**(Unfinished, and WIP)**

**Researching Methods and Algorithms for Phishing Prevention Specifically Regarding Impersonation**

**What we aim to build, and for what purpose:**

Our goal as a group is to develop an algorithm of scripts that process either an email entire, or an email address and assess whether there is some malicious intent behind it, whether that be the content contained within, i.e. a phishing link or any link that is otherwise malicious, or potentially some content that may try to trick the recipient into sending money, research, or otherwise important files, documents, or information.

The aim is to create a simple UI that anyone can use, regardless of technical proficiency. During research it was made clear that many new researchers at UWE face issues where they are contacted by criminals acting as publishers, the University may then fund the production and sending of research to these publishers, wherein nothing comes of it and nothing is printed, and the University ultimately loses money to a scam. This works as it is often difficult to discern real publishers from fake, especially if your academic field is entirely detached from IT in any way. Also, this evidently doesn’t just affect University researchers, as this kind of impersonation fraud is by far the largest (at 51% of all fraud scams) that occurs in the UK (Ofcom, 2023).

This idea of working against phishing also directly ties into our overarching theme of “IT Continuity Planning at UWE Bristol" and our supporting document going over our “Risk Assessment,” where phishing is documented as one of the main improvements to work towards regarding expanding safety and protecting IT at UWE, as well as the students and lecturers who work and attend there.

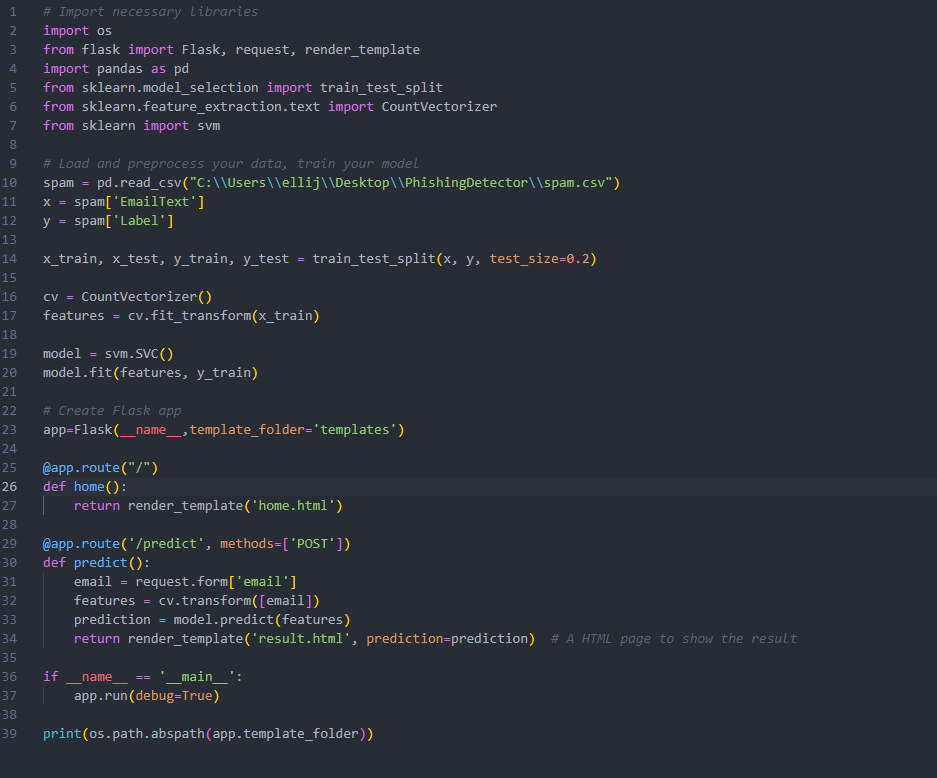
**What already exists:**

In researching how to create an algorithm or system than can be used as a prevention method to phishing scams and emails, or an interface that can manually check against common phishing address traits, there were a surprising number of prior examples and research articles into this exact topic, many that mostly matched against what our group was aiming for. It is important to note that we intended on using Python as our main language to develop this system in; this was not a limiting factor, but it becomes an easier task to take inspiration from pre-existing algorithms when they are programmed in the same language. “**HookPhish >-((())->”** is a Python script that is incredibly involved in that it checks against many already-established malware scanners, known unsafe URLs through Google’s own “Safe Site Status” website, and urlscan.io (Konate, 2023). This check is done by placing the email address in an interface that performs these checks, displaying to the user all the checks that are being completed, then spits out a result at the end. There are a few issues with this method however, as it presumes you have the capacity to, and are, running Python and understand it is way of working, furthermore, the UI is quite overwhelming in that it runs through a lot of technical jargon on-screen that the layman will not immediately understand, or even care to. While this is a very useful tool in regard to how it works, it doesn’t match up exactly to the use-case we are applying it to, at least in how it currently looks, (non-IT experienced users, looking to quickly check an email address), if this tool could be simplified down, in regards to its UI, for example if these inner-workings were made non-visible to the user, then this may be a path worth venturing down further.

**Building a machine learning detection system:**

It is a decision that is still yet to be decided, whether this algorithm will work as an extension to Outlook/Gmail, be a standalone app or hosted on its own website, this disregarded, the way in which it would work remains the same. Machine Learning felt like exactly the correct way to approach a system of this calibre, accessing data, and learning from it, and therefore allowing the algorithm to sort between spam, and non-spam emails, trained on a model of emails from both categories. Upon investigating into this, some challenges were encountered, the first of which was finding if there was some kind of API that could be hooked into Python that had the capacity for machine learning, and if there was, whether it was in my programming capacity to find a way to work that into the program we are trying to build. This goes further into this territory because as of current, I do not have a database or CSV of knowledge containing spam email examples, safe email examples, and the same for email addresses.

Because of these factors, I also began to question the scope here, was this a demonstrative block of code presented as a prototype, or was this meant to be a functioning and deployable piece of work?

Researching into machine learning, there were examples of programs and plugins specifically designed for Python, (Scikit, 2007), as well as direct tutorials on how to build an algorithm in Python for detecting spam emails, which would be learning from a CSV provided by the creator (LogRocket, 2021), and while those didn’t meet the exact specifications of what this project’s aim is, they did provide a basis for how an algorithm like this might function. Regarding Scikit, and this specific tutorial, a block of code was made and tested that did function in the way that was expected. 

To very minorly modify the code provided in the tutorial, I coded some basic HTML that would allow for HTML Form Entry and provide a result, whether the email entered was spam or not. This did work to the extent of following extremely strict criterion outlined in the CSV and without venturing much outside of that. So, while this was an impressive piece of code that was not exactly just checking against a database, it still did not have the open-endedness to be able to detect the nuance between an email from a publisher that was aiming to defraud a researcher and one that did not.

**Conclusion in a currently unfinished state:**

Though the algorithm is still in a place where it does not align with the goals of the project, it is visible how aspects from **HookPhish >-((())->** as well as aspects from the machine learning program could be collated into an algorithm that checks against not only the text included in the email, but a reverse search against the address of the email, and any URLs that exists within the contents of the email. At current, this is not a program that our group has finished coding, though the framework does exist, and we are in the process of working through a development cycle where we are working out any flaws and pitfalls of our algorithm before we commit it to building a minor but functional UI.

<https://www.ofcom.org.uk/__data/assets/pdf_file/0025/255409/online-scams-and-fraud-summary-report.pdf>

<https://asciinema.org/a/EkVgsFlj0vg8Wk4Z2c95hrO6u?source=post_page-----368216850d8-------------------------------->

https://scikit-learn.org/stable/about.html

https://blog.logrocket.com/email-spam-detector-python-machine-learning/