GENERATING SPEAR PHISHING EMAILS BASED ON WEB SCRAPED LINKEDIN INFORMATION USING OPENAI API

by

JUSTIN T. SNIDER CURTIS

Advisor
DANIEL PLANTE

A senior research proposal submitted in partial fulfillment of the requirements
for the degree of Bachelor of Science
in the Department of Mathematics and Computer Science
in the College of Arts and Science
at Stetson University
DeLand, Florida

Fall Term
2023

TABLE OF CONTENTS

TABLE OF CONTENTS ii

LIST OF FIGURES iii

ABSTRACT 1

1 Introduction 2

2 Related Work 5

3 Program Overview and Evaluation 8

3.1 Manual LinkedIn Input 8

3.2 Scrape LinkedIn Profile 9

3.3 Write Emails 11

3.4 Database Implementation 14

4 Program Design & Implementation 16

5 Results 23

6 Conclusion 29

REFERENCES 32

LIST OF FIGURES

Figure 1: JSON-LD Data..... 11

Figure 2: GUI to display created spear-phishing emails..... 14

Figure 3: Database Design..... 15

Figure 4: OpenAI API Prompt..... 20

Figure 5: Stacked bar chart depicting the overall frequency of each rating at various temperature levels..... 24

Figure 6: Average rating for temperatures..... 26

ABSTRACT

This paper presents a Python program utilizing the OpenAI API to generate convincing spear-phishing emails based on LinkedIn information. Users manually input the target's LinkedIn profile URL, prompting the program to web scrape and store data in a MongoDB database. The program generates six emails with varying persuasiveness and enables users to rank and store selected data. It serves as a valuable cybersecurity tool for testing an organization's susceptibility to phishing attacks, with ethical applications. Future enhancements could include prompt selection features and integration with tools like PhishGen and GoPhish for targeted phishing campaigns. The program's design and implementation, including web scraping and database management, are detailed, demonstrating its versatility and effectiveness.

1 Introduction

Phishing attacks are becoming increasingly widespread, posing a serious threat to cybersecurity today. In the third quarter of 2022, phishing attacks reached an all-time high, as reported by the Anti-Phishing Working Group (APWG) [3]. The APWG noted a total of 1,270,883 phishing attacks, representing the largest number of attacks in any quarter since its inception. Similarly, the Federal Bureau of Investigation's annual Internet Crime Report disclosed 847,376 complaints in 2021, leading to losses amounting to $6.9 billion [11]. The number of complaints and losses due to phishing attacks have consistently risen each year since 2017. Additionally, the annual Microsoft Digital Defense Report in 2022 revealed that over 710 million phishing emails are blocked every week [21]. Despite extensive efforts to research and tackle phishing through understanding, detection, and training, it remains a serious cybersecurity concern that demands attention.

Social engineering schemes that rely on deception and trust to exploit their victims often involve deceptive email addresses and messages, aimed at tricking unsuspecting individuals into interacting with counterfeit websites. One specific type of social engineering attack is phishing, which targets users by deceiving them into revealing valuable personal information, such as credit card numbers, passwords, and bank account details [3]. Phishing attacks rely on human behavior and psychological factors rather than software or server vulnerabilities, which can make certain users more susceptible to exploitation. While some phishing attempts may be easily identifiable, others can be highly convincing and difficult to detect. Spear phishing attacks,

which are tailored to target specific users with individualized content, pose an even greater challenge to detection efforts.

Businesses face significant risks when employees with access to valuable company data fall victim to phishing attacks. If the single sign-on (SSO) credentials of a senior employee are compromised, an attacker can gain access to substantial company resources. Whaling attacks, which target the most senior employees in a company, pose particularly significant threats, and their consequences can be devastating. To address these risks, many businesses provide their employees with phishing training and exercises aimed at improving their ability to detect and respond to phishing attacks. Some businesses conduct training without the employee's direct knowledge, such as simulated phishing emails sent at varying times and levels of complexity. When an employee clicks on a link in a simulated email, the business records the action and may require the employee to complete further training activities or watch training videos. Through increasingly sophisticated exercises, employees can improve their ability to identify actual phishing emails and reduce the risk of a successful attack.

As the complexity and frequency of cyber threats continue to grow, businesses are increasingly prioritizing the training of their employees to become more cyber aware. This includes educating personnel on identifying and responding to cyber threats, as well as the implementation of data protection and incident response policies and procedures. Human error, whether by employees or contractors, remains a significant cause of cybersecurity incidents, responsible for 56% of attacks, according to recent research [8]. The cost of such incidents can be substantial, with an average cost per incident estimated at $484,931 [8]. As a result, the

market for security awareness training solutions is expected to reach $10 billion per year by 2027. By investing in cyber awareness training, businesses can minimize the risks of cyber incidents and improve their overall security posture, ensuring the protection of their business and customers.

In order to increase the success rate of phishing attacks, it is important for attackers to create believable content that is tailored to the target. Simply copying and pasting a generic message is unlikely to be effective. In this article, we introduce a program that utilizes the OpenAI API to generate spear phishing emails based on information obtained from a target's LinkedIn profile. The aim of this program is to assist individuals in understanding how hackers can create persuasive spear phishing emails to increase the chances of a successful attack. Additionally, this program can be used to evaluate an organization's security measures. The program extracts relevant information from a target's LinkedIn profile and stores it in a database. The user can then select which information to use in the email, and the program generates 11 different emails with varying levels of "temperature" or persuasiveness. The user can then evaluate each email and select the one they believe is most convincing for their target.

The rest of the paper is organized in the following way. Section 2 looks at related work associated with phishing creation, detection, and awareness and training. Section 3 looks at the program overview and its functions. Section 4 describes the program design and implementation of the program. Section 5 discusses results from our user study. Section 6 is the conclusion of the paper with a discussion on future implications.

2 Related Work

Numerous research studies have been conducted to identify the characteristics of phishing emails. By analyzing the strategies used by experts in detecting these fraudulent messages, individuals can adopt similar cognitive processes to enhance their own ability to identify them [31]. Additionally, identifying the common errors made by non-expert end users can increase awareness of the email features that should be considered to detect a phishing attack [16, 30]. One study specifically examined the physical appearance of phishing web pages and how it can aid in detection [33]. Furthermore, there are efforts to develop automated detection algorithms that can intercept these emails before they reach the end user, which employ various technical detection methods such as deep learning, data mining, natural language processing, and URL analysis [2, 4, 10, 19, 23, 25, 26, 28, 37].

Several review articles offer comprehensive overviews of detection research and its future directions [1, 6, 9, 12]. The various techniques used for detection can be broadly categorized into Blacklist-based, Content-based, Heuristic-based, and Fuzzy rule-based approaches [1]. Blacklist methods rely on tools like Google Safe Browsing, OpenPhish, and PhishTank [34] to identify malicious URLs and warn users when they attempt to access them. While effective in detecting known phishing pages, these methods struggle to keep up with the continuous creation of new malicious sites that evade detection. Content-based approaches, on the other hand, scrutinize the content of web pages, including text, images, and frames [19, 25, 39]. However, these approaches can be resource-intensive due to the large number of potential features required for detection [1]. Heuristic-based approaches use features like footer links with

null value, zero links in the body of HTML, copyright content, title content, and website identity to distinguish between safe and malicious URLs [13, 32, 39]. Fuzzy rule-based approaches also leverage features like images, text, and frames, but use advanced learning systems like the Adaptive Neuro-Fuzzy Inference System (ANFIS) [26].

Despite advancements in detection methods, phishing attacks are still causing significant harm and financial loss to both individuals and businesses. Attackers are continually improving their techniques and making their attacks more sophisticated. Therefore, the end user is the ultimate line of defense. Their actions determine whether they click on a link, visit a malicious website, and share sensitive information with the attacker. It is crucial to train users on how to identify phishing emails to combat this issue effectively. By doing so, we can provide an additional layer of protection against phishing attacks. Studies investigating the human factors that contribute to users falling victim to phishing attacks can offer valuable insights for developing effective training materials [14, 18, 20, 27, 36]. Empirical evidence from case studies involving simulated phishing attacks and training can identify the characteristics that make users most susceptible to these attacks.

We found that *PhishGen* is one of the few programs that offers similar functionality to the program we were researching that gives us data we could use to create spear phishing emails [15]. This program generates phishing email addresses by requiring the user to provide a domain and their LinkedIn account credentials. *PhishGen* then identifies a company based on the domain and retrieves information on all employees from LinkedIn who work for that company. The employee data is saved into a csv file that includes email addresses, source, warning, search

term, occupation, profile heading, full name, first and last name, identifier, location, industry, connection, premium, job seeker, influencer, school, degree, field of study, school start, and school end. With this program, users can enter a company domain and retrieve a list of email addresses associated with that company on LinkedIn. This can be useful for email phishing training when a list of recipients is needed. While *PhishGen* is quite different from our program, it serves as a valuable tool for gathering intelligence on email phishing.

GoPhish is an open-source phishing framework that allows organizations to test their susceptibility to phishing attacks [38]. It is written in the Go programming language and can be easily downloaded and run. The framework enables users to create phishing email campaigns, either by designing or importing email templates, importing mailing lists, and including landing page URLs. *GoPhish* also tracks the campaign's progress by monitoring the email open rate, link clicks, and credential submissions on the landing page. Results can be exported to a CSV or Excel file for security experts to analyze and generate reports. It is a highly functional tool that is widely used by industry professionals.

KnowBe4 is a well-known service widely used for employee security training. It is a popular integrated Security Awareness Training and Simulated Phishing platform with a global user base of over 50,000 organizations [17]. *KnowBe4* specializes in educating and training employees to protect against social engineering attacks, and also provides compliance training as required. Organizations can opt to work with KnowBe4 directly or engage third-party consulting firms for testing. Since their services are designed for businesses, the prices vary based on seat and tier.

3 Program Overview and Evaluation

The Python program is designed to create spear phishing emails based on LinkedIn information using OpenAI API following a systematic approach to generate convincing emails for a specific target. The user would manually input the URL for the LinkedIn profile that they are targeting. Subsequently, the program web scrapes the required information from the URL and stores it in a database collection. The program generates 6 emails with varying degrees of persuasiveness or "temperature" and displays them through a graphical user interface. The user can then sort the emails based on their level of believability and save the data in another collection within the database by clicking on the "Insert Data" button. The program not only helps users understand how hackers can create believable spear phishing emails but also has potential ethical applications in assessing an organization's security measures.

The rest of this section shows all the features that are used in the process to create the spear phishing emails. It is arranged in subsections with Section 3.1 detailing how we get our URL, Section 3.2 detailing how we scrape the information off the target's LinkedIn profile, Section 3.3 detailing how the emails are created from the web scraped information, and Section 3.4 detailing the database implementation.

3.1 Manual LinkedIn Input

Our current function simply asks a user to input a URL for a target's LinkedIn profile and then passes this to our next function which scrapes the profile for all the relevant data. We used to have a function that would do a Google search and return the top five results based off the

person’s name and the organization they are affiliated with. These links were also restricted to only LinkedIn websites. We chose to remove this as sometimes you would have to manually check the links anyways due to the target not creating a custom LinkedIn URL which makes it much easier to figure out which link is theirs. We also ran into an issue with the Google search API throwing us errors that we could not figure out.

3.2 Scrape LinkedIn Profile

This function operates as a sophisticated conductor, systematically extracting crucial details from LinkedIn profiles and seamlessly integrating them into a MongoDB database, with a specific focus on MongoDB Atlas for streamlined cloud-based database management. At its core, the function harnesses the Chrome webdriver, a versatile tool designed for automating web browsing tasks. Initial configurations include setting up options for the webdriver, considering both headless mode and performance optimizations. Subsequently, the webdriver navigates to the specified LinkedIn profile URL, employing practices such as scrolling to the page's bottom to ensure the thorough loading of all page elements. Following this, the function captures the HTML source code, a pivotal step before responsibly closing the webdriver to release system resources. The HTML source code undergoes meticulous parsing using BeautifulSoup, a Python library tailored for web scraping. The primary objective is to locate the script tag housing JSON-LD data, a format frequently employed for expressing linked data in JSON. The function adeptly extracts key fields of interest, covering the individual's name, job title, current company,

location, education, and previous job positions. Importantly, the 'About' text is also extracted from the page source, with a careful focus on removing any extraneous LinkedIn profile information or default about sections. Figure 1 provides a visual representation of the JSON-LD data extracted during this process. A notable feature of the function lies in its proactive handling of potential security measures on the LinkedIn page. If the title suggests security measures, the function prompts the user for a manual bypass, ensuring adaptability and circumvention of any security measures in place. Post-extraction, the function meticulously organizes the obtained information into a dictionary, appropriately named profile_data. This data seamlessly integrates into a MongoDB database using PyMongo, with a specific emphasis on MongoDB Atlas over MongoDB due to the robust cloud backup capabilities. The designated collection within the database is suitably titled 'mycollection'. To confirm the successful extraction and offer insights, the function prints the information to the console. Ultimately, the function concludes by returning the profile_data dictionary, playing a pivotal role in subsequent stages of the comprehensive process. In cybersecurity or computer science contexts, the function's adept handling of potential security measures positions it as a valuable tool for researchers or analysts conducting web-based reconnaissance, where automated and secure extraction of profile data from platforms like LinkedIn is essential for understanding potential threats or assessing the security landscape.

Figure 1: JSON-LD Data

3.3 Write Emails

The write_email function orchestrates a user-driven process for crafting personalized emails through the OpenAI API and storing relevant data in a MongoDB database. Taking a 'profile_data' dictionary as its sole parameter, the function guides users to select specific variables of interest from the 'profile_data' dictionary, consolidating them into a list called 'variables.' These selected variables serve as crucial components for tailoring the content of a targeted email. Utilizing the OpenAI API, the function dynamically generates an email prompt

that seamlessly integrates essential details from the 'profile_data' dictionary and the user-selected variables. The resulting prompt is then presented on the console, offering users a comprehensive preview of the email content. To enhance user interaction, the function establishes a Tkinter window featuring text widgets for potential email responses, each accompanied by a corresponding combobox. These comboboxes enable users to assign numerical ratings on a scale of 1 to 6, indicating the perceived persuasiveness of the corresponding email response. To diversify response styles, the text widgets are generated with varying temperatures—0.0, 0.2, 0.4, 0.6, 0.8, and 1.0—each representing distinct levels of creativity and fluency in the generated content. Notably, the text widgets randomly select temperatures, ensuring that users cannot discern a specific order in which creativity is heightened. Subsequently, the function compiles a dictionary, 'email_data,' encompassing critical components such as the 'variables' list, the dynamically generated 'prompt' string, and an 'email_data_list.' The 'email_data_list' comprises dictionaries containing the email response text and the corresponding user-assigned ratings on the 1 to 6 scale. This comprehensive 'email_data' dictionary is then inserted into a MongoDB database collection named 'emailinformation.' The user's input, specifically the assigned ratings, undergoes real-time updates within the 'email_data_list' upon the user triggering the "Insert Data" button in the Tkinter window. This final step completes the process, persistently storing both the email content and user evaluations in the MongoDB collection for subsequent analysis

or utilization. Figure 2 provides a visual representation of the graphical user interface displaying the generated emails.

Our Tkinter GUI experiences a build time of approximately one and a half minutes, primarily attributed to the duration of the OpenAI API calls. The process involves a total of six API calls, averaging around 15 seconds per call. Each API call corresponds to the generation of an email, and the GUI is designed to await the completion of all calls before fully constructing. This approach ensures that the GUI is populated with the generated content, leading to the observed build time.

This function could be particularly beneficial in cybersecurity or computer science contexts, where tailored and dynamic email content generation, along with real-time user evaluations, can be pivotal for security awareness training or testing system responses to socially engineered threats. The integration of diverse response styles and the secure storage of email

information in MongoDB align with the needs of these fields, contributing to its potential usefulness in practical applications.

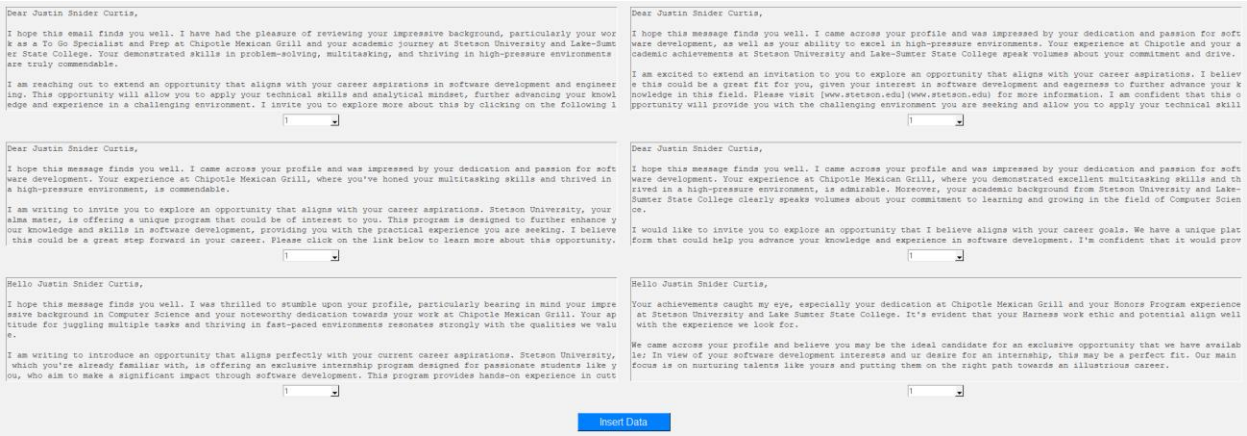

Figure 2: GUI to display created spear-phishing emails

3.4 Database Implementation

The database utilized for this project was MongoDB, hosted on MongoDB Atlas. MongoDB is a versatile, document-oriented NoSQL database engineered for the storage, retrieval, and management of semi-structured and unstructured data. In contrast to traditional relational databases, MongoDB employs a flexible, JSON-like document structure with dynamic schemas, facilitating quicker and more scalable data management, as well as agile development processes. BSON (Binary JSON) is employed by MongoDB to represent its documents and data structures

in a binary-encoded format [24]. In our program, the connection to MongoDB is established, and data is added using the PyMongo library [5]. The database comprises two collections, namely 'mycollection' and 'emailinformation.' The 'mycollection' collection stores LinkedIn profile information, including name, job title, current company name, location, education, previous jobs, the about section, and the URL. On the other hand, the 'emailinformation' collection stores crucial project data such as the variables used for OpenAI API prompts, the prompt employed, and a dictionary named 'email_data_list.' Within this list, we store the temperature, the email generated based on that temperature, and the user-selected rating within the Tkinter GUI. For a detailed visualization of the database design, please refer to Figure 4.

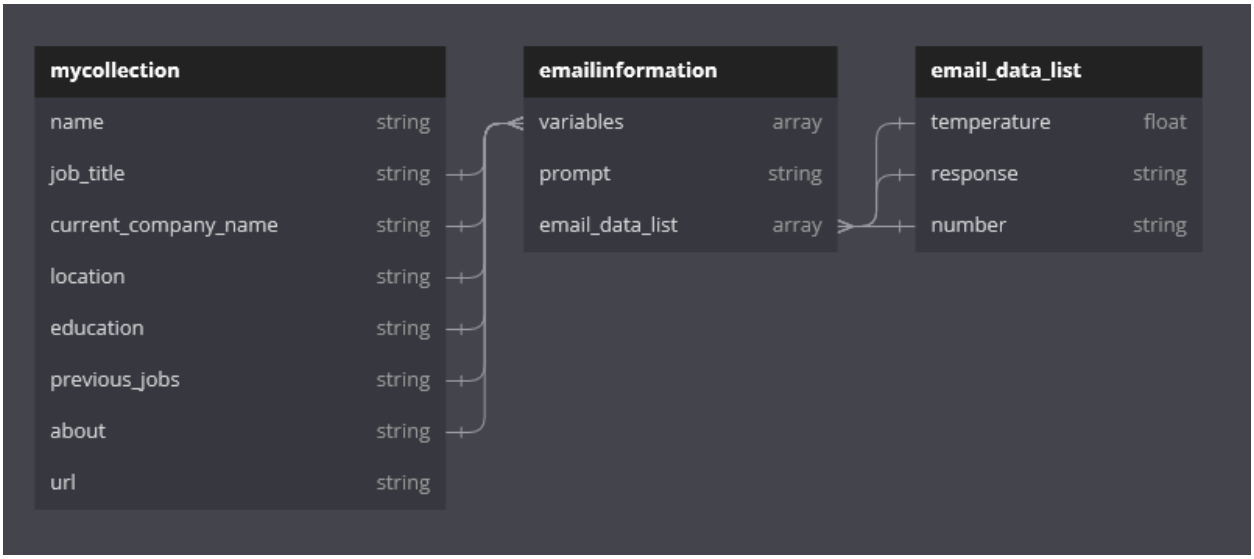

Figure 3: Database Design

4 Program Design & Implementation

Once the idea for the program was established, the first task was to figure out how to get the information off a target's LinkedIn profile. The original idea for this was to use the LinkedIn API since it allows developers to access and retrieve user profile data such as name, job title, work experience, education, and skills [22]. We ultimately did not use the LinkedIn API as we had to create an application and register it with the LinkedIn Developer Program. This would make it to where we must use OAuth 2.0 authentication and authentication protocol to gain access to the data we need. This would require the target user to log in to the application and grant permission to the application to gain access to the data. If a user grants permission, LinkedIn generates an Access Token and sends it back to the application. From there the application uses the Access Token to make API requests to LinkedIn's servers and returns the user's data in a standardized format, such as XML or JSON.

Instead of doing this we looked for alternatives to gain access to the data that we wanted without alerting the target that we were doing it. For this we looked at LinkedIn profiles without being logged into LinkedIn and viewed the page source in Google Chrome by right clicking and selecting "View page source". From there we looked at the HTML format of the webpage to see what data was available to scrape. This is when we found the "application/ld+json" script type that contained all the data that we needed to gain information we could use against our target. This script contains information on the location the target resides in, where they have worked, where they have gone to school, their name, job title, where they currently work, the about section, awards they have received, what languages they speak, and more. Using this

information, we can craft believable content for a target to make it more likely that they click on the link in our email. We also found that we can view LinkedIn profiles while in incognito mode on Google Chrome to ensure we didn't alert the target.

To scrape the information off the target LinkedIn profile, we chose to use Selenium and BeautifulSoup4. Selenium was chosen because it can automate web browser interactions, automate tasks, and supports multiple web browsers, which makes it perfect for web scraping [35]. In our program we have Selenium's webdriver set to Google Chrome running in headless mode, which is a way to make the web scraping invisible without the need for a GUI. BeautifulSoup4 is a Python library that allows developers to parse HTML and XML documents. [7] We also added an option to turn headless mode on or off, which we'll talk more about why in the next sentence. The main reason we chose BeautifulSoup4 was for its tools for searching and navigating HTML and XML documents. With both of these Python packages, scraping the information off the LinkedIn profile was much easier compared to other methods.

Sometimes we ran into an issue with web scraping where LinkedIn would ask us to complete a security check or log in, which is why we had to toggle headless mode on or off. We had to see what the issue was when loading LinkedIn, which was a problem we ran into in our first iteration of the program. To combat this, we checked the page title for the word "security" and then checked the page source for the words "sign in". We then would print which issue we ran into and manually addressed them before pressing enter to continue the program. The security check was a simple task of rotating an animal of some sort to point in the same direction as an arrow. For the sign in issue, we found that we had to sign in, then sign out, paste the link of

our target again into the address bar, navigate to it, and then go back to the program and hit enter. We're not sure if this notifies the user or not of who logged in to view their account, but this could be avoided by creating a dummy account or a batch of dummy accounts. The program would then scroll all the way to the bottom of the page, get the page source, and then close our Google Chrome window. We would then scrape the relevant information from "application/ld+json".

To store our web scraped information, the created spear phishing emails, and other relevant data, we chose to use MongoDB [24]. Specifically using MongoDB Atlas so that we had a cloud backup of our data in case our personal machine were to go down for any reason. For this program we specifically used PyMongo, which is a Python library that provides a convenient and easy-to-use interface for interacting with MongoDB. PyMongo also has some features that we found useful such as creating, reading, updating, and deleting.

The next step of the program focused on how we can use this information to craft the emails using OpenAI API. We initially began by creating prompts and hardcoding the information to see what the emails might look like. For our current iteration of the program, we thought the best idea would be to invite our target to an opportunity. We felt that a broader prompt would be best for the user study. We could make multiple different prompts, but we felt it was best to stick with one for the time being. Once we had a prompt that was generating believable emails a majority of the time, we switched our focus over to how to let a user find a target and craft the spear phishing emails.

Our current program starts by asking the user to provide the URL of the LinkedIn profile of the target they want to create emails for. Once this is inputted, the program passes this to our next function which scrapes the LinkedIn profile for all the information that is needed to create the spear-phishing emails and stores it in a MongoDB database inside of a collection named 'mycollection'. The next step of the program asks the user to select what group of data they'd like to use. Currently we're just using all of the data that we collect from a profile since in the real world we believe that an attacker would use all of the information that they could get in order to create the emails. The program then appends this information to the end of our prompt which is then used by the OpenAI API in the next function of the program.

The final part of the program creates six different emails with varying temperatures from 0 to 1 going up by .2 for each using OpenAI API. We decided to use OpenAI API for generating these emails because it provides state-of-the-art language models capable of generating highly realistic and convincing text. The API leverages cutting-edge machine learning techniques and vast amounts of training data to generate high-quality text output [40]. Currently we are using the GPT-4 model. We then created a GUI using Tkinter and added our spear phishing emails to the window inside of text widgets. The reason we chose Tkinter is because it is a standard GUI library for Python, it is easy to learn, and has cross-platform support [29]. The program also adds combo boxes with each email for the user to rate how convincing the email is. The data from the combo boxes is inserted into our MongoDB after a user presses the "Insert Data" button located at the bottom of the window. Figure 5 shows the prompt we are currently utilizing to create the spear-phishing emails.

```
# Prompt for the OpenAI API
prompt = f"Write an email to {profile_data.get('name', '')} inviting them to an opportunity, sign off" \
         f" the email with a random name, and embed this link 'www.stetson.edu'. Please keep it about" \
         f" two paragraphs long at maximum and do not include a subject line.; here's some information " \
         f"about {profile_data.get('name', '')}:"

for variable in groups:
    prompt += f"- {variable}\n"
```

Figure 4: OpenAI API Prompt

We originally wanted to test with multiple different prompts based on what a participant would tell us they were looking for or would be most willing to respond to. We thought it would be best, however, to stick to a broader prompt for the user study. There are numerous prompts and roles capable of achieving the same functionality as our current program. Showing our prompt allows for replication and ideas for how to create a prompt that is similar that could produce the same effectiveness.

The current content we have set for the “system” role in the ChatGPT completion code is “You are an assistant who specializes in writing emails and convincing clients to click a link.”. The first part of our prompt tells ChatGPT to write an email to our participants and invite them to an opportunity. We originally were thinking of inviting them to a possible job, career fair, or internship, but figured these were too specific for the user study so changed to something broader that would work for all participants. The second part where it tells the AI to sign off with a random name is to produce a bit more of a believable email. People usually trust an email from

someone if it is signed off by someone. We did find that with lower temperature it would default to John Doe. We embedded a link because we wanted to see if that would make participants believe the email more by wanting to click on it or reply. The next sentence on length and subject line were to keep the emails a bit more concise and only focused on email content. We found that not specifying the length of the email would create long emails and subject lines were hardly ever believable. The last sentence of the prompt simply passes all of the information that we scraped from the profile into the prompt, which ChatGPT then uses when crafting the spear-phishing emails.

To initiate our research, we started by submitting our project to Stetson University's Institutional Review Board (IRB), tasked with ensuring the welfare of human subjects in university research. Upon receiving approval, we gained clearance to begin our data collection. We recruited participants from the computer science and math department as well as from some outside participants. This resulted in the involvement of 10 participants in our study.

For IRB approval we had to take a CITI Social and Behavioral Research course, submit a protocol description form, and an informed consent form for participants. The protocol description form asked for a summarization of the purpose of our research and why it needed to be done, hypothesis, number of participants expected, eligibility requirements, how we planned on recruiting participants, compensation (which we did not have), formal relationship we the researcher may have had with participants, how we'll protect participants' privacy, and our methods to be used in the research. In the informed consent form we had to answer the following questions: what is the study about and why are we doing it, what will happen if you take part in

this study, how could you benefit from this study, what risks might result from being in this study, how will we protect your information, what will happen to the information we collect about you after the study is over, how will we compensate you for being part of the study.

During the implementation of our study, participants were presented with an informed consent form. Through a Google form, they were required to input their Stetson University email and affirm their agreement to participate by selecting a "Yes" button, confirming their understanding of the consent form. Following this, participants were instructed to locate and input their LinkedIn profile URL. The program then initiated data scraping, with a final confirmation prompt before proceeding. On occasion, a manual bypass of a security check was necessary, or we were prompted to log in where we simply logged out afterwards to continue data collection. Following this, six distinct spear-phishing emails were generated using the OpenAI API and the Tkinter GUI, with the generation process taking approximately one and a half minutes. The entire process, from obtaining consent to email generation, took around 5 minutes. Participants were then directed to review each generated email and rank them from 1 to 6 based on their perceived persuasiveness, with 1 denoting the most convincing and 6 the least convincing. Duplicate rankings were not permitted. Subsequently, the collected rankings were stored in the MongoDB database, along with the corresponding email temperature, following the completion of the ranking process, by activating the "Insert Data" button. In total this process took about ten minutes per participant.

5 Results

Our study aimed to determine the temperature at which spear-phishing emails were most convincingly crafted in comparison to others. This research is significant in the realm of cybersecurity as it seeks to uncover the environmental variables influencing the success of spear-phishing attacks. Identifying the optimal temperature conditions for crafting convincing emails is pivotal for developing targeted countermeasures and enhancing overall digital security. Since the use of AI is becoming more and more prominent in the creation and detection of everything today, understanding how temperature influences results is important. As attackers continuously adapt their strategies, our study delves into a specific factor that can influence their tactics—leveraging temperature as a potential factor in the effectiveness of spear-phishing. By unraveling the relationship between temperature and how it affects the persuasiveness of email generation, we aim to provide practical insights that can guide the development of adaptive security measures. This research could serve as a resource for ongoing discussions on AI-driven cyber threats, anticipating future challenges and informing cybersecurity strategies.

While our study initially aimed to include a more substantial participant pool, consisting ideally of 20 to 50 individuals, we encountered unforeseen challenges that limited our recruitment to 10 participants within a constrained time frame. The development of our program, essential for the study's execution, encountered unexpected hurdles that extended our timeline. Additionally, security concerns arising from interactions with LinkedIn presented unanticipated obstacles during the participant recruitment process. Despite these challenges, the insights gleaned from the gathered data provide valuable preliminary findings. As we acknowledge the

limitations imposed by our constrained participant numbers, future iterations of this research will benefit from a more extensive and diverse participant cohort, ensuring a more robust and comprehensive understanding of the dynamics explored in our study.

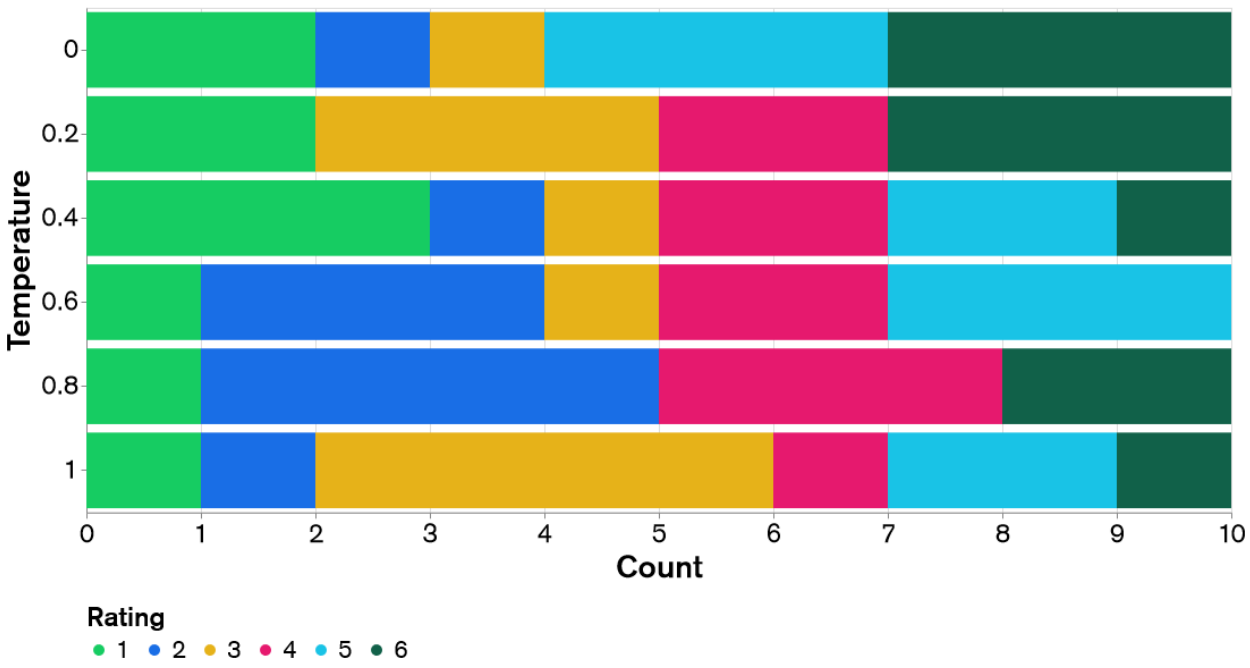

Figure 5: Stacked bar chart depicting the overall frequency of each rating at various temperature levels

In Figure 5, the cumulative count of each rating is depicted across a range of temperature levels from 0 to 1, increasing in increments of 0.2. The analysis of individual temperature points revealed notable variations in user ratings. At Temperature 0, the distribution of ratings was diverse, with two instances receiving a rating of 1, one instance rated at 2, another at 3, three instances at a rating of 5, and three instances at a rating of 6. This diversity underscores the

nuanced impact of Temperature 0 on user assessments, showcasing the variability in the perceived effectiveness of the generated content. Moving on to Temperature 0.2, the examination unveiled a unique pattern in the rating distribution. Two instances received a rating of 1, three instances were rated at 3, two instances at a rating of 4, and three instances at a rating of 6. This pattern emphasizes the varied responses to content generated at Temperature 0.2, suggesting potential nuances in the effectiveness of spear-phishing emails at this specific temperature level. Similarly, at Temperature 0.4, the analysis revealed a diverse distribution, including three instances with a rating of 1, one instance at a rating of 2, one instance at a rating of 3, two instances at a rating of 4, two instances at a rating of 5, and one instance with a rating of 6. This varied response indicates nuanced perceptions of spear-phishing email effectiveness at Temperature 0.4, underscoring the importance of environmental conditions in influencing user assessments. Temperature 0.6 displayed a distinctive distribution, encompassing one instance with a rating of 1, three instances at a rating of 2, one instance at a rating of 3, two instances at a rating of 4, and three instances at a rating of 5. This pattern suggests a nuanced range of perceived effectiveness for spear-phishing emails generated at Temperature 0.6, highlighting the impact of this specific environmental condition on user evaluations. At Temperature 0.8, the examination revealed a distribution with one instance receiving a rating of 1, four instances at a rating of 2, three instances at a rating of 4, and two instances at a rating of 6. This rating profile suggests a varied response to spear-phishing emails generated at Temperature 0.8, emphasizing the nuanced impact of this specific environmental condition on user assessments. Finally, at Temperature 1, the analysis unveiled a distribution encompassing one instance with a rating of 1,

one instance at a rating of 2, four instances at a rating of 3, one instance at a rating of 4, two instances at a rating of 5, and one instance at a rating of 6. This diverse rating profile underscores the nuanced nature of user perceptions toward spear-phishing emails generated at Temperature 1, emphasizing the importance of this specific environmental condition in shaping user evaluations.

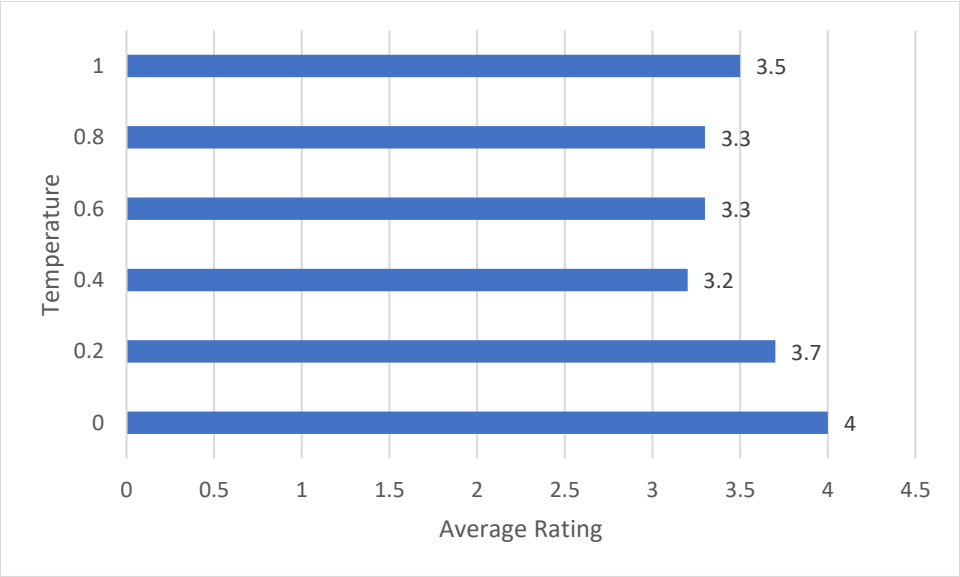

Figure 6: Average rating for temperatures

In Figure 6, the average ratings offer a straightforward insight into how different temperatures influenced the perceived persuasiveness of spear-phishing emails, using a scale from 1 to 6. At Temperature 0, the average rating was 4, suggesting a moderate level of perceived effectiveness. Temperature 0.2 showed a slightly lower average of 3.7, indicating a slight increase in persuasiveness. Temperature 0.4 had an average of 3.2, pointing to a shift towards a more convincing perception. At Temperatures 0.6 and 0.8, the averages remained

steady at 3.3, signaling a consistent trend. Notably, at Temperature 1, the average rating increased slightly to 3.5. It's important to note that, in our ranking system participants were asked to rate emails from most to least convincing, lower average scores reflect more convincing emails. Overall, spear-phishing emails that were generated with temperatures ranging from 0.4 to 0.8 resulted in the most convincing emails according to our study.

We opted not to conduct a statistical analysis for our study due to the limited number of participants involved. With a participant pool of 10, we recognized that the sample size might not be sufficient to yield statistically robust results. A larger and more diverse dataset is typically required for meaningful statistical analyses and generalizable conclusions. While our current findings provide valuable insights into the trends observed, we acknowledge the inherent limitations associated with the smaller sample size. Future iterations of our research, with a more extensive and varied participant cohort, would allow for a more rigorous statistical examination, enhancing the reliability and generalizability of our conclusions.

It's crucial to acknowledge the potential impact of personal bias on our study, given the subjective nature of individuals' preferences in email content. The variety of personal tastes and preferences among participants may have influenced their assessments of the spear-phishing emails generated in our study. Recognizing that people's perceptions of what constitutes a convincing email can be highly subjective, our limited set of prompts may not have fully accommodated the diverse range of preferences within the participant group. A more expansive selection of prompts might have allowed for a broader exploration of individual preferences and

biases, enhancing the study's capacity to capture a more comprehensive spectrum of user perceptions.

The implications of our results hold significance in enhancing the broader understanding of the relationship between temperature and the persuasiveness of spear-phishing emails. The observed trends, particularly the peak in convincing emails within the temperature range of 0.4 to 0.8, underscore the dynamic nature of environmental factors on user perceptions. However, it's essential to acknowledge the study's limitations, particularly the potential influence of personal bias on participants' assessments, as individual preferences in email content can vary significantly. This recognition prompts a thoughtful reflection on the scope of our findings. The key takeaway lies in the nuanced impact of temperature settings on perceived email effectiveness, providing valuable insights for cybersecurity strategies. Future research could delve deeper into mitigating personal bias, exploring a more extensive range of prompts to better accommodate individual preferences. Additionally, a larger participant pool would contribute to more robust statistical analyses, further enhancing the generalizability of our conclusions. In summary, our study serves as a foundation for understanding the interplay between environmental variables and spear-phishing email effectiveness, paving the way for future investigations in this evolving field.

6 Conclusion

In summary, the presented program, leveraging the OpenAI API to generate spear-phishing emails based on LinkedIn information, emerges as a robust and versatile tool for crafting convincing emails tailored to specific individuals or organizations. Employing web scraping techniques, the program extracts pertinent data from LinkedIn profiles, stores it in a database, and employs the OpenAI API to generate multiple emails with varying temperature settings, providing users with a diverse range of options. The program features a user-friendly graphical interface enabling users to rank emails based on their persuasiveness, with these rankings then stored for future reference. In essence, this tool proves invaluable in the realm of cybersecurity, empowering professionals to assess the vulnerability of individuals and organizations to phishing attacks. By doing so, it equips them with insights to fortify networks and safeguard critical data effectively.

Our investigation into the correlation between temperature settings and the efficacy of spear-phishing emails has yielded noteworthy insights. Notably, the temperature range spanning 0.4 to 0.8 emerged as a focal point for generating convincingly deceptive emails, suggesting a nuanced relationship between environmental variables and user perceptions. However, it's crucial to acknowledge the study's limitations, including a restricted participant pool and potential personal biases influencing perceived persuasiveness. Despite these constraints, our findings illuminate the dynamic nature of spear-phishing email effectiveness, emphasizing the significance of considering environmental conditions in cybersecurity. These observed trends underscore the need for future research endeavors with broader participant cohorts and

diversified prompt variations to refine our understanding of this intricate relationship. In essence, this study provides a foundational perspective on the influence of temperature settings on spear-phishing emails, offering valuable insights for ongoing discussions and future investigations in this rapidly evolving field.

For future developments, there is a prospect of expanding the range of prompts to incorporate a prompt selection feature. This would be particularly relevant considering that some LinkedIn profiles indicate openness to job opportunities or hiring. Such information could influence the choice of spear-phishing emails, tailoring them to specific recipients. For instance, sending a student an email about internship or research opportunities aligning with their interests could enhance the likelihood of engagement with the provided link or attachment. This approach would not only increase click-through rates but also facilitate more precise and targeted phishing campaigns.

One idea that could lead to interesting data would be to have ChatGPT rate the emails believability it created itself and then compare this to the data we get from our study. Comparative analysis between AI ratings and human study results would shed light on strengths and weaknesses ChatGPT when it comes to generating email content. Expanding on this idea, we could train another AI specifically on email datasets that focus on content structure to discern patterns that are related to believability. Using this we could put the feedback from this AI back into ChatGPT with the original email to create an even better email.

Another potential avenue for future enhancements involves integrating PhishGen with LinkedIn to generate a comprehensive list of employees within a company. Subsequently,

personalized spear-phishing emails could be crafted for each individual [15]. These emails, when employed in conjunction with GoPhish [15], could be part of a phishing campaign. Tracking features within GoPhish could then be employed to monitor email opens, link clicks, and any submitted credentials. Such data would be invaluable in assessing the effectiveness of an organization's phishing training program and evaluating the efficiency of spear-phishing endeavors facilitated by the OpenAI API.

REFERENCES

[1] AlMaha Abu Zuraiq and Mouhammd Alkasassbeh. 2019. Phishing detection approaches. In 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS). IEEE, Amman, Jordan, 1–6.

[2] Anindita Khade and S Shinde. 2014. Detection of phishing websites using data mining techniques. International Journal of Engineering Research and Technology 2, 12 (2014), 3725–3729.

[3] Apwg APWG. 2022. Phishing activity trends report: 3rd quarter 2022. Anti-Phishing Working Group (2022).

[4] Atharva Deshpande, Omkar Pedamkar, Nachiket Chaudhary, and Swapna Borde. 2021. Detection of PhishingWebsites using Machine Learning. INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT) Volume 10 (2021).

[5] Atlas MongoDB, Inc. "Pymongo Tutorial: Mongodb and Python." MongoDB, 2022, https://www.mongodb.com/languages/python/pymongo-tutorial.

[6] Ayman El Aassal, Shahryar Baki, Avisha Das, and Rakesh M Verma. 2020. An indepth benchmarking and evaluation of phishing detection research for security needs. IEEE Access 8 (2020), 22170–22192.

[7] Beautiful Soup. 2022. Beautiful Soup Documentation. https://beautiful-soup-4.readthedocs.io/en/latest/

[8] Braeu, David. "Security Awareness Training Market to Hit $10 Billion Annually by 2027." Cybercrime Magazine, 20 Apr. 2022, https://cybersecurityventures.com/security-awareness-training-market-to-hit-10-billion-annually-by-2027/.

[9] Brij B Gupta, Aakanksha Tewari, Ankit Kumar Jain, and Dharma P Agrawal. 2017. Fighting against phishing attacks: state of the art and future challenges. Neural Computing and Applications 28, 12 (2017), 3629–3654.

[10] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-scale automatic classification of phishing pages. (2010).

[11] FBI. 2021. Federal Bureau of Investigation Internet Crime Report 2021. (2021). https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf

[12] Gaurav Varshney, Manoj Misra, and Pradeep K Atrey. 2016. A survey and classification of web phishing detection schemes. Security and Communication Networks 9, 18 (2016), 6266–6284.

[13] Gitanjali Baral and Nalin Asanka Gamagedara Arachchilage. 2019. Building confidence not to be phished through a gamified approach: conceptualizing user's self-efficacy in phishing threat avoidance behaviour. In 2019 Cybersecurity and Cyberforensics Conference (CCC). IEEE, Melbourne, Australia, 102–110.

[14] Hossein Abroshan, Jan Devos, Geert Poels, and Eric Laermans. 2021. A phishing mitigation solution using human behaviour and emotions that influence the success of phishing attacks. In Adjunct Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization. Utrecht, Netherlands, 345–350.

[15] jamesm0rr1s. "Phishing-Email-Address-Generator: PhishGen." *GitHub*, 15 July 2019, https://github.com/jamesm0rr1s/Phishing-Email-Address-Generator.

[16] Jingguo Wang, Yuan Li, and H Raghav Rao. 2017. Coping responses in phishing detection: an investigation of antecedents and consequences. Information Systems Research 28, 2 (2017), 378–396

[17] KnowBe4. "Security Awareness Training." *KnowBe4*, 2022, https://www.knowbe4.com/.

[18] Lennart Jaeger and Andreas Eckhardt. 2021. Eyes wide open: The role of situational information security awareness for security-related behaviour. Information Systems Journal 31, 3 (2021), 429–472.

[19] Mahmood Moghimi and Ali Yazdian Varjani. 2016. New rule-based phishing detection method. Expert systems with applications 53 (2016), 231–242.

[20] Matthew L Jensen, Michael Dinger, Ryan T Wright, and Jason Bennett Thatcher. 2017. Training to mitigate phishing attacks using mindfulness techniques. Journal of Management Information Systems 34, 2 (2017), 597–626.

[21] Microsoft. 2022. Microsoft Digital Defense Report 2022. https://www.microsoft.com/en-us/security/business/microsoft-digital-defense-report-2022

[22] Microsoft. 2023. LinkedIn API Overview. https://learn.microsoft.com/en-us/linkedin/

[23] Mohammad Mehdi Yadollahi, Farzaneh Shoeleh, Elham Serkani, Afsaneh Madani, and Hossein Gharaee. 2019. An adaptive machine learning based approach for phishing detection using hybrid features. In 2019 5th International Conference on Web Research (ICWR). IEEE, Tehran, Iran, 281–286.

[24] MongoDB, Inc. "Why Use Mongodb and When to Use It?" MongoDB, 2022, https://www.mongodb.com/why-use-mongodb.

[25] Moruf Akin Adebowale, Khin T Lwin, and Mohammed Alamgir Hossain. 2020. Intelligent phishing detection scheme using deep learning algorithms. Journal of Enterprise Information Management (2020).

[26] Moruf A Adebowale, Khin T Lwin, Erika Sanchez, and Alamgir M Hossain. 2019. Intelligent web-phishing detection and protection scheme using integrated features of Images, frames and text. Expert Systems with Applications 115 (2019), 300–313.

[27] Mousa Jari. 2022. An Overview of Phishing Victimization: Human Factors, Training and the Role of Emotions. arXiv preprint arXiv:2209.11197 (2022).

[28] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. 2019. Machine learning based phishing detection from URLs. Expert Systems with Applications 117 (2019), 345–357.

[29] Python Software Foundation. 2023. Tkinter – Python interface to Tcl/Tk. https://docs.python.org/3/library/tkinter.html
[30] Rick Wash, Norbert Nthala, and Emilee Rader. 2021. Knowledge and Capabilities that {Non-Expert} Users Bring to Phishing Detection. In Seventeenth Symposium on Usable Privacy and Security (SOUPS 2021). online, 377–396.
[31] Rick Wash. 2020. How experts detect phishing scam emails. Proceedings of the ACM on Human-Computer Interaction 4, CSCW2 (2020), 1–28.
[32] Routhu Srinivasa Rao and Syed Taqi Ali. 2015. Phishshield: a desktop application to detect phishing webpages through heuristic approach. Procedia Computer Science 54 (2015), 147–156.
[33] Sadia Afroz and Rachel Greenstadt. 2011. Phishzoo: Detecting phishing websites by looking at them. In 2011 IEEE Fifth International Conference on Semantic Computing. IEEE, Palo Alto, California, USA, 368–375.
[34] Simon Bell and Peter Komisarczuk. 2020. An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In Proceedings of the Australasian Computer Science Week Multiconference. Melbourne VIC, Australia, 1–11.
[35] Software Freedom Conservancy. 2022. Selenium. https://www.selenium.dev/
[36] Steve Sheng, Mandy Holbrook, Ponnurangam Kumaraguru, Lorrie Faith Cranor, and Julie Downs. 2010. Who falls for phish? A demographic analysis of phishing susceptibility and effectiveness of interventions. In Proceedings of the SIGCHI conference on human factors in computing systems. Atlanta, Georgia, USA, 373–382.
[37] Wei Wei, Qiao Ke, Jakub Nowak, Marcin Korytkowski, Rafał Scherer, and Marcin Woźniak. 2020. Accurate and fast URL phishing detector: a convolutional neural network approach. Computer Networks 178 (2020), 107275.
[38] Wright, Jordan. “Announcing Gophish v0.1.1.” *Gophish*, 1 Feb. 2016, https://getgophish.com/blog/post/release-0.1.1/.
[39] Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. Cantina: a content-based approach to detecting phishing web sites. In Proceedings of the 16th international conference on World Wide Web. Banff Alberta, Canada, 639–648.
[40] OpenAI. “Introduction to OpenAI API.” 2023. https://platform.openai.com/docs/introduction/overview.