Abstract

Emails remain the most prevalent communication channel for social engineering attacks. As attackers increasingly use advanced tools to generate sophisticated phishing email content capable of bypassing detection systems, traditional methods often struggle to keep up. This poses a growing risk of sensitive information being compromised amongst various malicious outcomes. AI has gained significant attention and has evolved into a valuable tool for various applications, including phishing detection. LLMs can understand the context of language, enabling them to identify underlying intent even when certain features or parameters are modified.

This study assesses the effectiveness of a context-aware model in phishing email detection by utilizing a diverse dataset and leveraging the natural language processing capabilities of OpenAI's GPT-4. The system was evaluated using a range of performance metrics, such as accuracy, recall, precision, F1 score, and the average confidence score across all analyses conducted on the selected dataset. The system performed exceptionally well in classifying emails, achieving an accuracy score of 99.5%, a true positive rate of 99.7%, and an average confidence level of 90.2%. These findings show significant promise for enhancing email security.

1.0. Introduction

Phishing is one of the most prevalent and effective types of social engineering attacks in recent times (Sadiku, Shadare and Musa, 2016). Attackers use deliberate psychological deception through social interactions to manipulate, persuade, or trick victims, aiming to exploit vulnerabilities and ultimately compromise the confidentiality and integrity of their data and resources (Wang *et al.*, 2021). Attackers, often motivated by financial gain (Wolert and Rawski, 2023), impersonate legitimate organizations to exploit the security mistakes of victims for them to reveal sensitive information. Phishing is considered dangerous because it targets individuals, businesses, organizations, and even governments (Altwaijry *et al.*, 2024).

A report by the (Information Commissioner's Office, 2024) indicates that phishing may occur through various methods, including emails, texts (often referred to as smishing), and voice calls (known as vishing), with email being the most common method (Wolert and Rawski, 2023).

Phishing emails are fraudulent spam messages used to deceive recipients. Attackers use these messages to steal sensitive information or spread malware by disguising them as legitimate emails coming from trusted sources like businesses, banks or government agencies ultimately tricking victims into clicking links that appear genuine, downloading malicious attachments, or directly disclosing personal information (Roumeliotis, Tselikas and Nasiopoulos, 2024).

According to an annual report by the FBI's Internet Crime Complaint Center (IC3, 2024), phishing emails were the most frequently reported cybercrime in 2023, with over 298,000 complaints filed. Businesses have reported losses of over $50 billion worldwide due to these and similar attacks.

Also, an annual state of the phish report by (Proofpoint, 2024), shows that 74% of organizations in the UK were targeted by phishing email attacks in 2023. The report also indicates that fraudulent email attack volumes grew in the Middle East and parts of Asia by an average of over 30%, although this growth was previously limited by language barriers. However, with the assistance of generative AI, attackers can now create phishing emails in multiple languages,

making them more effective. Proofpoint detects an average of 66 million targeted email fraud attacks each month.

Lastly, a report by (Google, 2023) indicates that its email service, Gmail, blocks over 15 billion unwanted emails every day.

Detecting and blocking these fraudulent emails before they reach recipients is crucial for preventing phishing attacks. Conventional methods for detecting phishing emails, such as blacklisting and signature-based approaches, are effective but often rely heavily on human effort, which slows down the detection process. These methods have become less effective due to the increasing complexity of phishing attacks (Altwaijry *et al.*, 2024).

The integration of Artificial Intelligence and machine learning techniques has been instrumental in improving phishing email detection as these technologies can identify patterns and correlations that traditional methods may overlook, significantly improving accuracy.

However, while explicit feature-based machine learning classification models can be effective for certain phishing detection tasks, they have significant limitations. By relying on predefined patterns or specific features, these models become rigid and less adaptable to subtle or evolving phishing tactics. This approach often fails to capture the broader context of an email, focusing instead on isolated characteristics and missing subtle changes in language patterns or intent. Additionally, attackers can now leverage AI tools to modify email content, bypassing these fixed-feature models with ease. In cases where these limitations hinder detection, context-aware models, which analyze the entire email for patterns and intent, offer a more adaptable and robust solution.

This study focuses on leveraging the capabilities of OpenAI's large language model, GPT-4, along with a feedback loop, to develop an online web-based phishing email detection system. By capitalizing on the advanced features of GPT-4, such as contextual and natural language understanding, pattern recognition, and adaptive learning, this study aims to optimize its use in identifying phishing emails. The model is being modified to enhance its proficiency in recognizing email intent, distinctive writing styles, unusual requests, suspicious phrases, manipulative techniques, and urgency signals, thereby transforming it into a powerful tool for cybersecurity.

Furthermore, this research explores the multilingual and cross-cultural capabilities of the large language model, as it can be enhanced to identify phishing in various languages and cultural contexts. Through this investigation, the study aims to develop optimized methods for deploying GPT-4 effectively, making it a robust and adaptable solution for phishing email detection across diverse users.

2.0. Relevant Works and Literature Review

Traditional methods for detecting phishing emails include blacklist-based and signature-based approaches:

Blacklist-based detection: This method relies on databases of known malicious or suspicious URLs, domains, and IP addresses associated with phishing attacks, which are used to validate website addresses (Aljofey *et al.*, 2022). It is one of the most common approaches for blocking phishing emails (Sheng *et al.*, 2009). Embedded hyperlinks and sender information in emails are analyzed to determine if links redirect to potentially malicious or legitimate sites by comparing URLs against these databases (Sharma, Sharma and Sharma, 2024), if a match is found then the

email gets flagged as phishing. Commonly used blacklists include Google Safe Browsing and PhishTank. This method requires significant human effort to keep the list updated, as a large number of deceptive or fraudulent websites are created daily. This process is slow, prone to human error (Sheng *et al.*, 2009) and has proven inefficient for detecting zero-hour phishing URLs, as new URLs are often not added in time.

Signature-based detection: In cybersecurity, signatures are recognizable, distinguishable patterns associated with an attack. This approach involves using server-side algorithms implemented by email service providers to scan email content for specific patterns or signatures associated with phishing, such as suspicious keywords, attachments, or links. It also performs heuristic analysis, applying a set of rules to identify characteristics typical of phishing attempts. This includes scanning for common phishing phrases like “verify your account” or “urgent action required” (Sharma, Sharma and Sharma, 2024) and analyzing email metadata for inconsistencies in header information that may indicate spoofing. Signature-based or heuristic approach can detect phishing emails as they are sent, without heavy reliance on up-to-date blacklists (Sheng *et al.*, 2009). While this method can be effective for known threats, it has several drawbacks. It struggles to detect new or evolving phishing tactics, as signatures are predefined, and attackers can easily modify email content to bypass signature-based filters.

Given the limitations of traditional phishing email detection methods, machine learning models have been applied to enhance detection accuracy and adaptability. Using rule-based heuristic analysis, specific features are extracted from emails to build a training dataset for a feature-based machine learning classification model. However, this approach still has certain limitations.

(Smadi, Aslam and Zhang, 2018) proposed a solution for phishing email detection by utilizing a dynamic evolving neural network combined with reinforcement learning to develop an online phishing email detection system. This feature-based classification model extracts several types of information from both the header and content of emails. The system utilized reinforcement learning, allowing it to evolve and consistently develop the best model for addressing the task as phishing tactics change. The proposed solution achieved an accuracy score of 98.6%, however, it also has significant weaknesses. As a feature-based machine learning model, the system includes certain features that are irrelevant to the problem, as well as some with explicitly defined parameters. For example, it includes a feature called "*BlackListURL*", which is a binary feature that indicates whether any URLs in the email content are listed in the PhishTank database of reported phishing URLs. This feature updates every 60 minutes, which is not ideal given the large number of phishing URLs created each minute. This timeframe provides an opportunity for an attacker to orchestrate an attack and bypass the system. In fact, this feature should not exist, as it provides sufficient information to determine whether an email is intended for phishing or not. The system relies heavily on explicitly defined parameters, such as searching for specific English words that may have numerous synonyms. For instance, the feature "*SubjectVerifyWord*" checks whether the email subject contains the word "verify", which could easily be bypassed by using a synonym like "confirm." Another feature, "*BodyVerifyYourAccountPhrase*", checks whether the email body includes the phrase "verify your account" which can also be bypassed as well. In production, newly processed emails, after being classified by the model, are saved and used to train future models. However, false negatives present a challenge, when an attacker bypasses detection by exploiting gaps in the extracted features, there is no feedback loop to inform the model of this misclassification. As a result, the model may inadvertently incorporate features of these false negative emails into its training data, which, over time, can degrade its accuracy and

undermine its reliability. Also, based on the content of some extracted features, the researchers focused on a specific type of phishing email or tactics related to financial institutions, potentially overlooking other methods or variants of phishing email attacks that may also be relevant. Furthermore, the model is not trained on emails from diverse languages or cultural contexts, which limits its effectiveness in detecting phishing emails across different user bases. Lastly, the model lacked the ability to analyze attachments or other base64-encoded parts of the emails that represented binary files (e.g., images or PDFs) or text files.

A recent study by (R. Chataut, P. K. Gyawali and Y. Usman, 2024) introduced a novel approach to detecting phishing emails by utilizing a large language model. The researchers developed a custom variant of OpenAI's ChatGPT, called "CyberGPT", specifically designed for phishing email detection. This model leverages the natural language understanding and learning capabilities of OpenAI's latest large language model, GPT-4, to enhance detection accuracy. The study included a comparative analysis of OpenAI's GPT-3.5, GPT-4, and CyberGPT, with CyberGPT demonstrating superior performance by achieving a detection accuracy of 97.46%. A dataset of 828 emails, consisting of both legitimate (ham) and phishing examples from publicly available sources, was used to fine-tune, test, and evaluate the model's performance. However, there are several limitations, as the model lacks a feedback loop. This may result in static learning, which limits improvements in accuracy and adaptability over time. Without these mechanisms, CyberGPT may face challenges in adjusting to emerging phishing tactics, potentially leading to outdated detection strategies. Also, the researchers did not analyze other relevant parts of the emails, such as metadata, HTML formatting, or base64-encoded texts representing attachments or inline binary files like images, PDFs or even executables, as the emails used in their study had already been stripped of these elements. Furthermore, phishing emails in other languages were not considered.

An analysis of existing literature reveals several limitations in current phishing detection approaches. Blacklists are effective and have low false positive rates but are slow to update. Signature-based methods rely heavily on heuristic analysis and predefined signatures, which attackers can easily bypass. Additionally, feature-based machine learning models extract specific email characteristics through rule-based heuristic analysis to create a feature-based training dataset. However, these models also depend on predefined parameters and are typically trained to address only one type of phishing scheme at a time, limiting their ability to handle diverse linguistic and cultural contexts. Lastly, a context-aware model without a feedback loop may lead to static learning, hindering its improvement over time.

3.0. Methodology

The system was developed using an in-silico method, meaning the experiment was conducted using computer simulations. This section outlines the design methods applied in this study, which are detailed in the subsections below.

3.1. Research Problem

Through an in-depth investigation and analysis of the literature, it becomes evident that traditional methods and feature-based ML approaches for detecting phishing emails may lack the ability to quickly adapt to new behaviors or dynamically evolve to counter unseen tactics. Feature-based machine learning models for phishing email detection often rely on explicitly hand-engineered features extracted from emails, based on predefined rules or specific characteristics. These models are typically not designed to generalize well beyond the patterns

they were trained on, making them less effective at identifying new and evolving phishing techniques. Additionally, they focus on isolated features or patterns rather than considering the broader context. These limitations make it possible for attackers to bypass detection by retaining the phishing intent while altering language variation that the models may fail to detect. Phishing email detection often requires understanding various signs or how information is presented in an email, but these aspects may be overlooked by machine learning models. The proposed model will have the ability to detect new phishing email tactics by leveraging the power of OpenAI's large language model, ChatGPT, with its natural language understanding and context awareness. It can understand the broader context of an email in terms of intent, regardless of how the language is altered, because it analyzes phrases for various indicators, offering a much deeper understanding of the context.

3.2. Proposed Framework

The proposed framework is a web-based phishing email detection system that is powered by a context-aware model. The system extracts all parts of an email, such as metadata, plain texts, HTML, and even base64-encoded text representing text-based binary files, all separated by the specified boundary. These parts are then analyzed together to identify potential phishing indicators, such as inconsistencies in email headers, suspicious links, urgency signs, unusual requests, manipulative language or tone, and even suspicious phrases (Oles, 2023). Additionally, the system is designed to also detect inappropriate requests by understanding the language within the broader context of the email. For instance, if a phishing email pretends to be from a particular organization but uses language not typically associated with the style of communication for that organization, the system can recognize and classify it accordingly. Furthermore, the system is robust, as it can detect a wide range of phishing email tactics targeting various institutions like banks, social media platforms, e-commerce sites, and more. It can also identify phishing attempts across various linguistic and cultural contexts, making it valuable for diverse users.

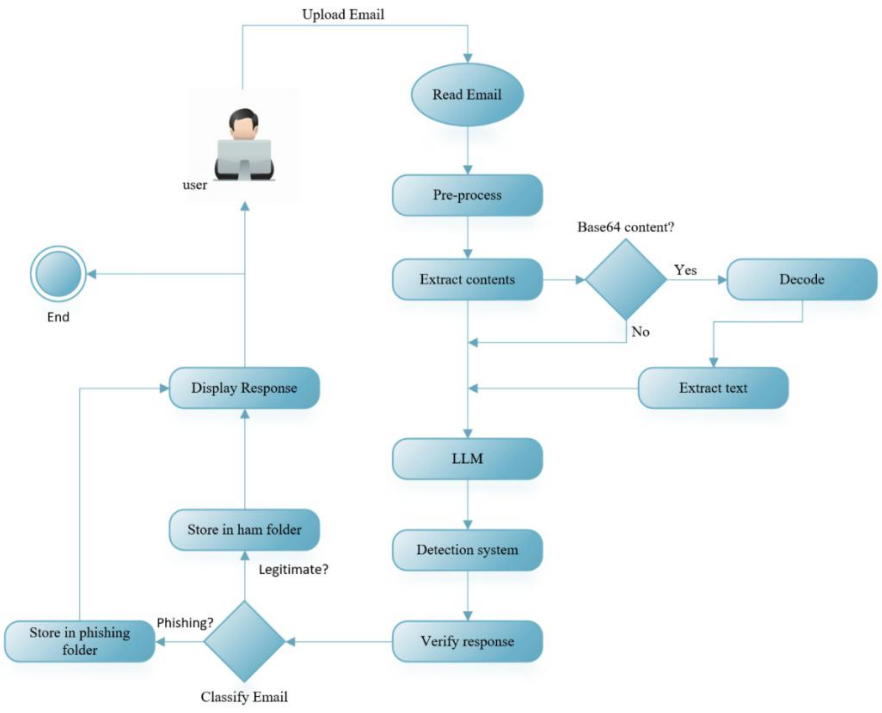

Fig. 1. Activity diagram of the proposed system

3.3. Data

The experiment was conducted using a combination of phishing and legitimate (ham) emails obtained from three publicly available datasets.

1. (SpamAssassin, 2006) public corpus is an organized collection of email messages published by SpamAssassin, suitable for testing spam filtering systems. It comprises a total of 6,047 emails, including 4,150 ham (legitimate) emails and 1,897 spam emails. From this collection, a total of 250 ham emails were selected.
2. (Nazario, J., 2024) phishing corpus by Jose Nazario is a collection exclusively comprising phishing emails, most recently updated in 2024. From this corpus, 250 phishing emails were randomly selected from recent years list.
3. (Arifa Islam, C., 2023) Phishing Email Curated Datasets compiled 11 datasets from 1995 to 2022. The datasets are provided in CSV format and have been stripped of the main email structure, such as metadata, HTML, or Base64 content. Some entries include only the sender, receiver, subject, body, and classification (phishing or legitimate). A total of 50 phishing emails and 50 legitimate emails were randomly selected from the dataset.

A total of 600 emails were used for the experiment. 300 phishing emails and 300 legitimate emails.

3.3.1. Structure Investigation

Email messages are entirely composed of strings and consist of header and content. The body may contain one or a mix of different content types. The Content-Type provides information about the format of the email body. This depends on factors such as text formatting or presence of attachments.

MIME (Multipurpose Internet Mail Extensions) allows emails to include multiple parts, with each part assigned its own MIME type (S. Turner, 2010). This enables the transmission of emails containing various types of content. Each part of the email body is separated by a boundary, which is a text string used to distinguish the different sections. When an email has no attachments, the Content-Type is usually set to text/plain or text/html, depending on the text formatting. In some cases, both formats may be included, and the Content-Type is set to multipart/alternative with the specified boundary. If the email contains attachments, the Content-Type is set to multipart/mixed, and the attachments are encoded in Base64 for easier transmission.

The MIME standard encodes binary files, such as images, audio, or application programs, into text strings that can be transmitted via email (S. Turner, 2010).

```
Date: Sun, 07 Jan 2024 18:29:02 +0000 (UTC)
From: "George Lancaster (via Onshape)" <alerts@onshape.com>
Message-ID: <1869925174.1693278947.1704652141950@smtp.sendgrid.net>
Subject: George Lancaster shared an Onshape folder with you
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_Part_16288_1273414476.1704652141950"
X-Mailer: sendhtml
```

Fig. 2. Content-Type field denoting multiple content sections

3.3.2. Quality Investigation and Email Preprocessing

The system processes emails uploaded in .txt or .eml formats. The uploaded email is then prepared for analysis to verify its status. First, the system checks the overall structure of the email

to ensure it meets basic requirements. This includes confirming that the email is not empty, its text content is properly encoded using valid character sets, and it is human-readable and printable. Also, the system detects the language of the email text content.

The built-in Python class BytesParser is used to parse binary MIME emails. This generates an in-memory object tree representing the email. By applying the walk() method on this object, the system iterates through all parts and subparts of the email's structure. During this process, the important header fields are extracted first, followed by the text content, which is decoded using the character set specified for that content. If the content is in HTML format, the HTML tags are stripped, retaining only the text. However, certain elements, such as anchor tags, forms, and scripts, are retained as well, as they may provide phishing-related insights. Parts with base64 content transfer encoding are decoded, and if they are text-based, the text is extracted. If the files are encrypted and cannot be analyzed, the system generates a warning.

All these steps then form the main content to be analyzed by the detection system.

```
Subject: You have a Document Via DOCUSIGN
Date: 12 Mar 2024 14:48:42 -0400
MIME-Version: 1.0
Content-Type: multipart/mixed;
        boundary="----=_NextPart_000_0012_9DB1A823.EF9CDCCD"

This is a multi-part message in MIME format.

------=_NextPart_000_0012_9DB1A823.EF9CDCCD
Content-Type: text/plain;
        charset="utf-8"
Content-Transfer-Encoding: quoted-printable

Dear jose@monkey.org,
=20=20
  Kindly view the attached document and revert back to us as soon=20
as possible.

------=_NextPart_000_0012_9DB1A823.EF9CDCCD
Content-Type: application/pdf; name="Wetransfer2024.pdf"
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename="Wetransfer2024.pdf"

JVBERi0xLjQKJeLjz9MKMSAwIG9iago8PAovVHlwZSAvUGFnZXMKL0NvdW50IDEKL0tpZHMg
```

Fig. 3. Structure of a multipart *phishing* email in MIME format

3.4. Large Language Models

Large language models (LLMs) are advanced AI systems developed using deep learning techniques (Hadi et al., 2023) and a wide range of datasets from multiple sources. LLMs have improved the ability of machines to process text, understand language context and the literal meaning of words, and generate clear, coherent text for various tasks (Naveed et al., 2024), in a human-readable format. LLMs have demonstrated remarkable capabilities in a wide range of language-related tasks, such as text generation, summarization, question answering, and more (Hadi et al., 2023). Advancements in LLM development are as a result of the introduction of transformer architecture (Vaswani et al., 2023), which laid the foundation for models like OpenAI's GPT, Google's BERT and more.

LLMs have advanced capabilities that extend to various specialized domains, including phishing email detection. These models excel in natural language processing, making them highly effective for identifying the complex language patterns often used in phishing emails.

3.4.1. OpenAI's GPT-4

GPT-4 is a high-performance large language model developed by OpenAI and is based on the transformer architecture outlined in (Vaswani *et al.*, 2023). GPT-4 is a large-scale multimodal

model with enhanced capabilities, surpassing the performance of its predecessors in the GPT series. GPT-4 is pre-trained on a diverse dataset of texts from publicly available sources, such as the internet, as well as third-party data (OpenAI *et al.*, 2024). It is further fine-tuned using reinforcement learning from human feedback (RLHF) and trained to predict the next word in a sequence (OpenAI *et al.*, 2024).

As a multimodal model, GPT-4 can process both image and text inputs and generate text outputs. This powerful model outperforms existing LLMs in various natural language tasks, making it suitable for a wide range of applications, including phishing email detection.

3.4.2. Prompt Engineering

A prompt is a text in natural language or a human-readable format, that defines the tasks a LLM should perform. Prompt engineering refers to the process of designing clear and effective prompts to achieve better and more accurate responses from the model. This process is vital for ensuring that the LLM performs the required tasks, produces the desired responses, and enhances its effectiveness in the chosen application.

In this system, the model assumes the role of an email security expert specializing in phishing email detection. GPT-4 analyzes various parts of an email as a whole to detect potential phishing indicators and classify the email accordingly. It generates a confidence score, expressed as a percentage (0 to 100), representing the likelihood that the email is phishing. Also, it provides a detailed explanation of its reasoning, offering a clear overview of the analytical process. The response is delivered in JSON format and includes the following key-value pairs:

1. "is_phishing": Boolean value (true or false) indicating whether the email is classified as phishing (true) or legitimate (false).
2. "confidence": Percentage indicating the level of confidence in its classification.
3. "reasoning": Detailed explanation outlining the model rationale, including specific signs and indicators that informed its decision.

main.py +

```
{
    "role": "system",
    "content": "You are an email security expert specialized in phishing detection."
},
{
    "role": "user",
    "content": prompt
}
```

Fig. 4. Model initialization

main.py +

```
"""
    Analyze the email thoroughly for potential phishing indicators.
    - Review the headers to ensure they originate from trusted sources.
    - Check all URLs in the email to confirm they point to legitimate domains.
    - Avoid classifying the email as phishing based solely on its content unless there are clear
    inconsistencies in the headers or indications that the links lead to malicious or suspicious domains.
    Entire response should consist of a single JSON object {{}}, and must NOT be wrapped within JSON md markers.
    Response format:
    {{
        "is_phishing": true or false,
        "confidence": A numeric confidence level from 0 to 100 indicating the likelihood of phishing or not,
        "reasoning": "Detailed explanation. Keep the response direct without additional commentary."
    }}
"""
```

Fig. 5. Model prompt

3.4.3. AI-Powered Phishing Email Detection System

This AI-powered phishing email detection system leverages the full computational power of OpenAI's latest GPT model, GPT-4. With its advanced natural language processing capabilities and ability to understand language in a broad context, GPT-4 offers a significant advantage in detecting phishing emails compared to known traditional techniques. In this system, the model is fine-tuned in order to optimize its performance in detecting phishing emails, including designing effective prompts to achieve the best results. As detailed in Fig. 4, during model initialization, the system assumes the role of an email security expert specializing in phishing email detection. It provides a comprehensive analysis of the email, including all decoded content. The prompt is refined to achieve optimal results, and the response is serialized in JSON format. Instructions relevant to phishing email detection are as follows:

1. Adopt the primary role of an email security expert specializing in thoroughly analyzing emails individually for potential phishing indicators. These include inconsistencies in email headers, suspicious links, urgency signs, unusual requests, and other warning signs (Oles, 2023).
2. Review the email headers to ensure they originate from trusted sources, verify all URLs to confirm they lead to legitimate domains, and then classify the email.
3. Review all texts extracted from binary files of decoded Base64-encoded sections of the email, if there are any. Disregard any scrambled or misspelled texts, as such errors may be a phishing indicator. However, it is also important to consider that these issues could arise from the text extraction process itself, where some content, while clear to the recipient, may not be accurately extracted or rendered by the system.
4. Provide a logical explanation, including any observed signs or lack that support the classification.
5. The response should be a single JSON object enclosed in {}, and must not be wrapped in JSON markdown markers, as the model tends to automatically wrap responses that way. The JSON object should include the following keys: 'is_phishing' (a boolean value: true or false), 'confidence' (a numeric value in percentage indicating the model's confidence in the classification), and 'reasoning' (a detailed explanation of the classification, including relevant signs).

Emails consist of headers, metadata, and body. This structure is commonly used to store and share emails. The headers include important details such as the sender, recipient, and information about the servers through which the email was routed. These details can help identify potential phishing attempts. Some important header fields are ‘FROM’, ‘TO’, ‘SUBJECT’, and ‘DATE’, which represent the sender, recipient, email subject, and the date it was sent or received, respectively. Other important header fields considered for analysis include:

1. Return-Path: This is the address where the email will be returned if it cannot be delivered to the recipient due to any issues. A mismatch between this address and the sender or reply-to addresses may suggest phishing.
2. Reply-To: This is the address where replies are sent. If this address differs from the sender’s address, it could indicate phishing, as replies may be redirected to malicious addresses, which is a common phishing tactic.
3. Received: The full information of this shows the servers that processed the email (Durumeric *et al.*, 2015). Inconsistencies in IP addresses, domains or timestamps may indicate phishing.

4. Content-Type: This specifies the type of content in the email, such as text/html or multipart/mixed.
5. DKIM-Signature: This field provides the results of DomainKeys Identified Mail (DKIM) verification (Durumeric *et al.*, 2015). A valid signature confirms that the email has not been tampered with during transmission.
6. Authentication-Results: This field summarizes the results of SPF, DKIM, and DMARC checks. These are various security authentication methods used to prevent domain impersonation, where attackers send emails from domains they do not own. Failures in these checks often indicate phishing attempts.
    a. SPF provides a list of servers and their corresponding IP addresses from which a domain is authorized to send emails.
    b. DKIM signature is a digital signature that uses PKI cryptography to confirm that an email originated from the domain it claims to be from, allowing domain owners to sign their outgoing emails (Durumeric *et al.*, 2015).
    c. DMARC specifies the actions the receiving email server should take based on the results of the SPF and DKIM checks. It can instruct the server to quarantine, reject, or deliver emails that fail SPF or DKIM validation (Durumeric *et al.*, 2015).

Base64 encoding is typically how binary files are transmitted in emails (S. Turner, 2010). The system can decode and extract text from base64-encoded content, including images and text-based binary files like PDFs, for analysis. This process is often used to counter a type of phishing tactic, where an image containing text is embedded directly in the email. Such tactics can bypass most known phishing detection methods, as the image may appear to be a harmless binary file but may contain texts with malicious intent. By decoding and extracting the text, the system gains a comprehensive view of the email's contents, enabling it to detect potential phishing threats in every section.

Legitimate emails from trusted organizations can sometimes contain phishing indicators. For instance, many bank statements sent via email may require users to input their date of birth as passwords to access encrypted attachments. While this is a standard banking practice, it can be flagged as a phishing indicator because it qualifies as an unusual request. Similarly, certain government or general policy changes may require institutions like banks to contact their customers to link specific documents to their accounts, sometimes with a sense of urgency, while providing links to access the platforms. These scenarios can also raise suspicion. In such cases, the system does not make decisions based solely on the content of the email, as that could lead to misclassifying a legitimate email as phishing. Additionally, some institutions use subdomains for campaigns or marketing messages, which the model may interpret as phishing indicators and misclassify the email. This highlights the importance of explicitly instructing the model not to classify emails as phishing unless there are clear inconsistencies in the email headers or the links point to malicious domains as it ensures a more balanced and accurate classification approach.

The implementation of these features aims to create a system that enhances the decision-making process, making it a powerful email security tool.

3.5. System Architecture

The AI-powered phishing email detection system is built using Python programming language. It leverages Django framework, which includes a lightweight localhost server to simulate a real-world deployment environment. The system provides a simple, user-friendly interface designed

with front-end development tools such as HTML, CSS, and JavaScript. It can be accessed through a web browser like Google Chrome or Mozilla Firefox. The system uses JSON for data serialization in multiple ways. First, during API calls to the LLM, responses are serialized in JSON format. JavaScript processes these JSON responses to deliver results to the user. Additionally, the system maintains a record of all analyzed emails in a JSON file and each analyzed email is stored.

Fig. 6. Shows the user-friendly interface of the phishing email detection system.

3.5.1. Django Framework

According to the Django documentation, Django is a free and open-source Python web framework that streamlines the process of web application development. It is based on the Python programming language and follows the Model-View-Controller (MVC) architectural pattern, often referred to as Model-View-Template (MVT) in Django's context. Django offers many features, including speed, security, and scalability. Its built-in CSRF protection safeguards applications against threats such as cross-site scripting (XSS) and cross-site request forgery (CSRF). Additionally, Django includes a lightweight, standalone web server for development purposes and a powerful template engine to render HTTP responses to users. With its comprehensive documentation, extensive community support, and robust ecosystem, Django is an ideal choice for developers seeking to build modern, high-performance web applications efficiently.

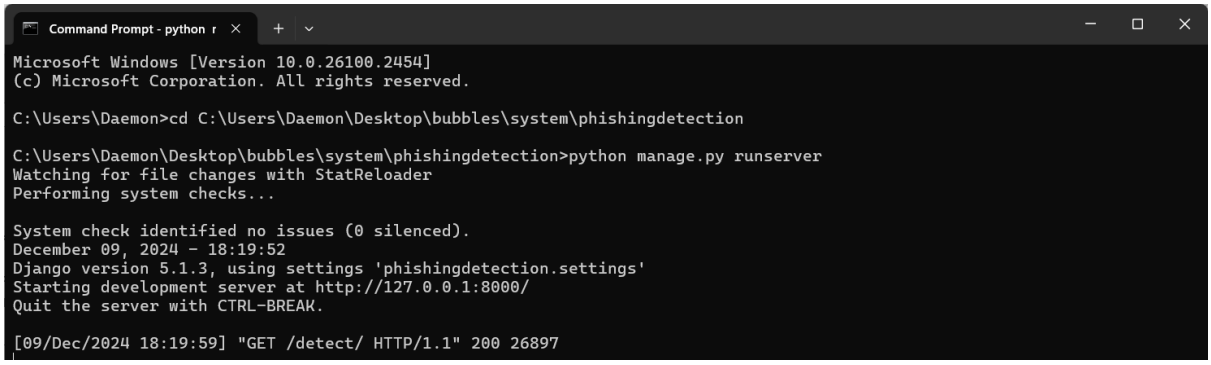

Fig. 7. Shows the server initialization.

4.0. Results and Discussion

The experiment was conducted on a dataset of 600 diverse emails, equally split into ham (legitimate) emails and phishing emails, and was repeated multiple times. The performance of the model was evaluated using major metrics, highlighting the effectiveness of large language models (LLMs) in detecting phishing emails. The evaluation consisted of two main parts:

- Confidence Score Analysis: The average confidence level of the model on all processed emails.
- Classification Metrics: Analysis of four key components of a confusion matrix, which are true positive (*TP*), false positive (*FP*), true negative (*TN*), and false negative (*FN*). These will then be used to compute the major metrics such as accuracy, precision, recall and F1 Score.

Since the primary goal of the system focuses on phishing email detection, a positive outcome indicates that an email is classified as phishing, while a negative outcome indicates it is classified as legitimate. The terms are defined as follows:

- *TP* denotes the number of phishing emails correctly classified as phishing.
- *TN* denotes the number of legitimate emails correctly classified as legitimate.
- *FP* denotes the number of legitimate emails incorrectly classified as phishing.
- *FN* represents the number of phishing emails incorrectly classified as legitimate.

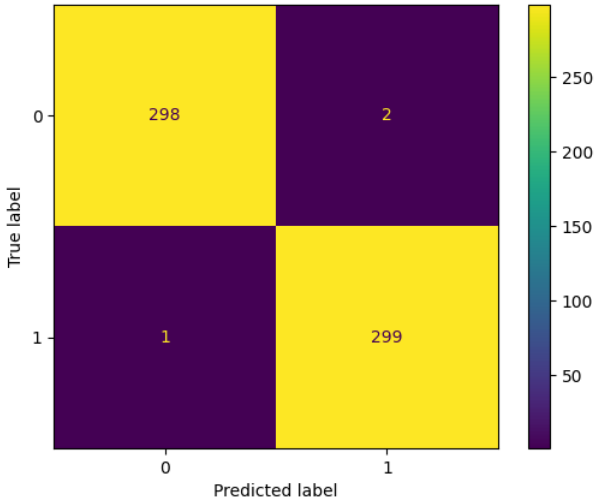

Fig. 8. Shows the confusion matrix of the system.

The confusion matrix in Fig. 8 shows that the system correctly classified 299 phishing emails (TP) out of 300 total phishing emails and 298 legitimate emails (TN) out of 300 legitimate emails. However, it misclassified 2 legitimate emails as phishing (FP) and 1 phishing email as legitimate (FN). Given that detecting phishing is of the greatest importance and is the primary objective of the system, this means the system is accurate 99% of the time.

Since the dataset is balanced, meaning it contains an equal number of emails for each class, we compute the most common performance metrics which are accuracy, recall (true positive rate), False positive rate, precision and F1 score.

Accuracy is the proportion of correct classifications (both positive and negative) and computed as:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

Recall, also known as the True Positive Rate, is the proportion of actual positives correctly classified as positives. It is calculated as:

$$\frac{TP}{TP + FN}$$

Precision is the proportion of all correct positive classification. It is computed as:

$$\frac{TP}{TP + FP}$$

F1 Score is the harmonic mean of both precision and recall. It is computed as:

$$\frac{2 * Precision * Recall}{Precision + Recall}$$

Accuracy	Recall (TPR)	Precision	F1 Score
99.5%	99.7%	99.3%	99.5%

Table 1. shows the different performance metrics.

Phishing	Legitimate
90.32%	89.97%

Table 2. shows the average confidence score for both classes.

With exceptionally high scores across all major performance metrics, the system demonstrates a robust ability to recognize signs of phishing. It effectively identifies unusual requests, spoofed headers, manipulative language, suspicious URLs, and more. The high confidence levels in its correct classifications further validate its accuracy and reliability. These attributes highlight the system's advanced effectiveness in detecting email threats, establishing it as a powerful and reliable tool for email phishing detection.

main.py +

```
{
    "is_phishing": false,
    "confidence": 95,
    "reasoning": """
    The email headers indicate that the email is sent from a legitimate Monzo domain (email.monzo.com),
    with proper DKIM and SPF authentication passing.
    The return-path and reply-to addresses are consistent with Monzo's email practices.
    The URLs in the email are formatted to redirect through clicks.monzo.com,
    which is a common practice for tracking email engagement by legitimate companies.
    There are no signs of domain spoofing or malicious redirects.
    The content is consistent with typical financial advice and promotional content from a bank like Monzo.
    Therefore, the email is unlikely to be phishing.
    """,
}
```

Fig. 9. Shows the JSON response from the LLM.

The system also incorporates a human feedback loop, allowing users to provide input on whether emails were correctly or incorrectly classified. This feedback is valuable for subsequent analysis, enabling the model to learn and improve its performance over time.

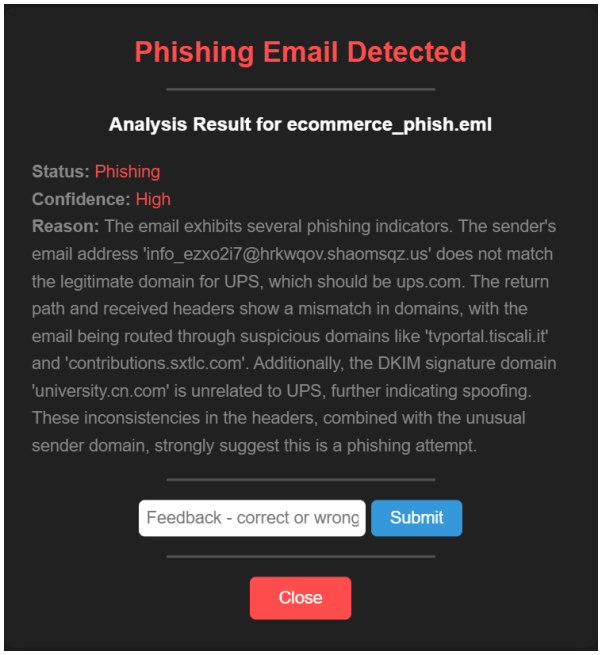

Fig. 10. Shows the analysis report of a logistics targeted phishing email along with a text input field for user feedback.

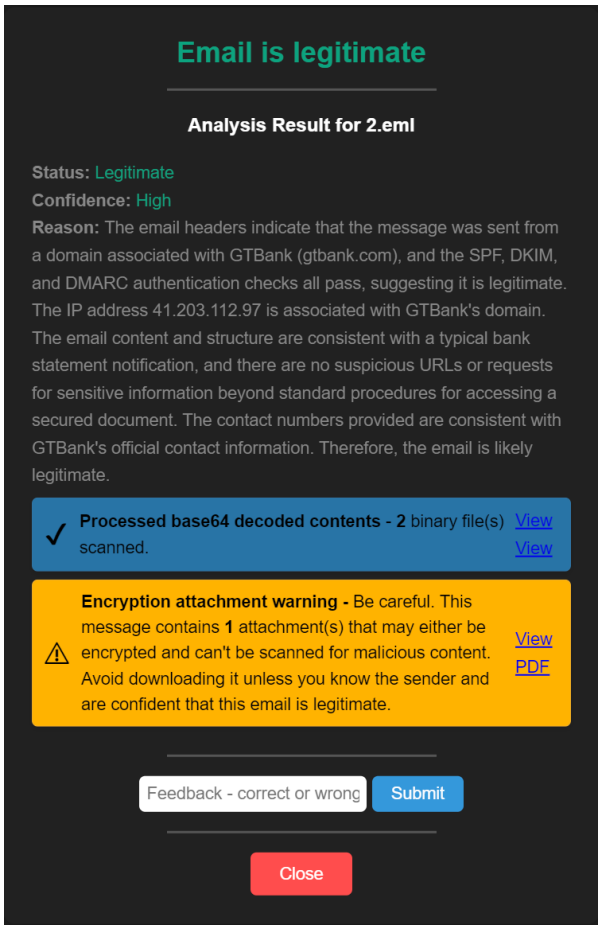

Fig. 11. Shows the analysis report for a statement email from a financial institution. The system shows processed text-based binary files and warning for encrypted pdf files that cannot be processed.

main.py +

```
"netflix_phish.eml": {
    "is_phishing": true,
    "confidence": 95,
    "reasoning": """The email exhibits several phishing indicators.
    The 'From' address uses a suspicious domain 'wryoqbw.xpjnclho.us'
    which does not align with Netflix's official domains. The 'Return-Path'
    and 'Received' headers show routing through unrelated and potentially
    compromised domains, such as 'nephcureaci.med.umich.edu' and 'crimestopperswv.com'.
    The DKIM signature domain '66571704.amnis.uk.com' is also unrelated to Netflix.
    These discrepancies in the email headers, combined with the context of a password
    change notification, strongly suggest phishing.""",
    "confidence_category": "High",
    "user_feedback": "correct"
}
```

Fig. 12. shows a high confidence score for a phishing email classification, confirmed by correct user feedback.

5.0. Conclusion and Future works

This project demonstrates how AI can significantly improve phishing email detection, as it can identify a wide range of highly sophisticated phishing tactics. It highlights the importance of natural language understanding and contextual awareness of OpenAI's GPT-4 model in detecting phishing email attempts across various industries and diverse languages and user groups. This ability, which is lacking in traditional models, allows LLMs to stay ahead of evolving phishing strategies, positioning it at the forefront of phishing email detection.

However, there are various limitations. Accessing OpenAI's GPT model is not free and requires payment. The pricing is based on the number of tokens used, typically charged per thousand or million tokens. The cost varies depending on the specific GPT-4 variant selected. GPT-4 offers different variants, ranging from the lightweight GPT-4 mini to more advanced models. Another limitation is the issue of data privacy. Since the GPT-4 model used in this project is owned and controlled by OpenAI, a third party, there is no guarantee of control over what data might be stored during the processing of emails. To address this concern, the system interface includes a warning that advises users to remove all sensitive information before uploading emails for analysis.

Future work would involve incorporating a means to detect malicious files, as GPT-4 is a text-based model capable of analyzing only text. It cannot inherently analyze binary or executable files for malicious intent. Identifying malicious behavior typically requires executing files in a controlled environment or scanning them with antivirus engines for behavioral analysis. A tool like VirusTotal, which is freely available, can be utilized for this purpose.

In conclusion, this project highlights the critical role of adopting a context-aware model for email security to enhance defenses against social engineering attacks. Leveraging LLMs for email security has shown significant potential in identifying and mitigating threats. LLMs can analyze email content, detect phishing indicators, and understand the context of human language, which is often exploited in social engineering attacks. By integrating AI-based solutions, organizations and individuals can proactively identify suspicious emails, minimize the risk of error, and strengthen their overall security against cyberattacks. This approach holds great promise for significantly mitigating email-based threats and paving the way for more secure email communication.

References

Information Commissioner's Office (2024) *Phishing.* Available at: https://ico.org.uk/about-the-ico/research-reports-impact-and-evaluation/research-and-reports/learning-from-the-mistakes-of-others-a-retrospective-review/phishing/ (Accessed 24 October 2024).

Proofpoint (2024) *Proofpoint's State of the Phish Report.* Available at: https://www.proofpoint.com/uk/newsroom/press-releases/proofpoints-2024-state-phish-report-67-uk-employees-willingly-gamble (Accessed 24 October 2024).

Federal Bureau of Investigation's Internet crimes complaint (2024), *FBI Releases Internet Crime Report.* Available at: https://www.fbi.gov/contact-us/field-offices/sanfrancisco/news/fbi-releases-internet-crime-report (Accessed 24 October 2024)

Google (2023), *New Gmail protections for a safer, less spammy inbox.* Available at: https://blog.google/products/gmail/gmail-security-authentication-spam-protection/ (Accessed 24 October 2024)

Arifa Islam, C. (2023) "Phishing Email Curated Datasets". Zenodo. Available at: https://doi.org/10.5281/zenodo.8339690.

Nazario, J. (2024) *Phishing Corpus*. Available at: https://monkey.org/~jose/phishing/ (Accessed 24 October 2024).

SpamAssassin (2006) *Public Corpus.* Available at: https://spamassassin.apache.org/old/publiccorpus/ (Accessed 24 October 2024).

Django Software Foundation (2013) *Django.* Available at: https://www.djangoproject.com/ (Accessed 9 December 9, 2024)

Aljofey, A. et al. (2022) 'An effective detection approach for phishing websites using URL and HTML features', Scientific Reports, 12(1), pp. 8842–5. Available at: https://doi.org/10.1038/s41598-022-10841-5

Altwaijry, N. et al. (2024) 'Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models', Sensors, 24(7), pp. 2077. Available at: https://doi.org/10.3390/s24072077

Azeez, N.A. et al. (2021) 'Adopting automated whitelist approach for detecting phishing attacks', Computers & Security, 108, pp. 102328. Available at: https://doi.org/10.1016/j.cose.2021.102328

Blank, I.A. (2023) 'What are large language models supposed to model?', Trends in Cognitive Sciences, 27(11), pp. 987–989. Available at: https://doi.org/10.1016/j.tics.2023.08.006

Durumeric, Z. et al. (2015) 'Neither Snow Nor Rain Nor MITM...', 10. Available at: 10.1145/2815675.2815695.

Hadi, M.U. et al. (2023) 'Large Language Models: A Comprehensive Survey of its Applications, Challenges, Limitations, and Future Prospects', Available at: https://doi.org/10.36227/techrxiv.23589741

Naveed, H. et al. (2024) A Comprehensive Overview of Large Language Models.

Oles, N. (2023) 'Phishing Tactics and Techniques', in 'Phishing Tactics and Techniques', How to Catch a Phish: A Practical Guide to Detecting Phishing Emails. Berkeley, CA: Apress, pp. 23–31.

OpenAI et al. (2024) GPT-4 Technical Report.

Pedregosa, F. et al. (2011) 'Scikit-learn: Machine Learning in Python', Journal of Machine Learning Research, 12(85), pp. 2825–2830. Available at: http://jmlr.org/papers/v12/pedregosa11a.html

R. Chataut, P. K. Gyawali and Y. Usman (2024) 'Can AI Keep You Safe? A Study of Large Language Models for Phishing Detection', 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC). Available at: 10.1109/CCWC60891.2024.10427626.

Roumeliotis, K.I., Tselikas, N.D. and Nasiopoulos, D.K. (2024) 'Next-Generation Spam Filtering: Comparative Fine-Tuning of LLMs, NLPs, and CNN Models for Email Spam Classification', Electronics, 13(11), pp. 2034. Available at: https://doi.org/10.3390/electronics13112034

S. Turner (2010) 'Secure/Multipurpose Internet Mail Extensions', IEEE Internet Computing, 14(5), pp. 82–86. Available at: https://doi.org/10.1109/MIC.2010.121

Sadiku, M., Shadare, A. and Musa, S. (2016) 'Social Engineering: An Introducction', The Journal of Scientific and Engineering Research, 3, pp. 64–66.

Sharma, S., Sharma, R. and Sharma, M. (2024) 'Phishing Emails Detection in Cyber Security', Available at: https://doi.org/10.20944/preprints202404.1655.v1

Sheng, S. et al. (2009) 'An Empirical Analysis of Phishing Blacklists',

Smadi, S., Aslam, N. and Zhang, L. (2018) 'Detection of online phishing email using dynamic evolving neural network based on reinforcement learning', Decision Support Systems, 107, pp. 88–102. Available at: https://doi.org/10.1016/j.dss.2018.01.001

Sokolova, M., Japkowicz, N. and Szpakowicz, S. (2006) 'Beyond Accuracy, F-Score and ROC: A Family of Discriminant Measures for Performance Evaluation', 01. Available at: 10.1007/11941439_114.

Vaswani, A. et al. (2023) Attention Is All You Need.

Wang, Z. et al. (2021) 'Social engineering in cybersecurity: a domain ontology and knowledge graph application examples', Cybersecurity, 4(1), pp. 31. Available at: https://doi.org/10.1186/s42400-021-00094-6

Wolert, R. and Rawski, M. (2023) 'Email Phishing Detection with BLSTM and Word Embeddings', International Journal of Electronics and Telecommunications, 69(3), pp. 485–491. Available at: https://doi.org/10.24425/ijet.2023.146496