Email Controls and Detection Engine

Final Year Project Report

by

Abdul Moiz Quddus

Kaleem Ullah

Usman Afzal

Muhammad Luqman

In Partial Fulfillment

Of the Requirements for the degree

Bachelor of Engineering in Software Engineering (BESE)

Military College of Signals

National University of Sciences and Technology

Islamabad, Pakistan

(2023)

DECLARATION

We hereby declare that this project report entitled “Email Controls and Detection Engine” submitted to the “CSE Department”, is a record of an original work done by us under the guidance of Supervisor “Dr. Waleed Bin Shahid” and that no part has been plagiarized without citations. Also, this project work is submitted in the partial fulfillment of the requirements for the degree of Bachelor of Computer Science.

Team Members	**Signature**
Abdul Moiz Quddus	_____
Kaleem Ullah	_____
Usman Afzal	_____
Muhammad Luqman	_____
Supervisor:	**Signature**
AP Dr. Waleed Bin Shahid	_____
Co-Supervisor:	**Signature**
Prof. Dr. Hammad Afzal	_____

Date:

_____

Place:

Military College of Signals

DEDICATION

To our parents, whose unwavering support and encouragement have been the foundation of our academic journey. This thesis is a testament to their love and belief in us. We thank them for everything.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

Our parents, colleagues, and most of all our worthy supervisor, Dr. Waleed Bin Shahid, without your guidance.

The group members, who through all adversities worked steadfastly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

With the increasing sophistication and frequency of email-based cyber attacks, it has become essential for organizations and individuals to implement effective security measures to protect their networks and sensitive information. However, many individuals lack the necessary knowledge and training to identify potential phishing attacks, malicious content, and attachments in their email inboxes, leaving organizations vulnerable to data breaches and other security threats.

To address this challenge, we have developed the Email Controls and Detection Engine. This product is designed to assist individuals in detecting and analyzing potential security threats in their email inboxes, providing an efficient and reliable tool for organizations to combat email-based cyber attacks.

The Email Controls and Detection Engine is a plugin designed to assist individuals in detecting and analyzing potential security threats in their email inboxes. This plugin is compatible with all modern browsers that use the Chromium platform and requires an internet connection for proper functionality. The analysis of email content is performed on the backend server, utilizing predetermined rules, with the results displayed on the plugin interface. Overall, the Email Controls and Detection Engine provides an efficient and reliable tool for organizations to combat potential security threats in their email communication.

Chapter 1

INTRODUCTION

The aim of this project is to provide a solution for the increasing cyber threats posed by email-based attacks. The Email Controls and Detection Engine is a Plugin-based solution designed to help both individuals and organizations to detect and analyze potential phishing attacks, malicious content, attachments, and hyperlinks in their email inbox. The plugin provides an efficient and reliable tool for organizations to combat potential security threats in their email communication.

With the increasing sophistication and frequency of email-based cyber attacks, many individuals lack the necessary knowledge and training to identify potential threats in their email inboxes, leaving organizations vulnerable to data breaches and other security threats. The Email Controls and Detection Engine addresses this challenge by offering a reliable tool to assist individuals in detecting and analyzing potential security threats.

The Email Controls and Detection Engine is comprised of three main modules: the Plugin, Reporting Panel, and Back-End Server. The Plugin is designed using HTML, CSS, and JavaScript, and offers features such as fetching and displaying emails, checking the connection to the server, and providing access to the reporting panel. The Back-End Server, AI-based modules, includes an email parser, body analysis, structure analysis, URL analysis, meta-analysis, and reports generation. After analyzing the email, the server sends the results back to the Reporting Panel, a simple landing page where users can view or download the complete detailed report of the received .eml file.

This project caters to both individuals and organizations looking for a reliable tool to combat email-based cyber-attacks. With the Email Controls and Detection Engine, individuals can feel confident in their ability to identify potential threats and protect their organization's sensitive information.

1.1 OVERVIEW

The Email Controls and Detection Engine is a proactive email security solution designed to address the increasing threat of email-based cyber attacks. This Plugin-based solution provides organizations and individuals with an efficient and reliable tool to detect and analyze potential phishing attacks, malicious content, attachments, and hyperlinks in their email inbox.

The system comprises three modules: the Plugin, Reporting Panel, and Back-End Server. The Plugin, designed in HTML, CSS, and JavaScript, fetches emails and provides users with a connection status to the Back-End Server. The Back-End Server, a Python-based server, analyzes emails using predetermined rules and sends the results to the Reporting Panel, where the complete detailed report of the email is displayed.

The Email Controls and Detection Engine aims to address the limitations of current email security solutions, which are reactive and rely on individuals to manually identify and report suspicious emails. By providing a proactive and reliable email security solution, the system enables organizations and individuals to protect their sensitive information, reputation, and ensure business continuity.

Overall, the Email Controls and Detection Engine is an essential tool for organizations and individuals looking to safeguard their email communication from potential cyber threats.

1.2 PROBLEM STATEMENT

Problem Statement Email communication is an essential part of business operations, but it has also become a prime target for cybercriminals looking to exploit vulnerabilities in an organization's security system. Phishing attacks, malware, and other email-based threats have become increasingly sophisticated, making them harder to detect and prevent. These attacks can result in significant financial losses, damage to an organization's reputation, and compromise of sensitive information.

Current email security solutions are often reactive, relying on individuals to manually identify and report suspicious emails. This approach is time-consuming and unreliable, as individuals may not have the necessary training or expertise to recognize these threats. This creates a pressing need for a proactive and reliable email security solution that can effectively identify and prevent email-based attacks.

Due to the increasing prevalence of email phishing attacks, many organizations are vulnerable to significant financial loss and reputational damage. The Belgian Bank lost $70 million to an email phishing scam, highlighting the severity of the issue. In fact, 91% of all cyber attacks start with a phishing email, and certain industries are particularly vulnerable. Therefore, there is an urgent need for a reliable tool to help individuals and organizations detect and analyze potential phishing attacks, malicious content, attachments, and hyperlinks in their email inbox.

The problem statement, therefore, is how can organizations proactively and reliably detect and prevent email-based cyber threats to safeguard their sensitive information and reputation, and ensure business continuity?

1.3 PROPOSED SOLUTION

Based on the identified problem, the proposed solution is the development of an Email Controls and Detection Engine, a plugin-based system that can detect and analyze potential phishing attacks, malicious content, attachments, and hyperlinks in email inboxes. The plugin would have features such as fetching emails, checking connections to the backend server, and displaying results on the reporting panel. The backend server would be an AI-based server with features such as an email parser, body analysis, structure analysis, URL analysis, meta-analysis, and reports generation, URLs analysis, and meta-analysis. The reporting panel would be a simple landing page where the complete detailed report of the email file would be displayed. The solution aims to provide organizations and individuals with an efficient and reliable tool to combat potential security threats in their email communication.

1.4 WORKING PRINCIPLE

The Email Controls and Detection Engine operates through a combination of client-server architecture and modular design, which allows for efficient and effective detection and analysis of potential email-based cyber threats. The working principle can be summarized in the following steps:

- *User Interface:* The user interacts with the plugin through a user interface designed to display the plugin and facilitate user interactions.
- *Email Fetching*: The Gmail API component handles incoming and outgoing requests to the Google server to fetch the emails for analysis.
- *Browser Plugin:* The plugin component is responsible for receiving and displaying incoming emails. It then sends requests to the backend server for further processing.
- *Backend Server Analysis:* The backend server, a Python-based server that is AI-powered, performs a comprehensive analysis of the emails, including scanning for malicious content, attachments, and hyperlinks. It utilizes AI-based techniques such as structure analysis, URL analysis, and meta-analysis to detect potential threats.
- *Report Generation:* After the analysis is complete, the backend server generates detailed reports on the findings and sends them back to the plugin.
- *Reporting Panel:* The reporting panel, a simple landing page, displays the complete detailed report of the analyzed email file for the user to view or download.

The decomposition of the Email Controls and Detection Engine into smaller modules and components allows for a more manageable and maintainable system. The modular design ensures that changes or updates to one component do not affect the others, making the system more flexible and adaptable to future needs.

The client-server architecture allows for centralized management of resources and services, enabling better control and security. This architecture also permits multiple

clients to access shared resources and services, facilitating data and information sharing across multiple devices and users.

1.5 OBJECTIVES

1.5.1 General Objectives:

To develop a state-of-the-art plugin-based email security solution, powered by artificial intelligence and machine learning techniques, to provide individuals and organizations with a reliable and efficient tool for detecting and analyzing potential email-based cyber threats.

1.5.2 Academic Objectives:

- Development of an intelligent Email Controls and Detection Engine
- Implementation of AI and machine learning techniques for efficient threat detection and analysis
- Enhancing collaboration and problem-solving skills by working in a team
- Designing a project that contributes to the cybersecurity of individuals and organizations, promoting the welfare of society.

1.6 SCOPE

In the era of digital communication, email spam has emerged as a significant concern for organizations and individuals. The increasing volume of unsolicited and potentially harmful messages demands a more effective solution to ensure email security and user confidence. This project aims to leverage AI/ML techniques and browser plugin technology to tackle this issue head-on.

The Email Controls and Detection Engine provides users with a reliable and user-friendly tool to authenticate their email accounts, fetch emails from their inboxes, and send them to a cloud-based detection engine for analysis. By parsing and examining each email component, the system can identify potential spam messages and generate a comprehensive PDF report of its findings.

This innovative approach not only enhances the email security of organizations and individuals but also promotes transparency and self-validation, empowering users to take control of their digital communication. By offering seamless integration with various email service providers, compatibility with Windows/Linux environments, and chromium-based browsers, the system caters to a broad user base with diverse requirements.

In summary, the Email Spam Detection System offers a cutting-edge solution for identifying and analyzing potential spam emails, ultimately reducing the risk of security breaches, data leaks, and user inconvenience in the ever-evolving landscape of digital communication.

1.7 DELIVERABLES:

1.7.1 User Accessibility and Control through Plugin:

The system offers user-friendly accessibility and control by providing a browser plugin and a reporting panel. Users can authenticate their email accounts, fetch emails, and send them for analysis through the plugin. The reporting panel enables users to view and download the generated PDF reports, promoting transparency and self-validation in the email communication process.

1.7.2 Email Analysis and Report:

The Email Controls and Detection Engine provides a comprehensive analysis of each email component, identifying potential spam messages and generating a detailed PDF report outlining its findings. This enables users to understand the credibility of the email sender and the security risks associated with the email content.

1.7.3 Email Classification and Scoring:

Utilizing AI/ML techniques and data sets, the Email Controls and Detection Engine classifies and scores the emails based on their potential spamming nature. This process helps users to quickly identify potentially harmful emails and prioritize genuine messages.

RELEVANT SUSTAINABLE ENVIRONMENTAL GOALS

1.8.1 Industry, Innovation, and Infrastructure

The Email Controls and Detection Engine contributes to the development of innovative cybersecurity solutions for both organizations and individual users. By integrating advanced AI/ML algorithms and modern technologies, this project enhances the security and efficiency of email communication systems. It promotes sustainable IT infrastructure by reducing the impact of spam and phishing emails, fostering an environment that prioritizes innovation and industry advancements.

Figure 1: SDG Goal 9 (Industry, Innovation & Infrastructure)

1.8 STRUCTURE OF THESIS

The structure of the thesis defined the distribution of chapters according to their topics.

- This Chapter 1 is *Introduction*.

- Chapter 2 contains *Literature Review*.
- Chapter 3 has Problem *Definition*.
- Chapter 4 describes *Methodology*.
- Chapter 5 talks about *Detailed Design and Architecture*.
- Chapter 6 has *Implementation and Testing*.
- Chapter 7 finalizes the *Results and Discussion*.
- Chapter 8 contains *Conclusion and Future Work.*
- Chapter 9 at last contains *References*.

Chapter 2

LITERATURE REVIEW

A new product is launched by modifying and enhancing the features of previously launched similar products. Literature Review is an important step for the development of an idea to a new product. For the Email Controls and Detection Engine, a detailed study regarding all similar projects is compulsory. Our research is divided into the following points:

- Existing email security solutions
- Spam detection techniques
- Phishing detection techniques
- Research papers

2.1 EXISTING EMAIL SECURITY SOLUTIONS

Email security solutions have been evolving to combat the increasing threats of spam, phishing, and other malicious content. Some of the existing solutions are:

2.1.1 Secure Email Gateways (SEGs):

These solutions provide protection against spam, phishing, and malware by filtering and scanning inbound and outbound emails. SEGs work by analyzing emails in real time, evaluating them against a set of rules and known threats, and blocking or quarantining suspicious messages. Examples of SEGs include:

2.1.1.1 Mimecast: Offers cloud-based email security with features such as URL protection, attachment protection, and impersonation protection, alongside archiving and continuity services.

2.1.1.2 Barracuda: Provides a comprehensive email security platform that includes spam filtering, virus protection, email encryption, data loss prevention, and advanced threat protection.

2.1.1.3 Proofpoint: Delivers a multi-layered email security solution that combines advanced threat detection, real-time intelligence, and policy enforcement to protect against spam, phishing, and malware attacks.

2.1.2 Advanced Threat Protection (ATP):

ATP solutions protect against sophisticated email-based threats that may bypass traditional email security measures. These solutions use machine learning, behavioral analysis, and sandboxing to detect and block advanced threats. Examples of ATP solutions include:

2.1.2.1 Microsoft Office 365 Advanced Threat Protection: Protects Office 365 users from advanced threats, such as spear-phishing and zero-day malware, by analyzing email attachments and URLs in a secure environment and providing real-time threat intelligence.

2.1.2.2 Symantec Email Security Cloud: Offers advanced email threat protection, including targeted attack protection, advanced URL defense, and sandboxing to detect and block sophisticated threats before they reach the recipient's inbox.

2.1.2.3 Cisco Advanced Malware Protection (AMP) for Email: Integrates with Cisco Email Security to provide advanced threat protection, using global threat intelligence, advanced sandboxing, and real-time malware blocking to prevent attacks.

2.1.3 Email Authentication Protocols: These protocols help prevent email spoofing and ensure the authenticity of the sender by verifying that an email has been sent from a legitimate source. Examples include:

2.1.3.1 Sender Policy Framework (SPF): A protocol that allows domain owners to specify which mail servers are authorized to send email on their behalf. Receiving email servers can then check the sender's IP address against the SPF record to verify that the email is from an authorized source.

2.1.3.2 DomainKeys Identified Mail (DKIM): A digital signature-based email authentication method that allows domain owners to sign their emails using a private key. The receiving email server can then verify the signature using the corresponding public key, which is published as a DNS record, ensuring the email has not been tampered with and that it comes from the claimed domain.

2.1.3.3 Domain-based Message Authentication, Reporting, and Conformance (DMARC): A protocol that builds on SPF and DKIM by allowing domain owners to define policies for how receiving email servers should handle unauthenticated emails, such as rejecting or quarantining them. DMARC also provides a reporting mechanism that helps domain owners monitor and improve their email authentication practices.

2.2 SPAM DETECTION TECHNIQUES

Various techniques have been developed to detect and filter spam emails. Here, we elaborate on each technique and provide examples of their use:

2.2.1 Rule-based Filtering:

This approach involves creating predefined rules to identify spam based on specific patterns. These patterns can include keywords, header information, sender

IP addresses, or other identifiable email characteristics. Rule-based filtering systems analyze incoming emails and apply these rules to determine if they should be flagged as spam.

Examples of rule-based filtering solutions:

2.2.1.1 SpamAssassin: An open-source email filter that uses a wide range of heuristic tests, including text analysis, Bayesian filtering, DNS blocklists, and collaborative filtering databases, to identify and flag spam emails.

2.2.1.2 SpamTitan: A cloud-based email security solution that uses rule-based filtering, along with other techniques such as Bayesian analysis, to block spam, phishing, and malware emails.

2.2.2 Machine Learning:

Machine learning techniques are used to classify emails as spam or non-spam based on various email features, including text, links, and attachments. Algorithms such as Naive Bayes, Decision Trees, and Support Vector Machines can be trained on large datasets of known spam and non-spam emails, allowing them to automatically identify patterns and characteristics associated with spam.

Examples of machine learning-based spam detection:

2.2.2.1 Naive Bayes: A popular probabilistic classifier based on Bayes' theorem, often used in spam filtering due to its simplicity and effectiveness. Naive Bayes classifiers can be trained on labeled email datasets to predict whether an email is spam or non-spam based on the frequency and co-occurrence of words and phrases in the email.

2.2.2.2 Support Vector Machines (SVM): A machine learning algorithm that finds the best hyperplane to separate spam and non-spam emails in a high-dimensional feature space. SVMs can provide high accuracy and generalization performance in spam detection tasks, particularly when combined with text preprocessing and feature selection techniques.

2.2.3 Deep Learning:

Advanced techniques like Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks are used to analyze the content and context of emails for spam detection. These deep learning models can capture complex patterns and relationships within email data, including the structure and semantics of text, allowing for more accurate and robust spam classification.

Examples of deep learning-based spam detection:

2.2.3.1 Recurrent Neural Networks (RNN): A type of neural network designed to process sequential data, such as text in emails. RNNs can be trained to detect spam by learning patterns and dependencies in the email content and can effectively model the contextual information present in emails.

2.2.3.2 Long Short-Term Memory (LSTM) networks: A type of RNN that can learn long-range dependencies in data, making them particularly suitable for spam detection tasks where context and relationships between words and phrases are important. LSTM networks can be trained on large email datasets to classify emails as spam or non-spam based on their content and structure.

By combining these spam detection techniques, email security solutions can achieve higher accuracy and better adapt to the evolving nature of spam and email-based threats.

2.3 PHISHING DETECTION TECHNIQUES

Phishing detection techniques aim to identify and block malicious emails attempting to deceive recipients into revealing sensitive information or performing actions that compromise security:

5.5.1 Heuristic Analysis:

This approach involves analyzing email content, structure, and sender information to identify patterns commonly associated with phishing emails.

5.5.2 Machine Learning:

Machine learning algorithms can be trained to detect phishing emails based on features such as URLs, sender information, and email content.

5.5.3 Blacklists and Whitelists:

Phishing detection systems may rely on blacklists of known phishing sites and whitelists of trusted websites to identify malicious emails.

2.4 RESEARCH PAPERS

Several research papers have proposed and evaluated various techniques for detecting spam and phishing emails:

[1] proposed a hybrid approach for spam filtering, combining rule-based filtering and machine learning techniques, achieving a high detection rate and low false positive rate.

[2] developed an LSTM-based deep learning model for spam detection, achieving an accuracy of 97.5% on a benchmark dataset.

[3] investigated the use of machine learning algorithms, including Naive Bayes, Decision Trees, and Support Vector Machines, for phishing email detection, finding that the ensemble-based approach provided the best results.

[4] introduced D-Fence, a flexible, efficient, and comprehensive phishing email detection system. D-Fence leverages natural language processing, machine learning, and a rule-based approach to analyze email content, context, and sender information. The system is designed to adapt to different types of phishing emails and offers a high detection rate while maintaining a low false positive rate. D-Fence demonstrates the effectiveness of combining multiple techniques to create a more robust and accurate phishing detection solution.

2.5 MOTIVATION TO WORK IN LIGHT OF THE LITERATURE REVIEW:

After conducting the literature review, it became evident that email security remains a critical concern, with spam and phishing attacks continuing to evolve in sophistication. Existing email security solutions, spam detection techniques, and phishing detection techniques have shown significant progress. However, there is always room for improvement in terms of accuracy, adaptability, and comprehensiveness.

Working on this project was motivated by the desire to build upon the existing research and develop a more effective and robust email controls and detection engine. The literature review highlights the strengths and weaknesses of various techniques, which provided valuable insights for our project. By learning from and combining the most successful approaches, such as rule-based filtering, machine learning, and deep learning, we aimed to create an email security solution that offers a higher detection rate and a lower false positive rate.

Furthermore, the literature review showcases the success of hybrid systems, like D-Fence, in achieving comprehensive phishing email detection. This project sought to emulate and build upon such successes by combining multiple techniques and considering various aspects of email data to create a more accurate and adaptable email security solution. The ultimate goal of this project is to contribute to the ongoing development of email security systems and help protect users from ever-evolving email-based threats.

Chapter 3

PROBLEM DEFINITION

The primary goal of this project is to develop an advanced email controls and detection engine capable of effectively identifying and blocking various email threats, including spam, phishing, and malware. To address the challenges posed by evolving email threats, inadequate detection, and false positives, the proposed solution must be adaptable, accurate, and comprehensive. Furthermore, the engine should combine multiple detection techniques, such as rule-based filtering, machine learning, and deep learning, to offer a robust defense against email-based attacks. Finally, the system should be scalable and efficient, ensuring its effectiveness in processing and analyzing large volumes of email data without compromising accuracy or performance.

3.1 CHALLENGES AND REQUIREMENTS:

To achieve these objectives, the problem definition can be broken down into the following subproblems and we will explain them in the form of challenges and requirements.

3.1.1 Evolving Email Threats and Adapting to Different Types of Threats:

Challenge: Email threats have grown more sophisticated over time, with attackers shifting from generic spam emails to targeted spear-phishing campaigns. Existing email security solutions must be continuously updated and enhanced to keep up with these evolving tactics and techniques. Email threats come in various forms, such as spam, phishing, and malware.

Requirement: Develop an email controls and detection engine that can adapt to changing attacker tactics and techniques, providing better protection to users. Incorporate diverse detection techniques in the engine that can adapt to various threat types.

3.1.2 Inadequate Detection, False Positives, and Enhancing Accuracy:

Challenge: Current email security solutions may struggle with accurately identifying all types of threats, leading to false positives and negatives.

Requirement: Create a more accurate and reliable email security solution that minimizes false positives and negatives while effectively detecting threats. Optimize the detection algorithms to improve accuracy and reduce false positives and negatives through continuous training and refinement of machine learning and deep learning models.

3.1.3 Combining Detection Techniques and Developing a Comprehensive Threat Detection Framework:

Challenge: An optimal email security solution should leverage multiple detection techniques to improve its accuracy and comprehensiveness.

Requirement: Develop an email controls and detection engine that harmonizes rule-based filtering, machine learning, and deep learning techniques to provide a robust defense against email-based attacks. Design a framework that effectively integrates these techniques, leveraging the strengths of each to cover the weaknesses of others, ensuring a more robust and accurate threat detection system.

3.1.4 Ensuring Scalability and Efficiency:

Challenge: The volume of emails and email-based attacks continues to grow, making it essential for email security solutions to be scalable and efficient.

Requirement: Design an email controls and detection engine that can handle high volumes of emails while maintaining accuracy and performance, ensuring the

email security solution remains effective as the organization and its email traffic grow. Build the engine with scalability and efficiency in mind, using optimized algorithms, parallel processing, and efficient resource management.

Chapter 4

METHODOLOGY

The methodology section outlines the steps and techniques used to develop the email controls and detection engine. This section provides a detailed explanation of the various stages and components involved in the process, along with examples for better understanding.

4.1 EMAIL FETCHING VIA PLUGIN

To fetch emails from the user's inbox, we will develop a plugin that integrates with popular email clients or services (currently Gmail). The plugin will leverage the email service's API to access the user's inbox, fetch emails, and send them to the backend server for processing. The plugin will handle authentication, API requests, and data transfer while ensuring user privacy and security.

4.2 AUTHENTICATION AND AUTHORIZATION

The plugin will implement OAuth 2.0 or a similar protocol to authenticate the user and obtain the necessary permissions to access their email data. This will ensure that the plugin operates within the boundaries of the user's privacy settings and adheres to the email service's security requirements.

4.3 API INTEGRATION

The plugin will use the email service's API to fetch emails from the user's inbox. It will handle API requests and data retrieval, ensuring that all relevant emails are fetched and sent to the backend server for processing.

4.4 DATA TRANSFER

The plugin will send the fetched email data to the backend server using a secure communication protocol, such as HTTPS or SSL/TLS. This ensures that the email data remains encrypted and protected during transit.

4.5 SERVER-SIDE PROCESSING

Upon receiving the email data, the backend server will preprocess the emails and extract relevant features using the data preprocessing and feature extraction subsystems. Then, the classification models subsystem will apply various detection techniques, such as rule-based filtering, machine learning, and deep learning, to classify the emails based on their threat levels.

Some of the AI/ML models that are used are described below.

4.5.1 Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are a class of deep learning models primarily used for image and text recognition tasks. A typical CNN architecture consists of a series of convolutional layers, followed by pooling layers, fully connected layers, and an output layer. In the context of text analysis, CNNs can automatically learn hierarchical representations of input data by scanning the text with filters of varying sizes, allowing the model to capture meaningful patterns and features.

In our project, we used a CNN to process email text for spam detection. The CNN learns to recognize patterns in the text that differentiate spam emails from legitimate ones. These patterns are then used to classify new, unseen emails as spam or not spam.

4.5.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) specifically designed to address the vanishing gradient problem in standard RNNs. This allows them to learn long-term dependencies in sequence data more effectively. LSTMs consist of memory cells that can store information over long sequences, along with input, output, and forget gates that regulate the flow of information into and out of the memory cells.

In our project, we used an LSTM model to process email text for spam detection. The LSTM learns to recognize patterns in the text and can capture long-range dependencies that may be indicative of spam emails. This helps the model to classify new, unseen emails as spam or not spam based on the patterns it has learned.

4.5.3 BERT

BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained deep learning model developed by Google AI for natural language understanding tasks. BERT is based on the Transformer architecture and utilizes self-attention mechanisms to process input data. The key innovation in BERT is its bidirectional training, which allows the model to learn contextual relationships between words in both directions, resulting in a deeper understanding of the text.

In our project, we fine-tuned BERT for the email spam detection task using our training data. BERT's ability to understand the context and semantic relationships between words in the text helps the model differentiate between spam and legitimate emails. This allows the model to classify new, unseen emails as spam or not spam based on the patterns it has learned.

4.5.4 Random Forest Classifier

The Random Forest Classifier is an ensemble learning method that constructs multiple decision trees during training and combines their predictions to produce a more accurate and robust classification. The key idea behind Random Forest is that a group of "weak learners" (individual decision trees) can work together to form a "strong learner" (the ensemble). During training, each decision tree in the Random Forest is built using a random subset of the training data, which helps to decorrelate the trees and reduce overfitting.

In our project, we used the Random Forest Classifier to process email features and identify spam based on the patterns learned from the training data. The ensemble nature of the Random Forest Classifier makes it less prone to overfitting and more accurate in classifying new, unseen emails as spam or not spam.

4.6 DATA COLLECTION AND PREPROCESSING:

Data is the foundation for building a robust email security solution. The first step in the methodology involves gathering a large dataset of emails, including legitimate emails, spam, phishing, and emails with malware attachments.

Example:

Collect emails from publicly available sources, such as the Enron email dataset and SpamAssassin dataset, as well as internal organizational archives with permission.

Preprocessing involves cleaning and transforming the raw data into a suitable format for further analysis. This may include removing unnecessary metadata, converting email contents to lowercase, tokenizing words, and removing stop words.

Example:

Use Natural Language Processing (NLP) libraries, such as NLTK or spaCy, to tokenize email text and remove stop words.

4.7 FEATURE EXTRACTION AND SELECTION:

The next step involves extracting relevant features from the preprocessed email data. These features can be used to train machine learning and deep learning models for threat detection.

Example:

Extract features like email header information, sender's domain, embedded links, attachment types, and word frequencies.

Feature selection involves selecting the most relevant features that contribute to accurate threat detection. This can be done using various techniques, such as information gain, chi-square test, or recursive feature elimination.

Example:

Use the chi-square test to identify the most relevant features for classification.

4.8 MODEL DEVELOPMENT AND INTEGRATION:

With the selected features, develop machine learning and deep learning models to identify and classify email threats. Split the dataset into training and testing subsets for model evaluation.

Example:

Train a Naive Bayes classifier, a Support Vector Machine (SVM) model, and a Long Short-Term Memory (LSTM) neural network on the training data.

Integrate the rule-based filtering, machine learning, and deep learning models into a comprehensive email controls and detection engine. Develop a framework that combines the outputs of these models to make a final decision on the email threat classification.

Example:

Use a weighted voting mechanism where each model's output contributes to the final decision based on its performance during testing.

4.9 MODEL EVALUATION AND OPTIMIZATION:

Evaluate the performance of the email controls and detection engine using various metrics, such as accuracy, precision, recall, and F1-score. Compare the results against existing email security solutions to determine the effectiveness of the developed engine.

Example:

Use a confusion matrix to calculate the mentioned metrics and identify areas for improvement.

Optimize the models and the overall engine by tuning hyperparameters, refining features, or adjusting the decision-making mechanism to minimize false positives and negatives.

Example:

Perform a grid search for optimal hyperparameters in the SVM model to improve its accuracy.

4.10 DEPLOYMENT AND CONTINUOUS IMPROVEMENT:

Deploy the email controls and detection engine in a real-world setting, such as an organization's email server, and monitor its performance. Continuously update and improve the models by incorporating new data and feedback from real-world deployments. This will help the email security solution adapt to evolving threats and maintain its effectiveness over time.

Example:

Implement a feedback loop where users can report false positives or negatives, allowing the models to learn from these instances and improve their accuracy.

Chapter 5

DETAILED DESIGN AND ARCHITECTURE

5.1 SYSTEM OVERVIEW

This section provides detailed system architecture of the Email Controls and Detection Engine. An overview of system modules, their structure, and their relationships are described in this section. User interfaces and related issues are also discussed.

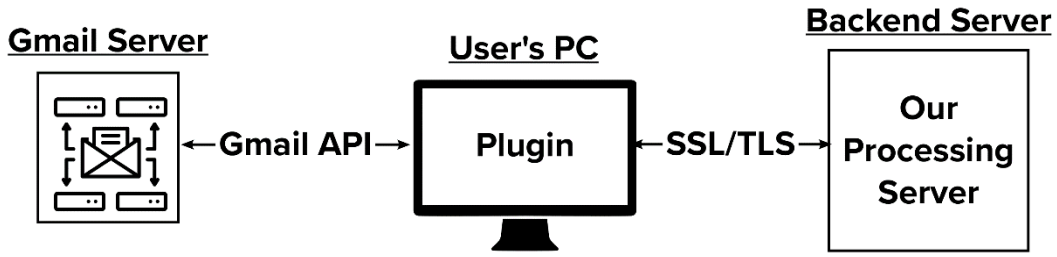

Figure 2: System Overview for Processing of an email.

5.1.1 Modules Overview

Email Controls and Detection Engine System requires several modules to work. These modules can be divided into three parts, the Plugin, Reporting Panel and Back-End Server. Following is a brief overview of all these modules. Detailed descriptions of these modules are presented in section 3.

5.1.2 Plugin

This module would be designed in HTML, CSS, and JavaScript. It has multiple features. Get emails, check connection to server and reports panel. The get emails will fetch the emails and show in your inbox, check connection to server return whether you have an active connection with the backend server or not, and on clicking the reports panel will move you to the reporting panel where the results are displayed.

5.1.3 Back-end Server

This module would be a Python based server. The backend server also has multiple features. Email Parser, Body Analysis, Structure Analysis, Reports Generation, URLs Analysis, and Meta Analysis. The email parser would parse the email into multiple tokens e-g from, To etc. The server finds out the attachment and URLs from the body of the email and then check the maliciousness of the URLs and attachment through VirusTotal API. After the complete analysis of the backend server the backend server would send the result back to the reporting panel.

5.1.4 Reporting Panel

This module would be a simple landing page designed with HTML, CSS and JavaScript where the complete detailed report of the email would be displayed which is received from the server. The user would view or download the report then from the Reporting Panel.

5.2 SYSTEM ARCHITECTURE

This section covers the overall architectural description of Emails Control and Detection Engine. It includes the high-level and low-level descriptions of the project including block diagrams of the application. Moreover, a complete object-oriented description includes class diagrams, sequence diagrams, and others. Finally, the rationale for the design pattern is provided.

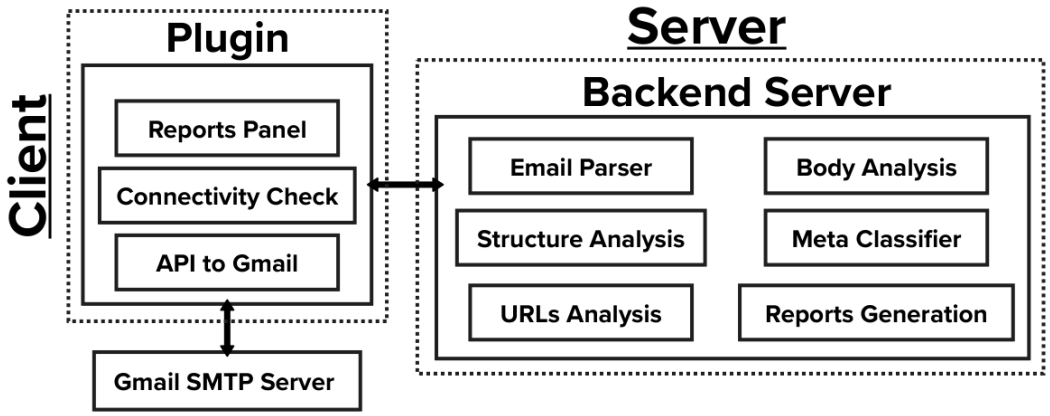

Figure 3: Client Server Architecture Diagram with sub components.

5.2.1 Architectural Design

The architectural design of an Emails Control and Detection Engine would describe the overall structure of the system, as well as the relationships between the different components or modules that make up the system. This might include information such as:

5.2.2 Decomposition Description

Decomposition is the process of breaking down a system or problem into smaller, more manageable parts. In the context of the Email Controls and Detection Engine, decomposition might involve dividing the plugin into separate components or modules that each handle a specific aspect of its functionality.

For example, the plugin could be decomposed into the following components:

5.2.2.1 User interface:

This component would handle the display of the plugin and any user interactions with it.

5.2.2.1 Plugin:

This component would be responsible for analyzing incoming emails for potential threats and sending requests to the backend server for further processing.

5.2.2.2 Gmail SMTP:

This component would handle the incoming and outgoing requests to google server for emails.

5.2.2.3 Backend Server:

This component would be responsible for analyzing, scanning, and generating reports of the emails.

By decomposing the Email Controls and Detection Engine into smaller parts, it becomes easier to design, develop, and maintain the plugin. It also allows for more modular and flexible design, as changes or updates to one component can be made without affecting the others.

5.2.2.4 Sequence Diagram

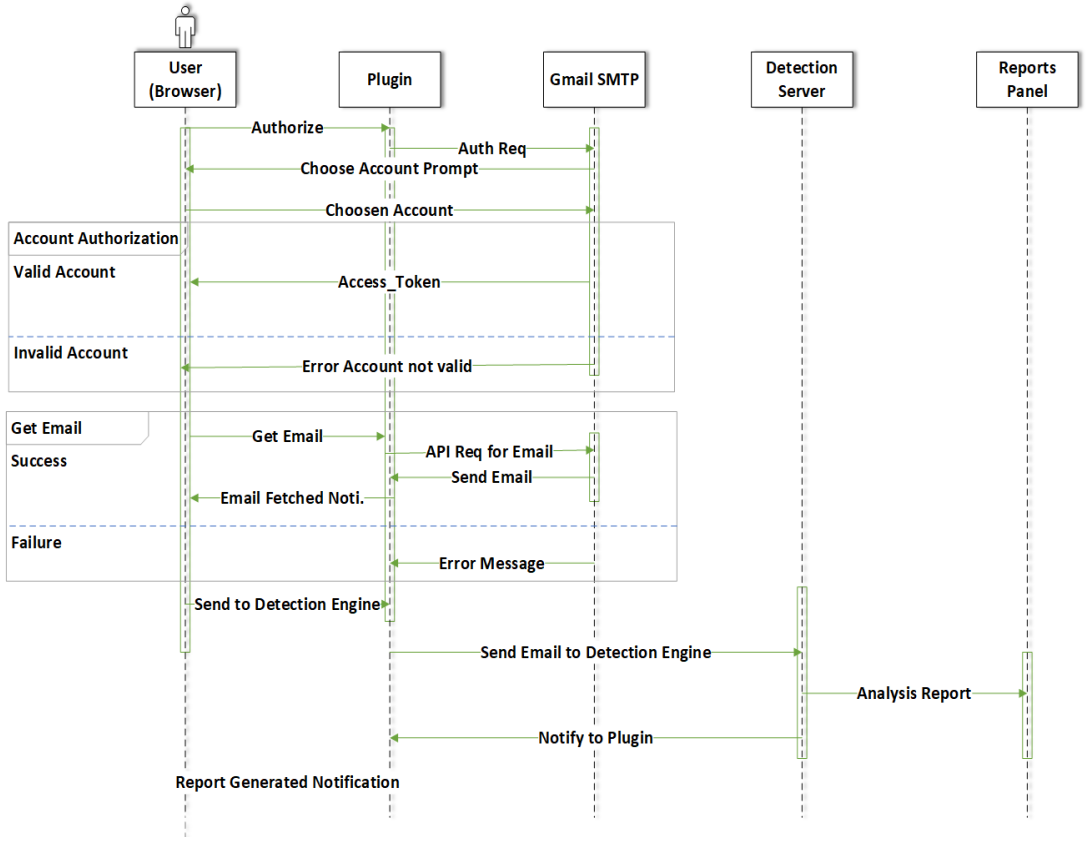

Figure 4: Sequence Diagram for Account, Authorization, Get Email, Send Email and Generate Report.

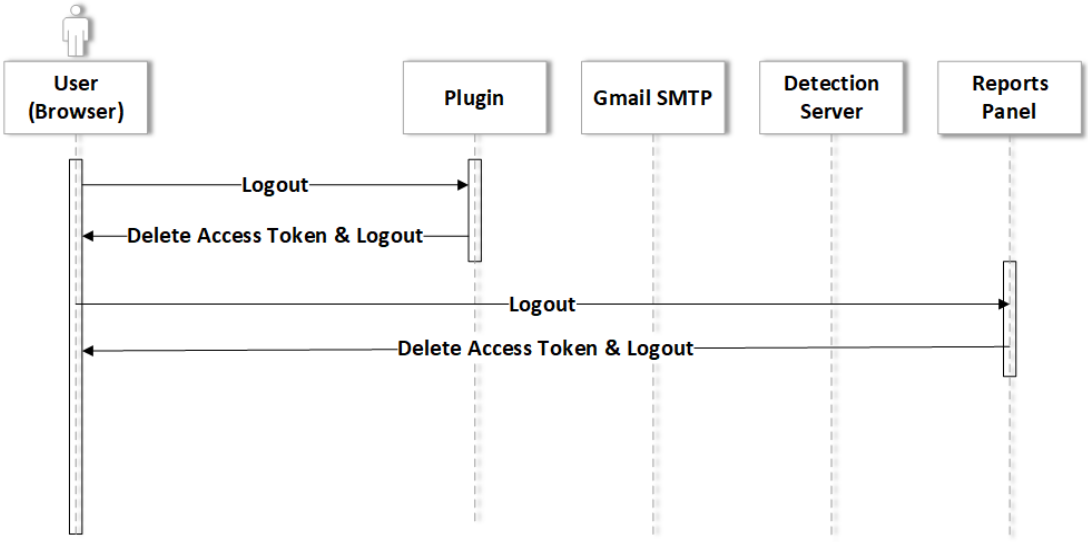

Figure 5: Sequence Diagram for logging out.

5.2.2.5 Dynamic View (Activity Diagrams)

Plugin Activity Diagram

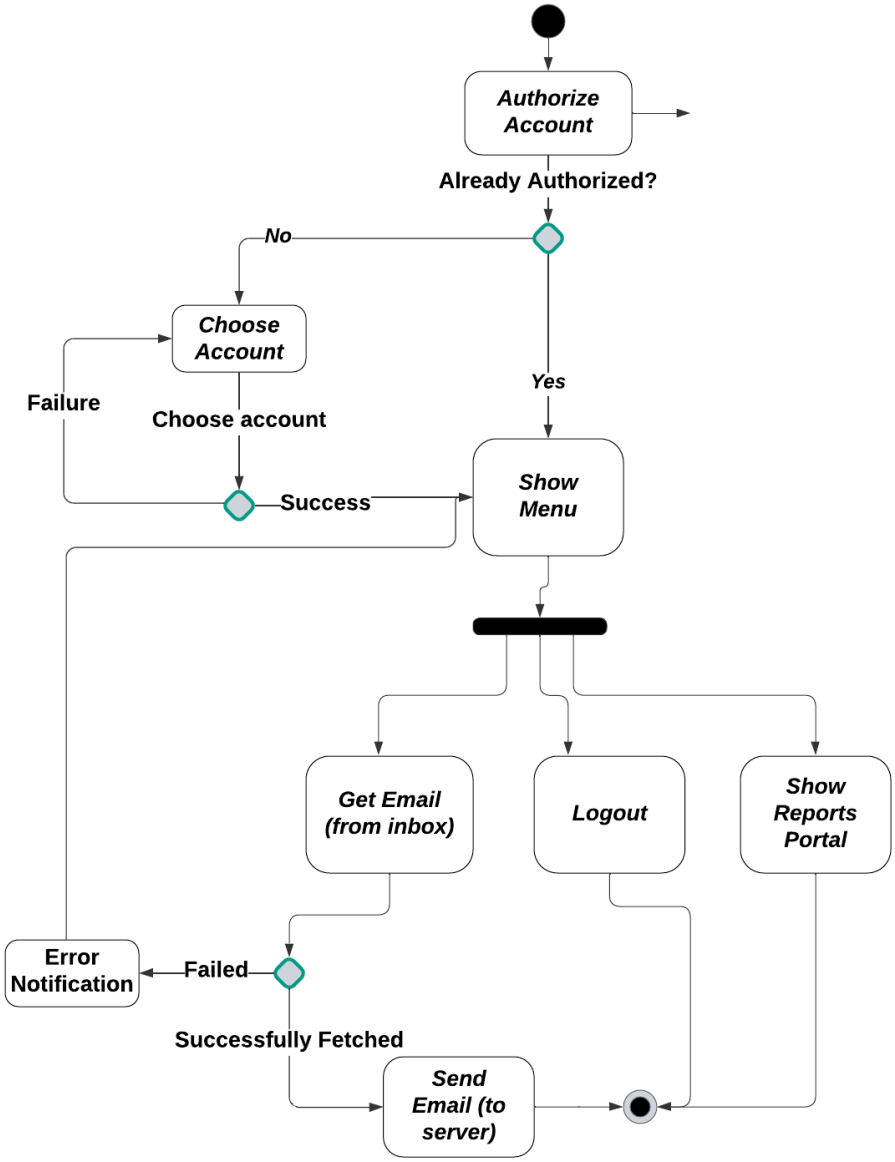

Figure 6: Activity Diagram for plugin.

Server Activity Diagram

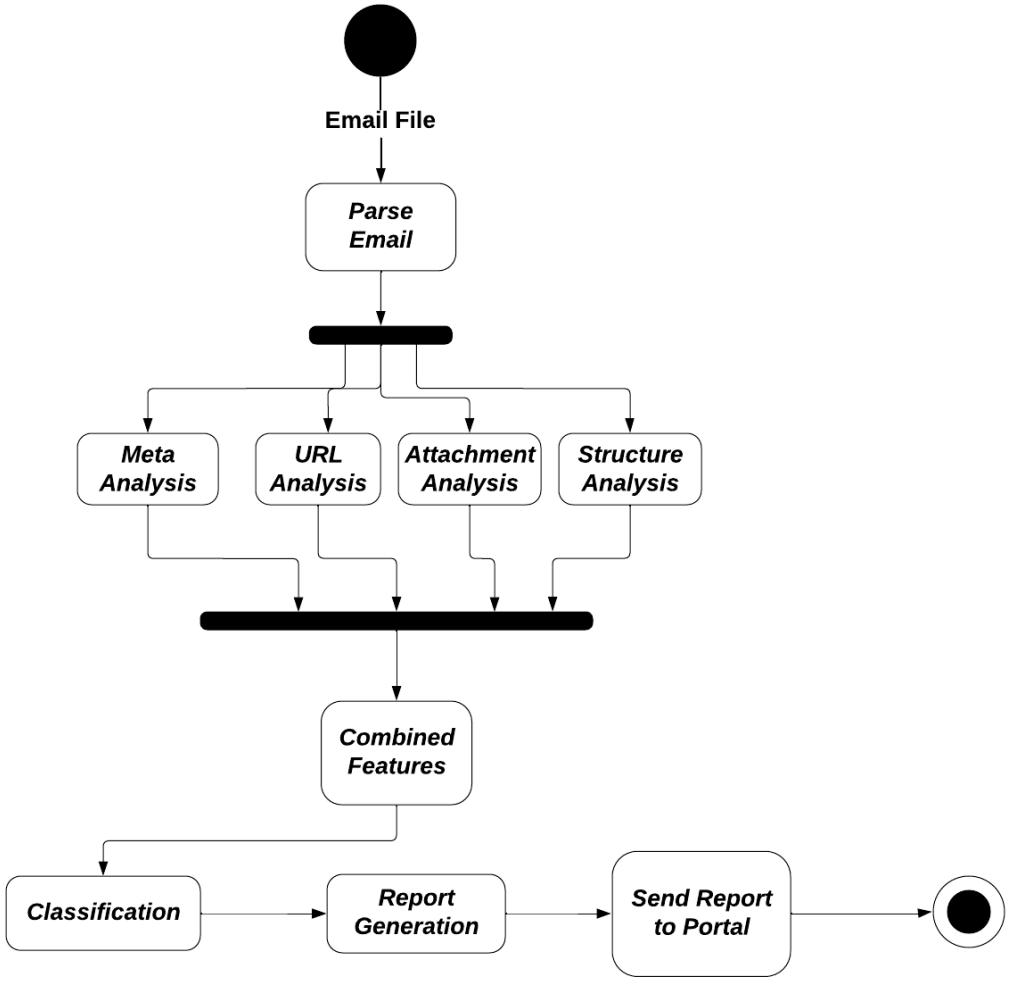

Figure 7: Activity Diagram for Server.

5.3.1 Design Rationale

Some of the reasons for choosing the Main and subprogram architecture are given below:

The Project is divided into smaller modules so it's easy-to-understand flow of data.

5.2.3.1 Client Server Architecture:

With a client-server architecture, a central server can be used to manage access to resources and services, allowing for better control and security. Also, it allows multiple clients to access shared resources and services, making it easier to share data and information between devices or users.

5.2.3.2 Backend Server:

It is a Python based server and will be responsible for analyzing, scanning, and generating reports of the emails. Also, it allows multiple clients to access shared resources and services, making it easier to share data and information between multiple devices.

5.2.3.3 Plugin:

It is any way to integrate our solution as it can be installed on any modern web browser. Hence, making it easy for users to continue with their work while using plugin.

5.3 DATA DESIGN

5.3.1 Data Description

Our system takes all the data from the email file which is in the form of .eml which the plugin will send to the backend server. The server parses the .eml file and analyze the multiple parts of the email separately. The multiple parts include meta, attachment, and structure of an email. The backend server after thoroughly analyzes the different parts of an email and sends the result back to the reporting panel in the form of Boolean.

5.3.2 Data Dictionary

Data used in our program is described with their type below:

- Delivered To: Boolean (Extracted from email)
- Received: string (Extracted from email)
- Return-Path: string (Extracted from email)
- From: string (Extracted from email)
- To: string (Extracted from email)
- Date: string (Extracted from email)
- Subject: string (Extracted from email)
- To: string (Extracted from email)
- Cc: string (Extracted from email)
- Content-Type: string (Extracted from email)
- Attachment: int (Extracted from email)
- text: string (Extracted from email)
- is_spam: boolean (used to tell whether the email is spam or ham)
- is_connect_server: boolean (used to check connection from the server)
- is_complete_report: boolean (used to notify after complete analysis of report)

5.4 COMPONENT DESIGN

For optimal functioning of the application, the induvial and various components of the system must be working at their best and their interactions should be going smoothly. Let's look at how the components interact with each other.

5.4.1 Component Diagram

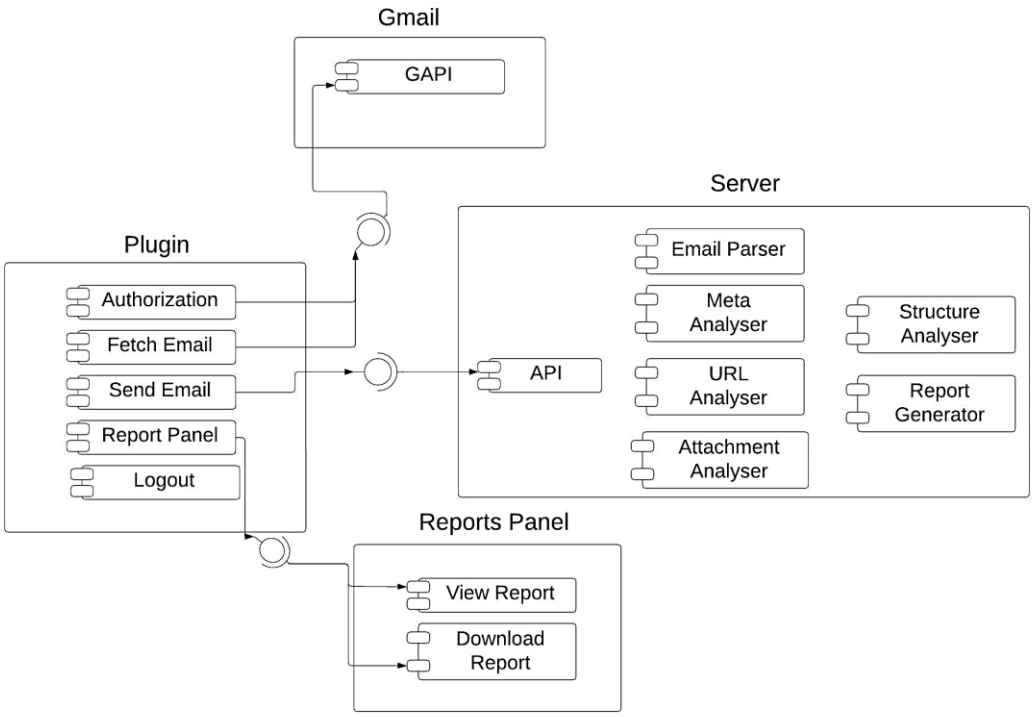

Figure 8: Component diagram for overall system.

5.4.2 Package Diagram

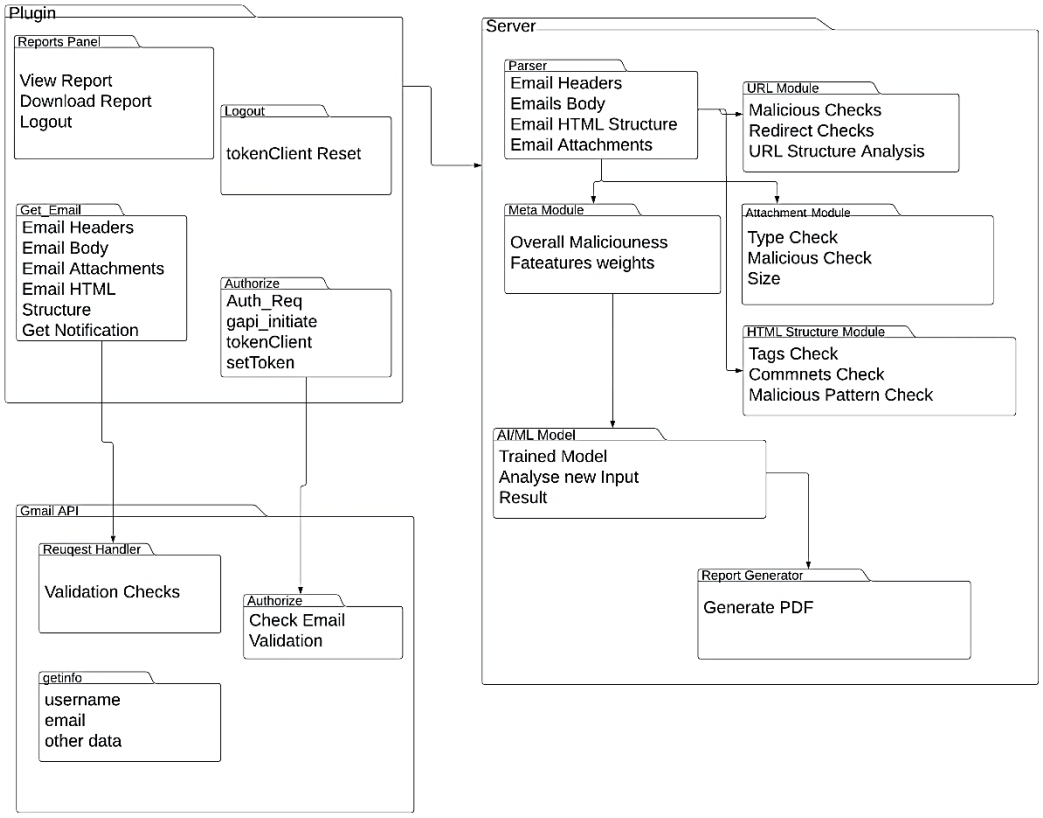

Figure 9: Package Diagram for overall system.

5.5 HUMAN INTERFACE DESIGN

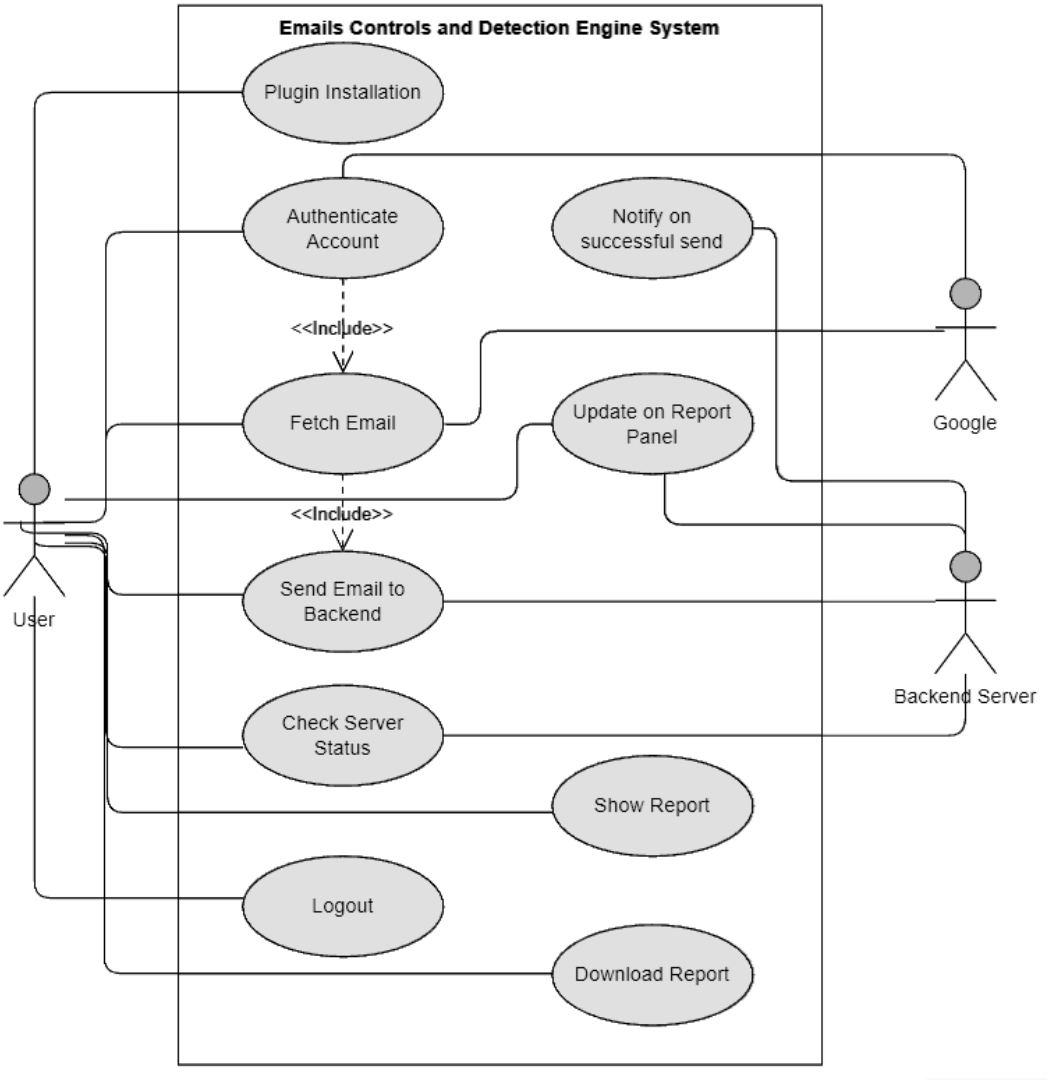

Figure 10: Usecase Diagram for overall system.

5.5.1 Use Cases

In this section, we cover, in detail, the major use cases of the Emails Control and Detection Engine.

Use Case Name	Plugin Installation
Actor	User

Trigger	The User Clicks on the Add to Chrome Button
Precondition	The User has a chromium-based web browser
Activity Flow	1. Open the browser in which they want to install the plugin. 2. Navigate to the plugin or extension manager of the application. This can usually be found in the settings or preferences menu. 3. Search for the plugin by name or browse the available plugin options. 4. Select the plugin and click the install button. 5. Follow any additional prompts or instructions for installing the plugin. 6. Once the plugin is installed, it may need to be activated or enabled within the application. 7. Restart the application to complete the installation process.
Postcondition	The Plugin will now be accessible from the top of the browser

Table 1: Usecase of Plugin Installation.

Use Case Name	Authenticate Account
Actor	User, Google
Trigger	The User clicked the Sign In with Google button
Precondition	The User has already installed the plugin
Activity Flow	1. The User opens the Plugin. 2. The plugin displays the login screen to the user.

	3. The User Sign In with google account. 4. The System proceeds with the login procedure.
Postcondition	The plugin receives the authentication token and displays the menu screen

Table 2: Usecase of Account Authentication.

Use Case Name	Fetch Email
Actor	User, Google
Trigger	The User clicked the Get Emails Button
Precondition	The User has Already logged in with Google Account
Activity Flow	1. Open the Plugin. 2. After Successful Authentication from your google account. 3. The user will click on the Get Emails Button. 4. After That All the Emails will be Shown On the plugin.
Postcondition	The plugin displays all the emails

Table 3: Usecase of Fetching Email.

Use Case Name	Send Email to Backend
Actor	User, Backend Server
Trigger	The User clicked the Send to Server
Precondition	The User has already logged in the plugin

Activity Flow	1. Open the Plugin. 2. After Successful Authentication from your google account. 3. The user will click on the Get Emails Button. 4. Once all the emails will be fetched. 5. The user will then select the emails and send the emails to the backend server.
Postcondition	The Backend sends the status.

Table 4: Usecase of sending email to backend.

Use Case Name	Notify on Successful Send
Actor	Backend Server
Trigger	The User Clicked the Send to Server
Precondition	The User has already logged in the plugin and Send the emails to the backend Server
Activity Flow	1. Open the Plugin. 2. After Successful Authentication from your google account. 3. The user will click on the Get Emails so the emails will display on the Pugin. 4. After successfully fetching the email then the user will send the emails to the backend server from plugin. 5. When the emails send successfully the plugin will notify you.
Postcondition	The status would be visible in the plugin

Table 5: Usecase notifying on successful send.

Use Case Name	Update on Reports Panel
Actor	User, Backend Server
Trigger	The User clicked the Reports Panel.
Precondition	The User has already logged in the plugin.
Activity Flow	1. Open the Plugin. 2. After Successful Authentication from your google account. 3. The user will click on the Report Panel. 4. The report panel will show all the completed reports.
Postcondition	The reporting panel would be displayed

Table 6: Usecase of updating Reports Panel.

Use Case Name	Show Report
Actor	User
Trigger	The User clicked the show reports in the reporting panel.
Precondition	The User has already logged in the app and the server would complete the analysis of the report
Activity Flow	1. Open the Plugin. 2. After Successful Authentication from your google account. 3. The user will click on the Report Panel. 4. The report panel will show all the complete reports and the reports which are under processing.

	5. After complete processing from the backend. The Reporting panel will show the complete report.
Postcondition	The report will be displayed in the reporting panel

Table 7: Usecase of showing report.

Use Case Name	Logout
Actor	User
Trigger	The User clicked the Logout Button
Precondition	The User has already logged in the app
Activity Flow	1. After Successfully login in the plugin. 2. You will be displayed with a logout Button. 3. When You click on the logout Button You will be moved To the Login screen.
Postcondition	The plugin removes the token from the local Storage

Table 8: Usecase of logging out.

5.5.2 User Interfaces

5.5.2.1 Sign In Page

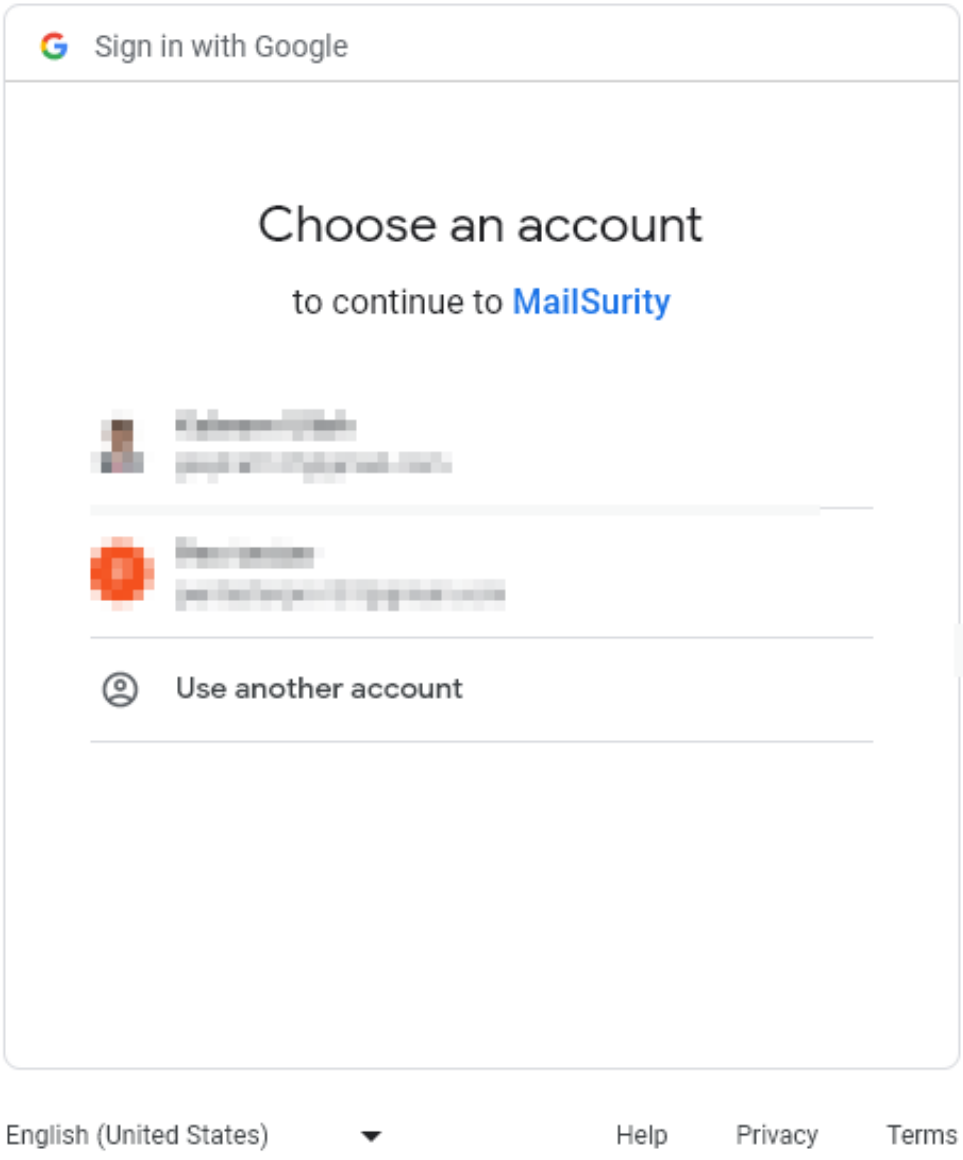

Figure 11: Sign In page of Google API.

5.5.2.2 Menu Page

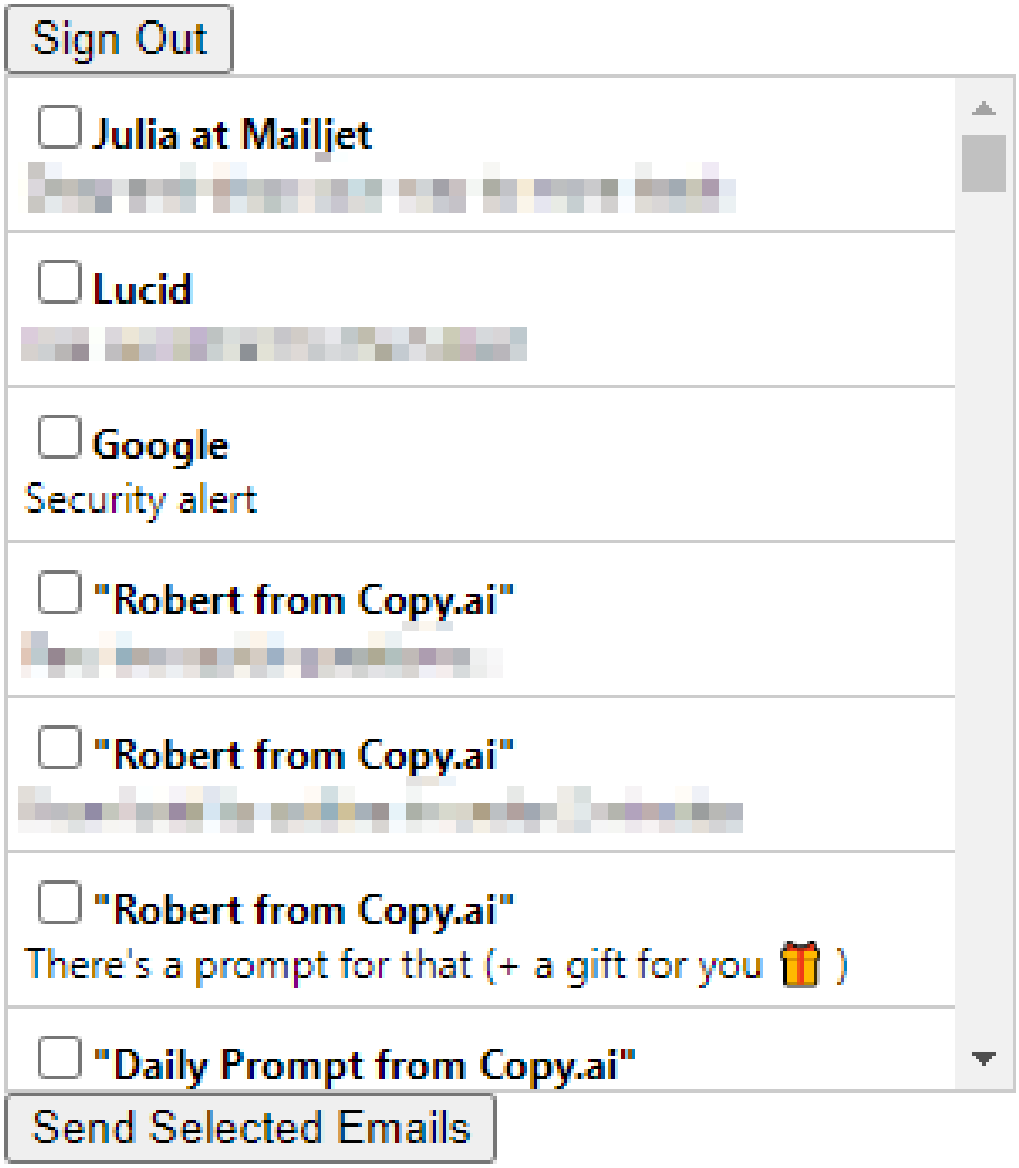

Figure 12: Menu Page of Plugin

5.5.2.3 Reporting Panel

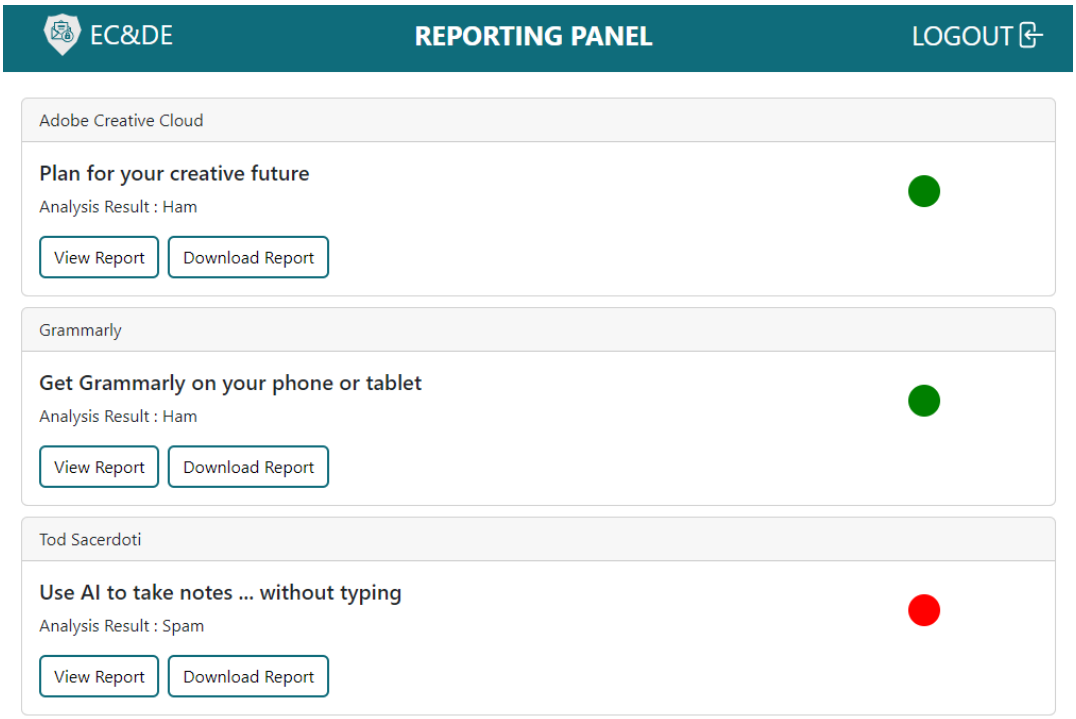

Figure 13: Reporting Panel

5.5.3 Screen Objects and Actions

5.5.3.1 Plugin: plugin consists of 3 components:

1. *Sign In Screen:*

   The Sign In Screen consists of Sign In with Google Button. When You Sign In with Google it moves to the menu screen.

2. *Menu Screen:*

   It consists of 3 buttons

   - Get Emails:

     After clicking on the get emails the plugin will show you the emails.

   - Reporting Panel:

After clicking the Reporting Panel, the reporting panel would appear and displays the result.

- Log Out:

  After Clicking the Logout button, the access token stored in the local storage would be removed.

3. *Display Email:*

The Display Email Screen lists all the emails with the checkbox on the side and includes a button named “Send to Server”. After clicking on this button, the emails select would be send to backend server.

5.5.3.2 Reporting Panel:

The Reporting Panel would display the complete detailed report of the .eml file which is received from the server after complete analysis of the report. The user would view or download the report then from the Reporting Panel.

Chapter 6

IMPLEMENTATION AND TESTING

6.1 INTRODUCTION:

The "Implementation and Testing" section aims to provide a comprehensive overview of the methods, tools, and techniques used during the development of the software, as well as the testing methodologies implemented to ensure its quality and adherence to the initial specifications. This section will cover aspects such as software development methodology, tools and technologies employed, and testing approaches of our software. By discussing these aspects, the section will offer insights into the development process, challenges encountered, and the measures taken to create a reliable and efficient software solution.

6.2 SOFTWARE DEVELOPMENT METHODOLOGIES:

Agile methodology has been chosen for this project because it is popular, iterative, and incremental approach to software development that emphasizes flexibility, collaboration, and customer satisfaction. Agile methodology positively impacts the development process and the quality of the final product.

In the context of our email analysis and reporting software, adopting Agile methodology provided significant benefits to our development process. Here's how we applied Agile specifically to our project:

6.2.1 Collaboration between teams:

Agile promoted close collaboration between us, the developers working on the email analysis functionality, the reporting panel team (including frontend and backend developers), and the Chromium extension developers. This ensured seamless integration between components and clear communication about requirements and changes.

6.2.2 Iterative development for each component:

We developed each component of the software (email analysis, reporting panel, and Chromium extension) iteratively. We focused on delivering small, functional parts of each component in sprints, allowing us to evaluate their effectiveness and make improvements based on user feedback.

6.2.3 Prioritization of features:

Agile enabled us to prioritize the most critical features for each component. For example, we prioritized the development of essential email analysis functionality, followed by basic reporting panel features, and finally the Chromium extension. This ensured that the most important aspects of the project were developed first, delivering maximum value to users.

6.2.4 Continuous feedback and improvement:

By releasing small increments of each component, we gathered user feedback on the email analysis results, reporting panel usability, and the Chromium extension's effectiveness. This feedback helped us identify issues, make improvements, and validate that the software was meeting user needs.

6.2.5 Adaptability:

Agile allowed us to adapt to changing requirements or new discoveries more efficiently. If we needed to add new analysis features or modify the reporting panel based on user feedback, Agile enabled us to adjust our development plan and priorities accordingly.

In summary, Agile methodology helped us develop our email analysis and reporting software more effectively by promoting collaboration, iterative development,

prioritization of features, continuous feedback, and adaptability. This ensured that our software met user needs and delivered value throughout the development process.

6.3 TOOLS AND TECHNOLOGIES:

For every component we have used different tools and technologies. This section will provide details of tools and technology used in each component.

6.3.1 Chromium Extension:

Programming Language:

JavaScript - The primary language used for the development of this Chromium extension. JavaScript is versatile and widely used for web development, making it suitable for creating browser extensions.

Framework:

None - This specific module does not utilize a particular framework. It relies on plain JavaScript to handle the extension's logic and interact with Google APIs.

Libraries:

Manifest V3: The latest version of the manifest file format used for Chromium extensions. It helps define the extension's metadata, permissions, and other necessary configurations.

6.3.2 Reporting Panel:

Programming Language:

JavaScript (JS)

Library:

React.js (A popular JavaScript library for building user interfaces)

UI Framework:

React-Bootstrap (A UI component library built with React and based on the Bootstrap framework)

Routing:

React Router DOM (A library for handling routing in React applications)

6.3.3 Backend Server:

Programming Language:

Python

Web framework:

Quart

Libraries:

- asyncio: Asynchronous I/O support for concurrent operations.
- csv: Reading and writing CSV files.
- base64: Encoding and decoding binary data using base64.
- collections: Container data types.
- os: Operating system interfaces.
- re: Regular expression operations.
- urllib.parse: URL parsing and manipulation.
- aiohttp: Asynchronous HTTP client/server framework.
- email: Parsing and handling email messages.
- bs4 (BeautifulSoup): HTML and XML parsing.

- string: Common string operations.
- google.oauth2.credentials: Google OAuth 2.0 authentication.
- googleapiclient.discovery: Google API client library.
- jwt: JSON Web Token support.
- functools: Higher-order functions and operations on callable objects.
- datetime: Basic date and time types.
- reportlab: PDF processing library.
- quart_cors: Cross-origin resource sharing (CORS) support for Quart.

Coding Conventions:

- Consistent indentation: The code consistently uses two spaces for indentation, providing a clean and organized code structure.
- Descriptive variable and function names: The variables and functions in the code are given meaningful names that describe their purpose, making the code more readable and maintainable.
- Proper use of async/await: The code makes good use of async/await to handle asynchronous operations, making the code easier to read and understand.
- Modularity: The code is organized into separate functions, each with a specific purpose, promoting modularity and maintainability.

6.3.4 Tools

IDEs:

- Visual Studio Code (VSCode): A popular and versatile code editor used for writing, editing, and debugging the JavaScript code for your extension.

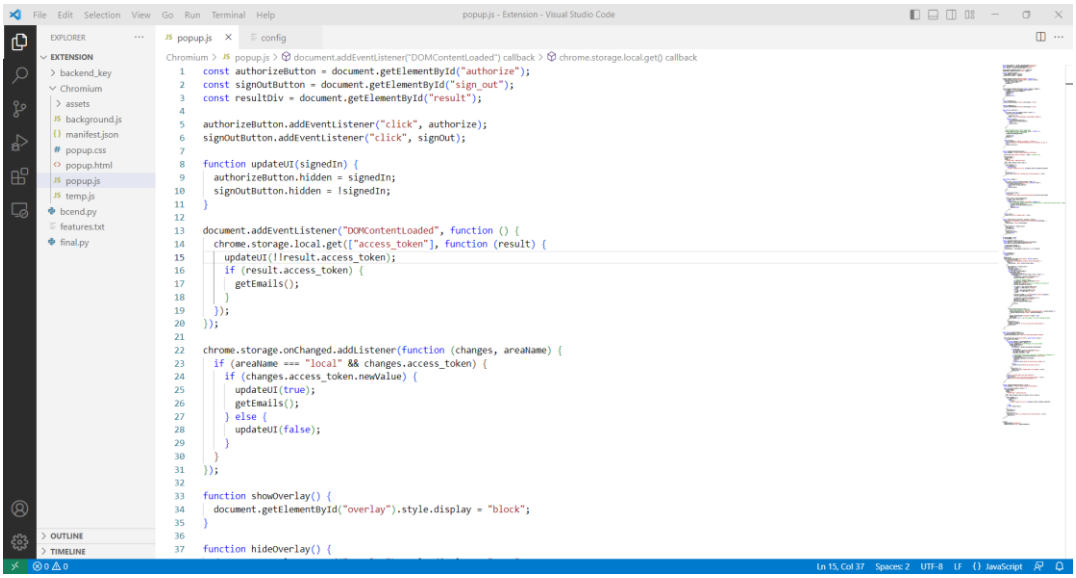

Figure 14: Visual Studio Code (just for showing UI of IDE)

Server:

- NPM (8.18.0)
- Python HTTP (3.10.2)

OS:

- Linux (Ubuntu 20.04.5 LTS)

6.4 TESTING METHODOLOGY:

In software development, testing is a critical step to ensure the quality and reliability of the application. There are several testing methodologies commonly used in the industry, including unit testing, integration testing, and system testing. we will briefly explain each methodology that we have used in our project are.

Here are three to four specific tests for each type of testing for our email analysis and reporting software project:

6.4.1 Unit Testing:

Unit testing focuses on verifying the functionality of individual components or functions in isolation. This methodology helps identify and fix bugs early in the development process. Some of the tests are written below.

6.4.1.1 Test 1: Fetch function

Description: Test the `fetch()` function to ensure it returns the correct JSON response when given valid input.

Expected Result: A JSON response containing the email data.

Status: Passed

6.4.1.2 Test 2: Email parsing function

Description: Test the email parsing function for extracting the correct sections and components from an email.

Expected Result: The email is parsed into its individual components (plain, HTML, application, and image sections).

Status: Passed

6.4.1.3 Test 3: PDF report generation

Description: Test the PDF report generation function to ensure it generates the report with the correct formatting and data.

Expected Result: A correctly formatted PDF report containing the analysis results.

Status: Passed

6.4.2 Integration Testing:

Integration testing focuses on verifying the interaction between different components or modules of the application. This methodology ensures that the integrated components work together as expected. Some of the tests are written below.

6.4.2.1 Test 1: OAuth token-based authorization

Description: Test the integration of the OAuth token-based authorization between the frontend and backend systems.

Expected Result: Only users with valid OAuth tokens can access the protected routes, while unauthorized users are denied access.

Status: Passed

6.4.2.2 Test 2: API endpoints and data flow

Description: Test the correct functioning and data flow between the frontend, backend, and various API endpoints.

Expected Result: Data is correctly sent and received between the frontend, backend, and APIs, with the appropriate processing applied at each stage.

Status: Passed

6.4.2.3 Test 3: Chromium extension integration

Description: Test the integration of the Chromium extension with the backend and the reporting panel.

Expected Result: The extension communicates effectively with the backend and reporting panel, sending email data for analysis and displaying the results.

Status: Passed

6.4.3 Functional Testing:

In this section, We tested the full workflow of analyzing an email, generating a report, and displaying it on the reporting panel to ensure the system met its functional requirements. We tested the authorization mechanism to ensure that only authorized users could access the reports. Some of the tests are written below

6.4.3.1 Test 1: Upload and analyze an email

Description: Test the ability to upload an email, which should trigger the analysis process and result in the generation of a PDF report.

Expected Result: The email analysis is successfully completed, and a PDF report is generated.

Status: Passed

6.4.3.2 Test 2: Access and download PDF report

Description: Test the ability to access and download the generated PDF report from the reporting panel.

Expected Result: The PDF report is accessible and downloadable from the reporting panel.

Status: Passed

6.4.3.3 Test 3: Filter functionality

Description: Test the filter functionality within the reporting panel, enabling users to filter email reports based on various criteria.

Expected Result: Users can successfully apply filters to the email reports, displaying only the relevant reports that meet the specified criteria.

Status: Passed

6.4.4 Security Testing:

In security testing, we tested the OAuth token-based authorization to ensure that the system was secure and protected against unauthorized access. We checked for potential vulnerabilities like SQL injections, XSS, or CSRF attacks by testing the input validation and sanitization in our backend API.

6.4.4.1 Test 1: Invalid OAuth token handling

Description: Test the handling of invalid or expired OAuth tokens to ensure that unauthorized users are denied access to the system.

Expected Result: Unauthorized users with invalid or expired tokens are denied access, and an error message is displayed.

Status: Passed

6.4.4.2 Test 2: Input validation

Description: Test the input validation mechanisms in the system to prevent common security vulnerabilities such as SQL injection, cross-site scripting, and buffer overflows.

Expected Result: The system as such does not takes any input but in URLs if an attacker uses the injections then it denies the request.

Status: Passed

6.4.4.3 Test 3: Secure data transfer

Description: Test the security of data storage mechanisms, such as encryption and access controls, to ensure the confidentiality and integrity of user data.

Expected Result: User data is securely transferred and protected from unauthorized access and tampering. No direct sniffing is possible.

Status: Passed

6.4.5 Performance Testing:

In performance testing we tested the system's performance when analyzing a large email with numerous attachments and sections. We tested the reporting panel's loading time when displaying a list of reports under heavy load.

6.4.5.1 Test 1: Load testing

Description: Test the system's ability to handle a large number of simultaneous requests without degrading performance.

Expected Result: The system performs well under heavy load, maintaining an acceptable response time and throughput.

Status: Passed

6.4.5.2 Test 2: Latency testing

Description: Test the latency of the system, ensuring that response times are within acceptable limits for users.

Expected Result: The system's latency is within acceptable limits, providing a good user experience.

Status: Passed

6.4.6 System Testing:

System testing focuses on verifying the application as a whole, from end to end. This methodology tests the complete functionality of the application, including the frontend and backend components, under different scenarios.

System testing is performed through a strong testing strategy and the test cases cover all the use cases.

6.4.6.1 Plugin Installation:

Test Description: Verify that the plugin can be successfully installed on a chromium-based web browser.

Expected Result: The plugin is installed, activated, and accessible from the browser toolbar.

Status: Passed

6.4.6.2 Authenticate Account:

Test Description: Verify that the user can log in using their Google account and the plugin receives the authentication token.

Expected Result: The user is authenticated, and the menu screen is displayed.

Status: Passed

6.4.6.3 Fetch Email:

Test Description: Verify that the plugin can fetch emails from the user's Google account and display them in the plugin.

Expected Result: Emails are fetched and displayed within the plugin.

Status: Passed

6.4.6.4 Send Email to Backend:

Test Description: Verify that the user can select emails and send them to the backend server for analysis.

Expected Result: Selected emails are sent to the backend server successfully, and the server confirms the receipt.

Status: Passed

6.4.6.5 Notify on Successful Send:

Test Description: Verify that the plugin notifies the user when emails are sent successfully to the backend server.

Expected Result: The plugin displays a status update indicating the successful sending of emails.

Status: Passed

6.4.6.6 Update on Report Panel:

Test Description: Verify that the reporting panel updates with the completed reports and any reports that are still being processed.

Expected Result: The reporting panel displays the report statuses accurately.

Status: Passed

6.4.6.7 Show Report:

Test Description: Verify that the user can view the completed reports in the reporting panel.

Expected Result: Completed reports are displayed in the reporting panel.

Status: Passed

6.4.6.8 Download Report:

Test Description: Verify that the user can download completed reports as PDF files to their local system.

Expected Result: The user can download the completed reports, and the files are saved in the local system.

Status: Passed

6.4.6.9 Logout:

Test Description: Verify that the user can log out of the plugin, and the authentication token is removed from local storage.

Expected Result: The user is logged out, and the token is removed from local storage.

Status: Passed

These tests cover every use case in your project, ensuring that each functionality works as intended and provides the desired results.

Chapter 7

RESULTS AND DISCUSSION

In this section, we present the results and discuss the performance of various machine learning models that were employed in our project for email spam detection. We used the following models for training.

7.1 STRUCTURE ANALYSIS:

This module analyzes the existence and values of email headers and HTML content structure. It uses features such as the number of hyperlinks, unique domain names in HTML assets, linked URLs, and the tree structure of the Document Object Model. It employs machine learning algorithms like RandomForest, XGBoost, SVM (SVC), and Naive Bayes for feature modeling.

Model	AUPRC	Recall (FPR=10^-3)	Train (s)	Test (ms)
RandomForest	0.8993	0.8933	5	0.01
XGBoost	0.8994	0.8884	10	0.01
SVM (SVC)	0.8969	0.8618	919	0.55
Naive Bayes	0.8940	0.0	2	0.01

Table 9: Performance Matrix of Structure Analysis Module.

7.2 TEXT ANALYSIS:

This module predicts the likelihood of sentences in an email belonging to a phishing email. It uses the Bidirectional Encoder Representations from Transformers (BERT) model for text analysis. Preprocessing includes language detection, and the module focuses on modeling English text.

Model (BERT+)	AUPRC	Recall (FPR=10^-3)	Train (s)	Test (ms)
RandomForest	0.8757	0.7796	61	0.01
XGBoost	0.8746	0.6995	560	0.02
SVM (SVC)	0.8310	0.0776	48392	8.44

Naive Bayes	0.7353	0.0	3	0.02

Table 10: Performance Matrix of Text Analysis Module.

7.3 URL ANALYSIS:

This module classifies URLs in phishing emails without external reputation sources. It uses a deep learning approach with a combined architecture of Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) to capture differentiating aspects of phishing URLs on a character/word level.

Architecture	AUPRC	Recall (FPR=10^-3)	Train (sec)	Test (ms)
CNN	0.8406	0.5775	302	0.76
LSTM	0.8149	0.5787	7728	14.41
CNN-LSTM	0.8851	0.7648	4247	7.85

Table 11: Performance Matrix of URL Analysis Module.

7.4 META-CLASSIFIER:

This module aggregates the predictions of the individual modules into a single prediction probability for each email. It takes the prediction probabilities from the independent modules as a feature vector, and a supervised learning algorithm is employed to build a classification model for predicting phishing emails. The meta-classifier uses XGBoost, as it is more flexible in configuring thresholds for extremely low false-positive rates.

As there are no significant differences among well-known classifiers, the table for meta-classifier performance is not provided. However, XGBoost is chosen for its flexibility and low false-positive rates.

These four modules work together to effectively classify emails as phishing or benign based on their structure, text content, and URLs.

Chapter 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

In our thesis, we presented an elaborate software solution that aimed to provide an efficient and accurate email analysis system, particularly focusing on spam detection. We carefully structured the backend of our software into various modules, including the URL Analysis module, Meta Classifier, Attachment Modules, and several other components. This modular approach enabled us to create a systematic and organized workflow, which significantly enhanced the email analysis process.

To further improve the capabilities of our software, we incorporated Artificial Intelligence and Machine Learning (AI/ML) techniques into the core of our system. We utilized state-of-the-art algorithms such as Long Short-Term Memory (LSTM) for sequence analysis, Convolutional Neural Networks (CNN) for feature extraction, Bidirectional Encoder Representations from Transformers (BERT) for natural language understanding, and Random Forest for ensemble learning. By integrating these advanced techniques, we were able to achieve a high level of accuracy and effectiveness in our spam detection system.

In addition to the technical aspects, we paid close attention to the design and implementation of our software. We ensured that the various components were developed in a modular and maintainable manner, which allowed for seamless integration and adaptability to future advancements in the field of email analysis and spam detection.

Throughout the development process, we continuously tested and refined our algorithms and system components. This iterative approach allowed us to identify

and address potential issues, optimize system performance, and further enhance the overall capabilities of our software.

In conclusion, our software project demonstrated the immense potential of combining advanced AI/ML techniques with a well-structured backend system to create a powerful and efficient solution for email analysis and spam detection. The resulting system not only significantly improved spam detection capabilities but also provided a robust and flexible platform for further exploration and development in the ever-evolving field of email analysis. We believe that our work has laid a solid foundation for future research and innovation in this area, ultimately contributing to the ongoing battle against spam and other malicious email activities.

8.2 FUTURE WORK

As for future work, there are several directions we will consider:

8.2.1 Improve the email analysis algorithm:

By incorporating advanced machine learning techniques such as natural language processing and deep learning, we can enable the system to adapt to new phishing tactics and improve accuracy over time. This will ensure that our solution remains effective and relevant in the face of evolving spam email techniques.

8.2.2 Enhance the reporting panel:

Introducing features like sorting, filtering, and searching capabilities can allow users to better manage and navigate the analysis results. This will improve the overall user experience and make it easier for users to find relevant information in the analysis reports.

8.2.3 Implement support for other popular email providers:

By extending our solution to support other popular email providers like Outlook and Yahoo, we can broaden the user base and applicability of our solution. This will help ensure that more users can benefit from our spam email detection system, regardless of their email provider.

8.2.4 Develop the software for enterprises:

We plan to deploy our solution on middle servers for proactive analysis in enterprise settings. This will enable organizations to better protect their employees from spam and phishing emails, enhancing overall cybersecurity and reducing the risk of data breaches.

8.2.5 Expand to other email services:

In addition to supporting popular email providers, we aim to expand our solution to other email services, ensuring that our spam detection system can be used by a wider range of users and organizations.

8.2.6 Develop for mobile platforms:

We plan to develop our solution for both Android and iOS devices, making it more accessible and convenient for users who rely on mobile devices for email communication. This will help protect users from spam and phishing emails on their smartphones and tablets, further enhancing the security of email communication.

By addressing these future directions, we can continue to refine and expand our spam email detection system, ensuring that it remains a valuable tool for protecting users from unwanted and malicious emails.

Chapter 9

LIMITATIONS

One of the significant limitations of this research lies in the dataset used for training the machine learning models for the spam detection system. In machine learning, the performance and effectiveness of models are strongly influenced by the quality, size, diversity, and freshness of the data they are trained on. Let's dive deeper into these aspects:

9.1 SIZE OF THE DATASET

The size of the dataset is one of the most critical factors influencing the effectiveness and reliability of machine learning models. Training on a large and varied dataset helps the model learn and generalize patterns more efficiently. For our study, this would mean exposure to a wider array of email headers, HTML structures, text content, and URL patterns related to spam emails.

When the size of the dataset is relatively small, the model may not encounter enough variations of email content, limiting its ability to accurately predict and classify new instances. For example, suppose our dataset comprised only a few hundred emails, most of which contained similar spam tactics like false lottery winning notifications. In this case, the machine learning model might excel at identifying these specific types of spam emails but might falter when exposed to spam emails with more sophisticated tactics, such as intricate phishing attempts or more subtle marketing ploys.

A larger dataset could encompass thousands or even millions of emails, providing a broader spectrum of spam types, email structures, and tactics. Consequently, this could lead to a more well-rounded model capable of detecting an extensive array of spam emails in real-world scenarios.

9.2 QUALITY OF THE DATASET

The quality of the dataset significantly affects the training and performance of machine learning models. Quality refers to various factors, including cleanliness (absence of errors, omissions, and duplications), relevance (degree to which the data fits the purpose), and accuracy (extent to which the data correctly describes the 'real world' object or event being described).

In our study, suppose the dataset had incorrectly labeled examples where non-spam emails were marked as spam or vice versa. In that case, it could have led the model to learn incorrect patterns and associations, thereby reducing its effectiveness. For instance, if a set of genuine emails from a reputable company had been incorrectly labeled as spam, the machine learning model might learn to associate certain legitimate business communication patterns with spam activity. This could lead to a higher rate of false positives, where legitimate emails are incorrectly flagged as spam.

Moreover, if the dataset had missing or inconsistent information—for instance, missing email headers or varying formats in representing the same information—it could have complicated the learning process for the models, making it harder for them to extract meaningful features and patterns.

9.3 DIVERSITY OF THE DATASET

Diversity in the dataset is crucial for training robust machine learning models. It ensures that the model is exposed to a wide range of instances and variations. In the context of spam detection, diversity would mean having a wide array of spam emails with different content, structure, language, and tactics.

A lack of diversity in the dataset can lead to biased models. For instance, if our dataset primarily consisted of spam emails written in English, the model might have become very proficient at detecting spam emails in English but might underperform when exposed to spam emails written in other languages. Similarly, if the dataset mostly contained spam emails from a particular sector, like retail, the models might struggle to accurately classify spam emails from other sectors, like banking or healthcare.

Hence, having a diverse dataset with various spam email types can significantly enhance the effectiveness of the machine learning model by providing a broader understanding of spam patterns across different scenarios and contexts.

9.4 IMBALANCED DATASET

Machine learning models strive to minimize error during their training process, and in doing so, they tend to focus on the majority class in an imbalanced dataset. This situation is common in spam detection, where the number of non-spam (ham) emails usually surpasses spam emails.

Let's take an example: if our dataset contained 95% ham emails and just 5% spam emails, a machine learning model could achieve 95% accuracy by merely classifying all emails as ham, completely ignoring the spam class. Such a model would be practically useless despite its high accuracy rate, as it would fail to detect any spam emails.

Addressing this issue is not trivial and often involves techniques such as over-sampling the minority class, under-sampling the majority class, or using synthetic minority oversampling techniques (SMOTE). However, these

techniques have their own trade-offs and limitations. Over-sampling might lead to overfitting, while under-sampling could result in loss of potentially useful data.

9.5 STATIC DATASET

The field of cybersecurity, particularly spam detection, is dynamic and continuously evolving. Spammers frequently change their tactics to bypass spam filters, making spam detection a moving target problem.

If our dataset was static and didn't account for these evolving spam tactics, our machine learning model might become obsolete over time. For instance, the model might have learned to detect spam emails based on keyword stuffing (an outdated spam tactic), but it might fail to detect newer spam tactics such as using homoglyphs (confusingly similar characters) or employing complex HTML structures to conceal spam content.

Therefore, the limitation of a static dataset can lead to less adaptable models, potentially decreasing their effectiveness over time in the real world where spam tactics continually evolve.

9.6 LIMITED EXTERNAL SOURCES

While our dataset and approach focus on the content and structure of the emails, they do not make use of external reputation sources or user-specific behavior. Some spam detection systems use blacklists, domain reputation systems, or analyze user-specific interaction with emails to increase their detection accuracy.

For instance, a spam email might not exhibit common spam characteristics and may appear as a legitimate email when observed in isolation. However, if

multiple users reported similar emails as spam or if the sending domain was known to be associated with spam activities, it could be classified as a spam email.

While our model focuses on email content analysis, this limitation of not using external sources might decrease its ability to detect certain sophisticated spam emails compared to systems that use a more holistic approach.

Chapter 10

REFERENCES

[1]

Julio Villena-Román, Sonia Collada-Pérez, Sara Lana-Serrano, José Carlos González. Hybrid Approach Combining Machine Learning and a Rule-Based Expert System for Text Categorization. *Published in The Florida AI Research. 20 March 2011*

Link:https://www.semanticscholar.org/paper/Hybrid-Approach-Combining-Machine-Learning-and-a-Villena-Rom%C3%A1n-Collada-P%C3%A9rez/6e2f9f203db02be6d7a90d7b6c6819c1c65ab6ae#related-papers

[2]

Panthagani Vijaya Babu, T. R. Rajesh, Anjaneyulu Nelluru. LSTM-based Deep Learning Model for Emotion Intensity Level by Enhanced Sentiment Classification. *Published 2020*

Link:https://www.semanticscholar.org/paper/LSTM-based-Deep-Learning-Model-for-Emotion-Level-by-Babu-Rajesh/f6ba9a54777c47c13706c0a3ef66ca3465eefb6f

[3]

Indika Wickramasinghe, Harsha Kalutarage. Naive Bayes: applications, variations and vulnerabilities: a review of literature with code snippets for implementation Indika. *Published February 2021*

Link:https://www.researchgate.net/publication/344194503_Naive_Bayes_applications_variations_and_vulnerabilities_a_review_of_literature_with_code_snippets_for_implementation_Indika

[4]

Abbasi, F., Jamil, N., & Shah, S. A. A. (2021). D-Fence: A Flexible, Efficient, and Comprehensive Phishing Email Detection System. In Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIACCS '21), *June 7–11, 2021, Virtual Event, Japan.*

Link:https://www.researchgate.net/publication/354424369_D-Fence_A_Flexible_Efficient_and_Comprehensive_Phishing_Email_Detection_System

[5]

Ferrara, E.: The history of digital spam. arXiv preprint arXiv:1908.06173 (2019)

Link: https://arxiv.org/abs/1908.06173

[6]

Ren, Y., Ji, D.: Learning to detect deceptive opinion spam: a survey. IEEE Access 7, 42934–42945 (2019)

Link: https://ieeexplore.ieee.org/abstract/document/8678638

[7]

Fang, Y., Zhang, C., Huang, C., Liu, L., Yang, Y.: Phishing email detection using improved RCNN model with multilevel vectors and attention mechanism. IEEE Access 7, 56329–56340 (2019)

Link: https://ieeexplore.ieee.org/abstract/document/8701426

[8]

Ji, S., Ma, H., Liang, Y., Leung, H., Zhang, C.: Correction to: a whitelist and blacklist-based co-evolutionary strategy for defensing against multifarious trust attacks. Appl. Intell. 48(7), 1891 (2018)

Link: https://link.springer.com/article/10.1007/s10489-018-1195-1

[9]

Caraffini, F., Neri, F., Epitropakis, M.: HyperSpam: a study on hyper-heuristic coordination strategies in the continuous domain. Inf. Sci. 477, 189–202 (2019)

Link: https://www.sciencedirect.com/science/article/pii/S002002551830851X

[10]

Alghoul, A., Al Ajrami, S., Al Jarousha, G., Harb, G., Abu-Naser, S. S.: Email classification using artificial neural network. Int. J. Acad. Dev. 2(11), 8–14 (2018)

Link: http://dstore.alazhar.edu.ps/xmlui/handle/123456789/225

[11]

Yu, S.: Covert communication by means of email spam: a challenge for digital investigation. Digit. Invest. 13, 72–79 (2015)

Link: https://www.sciencedirect.com/science/article/pii/S1742287615000432

[12]

Aleroud, A., Zhou, L.: Phishing environments, techniques, and countermeasures: a survey. Comput. Secur. 68, 160–196 (2017)

Link: https://www.sciencedirect.com/science/article/pii/S0167404817300810

[13]

Fang, Y., Zhang, C., Huang, C., Liu, L., Yang, Y.: Phishing email detection using improved RCNN model with multilevel vectors and attention mechanism. IEEE Access 7, 374–406 (2019)

Link: https://ieeexplore.ieee.org/abstract/document/8701426

[14]

Gupta, S., Deep, K.: Improved sine cosine algorithm with crossover scheme for global optimization. Knowl. Based Syst. 165, 374–406 (2019)

Link: https://www.sciencedirect.com/science/article/pii/S0950705118305951

[15]

Venkatraman, S., Surendiran, B., Kumar, P.A.R.: Spam e-mail classification for the Internet of Things environment using semantic similarity approach. J. Supercomput. 76, 756–776 (2020)

Link: https://link.springer.com/article/10.1007/s11227-019-02913-7

[16]

Asghar, M.Z., Ullah, A., Ahmad, S., Khan, A.: Opinion spam detection framework using hybrid classification scheme. Soft Comput. 24, 3475–3498 (2020)

Link: https://link.springer.com/article/10.1007/s00500-019-04107-y

[17]

Citlak, O., Dorterler, M., Dogru, I.A.: A survey on detecting spam accounts on Twitter network. SNAM 9(1), 35 (2019)

Link: https://link.springer.com/article/10.1007/s13278-019-0582-x

[19]

Shuaib, M., Adebayo, O.S., Osho, O., Idris, I., Alhasan, J.K., Rana, N.: Whale optimization algorithm-based email spam feature selection method using rotation forest algorithm for classification. SN Appl. Sci. 1(5), 390 (2019)

Link: https://link.springer.com/article/10.1007/s42452-019-0394-7

[20]

Mokri, M.A.E.S., Hamou, R.M., Amine, A.A.: New bio-inspired technique based on octopus algorithm for spam filtering. Appl. Intell. 49, 3425–3435 (2019)

Link: https://link.springer.com/article/10.1007/s10489-019-01463-y

[21]

Chikh, R., Chikhi, S.: Clustered negative selection algorithm and fruit fly algorithm based email spam classification. J. Ambient Intell. Hum. Comput. 10(1), 143–152 (2019)

Link: https://link.springer.com/article/10.1007/s12652-017-0621-2