PHISH-AWARE

Final Report

SEAN DOWLING
C00246571

Contents

A. Introduction

Email has become an essential communication tool in today's digital world, In 2020, the number of global e-mail users amounted to four billion and is set to grow to 4.6 billion users in 2025 (Ceci, 2022). However, with the increase in email usage comes an increase in email-related security threats, such as phishing attacks, malware, and viruses, phishing email statistics suggest that nearly 1.2% of all emails sent are malicious, which in numbers translated to 3.4 billion phishing emails daily (James, 2023). To address these security concerns, an Outlook add-in was developed to provide users with advanced email analysis features. The add-in allows users to analyze email headers, URLs, and attachments to detect potential threats and provides phishing guidance to help users identify and avoid phishing attacks. This final report presents an overview of the Outlook add-in's features, the methodologies used for email analysis, and the effectiveness of the add-in in enhancing email security. The tool can also be used as a guide to manually detect phishing emails, including common techniques used by attackers, common errors made and other useful information. From my own research of Outlook Add-Ins, there are a small number of similar tools, none of which go into detail of why an email may be malicious or not. There are also no add-ins found that provide analysis on attachments contained within an email. Finally, no tool provides guidance on common attacks and techniques used by attackers in today’s world.

B. Project Description

The Outlook Add-In itself is a powerful tool that enables users to analyze emails' header information, including SPF, DKIM, DMARC, To, From, and Return Path and other essential details. The add-in will use various Application Programming Interfaces (APIs) to scan URLs and attachments, detecting any potentially malicious activity. The Add-In is designed to improve email security by identifying suspicious emails, alerting users of any potential threats. The tool will enable users to view a detailed report of the email's header information, as well as any URLs and attachments. The report will provide guidance on whether the email is safe or potentially harmful. The Add-In is very user-friendly, easy to install, and will integrate seamlessly with Microsoft Outlook.

The key features of the Add-In include:

- Email header analysis: The Add-In will scan emails' header information and provide detailed information on SPF, DKIM, DMARC, as well as other essential details.
- URL and attachment scanning: The tool will use various APIs to analyze URLs and attachments, detecting any potentially malicious activity.
- Report function: The Add-In will generate a detailed report on the email's header information, URLs, and attachments, and will allow the user to send the report to IT Services.
- Guidance tool: The Add-In will provide a guide for users on whether the email is safe or potentially malicious, thus helping users make informed decisions about opening any links or attachments.

In conclusion, the Outlook Add-In is a standout tool that will provides users with an additional layer of security, allowing them to analyze emails' header information, URLs, and attachments. The tool will be user-friendly, easy to install, and will work hand in hand with Microsoft Outlook, both the application and web browser versions, providing users with a powerful tool to protect themselves from potential email threats.

C. Issues

Many issues arose along the way. Many of these issues were an easy fix while some required a more thorough investigation.

1. Python

My lack of knowledge with the Python language hindered me. So, to help alleviate this problem I switched to JavaScript, HTML and CSS. I done this as there many libraries available that work hand in hand with Microsoft applications like Outlook, libraries such Microsoft Office JavaScript API, which gave me the foundation for finding all the information for the header analysis, URLs and attachments. Due to my lack of knowledge of Python I did not want to get stuck and have to rethink my whole project and functional specification, so for convenience and saving time I switched to a language I have used before and have a good understanding of.

2. Cross Origin Resource Sharing

Cross-Origin Resource Sharing (CORS) is an HTTP-header based mechanism that allows a server to indicate any origins (domain, scheme, or port) other than its own from which a browser should permit loading resources (Mozilla , 2023)This feature is designed to prevent malicious scripts from stealing data or executing actions on behalf of a user without their consent. However, CORS can sometimes cause issues when developing web applications that need to communicate with APIs hosted on different domains. For this reason, any attempt made by my Outlook add-in to contact the API would result in a “Error 500: Forbidden” error. I found online a probable fix to this problem, by sending the API call through a “reverse-proxy” called cors-anywhere.herokuapp.com, I could in theory, get the information I was looking for, but this solution did not work as the use of reverse-proxies by all the APIs is forbidden. This was my biggest problem area until I found my own solution. By hosting nothing but the API calls on a web application, I could realistically contact from my Outlook add-in to my API web application, which would do nothing other than make the API calls. The results would then be sent back to the add-in in a nicely laid out format. Finding a place to host firstly the Outlook add-in was the next problem I had.

3. Web App Hosting

For testing purposes, I was hosting the Outlook Add-In on the localhost, but when I wanted to get it up and available for use by anyone, I needed a place for it to be stored and work. I tried many different hosting websites to no avail due to many different scenarios, but I finally found Azure Web Apps. I was able to store not only the API calls here, but the Outlook Add-In also, this was ideal as there was also a plugin on Visual Studio code that worked hand in hand with Azure so deploying code to the site was easy. I had concerns though about the Cross Origin Resource Sharing as I had a fear that due to Azure's high security standards I would not be able to bypass it, but to my delight, there is an area where you can allow certain domains to allow CORS, so I enabled it for my Outlook Add-In and disabled it for every other domain, keeping it as safe as possible.

4. URL and Attachment APIs

Getting the API calls to work was also a large problem I had. The largest, most well-known and best API that I used, VirusTotal, was the easiest to set up and get running, as there is plenty of documentation to support, from different languages to different libraries you can use, it's all there. Due to a lack of documentation for the other APIs I have used, I found it really challenging designing the code to make the successful API calls. Due to the increased trial and error period for this, it increased the time pressure I had already been feeling.

5. URLs

I had some minor problems with extracting URLs from an email. The problem I was facing was that the URL that was being extracted was the eur03.safelinks.protection.outlook.com URL. This link is Outlooks own phishing detection system in work, but for my scenario I didn't need this extra part of the link as it wouldn't be scanned by any API, so I created a piece of code that would bypass any protection URL and only scan the legitimate URL. Also, during my testing phase, I found that large corporation domains like Facebook, Youtube, Instagram, etc. would not be scanned by some API scanners, so I created another piece of code that would bypass them also so that it would not return false results to the user. Another problem I had with the URL scanning part was that the setu.ie URL is included in everybody's signature, meaning that it would take up resources to scan that link every time the Add-In was run, so once again I created a method to bypass the setu domain. The final problem I had with the URLs was anything in place after the domain name, for example www.example.com/sample. Anything after / will not be scanned by any API scanner, so I used Regex to filter out everything after the first / found in a link - var filteredUrl = url.replace(/^(?:https?:\/\/)?(?:www\.)?([^\/]+)\/?.*$/, "$1");. So once again, if I had an URL like www.example.com/sample then /sample would be stripped off and you would just be left with www.example.com. This new filtered URL will then be sent to the API scanners to be scanned.

6. Attachments

Due to security problems, allowing Add-Ins to extract potentially malicious attachments is not allowed. This means I was not able to use the same method for extract URLs from an email body. Without heightened privilege levels the only parts of the attachment any Outlook Add-In is able to have access to is the name, size and file type. This was a common feature that seemed to be missing when I was doing my research, no Add-In seemed to deal with attachments at all, therefore making my Add-In unique in that sense. I have successfully overcome this problem by the use of an iFrame. I was able to set up my second web app with nothing but a drag and drop area for the users to drop their potentially malicious files. I was then able to make the API call through this web app and show the results back to the user through the iFrame. Although it doesn't look as nice and flush as I might have expected it is still very functional in a sense that it does exactly what it is meant to.

D. Accomplishments

I have achieved everything I set out to achieve shown in my research paper and functional specification paper. This of course was to expand the functionality of Phish detection and guidance to a new level. Below is a table of the Outlook Add-In's functionality with a small description.

Functionality	**Description**
Show valuable info to user	Show information that is needed to the user such as to, from, subject, return-path, etc.
Check SPF, DMARC, DKIM	Check to see if the Sender Policy Framework (SPF), Domain Message Authentication, Reporting and Conformance, and DomainKeys Identified Mail are all passing, soft-failing or failing
Check if "From" and "Return-Path" match	A tick or X will appear if the "From" and "Return-Path" fields match or not
Locate URLs	Find any URL in the email body
Analyse URLs	Get a report on each URL from APIs
Detect Attachments	List information for Attachments – Name, Size, Type
Allow user to analyse Attachments	Allow the user to drag and drop the Attachment to a separate web app that will use APIs to scan the attachment
Provide Phishing Guidance	Area designed specifically for showing common techniques and common attacks used
Report Function	Allow the user to report the email to IT Services, containing the results along with all header information, URL and Attachment scans also

E. Non-Fulfilment

I completed all tasks I had outlined in my “Functional Specification” paper, although the only part I did not complete in full is the Attachment API Scanners. Due to the difficulty and time constraints, I was only able to get one API Scanner (VirusTotal) working to a high level. The testing phase also was not fully successful. I wanted to test both scenarios for the URL and Attachment analysis, this would have been done using tools like KingPhisher or GoPhish, due to time constraints and lack of knowledge using the tools, I was unable to test the malicious side of the tool. Its not common for websites to have documentation on creating phishing websites to test the URL API scanners or malicious Files to test the Attachment API scanners. I was still able to test it to the best of my ability, by sending normal or “clean” emails firstly for testing, which were passed successfully, the malicious files and URLs on the other hand was not complete.

F. Learning Outcomes

I have learned a lot throughout this project, not only from a technical standpoint but also from a personal perspective.

1. Technical

Throughout my time of developing this Outlook Add-In tool, I have had many learning opportunities, these include:

- NPM
- Outlook Add-In Development
- VirusTotal API
- URL Scan API
- IP Quality Score API
- Azure Web Apps
- Microsoft Office JavaScript API

2. Personal

Through exposure to new technologies, I've gained valuable insights into project management, including:

- Breaking down the main project scope into smaller, manageable segments.
- Expecting design changes and recalibrating the original scope as necessary.
- Being flexible in altering project scope for a better outcome.
- Expecting schedule changes and implementing workarounds to address issues.
- Managing time effectively, as it directly impacts the quality and scope of the final product.
- Recognizing the importance of research and planning in identifying project goals, focus, and methods.
- By adopting these principles, I've been able to successfully manage projects and deliver high-quality results.
- Enhancing problem solving, especially with time against you

G. Layout of Results

Below is a layout of the results that are shown to the user for each section – Header Information, URLs, Attachments and Guidance section.

1. Header Information

The header information is laid out in a fashion that is easily readable to the non-technical user, while also going more in depth for users who want to know more about potential phishing attacks.

Analysis of Email
To
Sender
Return Path
Received
Subject
SPF
DMARC
DKIM
SPF, DMARC, DKIM (Clickable Buttons)

- **To** – Contains who the email was sent to.
- **Sender** – Contains the sender address (that is visible to the user).
- **Return Path** – Contains the return path of the email.
- **Received** – Contains the time and date the email was received.
- **Subject** – Contains the subject of the message.
- **SPF** – Contains the result of the Sender Policy Framework authentication method.
- **DMARC** – Contains the result of the Domain Message Authentication, Reporting and Conformance authentication method.
- **DKIM** – Contains the result of the DomainKeys Identified Email authentication method.
- **SPF, DMARC, DKIM (Clickable Buttons)** – Allows the user to click on each for a mini description of each authentication method.

2. URLs

There may be more than one URL within an email, so for ease of reading each on is laid out below one another with safety level being the standout noticeable item as that will be the one thing most users look out for.

Domains found in email:
URL
Safety Level
URL Scan Results
IP Quality Score Results
VirusTotal Results

- **Domains found in Email** – Lists all URLs found within the body of the email.
- **URL** – Shows the current URL being scanned.
- **Safety Level** – Gives a risk score based on the scan results.
- **URL Scan Results** – Shows the results of the scan done by the URL Scan API.
- **IP Quality Score Results** – Shows the results of the scan done by the IP Quality Score API.
- **VirusTotal Results** – Shows the results of the scan done by the VirusTotal API.
- Users can also click on “Raw Data” below each scan for a more in-depth analysis.

3. Attachments

Since attachments are being delt with differently then everything else, its layout may be a little different, but none the less the functionality is still there.

File Upload:
Scan Attachment:
Scan Results:
Positives:
Total:
SHA256:

- **File Upload** – Allows the user to drag and drop any attachments within an email.
- **Scan Attachment** – Scans the attachment using the VirusTotal API.
- **Scan Results** – Shows the results of the scan.
- **Positives** – Gives you a number based of how many vendors marked the file as malicious e.g., 3.
- **Total** – Shows the total number of vendors that scanned the file e.g., 88.
- **SHA256** – Allows the user to click a button to copy the Hash of the file if they want to do a more in-depth analysis themselves.

The scan itself for new files may take some time, especially if they have never been seen by any vendor, so unfortunately the user will have to resubmit the file after around 1 minute of waiting due to the scan taken place, this is all portrayed to the user, so they know what is happening.

4. Guidance

The guidance part of the tool again includes a lot of clickable buttons for the user. The user can have it open alongside their potentially malicious email, comparing common techniques to the email body, making their judgements along the way. This is very handy especially for the non-technical users so that they can get a quick understanding of the techniques used, and it is always there open on the side, making it easier than having to open and close a document every time.

Overview:
Don't fall for their trap:
Common Techniques:
Clickable Buttons
Malicious Attachments
Phishing Websites
Dates and Times
Spoofed Emails
Urgent Emails
Grammar

- **Overview** – Gives a quick overview of what phishing is.
- **Don't fall for their trap** – Warns users not to click any potentially malicious links or attachments.
- **Common Techniques** – Shows users common techniques used.
- **Clickable Buttons** – Allows users to click on each different technique and gives them a brief description of each.

H. Testing

To test each functionality of my Outlook Add-In I created several test cases to see if it fulfilled the functionality that was outlined in my functional specification paper.

Description	Steps	Expected Result	Actual Result
Detect To, From, Subject, Date fields	Send a test email	Information laid out showing the To, From, Subject and Date	Pass
Detect SPF, DKIM and DMARC results	Send a test email	Should show results on whether SPF, DKIM and DMARC passed or failed	Pass
Detect if From and Return-Path match	Send a test email	Should show a tick or X if From and Return-Path match or not	Pass
Detect any URL contained within an email	Send a test email containing an URL	Should outline all URLs found within an email	Pass
Detect any Attachments contained within an email	Send a test email containing an Attachment	Should outline all Attachments found within an email – Showing the name, size and type	Pass
Send detected URLs to be scanned	Send a test email containing an URL	Should contact various APIs and show results of scans	Pass
Allow user to drag and drop Attachments to be scanned	Send a test email containing an Attachment	Should contact VirusTotal and show results of scans	Pass
Allow user to view common techniques used by attackers	Send a test email and open the Outlook Add-In	Should show common techniques by clicking on each button	Pass

I. User Testing

Once I had completed my project to the best of my ability, I began testing it out. I ran the test with my family members. They were divided into two groups based on their experience with using a computer. These groups were:

- Avid users (users that use the computer on a daily basis)
- Novice users (users that rarely use a computer)

Both groups have no experience in the field of phishing or using Outlook Add-Ins.

- Avid users were able to install, use each part of the tool (header information, URL and attachment analysis, and the guidance section) with relative ease.
- Novice users struggled to install the tool, but were able to use each part of the tool to the best of their abilities, frequently referencing back to the guidance section of the tool.

For the final part of my project, to accommodate non-technical users, a user manual has been created, stepping users through each component of the tool, from installation all the way to reporting.

J. What I would do differently

If I had to start this project from scratch again, I would focus on just header information and URLs. Due to the nature of security surrounding attachments contained within emails, it caused more hassle than was needed while doing the project. With the extra time I would then have I would be able to perfect a more distinct classification on whether an email was phishing or not. Like I said above, I would have liked to have done this project in Python, but due to my lack of knowledge and limited time, I felt more at ease using a language I am comfortable with and have used in the past.

K. Conclusion

Overall, I believe the end product I have produced has been a success. I completed all the main points outlined in my functional specification, creating a fully functional Outlook Add-In that allows users to perform phish analysis. If I was able to get some major problems figured out and dealt with earlier I feel like I would have more time to perfect the tool to a higher level. I am glad of the different technologies I used all the way through, and I am delighted with the knowledge I gained throughout the whole process. I was extremely happy with how I found solutions to seemingly impossible tasks, all while keeping in line with the time constraints that were given to us.

I also feel my tool will may a good impact on the phish protection market. In the fashion that I have gone about creating this tool it sets it apart from all the other tools freely available. As I have stated above, the tool can be used in many different ways, from header analysis, URL analysis and finally attachment analysis, which has not been seen in any other tool. This is why I feel my tool is a great impact as it deals with all possible aspects of phish threats in today's climate.

Although it may not mean much, but due to the extreme enjoyment (mostly) I got throughout this whole process, I do plan on creating more add-ins in my spare time as I know the possibilities are endless with them.

L. User Manual

The installation process is complex, so I will quickly run through the steps needed to use my tool to its max ability.

- Go to https://showcase.itcarlow.ie/ and click on Cybercrime and IT Security
- Locate “An Outlook Add-In for Detection and Guidance against Phishing Attacks”
- Open the webpage and locate “Download the Tool.”
- You can then follow the readme.md file at the bottom of the GitHub repository page for the complete the installation process.

L. Appendix

Below are what I feel a significant code snippets that have had a major contribution to the completion of this project.

The first code snippet was the largest problem I had in the project. You can see the xhr request that had to be made to my API proxy. As I said above I had great problems with Cross Origin Resource Sharing. Originally, I was requesting the API scanners straight from the Outlook add-in, but due to CORS, it was forbidden so I had to create my own API proxy where I hosted the API calls and made the request straight to there and got the results that way instead.

```
for (var i = 0; i < filteredUrls.length; i++) {
  if (scannedUrls.indexOf(filteredUrls[i]) === -1) {
    scannedUrls.push(filteredUrls[i]);
    (function (url) {
      var xhr = new XMLHttpRequest();
      var apiUrl = `https://api-phish-proxy.azurewebsites.net/scan?url=${encodeURIComponent(url)}`;
      xhr.open("GET", apiUrl, true);
      xhr.setRequestHeader("Content-Type", "application/json");
      xhr.setRequestHeader("Access-Control-Allow-Origin", "*");
      xhr.setRequestHeader("Access-Control-Allow-Headers", "Origin, X-Requested-With, Content-Type, Accept");
      xhr.onreadystatechange = function () {
        if (this.readyState === XMLHttpRequest.DONE && this.status === 200) {
          var responseDiv = document.createElement("div");
          responseDiv.innerHTML = `<div style="text-align: center;">
      <p><h3><b><u>URL:<br></b></u> ${url}</h3></p>
      </div> <b>${this.responseText}</b>`;
          document.getElementById("item-vtlinksresponse").appendChild(responseDiv);
        } else if (this.readyState === XMLHttpRequest.DONE) {
          console.error(`Request failed with status: ${this.status}`);
          var responseDiv = document.createElement("div");
          responseDiv.innerHTML = `Error for ${url}: This URL could not be scanned for some reason, please retry
          document.getElementById("item-vtlinksresponse").appendChild(responseDiv);
        }
      };
      xhr.send();
    })(filteredUrls[i]);
```

The next code snippet shows the reporting section, I originally anticipated using the “Forward” method but after further research I needed higher permission levels which I could not get access to. So instead, I was able to create a new email message are from there I was able to change the “To,” “Subject” and body of the email. In the email body then I added all the relevant information for the IT services team to be able to make their determination.

```
  // Create the new message and set the recipients and subject
  const message = Office.context.mailbox.displayNewMessageForm({
    toRecipients: ["itservices@itcarlow.ie"],
    subject: `PHISHING ATTEMPT: (${subject})`,
    htmlBody: `${emailBody}`,
  });
}
```

The final code snippet I want to speak about is the file upload section for attachments. As I spoke about, for security reasons attachments have to be dealt with differently then URLs. So, I designed a zone for users to drag and drop their attachments into for them to be scanned.

```
<h3>File upload</h3>
<div id="drop-zone">Drag and drop files here or click to upload</div>
<form id="upload-form" method="post" action="/upload" enctype="multipart/form-data" style="display: none;">
  <input id="file-input" type="file" name="file">
  <button type="submit">Submit</button>
</form>
<p>If the information is blank, please resubmit your file in 1 minute as the scan has not completed</p>
<div id="result-message"></div>
<script>
  var dropZone = document.getElementById('drop-zone');
  var fileInput = document.getElementById('file-input');
  var uploadForm = document.getElementById('upload-form');

  dropZone.addEventListener('dragover', function (event) {
    event.preventDefault();
    dropZone.style.backgroundColor = 'lightgray';
  });

  dropZone.addEventListener('dragleave', function (event) {
    event.preventDefault();
    dropZone.style.backgroundColor = '';
  });

  dropZone.addEventListener('drop', function (event) {
    event.preventDefault();
    dropZone.style.backgroundColor = '';
    fileInput.files = event.dataTransfer.files;
    uploadForm.style.display = 'block';
  });

  dropZone.addEventListener('click', function () {
    fileInput.click();
  });

  fileInput.addEventListener('change', function () {
    uploadForm.style.display = 'block';
  });
```

If you would like to see the rest of the code, it is available on GitHub. The code for the tool itself is available here. For the code for the API calls, you can check it out here.

M.Glossary

Glossary of Terms, Abbreviations and Acronyms

API:	Application Programming Interface
CSS:	Cascading Style Sheets
DKIM:	DomainKeys Identified Mail
DMARC:	Domain-based Message Authentication, Reporting and Conformance
Email:	Electronic Mail
HTML:	Hypertext Markup Language
IP:	Internet Protocol
JS:	JavaScript
Regex:	Regular Expression
SPF:	Sender Policy Framework
URL:	Uniform Resource Locator

N. Bibliography

Ceci, L., 2022. *Number of e-mail users worldwide from 2017 to 2025.* [Online]
Available at: https://www.statista.com/statistics/255080/number-of-e-mail-users-worldwide/
[Accessed 10 April 2023].

James, N., 2023. *81 Phishing Attack Statistics 2023: The Ultimate Insight.* [Online]
Available at: https://www.getastra.com/blog/security-audit/phishing-attack-statistics/#:~:text=Phishing%20email%20statistics%20suggest%20that,attack%20occurring%20every%2011%20seconds.
[Accessed 10 April 2023].

Mozilla , 2023. *Cross-Origin Resource Sharing (CORS).* [Online]
Available at: https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS
[Accessed 10 April 2023].

O. Acknowledgements

I would like to thank my supervisor, Keara Barrett, for providing guidance and feedback throughout this project. I would also like to thank Security Risk Advisors. I completed my 3rd year work placement there and inspired me to make this tool.

P. Declaration

*I declare that all material in this submission e.g., thesis/essay/project/assignment is entirely my/our own work except where duly acknowledged.

*I have cited the sources of all quotations, paraphrases, summaries of information, tables, diagrams or other material: including software and other electronic media in which intellectual property rights may reside.

*I have provided a complete bibliography of all works and sources used in the preparation of this submission.

*I understand that failure to comply with the Institute's regulations governing plagiarism constitutes a serious offence.

Student Name: Sean Dowling

Student Number(s): C00246571

Date: 17th April 2023