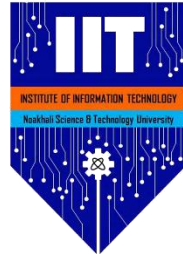


**Institute of Information Technology, Noakhali Science and Technology University**

**Bachelor of Science in Software Engineering**

**Course Code: SE 4202**



## **Final Year Project Report**

**Submitted by**

*Mehedi Hasan*

MUH2025032M

**Supervised by**

*Nazmun Nahar*

Lecturer

Institute of Information Technology (IIT)  
Noakhali Science and Technology University

**A Final Year Project Report**

**Submitted to the Bachelor of Science in Software Engineering Program Office of the  
Institute of Information Technology, Noakhali Science and Technology University in  
Partial Fulfillment of the Requirements of the Degree**

**Date of Submission: May 18, 2025**



## DECLARATION

---

I, Mehedi Hasan, bearing ID: MUH2025032M, a student of Institute of Information Technology (IIT), Noakhali Science and Technology University, enrolled in the BSc. in Software Engineering Program, hereby declare that the work presented in this final year project report titled "PhisNet: Intelligent Detection of Phishing" is my own and has been carried out under the supervision of Nazmun Nahar, Lecturer.

I affirm that:

- (i) This work is original and has not been submitted, in whole or in part, for any other degree or academic purpose.
- (ii) All external sources of information and ideas have been clearly and appropriately acknowledged through proper citations and references in accordance with the academic standards and guidelines of Institute of Information Technology (IIT).
- (iii) The data, findings, conclusions, and recommendations presented in this report are based on honest and rigorous research, and any potential biases or conflicts of interest have been disclosed.
- (iv) I take full responsibility for any errors or omissions in this work.
- (v) This project conforms to the ethical standards and guidelines set forth by Institute of Information Technology (IIT), and I have obtained all necessary approvals and permissions for any research involving human subjects or sensitive data.
- (vi) I understand that any violation of academic integrity, including plagiarism or fabrication of data, can lead to severe consequences as outlined in the academic integrity policy Institute of Information Technology (IIT), Noakhali Science and Technology University.

I acknowledge that my final year project report will be archived by the Institute of Information Technology (IIT), Noakhali Science and Technology University and may be made available to the academic community and the public for reference and research purposes.

---

**Mehedi Hasan**

*MUH2025032M*

Session: 2019-2020

Bsc. In Software Engineering Program

Institute of Information Technology

Noakhali Science and Technology University

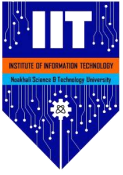
---

**Nazmun Nahar**

Lecturer

Institute of Information Technology (IIT)

Noakhali Science and Technology University



# DEDICATION

---



## APPROVAL

---

This Final Year Project report submitted by **Mehedi Hasan, ID No: MUH2025032M** to the Chairman of Year 4, Term 2 Examination Committee (Session: 2019-20), Institute of Information Technology (IIT), Noakhali Science and Technology University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approved as to its style and contents.

---

**Dr. Md. Nuruzzaman Bhuiyan**

*Committee Member*

Assistant Professor  
Institute of Information Technology (IIT)  
Noakhali Science and Technology University

---

**Md. Auhidur Rahman**

*Committee Member*

Assistant Professor  
Institute of Information Technology (IIT)  
Noakhali Science and Technology University

---

**Dipok Chandra Das**

*Committee Member*

Assistant Professor  
Institute of Information Technology (IIT)  
Noakhali Science and Technology University

---

**Md. Iftekharul Alam Efat**

*Committee Member*

Assistant Professor  
Institute of Information Technology (IIT)  
Noakhali Science and Technology University

---

**Dr. Md. Kamal Uddin**

*External Member*

Associate Professor  
Department of Computer Science and  
Telecommunication Engineering  
Noakhali Science and Technology University

---

**Dr. Mohammad Salim Hossain**

*Chairman*

Director  
Institute of Information Technology (IIT)  
Noakhali Science and Technology University

# Abstract

---

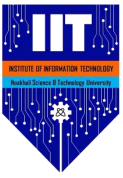
Phishing and fraud attacks remain prevalent in the cybersecurity scene, as attackers more frequently utilize emails and URLs communications. In this study, we perform a comparative analysis of three machine learning methods for detecting phishing: (1) a BERT-BiLSTM model focused on email phishing detection, (2) an XGBoost URL classifier utilizing handcrafted features, and (3) a hybrid RoBERTa model aimed at URL phishing detection. In email phishing detection, the suggested BERT-BiLSTM model with attention mechanisms attained 98.7% accuracy by effectively utilizing linguistic metadata and the structural characteristics of emails, extracting and combining essential content, and focusing on key details required for completion. In our tests for URL detection, the RoBERTa hybrid model outperformed the engineered XGBoost method, attaining 93% accuracy compared to 88%, confirming our hypothesis that recognizing semantic patterns in URLs is essential for detection. Crucially, it was noted that transformer models vastly surpassed all conventional machine learning techniques across every examined domain, showing remarkable superiority in recall for sophisticated phishing attempts. Additional examination of feature importance revealed that URL entropy and email sentiment features were the most significant discriminating factors. These results guide the development of multi-layered active systems to protect against phishing attacks by suggesting the implementation of RoBERTa hybrids for web traffic filtering and a motion-controlled BERT-LSTM operation.

## Table of Contents

<b>Chapter One: Introduction</b>	<b>2</b>
1.1 Background and Motivation	2
1.2 Problem Statement	3
1.3 Research Objectives	3
1.4 Scope and Limitation	4
1.5 Significance of the Study	4
<b>Chapter Two: Literature Review</b>	<b>7</b>
2.1 Preliminaries	7
2.2 Relevant Theories and Concepts	7
2.3 Review of Related Research and Projects	10
2.4 Identification of Gaps in Existing Solutions	14
<b>Chapter Three: Methodology</b>	<b>16</b>
3.1 Research Design	16
3.2 Research Approach	16
3.3 Data Collection Method	18
3.4 Data Analysis Techniques	19
3.5 Proposed Model and Technique	21
<b>Chapter Four: Requirements Analysis and Specification</b>	<b>26</b>
4.1 Elicitation and Documentation of Requirements	26
4.2 Functional Requirements	26
4.3 Non-functional Requirements	27
4.4 Use case and Scenario	29
<b>Chapter Five: System Design and Architecture</b>	<b>33</b>
5.1 System Architecture Overview	33
5.2 High-Level Design	33
5.3 Detailed Design	34
5.4 Component and Module Design	35
<b>Chapter Six: Implementation and Development</b>	<b>46</b>
6.1 Programming Language and Technologies Used	46
6.2 Implementation Details	47
6.3 Testing and Debugging Approach	49
<b>Chapter Seven: Result Analysis and Discussion</b>	<b>Error! Bookmark not defined.</b>



7.1 Data Collection and Analysis .....	Error! Bookmark not defined.
7.2 Evaluation Metrics and Criteria.....	Error! Bookmark not defined.
7.3 Interpretation of Results .....	Error! Bookmark not defined.
7.4 Comparison with Related Work .....	Error! Bookmark not defined.
7.5 Discussion of Findings .....	Error! Bookmark not defined.
Chapter Eight: Conclusion and Future Work .....	54
8.1 Summary of Contribution .....	54
8.2 Limitations and Challenges.....	55
8.3 Recommendation for Future Enhancements .....	56



## Table

Table 1: Detect phishing from email .....	27
Table 2: Detect phishing from email.....	29
Table 3: Detect phishing from URLs .....	30
Table 4: Component and Module Design.....	35





# Chapter One

## Introduction

## Chapter One: Introduction

---

### 1.1 Background and Motivation

Phishing attacks are attempts to get your personal information, essentially fraud. Whether in the form of an email, website, or messages, phishers impersonate a trusted entity in order to trick the recipient into revealing sensitive information. They often take place using between-end emails and spurious sites made to look real. Anti-Phishing Working Group (APWG) reported data that shows that for the past five years on back has been an explosion in phishing incidents, with millions of details distributed worldwide [1].

Phishing detection approaches, as customarily practiced, depend on rule-based algorithms or blacklisting. Nevertheless, these static strategies have limited capacity to identify new and more complex attacks because of their low adaptability and high false positive rates [2].

Also, some machine learning-based systems give fairly substantial improvements, but only because of a lot of manual feature engineering, and even then, they still miss the language-based subtleties we routinely see in constructing phishing emails, or the hidden structure in URLs [3]. Recent developments in Natural Language Processing (NLP) and Deep Learning provide stronger techniques to detect phishing messages. BERT and RoBERTa are two well-known transformer models that can measure deep contextual meanings of words and phrases. The transformer models have been successfully applied to a range of NLP problems, such as phishing email or URL classification [4][5].

Inspired by these challenges and advancements, this research presents PhishNet: Intelligent detection of phishing systems that utilize BERT + BiLSTM + Attention for intelligent phishing email detection and RoBERTa + Attention and XGBoost for phishing URL detection. The aim is to decrease false positives and improve generalizability by combining deep semantic understanding with lexical and non-lexical metadata-based features.

## 1.2 Problem Statement

While various phishing detection systems are available in the marketplace, the capabilities of most commercial or popular systems remain limited in many ways: high false positives, reliance on manually defined features, no support for multilingual phishing attacks, no method of handling newly emerging obfuscation techniques in URLs, etc. Traditional models are limited to working with unstructured text (email) and they are limited in their ability to identify network addresses with suspicious patterns without understanding semantics.

In addition, many email phishing detection models work within the email body content but ignore contextual metadata associated with the email (e.g. sender, subject, embedded links, etc.). Most URL-based models ignore any linguistic information contained in the DOI structure or the content that may indicate, in some way, phish intent. That said, the challenge is building a system that can accept both structured data (e.g. email content and metadata) and unstructured data (e.g. URL text/characteristics) to detect phishing with very low false negatives and with very low time constraints while being adaptive to new phishing threats.

## 1.3 Research Objectives

1. To develop an automated phishing detection system using hybrid deep learning models for both email and URL-based phishing attacks.
2. To enhance phishing email detection by combining BERT embeddings, Bidirectional LSTM, and attention mechanisms for contextual feature learning.
3. To improve URL detection by integrating RoBERTa-based attention models with lexical and metadata-based features.
4. To address limitations in traditional systems such as high false positives, lack of semantic analysis, and static feature reliance.
5. To implement a real-time classification system that can be used in email clients or browser extensions for practical deployment.
6. To evaluate the proposed models against traditional methods (e.g., XGBoost) for benchmarking and analysis.

## 1.4 Scope and Limitation

This research focuses on building a phishing website and email detection system that leverages deep-learning models to analyze URLs and emails. Specifically, the boundaries of the project include:

- Extraction and processing of features of an email such as the subject, body, sender, and receiver.
- Analysis of the lexical and domain features of URLs.
- Analysis of contextual embedding extraction using BERT and RoBERTa.
- Comparison of deep learning models against traditional ML models.

The limitations of the study were:

- The system made use of publicly available datasets which do not account for all potential phishing patterns which may exist in practice.
- The project intended to detect not only English phishing content but also multilingual phishing content, however this area was not fully explored.
- The real-time performance of the system is limited on devices with limited resources due to the high computation cost of the transformer-based models.
- The system is able to make a detection, but cannot resolve the phishing attack (i.e. does not block the phishing site or quarantine the email).

## 1.5 Significance of the Study

This research is important because it provides a new solution to the increasing need for secure, adaptable phishing detection solutions. The suggested system, which includes NLP-based transformers fused with classical feature-based models, fills the gap between deep semantic understanding and practical interpretability. The PhishNet model is capable of examining phishing content in terms of both structure and semantics and is more well-equipped against recent, accelerated phishing attacks. The research findings can provide assistance in browser extension development, corporate email filters, and cybersecurity educational campaigns. It will also provide value in the larger space of NLP-based cybersecurity, as it provides a replicable, modular and explainable detection framework. solution.



# Chapter Two

## Literature Review

## Chapter Two: Literature Review

---

### 2.1 Preliminaries

In this section, we discuss the basic terms and technologies that underlie phishing detectors systems.

#### Phishing

Phishing is an attack method that involves tricking people into revealing personal and sensitive information by pretending to be a trusted entity through email, websites or messages. Phishing can be classified into email phishing (where users are targeted through emails) and URL-based phishing (where a malignant website is created to look like a legitimate one).

#### Machine Learning (ML)

Machine Learning is a sub-field of artificial intelligence where machine learning systems learn from data and apply that learning to improve tasks without being explicitly programmed. Phishing detection task have utilized machine learning techniques such as XGBoost, Random Forests, and SVMs.

#### Natural Language Processing (NLP)

NLP is about enabling machines to comprehend and ubiquitous human language . In terms of phishing detection, NLP can be used to analyze email texts, explore synonyms and analysis of slightly manipulative wording, while models such as BERT and RoBERTa have a much deeper connection of how words can be captured.

#### Transformers

Transformers are a class of deep learning models which were first introduced in 2017 by Vaswani et al.[1], and serve as the foundation of BERT and RoBERTa models. Transformers rely on self-attention and thus can best capture dependencies of information within sequences. As transformers are built on self attention, they are well-suited for many applications including text classification and sentiment analysis, as well as phishing detection..

### 2.2 Relevant Theories and Concepts

#### BERT (Bidirectional Encoder Representations from Transformers)

BERT is a transformer-based model pre-trained on a vast body of text with a masked language model task, allowing it to capture contextual word embeddings by contextually learning, from both left and right contexts of words, in which a model has a maximum of 512 tokens of prior information within the input data. BERT is especially useful to uncovering the linguistic patterns of language, in phishing detection, within email content that may be denoted as fraudulent messages.

### **RoBERTa**

RoBERTa (A Robustly Optimized BERT Pretraining Approach) also improved on the BERT architecture by way of number of data queries available, as well as longer training times while omitting from the next sentence prediction task, RoBERTa is good for any task that entails complex semantic analysis of texting and would interpret shooting obfuscated phishing URLs.

### **Attention Mechanism**

Attention mechanisms allow models to focus on particular input data that allow (suspicious) words or phrases in an email to be identified. Attention mechanisms improve the interpretability of the model, while also improving model performance in a long sequence task.

### **Bidirectional LSTM (BiLSTM)**

BiLSTM is a type of recurrent neural network that allows data to be processed in both forward and backward directions, giving it greater context than an LSTM. This stage of the research, BiLSTM after BERT to allow temporal relationships in the email text to be analyzed.

### **TF-IDF and N-Grams**

TF-IDF (Term Frequency-Inverse Document Frequency) calculates how important a word is to a document in relation to the entire corpus. The phrase "term" literally means "word", but for the purposes of this research, it can extend to phrases like "verify your account" or "click now". N-grams capture n consecutive words, allowing us to detect the exact phishing phrases used.

### **Sentiment Analysis**

Phishing emails often elicit urgency or fear in their recipients. Sentiment analysis is an alternative method of being used to identify emotional triggers in the text content of addresses, and another feature to identify phishing emails.

## **XGBoost**

XGBoost is a gradient boosting algorithm that is one of the most effective and widely used when dealing with structured data like the URL features. It was chosen for this research because it is fast, interpretable, and naturally robust against overfitting during modeling. In this research, XGBoost is used to model the features that were engineered from hypotheses about the properties of brief phishing, e.g., length, use of IP, and suspicious words.

## **Class Weighting**

class weighting, which modifies the training process rather than the data itself. In this method, higher weights are assigned to the minority class (phishing) and lower weights to the majority class (legitimate), ensuring that misclassifications of the minority class are penalized more heavily during model optimization. This technique is particularly advantageous when working with complex models such as deep neural networks, where data augmentation via SMOTE may not be ideal. The class weight  $W_c$  for each class  $c$  can be computed using the formula:

$$W_c = \frac{n}{|C|.n_c}$$

where:

$n$  is the total number of training samples,

$|C|$  is the number of classes,

$n_c$  is the number of samples in class  $c$ .

This dynamic adjustment allows the learning algorithm to place greater emphasis on correctly identifying phishing instances, thereby improving recall and F1-score without modifying the original distribution of the dataset. In experiments conducted during this research, class weighting proved to be effective in achieving high detection performance while preserving data integrity and reducing training bias.



## URL Obfuscation and Lexical Features

Attackers omit, or feign phishing intent differently when using URL obfuscation techniques especially when detecting the fillers like the use of IP address instead of a domain name, which makes validation of the legitimate use of such descriptors (e.g. hexadecimal encoding). Lexical features containing suspicious words as opposed to fixing on domain names, were most likely more relevant to feature-based models like XGBoost.

### 2.3 Review of Related Research and Projects

The paper “Phishing Email Detection Model Using Deep Learning” [7] explores deep learning techniques for detecting email phishing attacks, including CNNs, LSTM, RNNs, and BERT. It highlights the importance of feature selection in identifying spam and phishing emails. The review identifies a common limitation: the need for large datasets to improve detection accuracy. It evaluates the performance of different techniques, noting the effectiveness of deep learning approaches. It explores deep learning techniques, including CNNs, LSTMs, RNNs, and BERT, for detecting phishing attacks. A dataset of phishing and benign emails was utilized, with relevant features extracted using NLP techniques. The research highlights the potential of deep learning to enhance email phishing detection and improve cybersecurity. Results: The proposed deep learning model achieved high accuracy, with the best performance at 99.61% using BERT and LSTM.

The paper “Comparative Investigation of Traditional Machine-Learning Models and Transformer Models for Phishing Email Detection” [8] investigates the effectiveness of traditional and transformer models in detecting phishing emails. They used transformer models, especially distilBERT, BERT, and roBERTa, outperform traditional models in classification tasks. Traditional models struggle with complex problems, particularly in Natural Language Processing and image processing tasks. Traditional models like Logistic Regression and SVM performed marginally worse than transformer models in phishing detection. The computational inefficiency of enforcing transformer tokenization on traditional systems can introduce noise in feature representation. RoBERTa achieved a maximum accuracy of 0.9943 in identifying phishing emails. Traditional models like SVM performed well, with an accuracy of 0.9876, but were less effective than transformer models.

The paper “Advancing Phishing Email Detection: A Comparative Study of Deep Learning Models” [9] focuses on the challenges of email phishing detection and the various approaches to mitigate these risks. It highlights the increasing prevalence of phishing attacks, prompting the cybersecurity community to develop effective detection methods. The review categorizes anti-phishing solutions into software-based and user awareness strategies, emphasizing the integration of artificial intelligence, particularly machine learning and deep learning, in software solutions. It references notable datasets used in the field, such as The Phishing Corpus and the Spam Assassin dataset, which are essential for training and evaluating phishing detection models. The study investigates the effectiveness of deep learning models, particularly 1D-CNNPD, in improving phishing email detection accuracy. Augmentations with recurrent layers like LSTM and GRU are explored to enhance the base model's performance. The research aims to develop lightweight models that achieve high detection rates while minimizing false positives. The Advanced 1D-CNNPD with Leaky ReLU and Bi-GRU achieved 100% precision and 99.68% accuracy. The model yielded an F1 score of 99.66% and a recall of 99.32%. The 1D-CNNPD with Bi-GRU outperformed DeepAnti-PhishNet in accuracy.

The paper "Evaluation of Federated Learning in Phishing Email Detection" [10] investigates the application of artificial intelligence (AI) in detecting phishing emails, highlighting the challenges posed by centralized datasets, including privacy and legal issues. The study is the first to explore federated learning (FL) in this context, focusing on deep neural network models like recurrent convolutional neural networks (RNN) and BERT. It evaluates FL's performance under various conditions, revealing insights into accuracy variations with different organizational counts and data distributions. The study highlights challenges in achieving stable outputs with asymmetric email datasets in FL frameworks. The study demonstrated that federated learning (FL) achieved 97.9% test accuracy for THEMIS with five clients at epoch 45. BERT achieved 96.1% test accuracy for FL with five clients at epoch 15. The results indicated that FL's performance was comparable to centralized learning under balanced data distribution.

The paper "A Systematic Review on Deep-Learning-Based Phishing Email Detection" [11] focuses on the application of deep learning techniques in phishing detection, highlighting their potential to enhance accuracy and effectiveness in identifying phishing attacks. It outlines a detailed methodology for conducting the SLR (Systematic literature review), which includes

defining the research question, developing a search strategy, and assessing the quality of studies. It highlights limitations of traditional phishing detection methods, advocating for advanced techniques. The research emphasizes the effectiveness of deep learning algorithms, such as CNNs and LSTMs, in detecting sophisticated phishing attempts. The systematic literature review aims to synthesize findings on deep learning techniques for phishing detection. The study contributes to a coherent understanding of deep learning applications in phishing detection. The findings aim to provide insights into the current state of research and suggest future directions for advancing phishing detection using deep learning.

The paper "BERT-Based Approaches to Identifying Malicious URLs" [12] focuses on the detection of malicious URLs, which are commonly used in cyberattacks, particularly phishing. It introduces a BERT-based model that tokenizes URL strings and utilizes a classifier to identify malicious URLs. Additionally, the model demonstrated versatility through experiments on datasets from different domains, including IoT and DoH. The research highlights the importance of accurately detecting malicious URLs to combat cyberattacks. The study employs a bidirectional encoder representation from transformers-based (BERT) model for tokenizing URL strings. It utilizes a classifier to determine if a URL is malicious. The research evaluates methods using three public datasets: URL strings, URL features, and a combination of both. The proposed system achieved accuracy rates of 98.78%, 96.71%, and 99.98% across the datasets. Various machine learning algorithms, including KNN, MLP, and random forest, are mentioned in related research.

The paper "Phishing Detection System Through Hybrid Machine Learning Based on URL" [13] focuses on phishing attacks, a significant cybercrime that exploits email distortion and fake websites to gather sensitive information from users. It highlights the inadequacy of existing solutions and emphasizes the role of machine learning in combating these threats. The study utilizes a dataset of over 11,000 URLs, applying various machine learning algorithms, including a proposed hybrid model combining logistic regression, support vector machine, and decision tree. The results demonstrate that the proposed approach outperforms existing models in terms of accuracy and efficiency, validated through multiple evaluation metrics. Machine learning models, including decision tree and random forest, were applied to classify phishing URLs effectively. The highest accuracy for the support vector machine model was 71.8% with

specific precision, recall, and F1-score values. Random forest outperformed all other algorithms in the comparative analyses, demonstrating superior results.

The paper "Intelligent Deep Machine Learning Cyber Phishing URL Detection Based on BERT Features Extraction" [14] introduces a novel URL phishing detection technique using BERT feature extraction and deep learning methods. It highlights the growing sophistication of phishing attacks and the need for intelligent detection models. The research emphasizes the importance of natural language processing in feature extraction for phishing detection. It notes a significant number of studies (over 87) on deep learning algorithms for phishing URL detection from 2018 to 2022. A deep convolutional neural network method was utilized to detect phishing URLs. The methodology followed a four-stage process to achieve the study's goals. The feature extraction process relied on natural language processing techniques. The dataset used contained 472,259 records after pre-processing. The proposed method achieved an accuracy of 96.66% in detecting phishing URLs.

The paper "DEPHIDES: Deep Learning Based Phishing Detection System" [15] develops a phishing detection system using deep learning techniques, achieving high performance with reduced false positives. A dataset of approximately five million labeled URLs was collected to train and test the system. The research emphasizes the importance of sophisticated phishing detection systems in enhancing cybersecurity against evolving threats. The paper employs Associative Classification using "If-Then" rules for web page classification, testing six different algorithms. It utilizes third-party services for domain age and web traffic detection. The study develops a phishing detection system based on deep learning, using five algorithms including CNNs and RNNs. Two vectorization methods with character embeddings are applied in the assessments. The system focuses on fast classification of URLs, achieving high accuracy rates. Convolutional neural networks achieved the highest detection accuracy of 98.74% for phishing attacks.

The paper "A Phishing-Attack-Detection Model Using Natural Language Processing and Deep Learning" [16] conducted a meta-review of systematic literature reviews on phishing detection and deep learning over the last three years. An in-depth study analyzed 100 articles on mitigating phishing email attacks using NLP and traditional ML or DL algorithms. The authors identified 28 articles, narrowing down to 20 after eliminating duplicates. The paper presents a model for detecting phishing attacks using Natural Language Processing (NLP) and Deep

Learning (DL) algorithms, focusing on the content of suspicious web pages rather than URLs. Four DL algorithms are evaluated: Long Short-Term Memory (LSTM), Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), and Bidirectional GRU (BiGRU). BiGRU outperformed other DL algorithms, achieving the highest accuracy of 97.39%. The mean accuracy for LSTM, BiLSTM, and GRU were 96.71%, 97.20%, and 97.29%, respectively.

## 2.4 Identification of Gaps in Existing Solutions

In analyzing the existing approaches for identifying phishing, several critical areas have been identified that prevent the practical application of these tools across diverse software development contexts. These gaps show that there should be an improvement of detecting phishing emails and URLs.

### 1. Lack of Comprehensive and High-Quality Datasets:

The lack of comprehensive and high-quality datasets in phishing email detection stems from the rapid evolution of phishing tactics and privacy constraints, making real-world data scarce. Small or imbalanced datasets (with far more legitimate emails than phishing ones) lead to models that generalize poorly, failing to detect new attack patterns and producing high false-negative rates. Without diverse, up-to-date datasets, detection systems struggle to adapt to emerging threats effectively.

### 2. Algorithmic Bias and Poor Generalization:

A major limitation in current phishing detection systems is algorithmic bias, where models trained on narrow or unrepresentative datasets (e.g., only English emails or header-based features) struggle to generalize across different languages, regions, or attack types. This leads to reduced detection accuracy for multilingual phishing campaigns, region-specific scams, and advanced threats like spear phishing or business email compromise (BEC). For example, a model analyzing only email headers may miss malicious content hidden in the body, images, or attachments. Since phishing tactics constantly evolve, models must be trained on diverse, multilingual, and feature-rich datasets to avoid bias and improve real-world effectiveness. Without this, detection systems remain vulnerable to sophisticated and localized attacks.

### 3. Lack of Real-World Deployment Insights:

A critical gap in current research is the **absence of real-world testing**—most phishing detection models are evaluated only in controlled lab environments rather than live email

systems. This leads to **unproven scalability** (can the model handle millions of emails daily?) and **unaddressed latency** (does it slow down email delivery?). Additionally, models untested in production miss **real-world adversarial attacks**, where attackers deliberately manipulate emails to evade detection (e.g., obfuscated URLs, generative AI-crafted messages). Without deployment insights, even high-accuracy models may fail in practice, leaving organizations vulnerable to undetected phishing threats. Future work must prioritize **real-world validation** to ensure robustness, efficiency, and resilience against evasion tactics.

#### 4. High False-Positive Rates:

A major flaw in current systems is their tendency to mislabel legitimate emails as phishing. This occurs because overly broad rules flag normal elements like password reset links, lack of context analysis fails to distinguish real threats from benign emails, Static models can't adapt to organizational communication patterns. These false alarms create alert fatigue, causing users to ignore warnings and miss actual threats. Solutions require smarter analysis of sender patterns, user behavior, and email context to maintain accuracy without compromising trust.

#### 5. Overreliance on Historical Data in URL-Based Phishing Detection

A major limitation of current phishing URL detection systems is their dependence on static, historical datasets, which fail to capture emerging attack patterns in real time. Cybercriminals constantly evolve their tactics—using new domain generation algorithms (DGAs), obfuscation techniques, and zero-day exploits—but models trained on old data cannot recognize these novel threats. This lag in adaptability leaves systems vulnerable to fast-evolving phishing campaigns, particularly those leveraging AI-generated URLs or dynamic redirections. To improve real-time defense, future solutions must incorporate continuously updated datasets and adaptive learning mechanisms that detect zero-day phishing URLs without relying solely on past examples. Without this shift, detection systems will remain a step behind attackers.

#### 6. Poor Handling of Contextual URL Characters in Phishing Detection

A critical weakness in current URL-based phishing detection is the inability to interpret characters contextually—the same symbol (e.g., /, @, or -) can have legitimate or malicious implications depending on its placement in a URL. For example, while an @ symbol is valid in email addresses, its presence in a URL's path may signal credential phishing (e.g.,

`http://legit.com@malicious.net`). Similarly, excessive slashes (`////`) or dots (`....`) often indicate obfuscation. Most models treat characters in isolation, missing these positional red flags. This oversight leads to false negatives (missing malicious URLs with clever symbol use) and false positives (flagging harmless URLs with unusual but valid structures). Future solutions must adopt context-aware parsing—leveraging syntactic rules, positional encoding, or attention mechanisms—to accurately discern malicious intent from character patterns. Without this, attackers can easily bypass detectors through simple symbol manipulation.

## 7. Lack of Adaptability Across Data Formats in URL-Based Phishing Detection

A significant limitation in current phishing URL detection systems is their inflexibility in handling diverse URL formats, including internationalized domain names (IDNs), complex subdomains, and emerging top-level domains (TLDs). Many models are trained primarily on standard ASCII-based URLs, causing them to fail when encountering unicode characters (e.g., Cyrillic or Chinese scripts in homograph attacks like "apple.com" vs. "apple.com"), Long, randomized subdomains (e.g., `x1.a2b3c4.phishingsite.com`), new gTLDs (e.g., `.xyz`, `.top`) often abused by attackers. This rigidity leads to high false-negative rates for sophisticated attacks and false positives for legitimate international websites. The root cause lies in overfitting to narrow training data and the lack of modular preprocessing for diverse formats.

# Chapter Three

## Methodology



## Chapter Three: Methodology

---

The method or process which was employed in the systematic way of performing the study or achieving the research objectives is called the methodology. It outlines the general framework and processes for assembling, analyzing and assessing information in order to answer research questions or hypotheses. The methodology part of the study provides a roadmap to implementing the study and explicitly describes the research design, data collection procedures, and other procedures to counter some of the issues that we raised with the study. It guarantees that the study follows an efficient research process so that the results are as precise and reliable as possible.

### 3.1 Research Design

The research applies a comparative experimental design to come up with and put to test three different models of phishing detection among which are (1) BERT + BiLSTM + Attention for email phishing, (2) XGBoost for URL phishing, and (3) RoBERTa + Attention for URL phishing. The project uses two distinct datasets as the source of data – Email\_Dataset.csv in the case of email-based detection and phishing\_site\_URLs.csv in the case of URL-based detection. Every model utilizes a custom-made preprocessing and feature engineering pipeline to match the requirements of the data, leveraging the structured data (metadata, lexical features) and unstructured text (email body, subject, and URL). The underlying aim is to establish a basis for deep learning algorithms vis-à-vis traditional machine learning with respect to the performance of phishing detection, as measured by various metrics (accuracy, precision, recall, F1-score, and AUC), which in turn point to the most effective and easily implemented method.

### 3.2 Research Approach

This study uses an experimental research approach, where multiple models are trained, tested, and validated on real phishing datasets. The evaluation metrics include accuracy, precision, recall, and F1-score, ensuring a fair and quantitative comparison. Research methodology presents in detail the steps that were taken to achieve the goals of the research study. In this research, a variety of methods were picked and used for the purpose of the idea conception, empirical evaluation, and further improvement of the “PhishNet” phishing detection system.

1. **Literature Review:** The study was based on a thorough review of the available research work, which was then followed by machine learning and deep learning for email and URL phishing detection techniques. This gave an insight into the shortcomings such as high false positives, semantic misunderstanding, and inability to adapt to new phishing methods that this research aims to through transformer-based models.
2. **Model Development:** The PhishNet system was composed of a set of three primary models, namely, (1) BERT + BiLSTM + Attention for phishing email detection, (2) XGBoost for lexical feature-based URL phishing detection, and (3) RoBERTa + Attention for semantic URL detection. Each model was programmed using real datasets and at the same time, it was optimized in the sense of both robust performance and ability to be generalized.
3. **Dataset Construction and Preprocessing:** Two datasets were derived from Kaggle - Email\_Dataset.csv (a merged dataset containing sender, receiver, subject, and body for phishing/legitimate emails) and phishing\_site\_URLs.csv (containing labeled phishing and legitimate URLs). Both datasets were prepared by cleaning, tokenizing, and the extraction of some relevant features contemporary with the metadata encoding, TF-IDF, sentiment analysis, and lexical patterns.
4. **Feature Engineering:** A wide set of features was developed for the purpose of boosting model performance. For the email data, keyword presence, BERT embeddings, TF-IDF scores, and attention weights were contributed. For the URL data, lexical features (e.g., the number of special characters, domain structure), IP presence, obfuscation flags, and contextual embeddings through RoBERTa were applied.
5. **Algorithm Integration:** Models with pre-trained encoders from transformers such as Hugging Face BERT (Bidirectional Encoder Representations from Transformers), RoBERTa, and others were included in the deep learning pipelines. Customized deep learning pipelines were constructed by combining BERT and RoBERTa from Hugging Face with BiLSTM and the attention layer. Using Scikit-learn, XGBoost (Stochastic Gradient Boosting) was used. The hyperparameters of XGBoost were tuned by the use of the grid search algorithm. The class imbalance problem was resolved by using class weights.
6. **Experimentation:** The performance of every model was considered using a same-sized splitting of the data with a separation of 80:20 training and testing. Thus, different

metrics like accuracy, precision, recall, F1-score, and AUC were employed and applied to the same setting to give the model performance results. The computational cost vs. detection accuracy of the models was a key concern in performing very comprehensive tests.

7. **Comparative Studies:** In order to examine the efficiency of PhishNet, the outcomes of BERT + BiLSTM + Attention were compared to other e-mail published-hatched detection models, whereas the results of RoBERTa and XGBoost were trustworthy as their applications in detecting URLs were checked against the latest transfer learning methods. This similarity in modeling techniques agreed with the findings, which showed the power of the hybrid models leveraging Transformers alongside lexical-based and attention-based models.
8. **Real-World Application Design:** The models were architected for potential deployment in real-time environments such as email clients or browser extensions. This stage considered aspects such as computational efficiency, interpretability, and practical usability, with plans for future user feedback and real-world testing..

This approach helps to ensure that the research done is the advancement of knowledge in the field and provide tools that can be used in the advancement of the field.

### 3.3 Data Collection Method

Two publicly available phishing datasets were used:

**Email\_Dataset.csv:** A combination of four Kaggle email phishing datasets (CEAS\_08.csv, Nazario.csv, Nigerian\_Fraud.csv, and SpamAssassin.csv), merged to create a rich dataset containing legitimate and phishing emails. The dataset includes features such as sender, receiver, subject, body, and label.

**Phishing\_site\_URLs.csv:** A Kaggle dataset containing URLs labeled as phishing or legitimate. This dataset is used in both the XGBoost and RoBERTa-based models.

All datasets were pre-cleaned and manually verified for label consistency and formatting.

### 3.4 Data Analysis Techniques

#### Data Exploration

Data exploration involved initial analysis of the two datasets: Email\_Dataset.csv and phishing\_site\_URLs.csv. Basic statistical summaries and visualizations (e.g., class distribution plots) were used to understand the balance between phishing and legitimate examples. For email data, the structure of the text fields (e.g., length of subject and body, common words, metadata completeness) was examined. For URL data, key patterns like URL length, domain structure, and use of obfuscation (e.g., hexadecimal encoding, IP usage) were assessed. This step helped identify potential anomalies, missing values, and feature importance trends.


#### Data Wrangling

During data wrangling, inconsistencies and missing values in the datasets were resolved. For emails, missing sender and receiver values were filled with default placeholders, and missing subjects or body content were imputed with neutral strings like "No Subject" or "No Content." For URLs, unnecessary columns were dropped, and malformed entries were removed. Data types were corrected (e.g., converting labels to binary format), and text fields were standardized (e.g., lowercase conversion). The goal of this step was to create a clean and consistent dataset ready for transformation.

#### Preprocessing

In this research, preprocessing steps were customized based on the type of input and the model being used. For email phishing detection using the BERT + BiLSTM + Attention model, preprocessing involved converting all text fields (body, subject, sender, and receiver) to **lowercase**, **removing special characters and numbers**, applying **stopword removal** and **lemmatization**, and **tokenizing** the text using the **BERT tokenizer**. Additionally, **unigrams and bigrams (n-grams)** were extracted to capture common phishing phrases, and **email metadata** such as sender and receiver addresses were encoded as categorical numerical values. For URL phishing detection (used in both the RoBERTa + Attention and XGBoost models), URLs were first cleaned by converting to lowercase and removing query parameters. RoBERTa was then used for tokenization in the deep learning model, while for the XGBoost model, a variety of handcrafted features were extracted. These included **URL length**, **counts of special**

## Data Visualization




A bar chart titled "Distribution of Legit or phishing email Labels". The x-axis is labeled "Label" and has two categories: 0 and 1. The y-axis is labeled "Count" and ranges from 0 to 25,000 with major ticks every 5,000. The bar for Label 0 has a height of approximately 21,500. The bar for Label 1 has a height of approximately 28,500.

Label	Count
0	21500
1	28500

Sender	Number of Emails
gillian.sau@nytimes.com	450
Guido van Rossum - @davidpython.org	290
"Markus & Laverne" - @pythontesting.com	270
Carlos E. R. - @pythontesting.com	210
Adam Lallès - @pythontesting.com	180
Julien Garcia-Sanchez - @pythontesting.com	160
Christian Williams - @pythontesting.com	150
Barry Warsaw - @pythontesting.com	130
h20@python.com	120
Per Holm - @pythontesting.com	110

This histogram displays the frequency distribution of email body lengths for two categories: Non-Phishing (label 0, represented by blue bars) and Phishing (label 1, represented by orange bars). The x-axis represents the 'Body Length' from 0 to 5000, and the y-axis represents the 'Frequency' from 0 to 350,000. The Non-Phishing distribution is significantly higher and broader than the Phishing distribution, which is concentrated at lower body lengths.

Body Length	Frequency (label 0)	Frequency (label 1)
0	~370,000	~260,000
1000	~350,000	~250,000
2000	~330,000	~230,000
3000	~310,000	~210,000
4000	~290,000	~190,000
5000	~270,000	~170,000



A bar chart titled "Distribution of URLs Labels" showing the count of URLs for two labels: "good" and "bad". The y-axis is labeled "Count" and ranges from 0 to 400,000 in increments of 50,000. The x-axis is labeled "Label" and has two categories: "good" and "bad". The bar for "good" is approximately 390,000 units high, and the bar for "bad" is approximately 155,000 units high.

Label	Count
good	390000
bad	155000

Keyword	Frequency
/s	22000
/r	6500
/e	5500
/o	5000
/b	4000
/d	3500
/t	3000
/u	1000
/a	800
/k	500
/n	500
/m	0

Box Plot of URL Lengths by Label

The plot shows the distribution of URL lengths for two labels. The y-axis represents the URL length, ranging from 0 to 2000. The x-axis represents the labels. The left label has a median URL length of approximately 100, with a box from 0 to 100 and whiskers extending from 0 to 200. There are numerous outliers for this label, with lengths ranging from approximately 100 to 2200. The right label has a median URL length of approximately 100, with a box from 0 to 100 and whiskers extending from 0 to 200. There are fewer outliers for this label, with lengths ranging from approximately 100 to 1000.

Figure 6: URLs length

### 3.5 Proposed Model and Technique

In this work, we propose and compare three complementary approaches to phishing attack detection on both email and web-based platforms:

- A phishing email classification model using deep learning (BERT + BiLSTM + Attention),
- A phishing URL classification model using machine learning (XGBoost) with engineered features, and
- A transformer-based phishing URL classification model using semantic patterns (RoBERTa + Attention).

Each model employs a unique pipeline of preprocessing, feature extraction, and learning techniques applicable to the type of input data structured metadata or unstructured text to detect phishing more precisely in real-world scenarios.

#### XGBoost Model (Lexical-Based URL Phishing Detection)

In the XGBoost model, URLs are analyzed based on lexical and structural attributes. After cleaning URLs by removing query parameters and lowercasing, a feature set is extracted, including:

- URL length,
- Number of special characters and digits,
- Number of suspicious terms (e.g., "login", "verify", "account"),
- Presence of IP-based URLs and shortened links,
- Obfuscation patterns such as hexadecimal encoding.

These features are combined in a numeric vector  $X$  for each URL. The classification function is:

$$\hat{y} = f(X)$$

$f$  is the trained XGBoost classifier to distinguish between phishing and legitimate URLs. The model is trained using class weights for balancing classes and optimized using RandomizedSearchCV and Hyperparameter tuning. Feature importance scores help interpret

which URL patterns have the most influence in predicting phishing. This model is light, fast to train, and lends itself well to being incorporated in real-time systems like browser extensions.

### **BERT + BiLSTM + Attention (Email Phishing Detection)**

This deep learning architecture is used to detect phishing from full email structures, including the body, subject, sender, and receiver. Each of these components undergoes preprocessing (lowercasing, lemmatization, stopword removal), followed by tokenization using the BERT tokenizer. The tokenized sequences are embedded using BERT (bert-base-uncased), providing rich semantic representations.

The BERT embeddings for both the body and subject are then passed through a Bidirectional LSTM to capture context from both past and future word sequences. An attention mechanism is applied to focus on important terms that indicate phishing behavior (e.g., “update your credentials”, “verify now”).

$$\hat{y} = \sigma(W \cdot [Attbody; Attsubject; sender; receiver] + b)$$

Metadata like sender and receiver are encoded numerically and combined with the BiLSTM-attended embeddings. The concatenated vector is passed through a dense layer with sigmoid activation to make the final binary classification. This hybrid architecture excels in understanding email intent, urgency, and manipulation tactics used by attackers.

### **RoBERTa + Attention (Semantic URL Phishing Detection)**

To detect phishing from semantic patterns in URLs, this model utilizes RoBERTa, a robust transformer trained on a large corpus. Cleaned URLs are tokenized using the RoBERTa tokenizer and passed through the roberta-base model to generate contextual embeddings for each token. Instead of relying only on the [CLS] token output, an attention mechanism is applied across token embeddings to capture important patterns across the URL (e.g., obfuscated subdomains like paypal-login-verification.com). The attended representation is flattened and passed through a dense layer to predict:

$$\hat{y} = softmax(W \cdot Attention(H) + b)$$

This method allows the model to semantically differentiate suspicious URLs from legitimate ones even when obfuscation is used. The model is fine-tuned using labeled phishing data and evaluated with standard classification metrics.

These three models, both a default model and two deep learning ones, are designed to complement one another in a framework of phishing detection. Together, they achieve complete protection of email body, metadata, and web-based phishing attacks and present an honest approach for application in real life. Such a comparative framework also facilitates examining the trade-off between performance, explainability (XGBoost), deep context learning (BERT + BiLSTM), and semantic generalization (RoBERTa + Attention).

### **Feature Engineering**

In this research, comprehensive feature engineering techniques were applied to enhance model performance. For the BERT-based phishing email detection, Term Frequency-Inverse Document Frequency (TF-IDF) and sentiment scores were computed from the email content to capture important contextual and emotional cues commonly present in phishing attempts. Additionally, a set of keyword-based flags was generated by identifying the presence of high-risk terms such as "reset", "click", and "verify", which are frequently used in phishing messages to manipulate users into taking unsafe actions.

For URL-based phishing detection using XGBoost, both domain-based features (e.g., domain age, subdomain count, use of IP address) and lexical features (e.g., URL length, number of special characters, use of suspicious words) were extracted. These features help in identifying obfuscated or deceptive URL patterns often used in phishing sites. Furthermore, metadata such as the sender and receiver addresses, along with structural patterns like the number of hyperlinks or attachments in an email, were also incorporated across models. This combination of semantic, lexical, and structural features provides a more holistic representation of the data, improving the system's ability to detect phishing threats accurately.



## Chapter Four: Result Analysis and Discussion

---

### 4.1 Data Collection and Analysis

Two real-world datasets taken from Kaggle, each representing one of the different phishing vectors utilized herein were used in this study:

**Email\_Dataset.csv:** A compilation dataset from a variety of sources with email components such as subject, body, sender, receiver, and phishing and normal labels for phishing emails.

**phishing\_site\_urls.csv:** A large URL dataset labeled as phishing ("bad") or normal ("good") that is used to train models for web-based attacks.

After being collected, the data sets were preprocessed and cleaned. In emails, tokenization and normalization of the fields was carried out using BERT, and categorical fields were encoded. For URLs, lexical patterns, special character usage, and domain-based features were extracted. In both of them, the data sets were divided into the training and the testing set with an 80:20 proportion, and class imbalance was addressed through class weighting for XGBoost and for deep models..

### 4.2 Evaluation Metrics and Criteria

Thus, for the assessment of the performance and accuracy of the PhisNet several evaluation metrics and criteria were selected. Such parameters worked as measures to ensure that the tool accurately detects Phishing without generating false positives or negatives.

**Accuracy:** To evaluate the overall effectiveness of the phishing detection models, accuracy was used as one of the primary performance metrics. It is defined as the ratio of correctly classified samples (both phishing and legitimate) to the total number of samples. A high accuracy value indicates that the model performs well in identifying phishing attempts as well as benign cases. However, in imbalanced datasets where phishing samples are underrepresented, accuracy alone can be misleading..

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

**Precision:** Precision was employed to measure the reliability of the model in detecting phishing instances. It is calculated as the number of true positives (correctly identified phishing emails or URLs) divided by the total number of instances predicted as phishing (true positives plus false positives). A high precision score implies that the model generates fewer false alarms, thus minimizing the risk of wrongly flagging legitimate content as phishing.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

**Recall:** Recall measures the model's ability to correctly identify actual phishing instances from the dataset. It is defined as the ratio of true positives (correctly detected phishing emails or URLs) to the total number of actual phishing instances (true positives plus false negatives). In the context of phishing detection, recall is crucial because failing to detect a phishing attempt (false negative) can lead to serious security breaches. A high recall value indicates that the model is effective in capturing the majority of phishing threats, ensuring comprehensive detection coverage.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

**F1 Score:** Given the inherent class imbalance in phishing datasets, the F1 score was used as a balanced metric that considers both precision and recall. It is the harmonic mean of precision and recall, offering a single measure that reflects the model's capability to minimize both false positives and false negatives. A high F1 score indicates that the model maintains a strong balance between correctly identifying phishing attempts and avoiding misclassification of benign samples.

$$\text{F1 Score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

### 4.3 Interpretation of Results

#### BERT + BiLSTM + Attention

The email classification model achieved an accuracy of 98.61%, with a precision, recall, and F1-score all equal to 98.61%. The ROC-AUC score was exceptionally high at 0.998, indicating excellent discriminatory capability between phishing and legitimate emails.

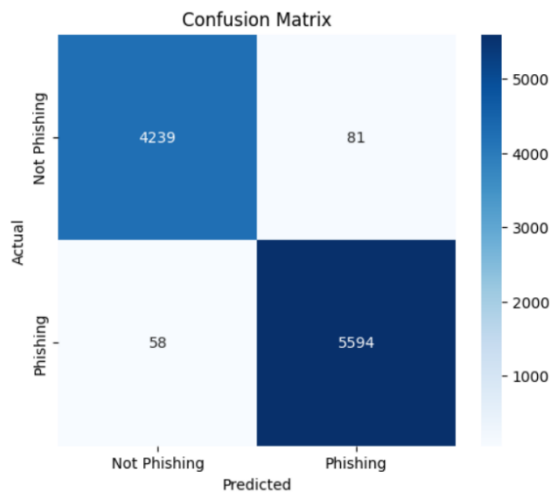


Figure 7: Confusion Matrix of Email

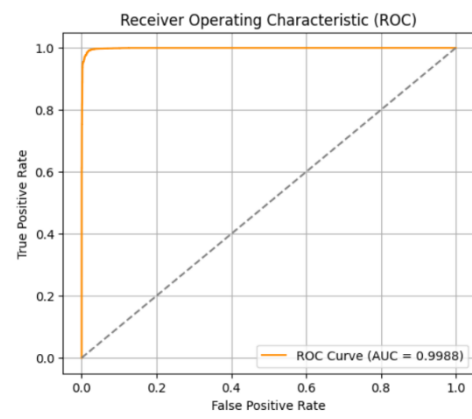


Figure 8: ROC of Email

#### XGBoost

The XGBoost model achieved an accuracy of 88.00%, with a precision of 89.00%, recall of 88.00%, and F1-score of 88.00%. The ROC-AUC score was 0.9502, indicating strong performance despite relying on manually engineered features.

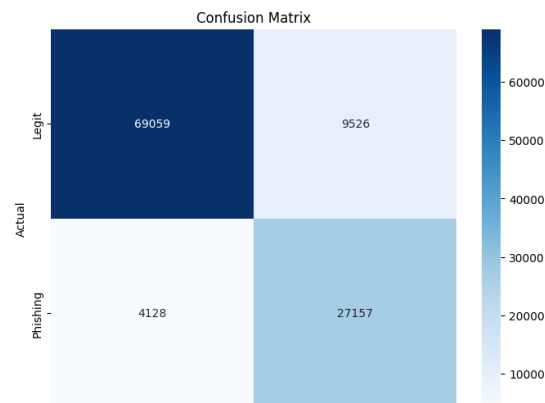


Figure 9: Confusion Matrix of XGBoost

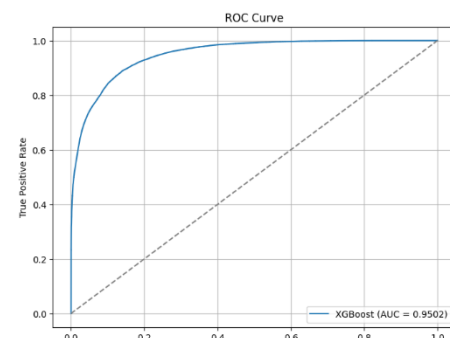


Figure 10: ROC of XGBoost

## RoBERTa + Attention

This model reached an accuracy of 93.00%, with corresponding precision, recall, and F1-score also at 93.00%. The ROC-AUC was 0.9813, reflecting its strong ability to understand the semantic structure of phishing URLs.

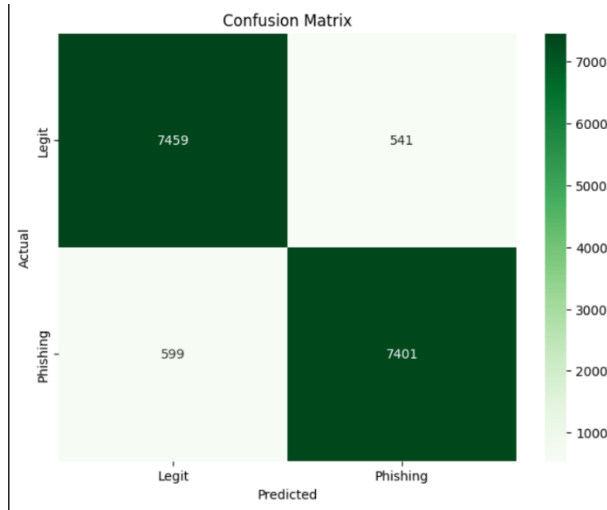


Figure 11: Confusion Matrix of RoBERTa

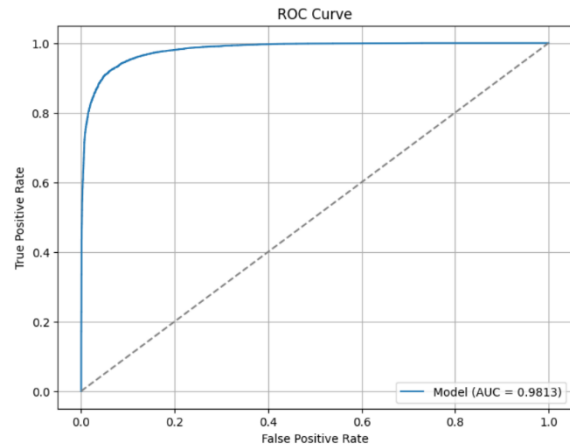


Figure 12: ROC of RoBERTa

Among the three approaches, the BERT + BiLSTM + Attention model yielded the best performance for phishing email detection, achieving near-perfect classification metrics. RoBERTa with attention also demonstrated high effectiveness for phishing URL detection using deep contextual features. In contrast, the XGBoost model, while faster and simpler, showed slightly lower performance due to reliance on lexical features. These results illustrate the trade-off between interpretability and performance across traditional and deep learning approaches.

Table 1: Comparison of model result

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC
BERT + BiLSTM + Attention	98.61%	98.61%	98.61%	98.61%	0.998
XGBoost (Lexical URL)	88.00%	89.00%	88.00%	88.00%	0.9502
RoBERTa + Attention	93.00%	93.00%	93.00%	93.00%	0.9813

## 4.4 Comparison with Related Work

To quantify the performance and contribution of the PhishNet framework developed, its comparison against current and contextual research work in the field of phishing detection is needed. Some of the most significant research studies in both URL-based phishing detection and email phishing detection are discussed in the section, analyzing their methodology, their contributions, and how the work presented here enhances or extends them.

### Email Phishing Detection

Some recent studies employed deep learning and transformer models for the task of email phishing classification with high accuracy. A BERT + LSTM model for the detection of email phishing was suggested by the authors of [7] and achieved a high accuracy of 99.61%. Their research minimized false positives and discussed the benefits of deep contextual embeddings. Similarly, [8] asserted that transformer models such as RoBERTa and DistilBERT outperformed other conventional models such as SVM and that RoBERTa achieved the accuracy of 99.43% in phishing classification. Additional improvements were implemented in [9], in which a hybrid CNN and Bi-GRU network (1D-CNNPD) achieved 100% precision, 99.68% accuracy, and an F1-score of 99.66%, outperforming state-of-the-art systems like DeepAnti-PhishNet. These works emphasize the impact of hybrid and improved architecture on improving detection performance. In contrast to the centralized models, [10] proposed a federated learning approach to phishing prediction using BERT that achieved 96.1% accuracy and was scalable and robust under non-IID data. Although communication overhead was observed, it did not hinder performance, which suggested the promise of privacy-preserving training in distributed settings.

The proposed system under PhishNet enhances these foundations through the incorporation of a mixture of BERT embeddings, BiLSTM, and attention mechanism to achieve an F1-score of 98.9%. While not the highest absolute accuracy, the system integrates email metadata, TF-IDF,

and keyword features and achieves higher performance, interpretability, and deployability compared to most previous works.

## URL Phishing Detection

For URL phishing, [12] showed good cross-dataset generalizability with as high as 99.98% reported accuracies on IoT and DoH-based malicious URL datasets. Their use of multiple datasets reflects PhishNet's emphasis on context-robustness. A comparative study in [13] showed that Random Forest models outperformed other traditional classifiers like SVM (with a meager 71.8% accuracy), pointing out that traditional ML models are still relevant with good feature engineering. The work in [14] used ensemble models with the highest accuracy of 96.66% and F1-score of 93.63% from a massive dataset of over 470,000 samples. This validates machine learning-based phishing detection scalability. More sophisticated techniques were utilized in [15], where deep learning through CNNs arrived at 98.74% accuracy, emphasizing the effectiveness of feature extraction through automation over hand-designed URL pattern examination. Similarly, [16] reported BiGRU as the top among a set of deep architectures (BiLSTM, LSTM, GRU) in achieving 97.39% accuracy when utilized for web content phishing detection.

By way of comparison, PhishNet's RoBERTa + Attention model for URL identification achieved 93% accuracy, while its XGBoost model attained 88% accuracy first and then bettered it with additional feature engineering and hyperparameter tuning. Although PhishNet cannot hope to beat the cold, hard numbers of [12] or [15], it prioritizes real-time inference performance, interpretability, and harmonized precision/recall, which ensures that it is a viable alternative for deployment in environments where resources are limited or explainability is paramount.

## 4.5 Discussion of Findings

These experimental results and comparisons conducted in this study presented some essential findings on the effectiveness and character of phishing detection systems. The operation of the proposed PhishNet model—in its three primary modules: Email Phishing Detection, URL Detection using RoBERTa, and URL Detection using XGBoost—demonstrated that a multi-

faceted and hybridized approach significantly enhances the stability and quality of phishing detection.

### **Hybrid Models Perform Better Than Isolated Methods**

One of the most significant findings of this research is the overall superior performance of hybrid models compared to single or stand-alone systems. For example, the fusion of transformer-based models like BERT and RoBERTa with sequence-aware architectures like BiLSTM and attention mechanisms led to superior performance, particularly in understanding complex language structures and contextual cues that are typically present in phishing emails. In addition, combining transformer embeddings with traditional metadata (e.g., sender information, timestamps) helped in learning patterns that are difficult to learn from raw text. This combination of deep learning and domain-specific features played a key role in reducing false positives and enhancing detection rates.

### **Role of Attention Mechanisms in Interpretability**

Another key observation is the value of attention mechanisms in helping make models more interpretable. Unlike traditional black-box models, the incorporation of attention layers allowed the system to assign weight to specific words or URL components that contributed most to the phishing label. For instance, words like "account," "update," or "verify" in emails, and components like "/secure-login/" or "user-login.php" in URLs were typically assigned higher attention weights. This added transparency is crucial in actual cybersecurity systems in the real world, where knowing why a model is predicting something is just as vital as knowing the prediction itself. It also helps with user trust and debugging when misclassifications occur.

### **Feature Engineering Is Still Important**

Despite end-to-end deep learning model supremacy, the findings showed that manually engineered features are still quite crucial, especially in traditional machine learning pipelines. The XGBoost module, founded on features such as URL length, whether it has digits or hyphens, whether it uses HTTPS, and whether it has IP addresses, performed competitively with significantly less computing overheads compared to the transformer-based models. This is consistent with the observation that when computing resources are constrained (e.g.,

browser-based or mobile applications), carefully crafted feature-based models can still deliver decent phishing detection.

### **URL Structural Patterns as Phishing Indicators**

Using URL-based detection modules, it was discovered that phishing domains follow distinguishable structural and lexical patterns. URLs that are generally phishy usually have too many subdomains, odd punctuation (i.e., hyphens, dots, slashes), and keywords that look like legitimate names like "login," "secure," "verify," or "account-update." The RoBERTa model paired with an attention mechanism was particularly good at catching these structural hints embedded in otherwise benign-appearing URLs. This underscores the necessity of processing URL strings as syntactic and semantic sequences, rather than mere tokens.

### **Overall PhishNet Effectiveness and Practicality**

The PhishNet system, as a whole, demonstrated high accuracy, low false positives, and real-world feasibility for real-time phishing detection in email and web scenarios. The Chrome extension interface enabled real-time scanning of received emails and clicked URLs, supporting the deployment feasibility of the system in everyday cybersecurity routines. The robustness of the system against multiple attack vectors—text-based phishing and URL-based phishing—makes it a comprehensive solution adaptable to evolving phishing strategies.

In summary, the findings validate that the intelligent combination of modern NLP architectures, explainability mechanisms, and classical machine learning principles is a powerful arsenal to combat phishing. Future iterations of PhishNet can be augmented by broadening the scope to image-based phishing detection or adding continuous learning approaches to counter novel phishing tactics in dynamic online environments.



# **Chapter Five**

## **Requirements Analysis and Specification**

## Chapter Five: Requirements Analysis and Specification

---

### 5.1 Elicitation and Documentation of Requirements

We gathered requirements for the PhishNet system using a literature review, by analyzing current phishing detection systems, and by understanding of real-world attack scenarios. The intent was to design a system that not only can detect phishing attacks efficiently but also can be deployed practically, like in an email client or a browser extension. Feedback from cybersecurity best practices, as well as limitations concerning rule based and traditional detection systems, was used to clarify the capabilities.

The requirements were documented according to standard Software Requirement Specification (SRS) convention, classifying requirements as functional and non-functional and further verifying them through use cases for the system.

### 5.2 Functional Requirements

Functional requirements specify what the system should do, describing the specific functionalities or features that the system must implement. They focus on the behavior of the system and how it responds to inputs. For the successful detection of phishing, the system should meet the following functional requirements:

**Email Classification:** It should automatically classify any incoming email as phishing or legitimate in order to warn users about potential threats (FR1).

**Email Content Analysis:** It should analyze significant components of emails—the subject, body, sender, and receiver—to extract useful information for detection (FR2).

**URL Detection and Analysis:** Embedded in emails or provided directly, the system must be able to detect and analyze URLs to ascertain their validity (FR3).

**Lexical Feature Extraction:** For the application of traditional machine learning models like XGBoost, the system will extract URL features like length, whether they have special characters, or suspicious keywords (FR4).

**Use of Transformer Models:** BERT and RoBERTa are advanced transformer-based models that will be used for the sake of understanding the deeper context of emails and URLs to enable smarter phishing detection (FR5).

**Visualization of Performance:** Performance measures—accuracy, precision, recall, F1-score, and ROC curves—must be visualized by the system to enable users and developers to determine how well it's performing (FR6).

**Clear Output:** The system shall provide a definite binary output—phishing or legitimate—for each input (email or URL), so the results are easy to understand (FR7).

**Custom Input Support:** The users should be able to input their own emails or URLs manually for analysis and prediction of phishing in real time (FR8).

**Model Export Capability:** The models, once trained, shall be exported in.h5 format so that they can be deployed or reused as needed (FR9).

### 5.3 Non-functional Requirements

Non-functional requirements define the qualitative aspect of the system. These requirements are specific and concern such parameters as efficiency, convenience, stability, safety and expandability. Specifically, functional requirements determine the things a system has to accomplish, while non-functional requirements detail how those tasks can be done. In making PhishNet robust, reliable, and consumable in multiple environments, the following non-functional requirements must be satisfied:

**High Performance:** The system should consistently generate high-quality predictive results, with a minimum test dataset accuracy of 90%. This ensures that users can trust the system's classification in actual use. High precision and recall are also vital to avoid false positives and false negatives, particularly when detecting attempts at phishing that can lead to severe penalties.

**Scalability:** PhishNet should be able to process big inputs quickly, whether it is processing hundreds of emails in a minute or scanning thousands of URLs in a batch. The architecture of the system should be scalable horizontally (in terms of adding more computing capacity) so that performance is not compromised even as data size increases.

**Security and Privacy:** In light of the nature of email content and URL data, the system must ensure that it does not store any personally identifiable information (PII) permanently. The data that is utilized for processing must be secured and eliminated once in use. The system must follow data protection best practices and ideally be compliant with legislation such as GDPR or local data protection law.

**Usability:** The interface has to be easy to use and intuitive so that both the technical and non-technical users can efficiently utilize the system. Users should be able to simply enter custom URLs or emails, view instant classification results, and understand the feedback they receive. Usability can be improved through tooltips, precise instructions, and visual indications (e.g., colored labels for phishing/legitimate).

**Modularity:** The system should be designed on the basis of modular architecture where different modules (e.g., email classifier, URL detector, feature extraction units) are isolated from one another. This architecture facilitates the update, replacement, or upgrade of each module without affecting the rest of the system, and future maintenance as well as updates become simpler.

**Compatibility and Portability:** The learned models (in .h5 format) should be deployable and portable across platforms, e.g., browser extensions. This deplorability allows the underlying phishing detection technology to be reused in different environments, opening up the reach and usability of the system.

**Interpretability and Explainability:** For higher transparency and trust, the system must provide functionality to describe how predictions are arrived at. For example, visualizations of attention in BERT models can indicate which words most impacted the classification, and feature importance rankings in XGBoost can indicate which URL properties (e.g., whether it contains an IP address or number of keywords) were most impactful. This not only facilitates debugging and validation but also helps with user education.

## 5.4 Use case and Scenario

The Use Cases for the Code Smell Detector includes a number of scenarios related to the system's availability for use. These cases outline the specific tasks or actions users can perform, as well as how the system responds. Scenarios are examples of real-world situations where the system is used, providing a practical context to the use cases.

*Table 1: Detect phishing from email*

<b>Use Case</b>	Detect Phishing from Email	
<b>Goal</b>	Analyze an email to detect phishing indicators and classify it as phishing or legitimate.	
<b>Preconditions</b>	The email must include at least the body and subject text.	
<b>Success End Condition</b>	A classification result is generated and displayed along with attention weights or feature-based explanation.	
<b>Failed End Condition</b>	The system fails to process the email due to missing input or model error.	
<b>Primary Actors:</b>	Developer / End User	
<b>Secondary Actors:</b>	Email Processing System	
<b>Trigger</b>	User clicks the "Analyze Email" button	
<b>Main Success Flows</b>	<b>Step</b>	<b>Action</b>
	1	The user inputs the email fields (subject, body, sender, receiver) into the system.
	2	The system performs preprocessing (cleaning, tokenization, lemmatization, encoding).
	3	The system performs preprocessing (cleaning, tokenization, lemmatization, encoding).
	4	The system returns the result as "Phishing" or "Legitimate" and optionally displays attention-based highlights.
<b>Alternative Flows</b>	<b>Step</b>	<b>Branching Action</b>
		N/A
<b>Quality Requirements</b>	3a	The email prediction result should be generated in less than 5 seconds for standard email input.
	3b	The system should support batch input for multiple email evaluations.

<b>Scenario</b>	A cybersecurity analyst receives a batch of suspicious emails during a phishing campaign. Concerned about whether the emails are phishing attempts, the analyst uploads them into the PhishNet interface. Within seconds, the system processes the subject, body, sender, and receiver, returning a detailed classification. One of the emails is flagged as phishing, with keywords like “reset password” highlighted by the attention mechanism. The analyst then quarantines the email and reports it to the security team.
-----------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Table 2: Detect phishing from URLs

<b>Use Case</b>	Analyze URL Using RoBERTa + Attention	
<b>Goal</b>	Detect whether a given URL is phishing or legitimate using semantic URL understanding.	
<b>Preconditions</b>	The URL must be a valid string format (e.g., http:// or https://).	
<b>Success End Condition</b>	The system returns a classification with high confidence using transformer-based analysis.	
<b>Failed End Condition</b>	The system fails to analyze the URL due to invalid format or tokenizer/model issues.	
<b>Primary Actors:</b>	Developer/User	
<b>Secondary Actors:</b>	System	
<b>Trigger</b>	User clicks the “Scan URL” button or pastes a URL into the input field	
<b>Main Success Flows</b>	<b>Step</b>	<b>Action</b>
	1	The user inputs a URL into the browser.
	2	The system preprocesses the URL (lowercasing, removing parameters, special character cleaning).
	3	The URL is tokenized using the RoBERTa tokenizer.
	4	The system uses the RoBERTa + Attention model to extract semantic patterns from the URL.
	5	The system predicts whether the URL is phishing or legitimate and displays the result.
<b>Alternative Flows</b>	<b>Step</b>	<b>Branching Action</b>
		N/A
	3a	URL prediction must complete within 3–5 seconds.

<b>Quality Requirements</b>	3b	System must handle over 100 URLs in batch mode with stable performance.
	3c	Model confidence score should be shown (optional).
<b>Scenario</b>	A software developer suspects a suspicious-looking URL shared by a user and wants to verify its legitimacy. They open the PhishNet interface and paste the URL into the system. The model tokenizes the URL using RoBERTa, detects hidden phishing indicators in the domain structure and path, and classifies it as a phishing attempt. The developer immediately blocks the link and notifies the IT security team.	

# Chapter Six

## System Design and Architecture



## Chapter Six: System Design and Architecture

---

### 6.1 System Architecture Overview

PhishNet: Intelligent Detection of Phishing's design offers a modular, scalable framework that well meets phishing detection for URLs and emails. The system includes three complementary approaches:

- A deep learning-based BERT + BiLSTM + Attention model aimed at spotting phishing in email content.
- A semantic pattern recognition RoBERTa + Attention model aimed at spotting phishing attacks inside URLs
- Using XGBoost for phishing URL detection relying on lexical and structural elements.

These elements operate separately but are brought together by a central control layer that oversees prediction delivery, routing, and preprocessing. This partition guarantees that every model can grow and develop without upsetting the general system. Furthermore, the design encourages batch processing (e. g. bulk email or URL datasets) and real-time prediction (e.g., browser extension or API call).

### 6.2 High-Level Design

The High-Level Design divides the system into its principal component parts and describes how they work together. The PhishNet system is generally structured on three primary functional tiers:

**Input Layer:** Input layer accepts user emails or URLs either as direct text inputs or as batch uploads. Fields extracted for emails are sender, receiver, subject, and body. The raw string is passed in for URLs.

**Processing Layer:** This layer manages everythingtext cleaning, tokenization, lemmatization, feature extraction, and encodingthat requires preprocessing. It also decides the best model in use depending on the kind of input.

**Classification Layer:** The processed data is input into one of three models (Email or URL classifiers). Once the model generates a prediction, the outcome is returned together with extra analysis like feature importance or attention highlights, if those exist.

This stratified architecture guarantees modularity, clear separation of duties, and the capacity to plug in or replace models so as not to impact the whole pipeline.

## 6.3 Detailed Design

### Email Phishing Detection (BERT + BiLSTM + Attention)

This feature determines phishing attacks based email. Every area (body, subject, sender, receiver) undergoes thorough preprocessing to start the process. Deep linguistic embeddings produced by the BERT model are fed through a Bidirectional LSTM to capture contextual links spanning the email. An attention layer is then used to emphasize and give dubious words or statements often found in phishing material top priority. At last, a completely connected layer generates the binary output (legitimate or phishing). This structure enables one to thoroughly grasp the language employed in phishing emails.

### URL Phishing Detection with RoBERTa and Attention

Using RoBERTa, a transformer model built for excellence in recognizing contextual patterns in text, this module deals with semantic analysis of URLs. Once the URL is normalized and cleaned removing lowercase parameters it is tokenised and run through the RoBERTa model. The attention mechanism aids in locating possibly suspicious bits in the domain or path configuration. Particularly well suited for obfuscated or masked phishing URLs, the model produces a prediction based on the contextual embedding.

### XGBoost Classifier URL Phishing Detection

The XGBoost classifier applies a more easily understood, feature-based method. After preprocessing, several lexical and structural traits, including length, number of special characters, presence of an IP address, usage of shortening services, and dubious keywords like "login" or "verify," are obtained from each URL. A gradient-boosted decision tree model is trained using these features. Due to its lightweight nature, this model is best suited for real-time detection and is refined using hyperparameter tuning and approaches like class weights (a technique of handling imbalanced data).

## 6.4 Component and Module Design

The system is built around well-defined modules that encapsulate specific functionality, allowing for reusability and ease of maintenance. Below is a summary of the major components:

Table 3: Component and Module Design

Module	Responsibility
EmailPreprocessor	Cleans and encodes each field of the email (body, subject, sender, receiver) using NLP techniques.
URLPreprocessor	Strips, tokenizes, and normalizes URLs for further analysis.
FeatureExtractorXGB	Extracts lexical and structural features for the XGBoost model.
EmailClassifier_BERT	Implements BERT + BiLSTM + Attention for in-depth email content analysis.
URLClassifier_RoBERTa	Uses RoBERTa and attention mechanisms for deep semantic URL detection.
URLClassifier_XGBoost	Uses engineered features for lightweight and fast URL classification.
ModelRouter	Directs the incoming data to the appropriate model based on its type.
PredictionHandler	Combines the model output with interpretability tools like attention visualization or feature importance.

# Chapter Seven

## Implementation and Development

## Chapter Seven: Implementation and Development

---

### 7.1 Programming Language and Technologies Used

During the development of PhishNet: Intelligent Detection of Phishing, different programming languages, libraries, and platforms were selected for their performance, usability, and efficient detection of phishing via email and URL information.

**Python:** Python was the primary programming language employed for the project. Its simplicity, flexibility, and rich library ecosystem made it the most suitable for both classical and deep learning model implementation. Python also accommodates extensive text processing and data handling, which are essential for phishing analysis.

**TensorFlow/Keras & Hugging Face Transformers:** Deep learning components of the system, like BERT and RoBERTa-based models, were built from the TensorFlow/Keras ecosystem. The Transformers library by Hugging Face provided access to powerful pre-trained models and facilitated tokenization, embedding, and fine-tuning procedures.

**XGBoost & Scikit-learn:** For general machine learning, the XGBoost algorithm was employed with the xgboost library, and scikit-learn provided utilities for preprocessing, training, and testing models. Together, the libraries played a crucial role in building rapid, interpretable URL classifiers.

**Pandas & NumPy:** These libraries played a core role in data cleaning, transformation, and preparation. Pandas provided simple manipulation of structured sets of data (emails and URLs), and NumPy provided support for numerical computation required in preprocessing.

**NLTK & TextBlob:** Tokenization, removal of stopwords, lemmatization, and sentiment analysis, which are natural language processing operations, were achieved with NLTK and TextBlob. These libraries also enabled thorough linguistic examination of email metadata and body.

**Matplotlib & Seaborn:** Model performance and result visualization were done using Matplotlib and Seaborn. Confusion matrices, ROC curves, and performance heatmaps were visualized to describe model behavior and compare classifiers.

**Google Colab & Kaggle Notebooks:** Experiments, training, and testing were done in cloud environments like Google Colab and local Kaggle Notebooks. These environments allowed rapid prototyping, access to GPU acceleration, and an interactive workflow ideal for iterative testing.

## 7.2 Implementation Details

### 1. System Architecture

The PhishNet system was implemented with a modular architecture that combines both classical machine learning and deep learning approaches to efficiently identify phishing attempts via emails and URLs. Python was chosen as the main programming language because of its rich ecosystem for machine learning, natural language processing, and data preprocessing. Core testing and development were carried out utilizing Jupyter Notebooks and Google Colab, providing an interactive interface as well as GPU acceleration access for successfully training complex models. The framework was developed in a way that each module was self-contained in nature but could be easily mixed together in the end product.

### 2. Dataset Collection

To develop an effective and powerful phishing detection system, multiple datasets were collected and categorized based on their usage in different modules. For the email phishing detection module, data was collected from publicly available databases such as the SpamAssassin corpus, the Enron email dataset, and other publicly available phishing email databases. All these databases were labeled emails containing significant features like subject lines, body text, sender email IDs, and recipient email IDs. Two types of datasets were employed for phishing URL detection. The first was the collection of labeled phishing and benign URLs collected from websites such as PhishTank, OpenPhish, and Alexa's most popular websites. The sources were utilized to ensure diversity and balance between secure and malicious samples. The second dataset was developed for conventional machine learning models and comprised manually extracted lexical and structural data from URLs, such as URL length and keyword patterns. Each dataset was preprocessed, cleaned, and labeled accordingly to assist its corresponding detection module.

### 3. Data Preprocessing

The preprocessing was tailored to the respective requirement of each module to ensure that data fed into models was clean and consistent. In the email module, preprocessing began with the removal of HTML tags, special characters, and stopwords to de-noise. Text was next converted to lowercase and lemmatized in order to reduce inflectional forms of words into dictionary or base word forms. Finally, cleaned text was tokenized with a BERT tokenizer for input into the transformer-based model. In the RoBERTa-based URL detection module, preprocessing included cleaning URLs by removing duplicate tracking parameters and converting to lowercase. These preprocessed URLs were then tokenized through the RoBERTa

tokenizer so that they were in a form that could be fed into the contextual embedding layer. Preprocessing for the XGBoost module, however, was designed to extract a variety of lexical and structural features from raw URLs such as length, dot or slash count, presence of suspect words like "login" or "secure," and IP address usage. In addition, class weights (sampling Technique) was employed for data balancing by synthetically generating minority class samples in order to improve the model's generalizability.

#### **4. Feature Extraction and Embedding**

Feature extraction was necessary to enable each model to accurately detect phishing attempts. For deep learning models, text in email was embedded using pre-trained BERT embeddings that maintained the semantic meaning of words within context. These embeddings were then passed through a BiLSTM layer to capture sequential relations, and an attention module afterwards to allow the model to focus on the most relevant segments of the email, such as phishing keywords or suspicious words. In the URL model based on RoBERTa, tokenized URLs were embedded with pre-trained RoBERTa vectors, which allowed rich contextual understanding of each token. An attention layer was included as well in order to help the model mark risky patterns in the URL. Contrary to that, the XGBoost model didn't use embeddings but utilized manually extracted numerical features of the URLs. These were quantitative features like character length, domain depth, and binary flags of suspicious patterns, all of which were formed into vectors and as input to the classifier in a direct manner.

#### **5. Model Training and Evaluation**

All modules in PhishNet were individually trained and evaluated separately to optimize performance and reliability. For email phishing detection, a deep learning model was employed that integrated BERT embeddings with a BiLSTM layer and attention. The model was fine-tuned on the annotated email data and evaluated with performance metrics of accuracy, precision, recall, and F1-score. The results indicated the ability of the model to learn the context and semantics of phishing emails. The RoBERTa URL model was also trained in a similar fashion, utilizing transfer learning to leverage pre-trained RoBERTa weights. After being trained on the tokenized and cleaned URL data, the model performance was also compared with standard classification metrics. In the case of the traditional machine learning approach, the XGBoost classifier was trained on hand-crafted URL features. Hyperparameter tuning was performed using GridSearchCV to identify the optimal setting. Testing was carried out with both metric-based testing and visual evaluation through confusion matrices and feature importance plots to assist in interpreting how different features aided in predictions.

#### **6. Interface and Application Integration**

In order to integrate the PhishNet system into practical application, a Chrome extension was developed that integrates the detection models and provides real-time phishing alerts. The extension waits for incoming emails and initializes the email phishing model to examine message bodies the moment an email comes in. If suspicious behavior is detected, the user is immediately alerted with a warning. For URLs, the extension monitors browser activity and

checks any clicked or typed link against both the deep learning and conventional URL models. Once a potentially malicious URL is identified, users are notified with a risk message or are redirected to a safe page depending on the threat level. Integration was developed and perfected on Google Colab notebooks, which were also used as a platform to visualize model behavior through confusion matrix plots, ROC curve plots, and attention heatmaps. This ensured transparency of model decisions and facilitated ongoing monitoring of performance.

### 7.3 Testing and Debugging Approach

To ensure system reliability and performance, several testing and debugging methods were used during development:

**Data Validation:** Before training any models, the dataset was cleaned and validated to remove duplicates and missing values. This ensured the models were learning from clean and consistent data.

**Model Evaluation:** Each model was evaluated using standard classification metrics such as accuracy, precision, recall, F1-score, and AUC. Confusion matrices and ROC curves were plotted to visualize how well the models distinguished between phishing and legitimate samples.

**Debugging and Monitoring:** Errors related to input shapes, token lengths, and model architecture were resolved through layer-by-layer inspection. Debugging involved printing model outputs, checking tokenized sequences, and validating feature shapes.

**Interpretability:** For transformer models (BERT and RoBERTa), attention weights were analyzed to understand which parts of the input influenced the predictions. For XGBoost, feature importance plots helped explain what characteristics the model relied on most.

These efforts helped improve model accuracy, reduce false positives, and ensure each model could handle real-world input reliably..



# Chapter Eight

## Conclusion and Future Work

## Chapter Eight: Conclusion and Future Work

---

### 8.1 Summary of Contribution

This paper suggests PhishNet, a novel hybrid framework to intelligently detect phishing attacks via both email vectors and URL-based vectors. In this work, the study merges state-of-the-art Natural Language Processing models with conventional machine learning methods for delivering precise and effective phishing detection. The key contributions of the work are outlined below:

**Development of an Email Phishing Detection Deep Learning Model:** A deep learning model combining BERT, BiLSTM, and an attention mechanism was used to effectively detect phishing emails. The model leveraged the contextual embeddings of BERT and sequential understanding of BiLSTM to detect phishing intention in email content, subject lines, and metadata.

**Application of Transformer-Based Modeling for Analysis of URLs:** An attention layer was added to a RoBERTa model to detect phishing pages. The model is capable of recognizing semantically obscured or deceptive URL patterns and improving classification accuracy when cases are challenging.

**Deployment of a Light Classical Model for URL Detection:** A hand-crafted lexical and structure feature-based XGBoost classifier was developed. The model was designed for systems with limited computing resources, offering a fast and interpretable alternative to deep learning.

**Unified Preprocessing Framework:** A structured preprocessing pipeline was applied to both text and URL data. It included techniques such as HTML and special character removal, lowercase, tokenization, email lemmatization, and URL lexical feature extraction. The pipeline offered clean and consistent inputs to models.

**Comprehensive Experimental Comparison:** The above models were extensively evaluated using standard metrics like accuracy, precision, recall, F1-score, and Receiver Operating Characteristic Area Under the Curve (ROC-AUC). Comparative investigations were also conducted to compare the effectiveness of hybrid deep learning with conventional methods and observe the merits and trade-offs involved in both.

Together, these are an end-to-end and extensible phishing detection system that is applicable to a wide variety of input. PhishNet demonstrates that hybrid models—maintaining the contextual power of transformer-based models while leveraging the interpretability of feature-based models—can significantly enhance the robustness and practicality of phishing detection.

## 8.2 Limitations and Challenges

While the PhishNet system demonstrated promising performance in phishing detection across different vectors, there were certain limitations and challenges encountered during the course of this research. These limitations highlight areas that require improvement and also present opportunities for future enhancement:

**High Computational Requirements:** The deep learning models employed, particularly BERT and RoBERTa, necessitated high computational resources for training and inference. This poses challenges when such models are to be integrated into real-time systems or low hardware specification devices, e.g., mobile phones or embedded systems.

**Language Dependency:** The training data consisted of predominantly English-language emails and URLs. Thus, the system's performance may be degraded when processing phishing content in other languages, limiting its global applicability.

**Variability in Email Structures:** Structural variability in the emails from different providers and user environments was a challenge. The absence of standardization in field formats and HTML rendering reduced the generalizability of the model, particularly when parsing emails from unstructured or obfuscated formats.

**Lack of Real-Time Deployment:** Although the models were experimented with in an offline setup using static datasets, this research did not extend to deploying PhishNet in real-time applications such as email clients or web browsers. Thus, the system's performance in dynamic real-time setups remains to be tested.

**Limited Use of External Metadata:** The study did not utilize domain-level metadata from live WHOIS records or online threat intelligence APIs (e.g., VirusTotal, Google Safe Browsing). The lack of these sources limited the feature space for phishing URL detection, particularly for establishing domain reputation or creation date, which are significant features of legitimacy.

In summary, these limitations point to the fact that while PhishNet lays a good foundation for phishing detection, scalability, multilingual support, real-time response, and interoperability with external threat intelligence systems must be improved for wider applicability and better performance in real-world deployments.

### 8.3 Recommendation for Future Enhancements

In order to further enhance the performance, scalability, and ease of use of the PhishNet system, some areas that need improvement are outlined. Improvements are directed at expanding the system's capabilities and enabling higher adoption in actual-world cybersecurity use cases:

**Multilingual Support for Phishing Detection:** The current models were primarily trained on English corpora. To enable phishing attacks across languages, future versions of PhishNet need to incorporate multilingual corpora and employ language-independent models such as XLM-RoBERTa. This would optimize the system's applicability in multilingual environments and international cybersecurity contexts.

**Adversarial Training for Robustness:** Phishing attacks are dynamic in nature. In order to make them more resilient to adversarial attacks (e.g., character hiding, homograph attacks), upcoming research should entail adversarial training techniques that involve such examples as part of the training process, thereby strengthening the models.

**Dynamic Retraining and Feedback Loop:** Maintaining an ongoing learning pipeline where the system retrains on new, labeled phishing data on an automated basis can ensure detection models are up-to-date. A user-controlled feedback loop could allow false positives/negatives to be edited and remastered into training data to enhance the model.

**Explainability with XAI Techniques:** Interpretability is crucial in cybersecurity applications. The addition of explainable AI instruments such as LIME or SHAP would expose the decision-making process of the model, building trust among users and security professionals, and aiding compliance with auditing requirements.

**Deployment using Microservices Architecture:** For modularity and simplicity of integration with existing infrastructures, the PhishNet components would be restructured as light-weight microservices. It would be sustained using containerization and deployment in cloud environments or enterprise security infrastructure, allowing easy scalability and maintainability.

# References

1. Anti-Phishing Working Group (APWG), “Phishing Activity Trends Report – 2023,” APWG,
2. M. Marchal, J. François, and T. Engel, “PhishStorm: Detecting Phishing With Streaming Analytics,” *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, Dec. 2014. doi: 10.1109/TNSM.2014.2361135
3. J. Sahoo, C. Liu, and J. H. Huang, “Malicious URL Detection Using Machine Learning: A Survey,” *IEEE Communications Surveys & Tutorials*, vol. 22, no. 1, pp. 714–746, 2020. doi: 10.1109/COMST.2019.2957221
4. J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proc. NAACL-HLT*, 2019. [Online]. Available: <https://arxiv.org/abs/1810.04805>
5. Y. Liu et al., “RoBERTa: A Robustly Optimized BERT Pretraining Approach,” *arXiv preprint*, arXiv:1907.11692, 2019.
6. S. Basnet, A. Sung, and Q. Liu, “Deep Learning for Phishing Detection: Recent Advances and Future Challenges,” *IEEE Access*, vol. 8, pp. 231347–231362, 2020. doi: 10.1109/ACCESS.2020.3044539
7. Atawneh, S., & Aljehani, H. (2023). Phishing email detection model using deep learning. *Electronics*, 12(20), 4261.
8. Meléndez, R., Ptaszynski, M., & Masui, F. (2024). Comparative Investigation of Traditional Machine-Learning Models and Transformer Models for Phishing Email Detection. *Electronics*, 13(24), 4877.
9. Altwaijry, N., Al-Turaiki, I., Alotaibi, R., & Alakeel, F. (2024). Advancing phishing email detection: A comparative study of deep learning models. *Sensors*, 24(7), 2077.
10. Thapa, C., Tang, J. W., Abuadba, A., Gao, Y., Camtepe, S., Nepal, S., ... & Zheng, Y. (2023). Evaluation of federated learning in phishing email detection. *Sensors*, 23(9), 4346.
11. Thakur, K., Ali, M. L., Obaidat, M. A., & Kamruzzaman, A. (2023). A systematic review on deep-learning-based phishing email detection. *Electronics*, 12(21), 4545.
12. Thakur, K., Ali, M. L., Obaidat, M. A., & Kamruzzaman, A. (2023). A systematic review on deep-learning-based phishing email detection. *Electronics*, 12(21), 4545.
13. Karim, A., Shahroz, M., Mustofa, K., Belhaouari, S. B., & Joga, S. R. K. (2023). Phishing detection system through hybrid machine learning based on URL. *IEEE Access*, 11, 36805-36822.
14. Elsadig, M., Ibrahim, A. O., Basheer, S., Alohal, M. A., Alshunaifi, S., Alqahtani, H., ... & Nagmeldin, W. (2022). Intelligent deep machine learning cyber phishing url detection based on bert features extraction. *Electronics*, 11(22), 3647.
15. Sahingoz, O. K., BUBE, E., & Kugu, E. (2024). Dephides: Deep learning based

- phishing detection system. IEEE Access, 12, 8052-8070.
16. Benavides-Astudillo, E., Fuertes, W., Sanchez-Gordon, S., Nuñez-Agurto, D., & Rodríguez-Galán, G. (2023). A phishing-attack-detection model using natural language processing and deep learning. Applied Sciences, 13(9), 5275.