and IP* address. All of this is visible in an Electronic Message's properties. Analysis of headers helps to identify suspicious senders (Info Security, 2020). Email header analysis can be used as one indicator in detecting

[16] Stanford Named Entity Recognizer (NER) software: https://nlp.stanford.edu/software/CRF-NER.html#:~:text=Stanford%20NER%20is%20a%20Java,or%20gene%20and%20protein%20names.

whether or not an email is Phishing. An example of an email header can be viewed in Figure 7.

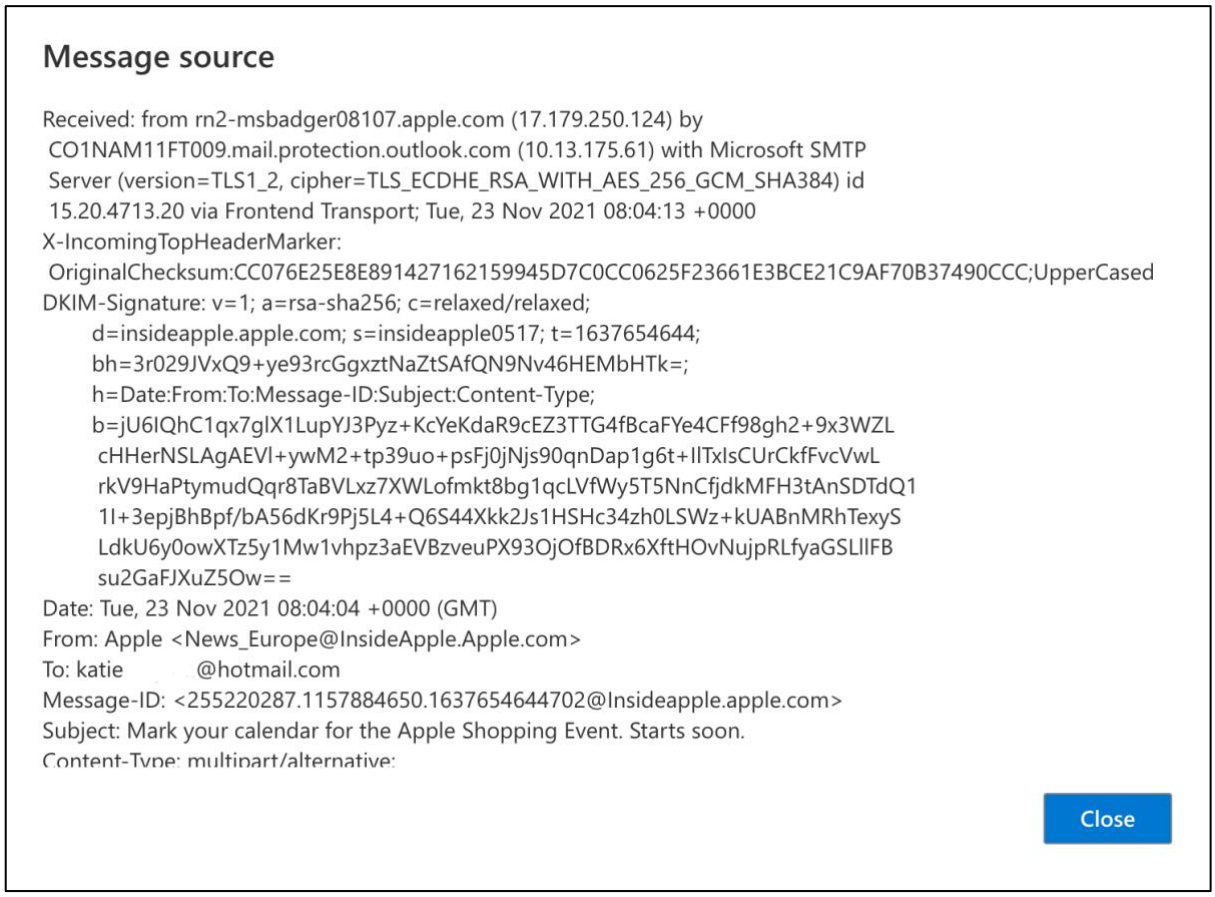

Figure 7: Email Header Analysis Example 3

Evaluation Approach

Due to the nature of the proposed system, an evaluation approach is necessary to determine the accuracy of the system.

One method for evaluating this system is configuring a Simple Mail Transfer Protocol (SMTP) server, which will allow for customizable addresses for the outgoing mail (SendGrid, 2019). The need for customizable addresses links back to a choosing a dataset which contains both Phishing and Non Phishing emails, including their header information, to revert them back to incoming mail and receive them within the Gmail Application. As Phish Alert would begin to receive the dataset emails in a typical way, the level of accuracy of the system could be deduced. This accuracy reading would then determine the need to tweak the system in any way.

Go Phish

As an alternative option to the evaluation approach mentioned above, I decided to research a number of different Phishing frameworks to potentially use to test the system.

Go Phish is an open-source Phishing framework that makes it easy to test an organization's exposure to Phishing (GoPhish, 2022). As Go Phish requires SMTP relay details to send

from, the customized SMTP server setup mention in the evaluation approach above could be used.

Google Custom Search API

In order to potentially help identify illegitimate URLs, I researched various APIs, including the Google Custom Search API.

The Google Custom Search API is a RESTful API that allows developed applications to get and show search results from Google programmatically (Expertrec Blog, 2022). This could be utilised to determine the number of search results returned (if any) for a URL found in the body of an email.

DNSTwister API

In order to potentially help identify illegitimate URLs, I researched various APIs, including the DNSTwister API.

The DNSTwister API is a public API that can be used to perform DNS related queries, such as a WHOIS lookup, which would assist in getting a domains registration date (DNSTwister, 2022).

Virus Total API

In order to potentially help identify malicious attachments, I researched various APIs, including the VirusTotal API.

The VirusTotal API allows an application to programmatically interact with VirusTotal, which inspects attachments with over 70 antivirus scanners to determine the legitimacy of a given attachment (VirusTotal, 2022). Although out of scope, analysing an email attachment can further protect the mail recipient.

X-Google-Original-From

Looking at the visible ‘FROM’ field of an email may not display the true origin of the email, it can be an alias or a generated value in the header (e.g. by a program) attached to the email. If someone is sending an email from a Gmail SMTP configuration and the alias is unverified, it will add an ‘X-Google-Original-From’ header, where the ‘FROM’ address displayed to the recipient will be the address the email is actually being sent from, not the alias (GoldFynch, 2021).

DMARC, DKIM & SPF

DMARC, DKIM and SPF are a set of email authentication methods to prove to receiving mail servers that the senders are authorized to send email from a particular domain and, are a way of verifying your email sending server is sending emails through your domain. This is a beneficial way of preventing someone else sending emails on your behalf of you by using your domain address (Net Core, 2021).

Typo-squatting

Typo-squatting is a type of social engineering attack which targets everyday users who incorrectly type a URL into their web browser. This type of attack typically involves users being tricked into visiting malicious sites with URLs that are common misspellings of legitimate websites. The illegitimate websites may then get users to enter sensitive information. The 'typo' in typo-squatting refers to common small mistakes people make when typing on a keyboard. An example of this attack would be CSOnline.com vs CSOOnline.com (Kaspersky, 2022).

Top Level Domain (TLD)

Domains are some of the most abused tools a bad actor can use in order to manipulate victims and execute successful Phishing attacks. Although legacy or generic TLDs (such as .com) are common for targeting enterprises, country code TLDs used in Phishing attacks are beginning to gain popularity. Some of these country code TLDs are .ml, .ga, .cf and .gq (PhishLabs, 2021).

Summary

To successfully implement my project specifications, there is many technologies to be considered, compared and contrasted in order to select the appropriate segments which will come together to produce my desired output, a browser extension which has the capability to inform mail recipients of potential malicious URL's contained in an email.

First and foremost, the biggest decision that has to be decided upon is whether to go down the Machine Learning or Conventional Programming route. This decision will subsequently impact and influence the remainder of the decisions needed in regards to underlying technologies. Machine Learning automates the development process of an application by accepting an extensive set of data and deriving an algorithm based on that dataset.

Conventional Programming is a very manual process where a person must create and formulate rules, therefore there is no automatic programming of logic (Kaggle, 2020). If a Conventional Programming implementation is selected then a Programming language must also be selected. Two programming languages known for their extensive libraries and cross-platform support are Java and Python (Imaginary Cloud, 2021).

In terms of my project deliverables, both these languages have the ability to manage electronic mail within a developed application quite well. This is with thanks to built-in libraries and API's. Both languages also support Domain Name System (DNS) providers which will aid the internal workings of a created algorithm to detect Phishing URL's within an email.

To assist the decision of selecting a Programming language to use with Conventional Programming, research had to be done to see if any Operating System or Hardware Requirements were needed. In regards to both Java and Python development, selecting an Operating System is done with respect to preference, not requirement. In terms of Hardware Requirements, the necessary requirements are very likely to be present in newer computers as standard features.

Comparing Gmail and Outlook to see how both of these Email Clients interact with browser extensions was an important part of the research phase as my project deliverables heavily rely on this outcome. Outlook does not permit extensions on their web interface as opposed to Gmail which does (Spike, 2021).

No matter which underlying technologies I choose to use, a dataset of Phishing emails is needed to either deduce an algorithm in the case of Machine Learning, or test the accuracy of a Conventionally Programmed algorithm. Phish Tank, an online repository of Phishing emails, can be used as this dataset. Phish Tank offers a convenient API for developers to access the Phishing metadata stored in the repository (Phish Tank, 2021).

An extract from a book called *A Machine-Learning Approach to Phishing Detection and Defense,* provided an insight into using different methods for Phishing detection, whether or not a Machine Learning or Conventionally Programmed system is implemented. This will assist in the various indicators needed for successfully detecting Phishing. This is spoken about in detail below in the Appendix section.

One specific and important Phishing detection method is known as Email Header Analysis. An email header contains a lot of important information about an email, specifically about its sender and route taken to the recipient. Email header analysis can be used as one indicator of determining an emails authenticity.

All of these areas of research began to shed a light on what underlying technologies could successfully be used in order to obtain my desired end-goal.

Conclusion

After extensive research and information gathering of all the technologies mentioned above, I have successfully decided on a number of options based on the information contained in this report. My decision is based upon technological reasoning, potential security risks and proficiency.

I have decided to implement the deliverables of this project using Conventional Programming as opposed to Machine Learning. One of the big downfalls to Machine Learning is the limitations that go alongside it. Some big Machine Learning limitations are as follows:

- Data
- Security Risks
- Training Time

Limitations concerning data and accuracy were most concerning to me due to my project specifications. This meant that if I did not include all possible scenarios of Phishing URL's / Emails in my dataset, which is impossible due to the everyday advancements of these attacks, then I could only expect poor results. The end goal of my project is to be able to inform an end-user of a potential malicious email, as an untrained eye is more partial to succumb to a Phishing email, therefore a flaw as big as this leaves a large gap in a system developed using Machine Learning techniques. In terms of Security, a simple reconnaissance attempt undertaken by a bad actor on a system developed like this would enable that bad actor to

simply create a Phishing email which is an exception to the underlying algorithm and it would leave the mail recipient extremely vulnerable if this system is the only protection in place to alert Phishing attempts.

This brings me to the next concerning limitation of Machine Learning; Security Risks. The first security risk associated with Machine Learning is known as Fooling the Systems. This, being one of the most common attacks on Machine Learning systems is to fool these systems into making false predictions by giving malicious inputs. These are known as optical illusions for machines, which feed the machines with input that does not exist and force then to make a decision. This attack targets machine learning models. The next security risk is Data Poisoning. As the name suggests, attackers try target the data used by Machine Learning systems. This is down to the fact Machine Learning systems depend on data for the purposes of learning. In a Data Poisoning attack, the bad attacker will manipulate, corrupt and poison data in a way that will bring the system down to a state of weakness or submission. The final security risk I have looked at in regards to Machine Learning is a Transfer Learning Attack. This occurs due to the fact most Machine learning systems leverage an already-trained model. These generic models are then tweaked to fulfil the required purposes by providing specialized training. If a popular Machine Learning model is chosen to begin with, this can be deadly, leading to suspicious and unanticipated behaviours.

The final limitation that is of a concern for my project is the training time associated with teaching and training a Machine Learning system. The more accurate and reliable a Machine Learning system is required to be, the more time that has to be put into training and teaching the machine. Even after the training stage, if the accuracy or reliability score is below the expectations, corrections would need to be made which requires more time.

I reached the decision to not use Machine Learning as a means of implementation after learning about the various limitations associated with Machine Learning systems. The main limitation which posed the most concern was the Security Risks. Under no circumstance was I willing to compromise on the Security of my proposed system. A known vulnerability in a system used as an intermediary between an untrained eye and a Phishing attempt is reckless. The other deciding factor to rule out this technology was the limitation around data. As the more a Machine Learning system is fed (data) and taught, the more accurate the results produced will be, the bigger this limitation becomes in terms of my project specifications. It is impossible for me to include every and all Phishing email behaviours in a dataset to feed to a Machine Learning system, meaning the level of accuracy is significantly reduced. Using Conventional Programming methods as a means of implementation will mean I can code and formulate rules to deal with a more generic view on all electronic mail which will hopefully allow a high level of detection accuracy.

A factor I also considered when selecting a Conventionally Programmed system over a Machine Learning system was the ability to be innovative, due to the lack of published papers introducing a system of this nature.

After I made the decision to use Conventional Programming, the next decision I made was based on a programming language that would complement the proposed system. I subsequently chose to use the programming language Java for implementation. I already had a proficiency advantage when it came to Java, and the decision was amplified given Java has a better performance than Python due to it being a compiled language. Another benefit to using Java is the need for programs to be well written in order to compile, which means a

developed system is stable, and less likely crash. The most important benefit to Java programming is how secure the programming language is. Java has fewer identified vulnerabilities than some of the other well-known languages, including Python. I acknowledge the various security risks associated with Conventional Programming but I can control this during the developmental phase by securely developing my system and performing various testing in a pre-deployment phase to catch any gaps in the system before it goes live. I also anticipate a higher accuracy with a Conventionally Programmed system as it will have an algorithm based on rules to deal with all Phishing possibilities as opposed to developing an algorithm based on a dataset.

In terms of Operating Systems and making a preference-based decision on this, I have decided to use MacOS for implementation. As Java requires rudimentary features when it comes to System Hardware, this is satisfied with most if not all devices purchased in recent years. I also intend to use the Integrated Development Environment (IDE*) Eclipse[17] to aid with my Java implementation.

To satisfy the specification to develop an Email Client extension or add-on, I have chosen to use Gmail as an Email Client. This was an easy decision based on Outlook's limitation to include extensions in a browser or web interface setting. This is predominately based on Outlook being a desktop application, with a web interface option available with a Microsoft 365 subscription. Gmail, by default is a web application.

As I have chosen Java as my programming language, I will be using the Java Mail API to gain visibility into electronic mail through a Java program. This will allow me to interact with and perform the necessary workings to determine if links contained within an email are malicious. I will also be able to access the Domain Name System (DNS) provider which will allow Java Naming Directory Interface (JNDI) applications to access information stored in the Internet Domain Name System. This will provide me with more opportunities to correctly determine the nature of a link within an email.

I have chosen to use the Phish Tank repository as my dataset of Phishing emails for this implementation. This was an easy decision based on the versatility of the dataset to be used with all technological approaches.

As I am Conventionally Programming my system, one of the detection methods I will be using as an important indicator is Email Header Analysis. This is an important indicator due to the amount of vital information an email header contains, specifically sender and route to the recipient information. Any inconsistencies in these two pieces of information will indicate early on the authenticity of an email.

This exhaustive list of chosen technologies will allow me to successfully implement my project deliverables in as much of an accurate, secure and timely manner.

[17] Eclipse: https://www.eclipse.org/ide/

Appendix

Appendix A: Algorithm

As I have yet to come across a similar algorithm previously published to tailor to my expected end-goal, I intend to utilize an algorithm I formulate to detect Phishing emails.

In the flowchart (Figure 7) below, at a very high level I have indicated the main approaches I intend to implement.

The implementation begins when the intended mail recipient receives an email. Email header analysis is then performed to check for any inconsistencies in relation to the 'Header From' and 'Mail From' addresses and the domains included in these addresses. The result of the Email header analysis would be stored before the system moves onto the next method of analysis, URL extraction. Any hyperlinks or URLs contained in the email are extracted to check for common Phishing indicators such as a young domain age – checked with DNS. Amiri et al, (2014) describes other indicators of Phishing that can be gathered from the URL:

- A long URL: this can be used to hide the suspicious part of the URL in the address bar
- Dots: more than five dots present in a URL. Legitimate URLs contain at most five dots). For example: http://www.website1.com.my/www.phish.com/index.php
- IP address: if an IP address is present instead of a fully qualified domain name, this is very suspicious as legitimate websites no longer use this method due to security reasons.
- '@' Symbol: an '@' symbol in the URL may indicate Phishing. This is because a web browser ignores everything to the left of an '@' symbol in a URL, therefore, the address ebay.com@fake-auction.com would actually be "fake-auction.com."
- Hexadecimal: Hex-encoded URLs can indicate Phishing attempts as most mail user agents, web browsers and HTTP servers understand basic hex-encoded character equivalents. This is used to evade anti-spam filters and black-lists.

The webpage associated with the hyperlinks can also be checked for Phishing indicators by extracting the source code of the webpage. This will indicate if any suspicious page redirects are occurring unbeknownst to the user. Frames are also a popular method of hiding attack content due to their uniform browser support and easy coding style (Amiri et al, 2014).

The next method of analysis is simply to check attachment extensions, if present. This isn't necessary to determine if an email is of a Phishing nature, but it might help with accuracy levels. Any attachments with the extensions .exe, .iso, .zip, .rar, and .dmg, would indicate the potential presence of Phishing.

The final step in the algorithm is finally classifying the email based on the results of the methods checked. If all methods produced a result which indicated Phishing, then it is safe to inform the mail recipient that the nature of the email is in fact Phishing.

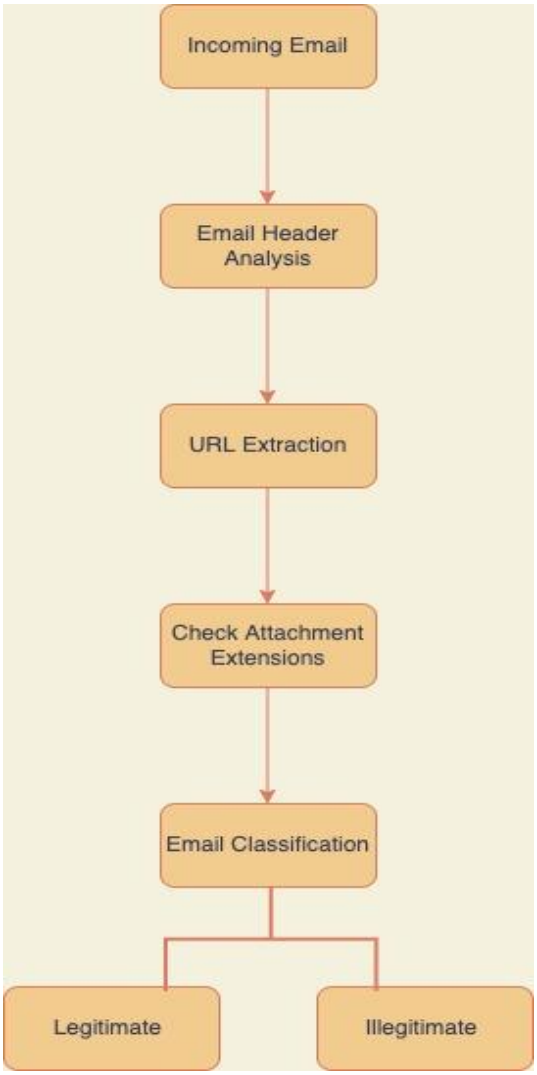
Incoming Email
Email Header Analysis
URL Extraction
Check Attachment Extensions
Email Classification
Legitimate
Illegitimate

Figure 7

Glossary

Glossary of Terms, Abbreviations and Acronyms

AI:	Artificial Intelligence
AMD:	Advanced Micro Devices
API:	Application Programming Interface
CPU:	Central Processing Unit
CRUD:	Create, Read, Update, Delete
DKIM:	DomainKeys Identified Mail
DMARC:	Domain-based Message Authentication, Reporting and Conformance
DNS:	Domain Name System
Email:	Electronic Mail
HTTP:	Hypertext Transfer Protocol
ID:	Identity
IDE:	Integrated Development Environment
IMAP:	Internet Message Access Protocol
IP:	Internet Protocol
JNDI:	Java Naming and Directory Interface
MIME:	Multiple Internet Mail Extension
NER:	Named Entity Recognition
OS:	Operating System
PC:	Personal Computer
POP:	Post Office Protocol
RAM:	Random Access Memory
SMTP:	Simple Mail Transfer Protocol
SPF:	Sender Policy Framework
TLD:	Top Level Domain
TLS:	Transport Layer Security
UI:	User Interface
URL:	Uniform Resource Locator

Bibliography

Amiri, I.S., Akanbi, O.A. and Fazeldehkordi, E., 2014. A Machine-Learning Approach to Phishing Detection and Defense. Syngress. https://www.sciencedirect.com/topics/computer-science/phishing-detection[Accessed: 5 November 2021]

Arxiv (2021) *Detecting Phishing Sites - An Overview.* Available at: https://arxiv.org/pdf/2103.12739.pdf#:~:text=There%20are%20various%20phishing%20attacks,%2D%20similarity%20and%20machine%2Dlearning [Accessed: 5 November 2021]

Britannica (2019) *Mac OS.* Available at: https://www.britannica.com/technology/Mac-OS [Accessed: 7 November 2021]

Britannica (2021) *Microsoft Windows.* Available at: https://www.britannica.com/technology/Windows-OS[Accessed: 7 November 2021]

Career Karma (2020) *The Best Operating System for Programming: Choose the Best OS for Coding.* Available at: https://careerkarma.com/blog/best-operating-system-for-programming/ [Accessed: 12 October 2021]

DNS Twister, 2022. *APIs*. Available at: https://dnstwister.report/documentation/dnstwister_apis [Accessed 18 April 2022]

Enthought (2020) *Enthought Python Minimum Hardware Requirements*. Available at: https://support.enthought.com/hc/en-us/articles/204273874-Enthought-Python-Minimum-Hardware-Requirements [Accessed: 14 October 2021]

Expert AI (2020) *What is Machine Learning? A Definition.* Available at: https://www.expert.ai/blog/machine-learning-definition/ [Accessed: 7 October 2021]

Expertrec Blog, 2021. *Google custom search JSON API explained*. Available at: https://blog.expertrec.com/google-custom-search-json-api-simplified/ [Accessed 18 April 2022]

Go Coding (2020) *Java Mail API.* Available at: https://gocoding.org/java-mail-api/ [Accessed: 18 October 2021]

Google Apps Script (2021), *Extending Gmail with Google Workspace add-ons*. Available at: https://developers.google.com/apps-script/add-ons/gmail[Accessed: 1 November 2021]

GoPhish, 2022. *Sending Profiles*. Available at: https://docs.getgophish.com/user-guide/documentation/sending-profiles [Accessed 18 April 2022]

Gold Fynch, 2021. *Why does email metadata sometimes appear inconsistent?* Available at: https://goldfynch.freshdesk.com/support/solutions/articles/2100033157-why-does-email-metadata-sometimes-appear-inconsistent-#:~:text=If%20the%20alias%20is%20unverified,email%20from%2C%20not%20the%20alias.

[Accessed 18 April 2022]

Google Safe Browsing (2021) *Safe Browsing Lookup API (v4).* Available at: https://developers.google.com/safe-browsing/v4/lookup-api?csw=1[Accessed: 1 November 2021]

Google Safe Browsing, 2022. *What are the Safe Browsing APIs?* Available at: https://developers.google.com/safe-browsing/v4
[Accessed 18 April 2022]

Google, 2021, *Google Workspace Add-ons*, image. Available at: https://workspace.google.com/products/add-ons/
[Accessed 1 November 2021]

Guru 99 (2021) *What is Java? Definition, Meaning & Features of Java Platforms.* Available at: https://www.guru99.com/java-platform.html [Accessed: 8 October 2021]

Humberto Rocha (2018) *Sending and receiving emails with Python.* Available at: https://humberto.io/blog/sending-and-receiving-emails-with-python/ [Accessed: 18 October 2021]

Kasper Sky, 2022. What is Typosquatting? – Definition and Explanation. Available at: https://www.kaspersky.com/resource-center/definitions/what-is-typosquatting[Accessed 18 April 2022]

IBM (2020) *Artificial Intelligence (AI)*. Available at: https://www.ibm.com/cloud/learn/what-is-artificial-intelligence#:~:text=Artificial%20intelligence%20leverages%20computers%20and,capabilities%20of%20the%20human%20mind [Accessed: 7 October 2021]

Imaginary Cloud (2021) *Python Vs Java: Key Differences and Code Examples.* Available at: https://www.imaginarycloud.com/blog/python-vs-java/ [Accessed: 12 October 2021]

Info Security (2020) *How it Works: Machine Learning Against Email Phishing*. Available at: https://www.infosecurity-magazine.com/blogs/machine-learning-email-phishing/[Accessed: 7 October 2021]

Info World (2020) *How secure is Java compared to other languages?* Available at: https://www.infoworld.com/article/3537561/how-secure-is-java-compared-to-other-languages.html [Accessed: 21 October 2021]

Kaggle (2019) *Difference between Traditional Programming and Machine learning.* Available at: https://www.kaggle.com/getting-started/150361 [Accessed: 8 October 2021]

Mail Trap (2019) *Guide to Send Emails in Java.* Available at: https://mailtrap.io/blog/sending-email-using-java/ [Accessed: 7 October 2021]

Meta Compliance (2021) *The Ultimate Guide To Phishing*. Available at: https://www.metacompliance.com/lp/ultimate-guide-phishing/[Accessed: 15 November 2021]

Microsoft (2021) *Build your first Outlook add-in*. Available at: https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/outlook-quickstart?tabs=yeomangenerator[Accessed: 1 November 2021]

Microsoft, 2021, *Build your first Outlook add-in.* Available at: https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/outlook-quickstart?tabs=yeomangenerator [Accessed: 1 November 2021]

National College of Ireland (2021) Detection of Phishing URL using Ensemble Learning Techniques. Available at: http://norma.ncirl.ie/4509/1/sharadrajendraparmar.pdf [Accessed: 25 October 2021]

Net Core, 2021. SPF DKIM and DMARC Explained With ISP Support. Available at: https://netcorecloud.com/tutorials/spf-dkim-dmarc/#:~:text=By%20Hitesh%20Pandey%F0%9F%92%BB,sending%20emails%20through%20your%20domain. [Accessed 18 April 2022]

Onix (2021) *What are the Limitations of Machine Learning?.* Available at: https://onix-systems.com/blog/what-do-you-need-to-know-about-the-limits-of-machine-learning[Accessed: 7 October 2021]

Oracle (2021) *DNS Service Provider for the Java Naming Directory Interface™ (JNDI).* Available at: https://docs.oracle.com/javase/7/docs/technotes/guides/jndi/jndi-dns.html [Accessed: 18 October 2021]

Oracle (2021) *What is Machine Learning?* Available at: https://www.oracle.com/ie/data-science/machine-learning/what-is-machine-learning/ [Accessed: 28 October 2021]

Phish Labs, 2021. Top 10 TLDs Abused. Available at: https://www.phishlabs.com/blog/top-10-tlds-abused/[Accessed 18 April 2022]

Phish Tank (2021) *Join the fight against phishing*. Available at: https://phishtank.org/[Accessed: 28 October 2021]

Pitch Funnel (2020) *Email Usage Statistics in 2021*. Available at: https://pitchfunnel.com/blog/email-usage-statistics/ [Accessed: 21 October 2021]

Proofpoint (2021) *What is Email Security?* Available at: https://www.proofpoint.com/us/threat-reference/email-security [Accessed: 11 November 2021]

Python (2021) *What is Python? Executive Summary.* Available at: https://www.python.org/doc/essays/blurb/ [Accessed: 8 October 2021]

Refresh Java (2021) *Hardware and Software requirements for Java.* Available at: https://www.refreshjava.com/java/geting-started-with-java [Accessed: 14 October 2021]

Sendgrid, 2019. *What Is an SMTP Server?* Available at: https://sendgrid.com/blog/what-is-an-smtp-server/ [Accessed 18 April 2022]

Snap Logic (2021) *Python vs. Java Performance.* Available at: https://www.snaplogic.com/glossary/python-vs-java-performance [Accessed: 8 October 2021]

Spike (2021) *Gmail vs Outlook: What's the Best Email Service?* Available at: https://www.spikenow.com/blog/team-collaboration/gmail-vs-outlook-whats-the-best-email-service/ [Accessed: 18 October 2021]

SUSE (2021) *Linux Operating System*. Available at: https://www.suse.com/suse-defines/definition/linux-operating-system/[Accessed: 7 November 2021]

Tech Radar (2020) What is phishing and how dangerous is it? Available at: https://www.techradar.com/news/what-is-phishing-and-how-dangerous-is-it [Accessed: 25 October 2021]

The EMEA (2021) *Tomorrow's Digital Workplace*. Available at: https://www.ifsskillnet.ie/wp-content/uploads/2021/07/EMEA-Report_Tomorrows-Digital-Wokrplace_English_final.pdf [Accessed: 21 October 2021]

The Stanford Natural Language Processing Group (2021) *Stanford Named Entity Recognizer (NER).* Available at: https://nlp.stanford.edu/software/CRF-NER.html [Accessed: 1 November 2021]

Towards Data Science (2019) *The Limitations of Machine Learning.* Available at: https://towardsdatascience.com/the-limitations-of-machine-learning-a00e0c3040c6 [Accessed: 7 October 2021]

Towards Data Science (2020) *5 Common Machine Learning Security Risks and How to Overcome Them.* Available at: https://towardsdatascience.com/5-common-machine-learning-security-risks-and-how-to-overcome-them-2f90115a699d [Accessed: 12 October 2021]

Tutorials Point (2021) *Python - DNS Look-up.* Available at: https://www.tutorialspoint.com/python_network_programming/python_dns_look_up.htm [Accessed: 18 October 2021]

VirusTotal, 2022. *How it works*. Available at: https://support.virustotal.com/hc/en-us/articles/115002126889-How-it-works [Accessed 18 April 2022]