Leveraging Metadata to Detect Phishing Where It Spreads

by

Mina Erfan

Thesis submitted to the University of Ottawa
in partial fulfillment of the requirements for the degree of
Master of Computer Science (Concentration in Applied Artificial Intelligence)

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Examining Committee

The following served on the Examining Committee for this thesis.

Carleton Member: Olga Baysal
Associate Professor, School of Computer Science
Carleton Univeristy

Internal Member(s): Paria Shirani
Assistant Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Supervisor(s): Paula Branco
Associate Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Guy-Vincent Jourdan
Professor, School of Electrical Engineering & Computer Science
University of Ottawa

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University of Ottawa regulations concerning plagiarism, including those regarding consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Phishing is one of the most persistent and evolving attacks in cybersecurity. Attackers are exploiting legitimate domains and various hosting and third-party platforms, making detection challenging. Many existing proactive detection methods still focus on identifying newly registered domains, often by monitoring data sources that signal suspicious domain activities. Although these approaches are effective at capturing a subset of attacks – specifically those involving attacker-owned domains – they leave open the question of how many phishing campaigns actually rely on attacker-controlled infrastructure compared to compromised or third-party platforms. If attacker-owned domains are not the dominant category, then relying only on infrastructure-level data sources may miss a large portion of phishing activities. In such cases, effective detection needs to focus on other dimensions of phishing behavior, such as how they propagate across platforms and how users interact with them.

We begin with a phishing domain ownership taxonomy that distinguishes between attacker-owned, compromised, and third-party platforms. Using our large-scale datasets, PhishXtract, collected over a year from multiple reporting feeds, we analyze how attackers leverage these different infrastructures. Our findings show that most phishing attacks are not hosted on attacker-owned domains. Instead, attackers often abuse legitimate infrastructures and third-party platforms to create their campaigns, which makes infrastructure-level indicators alone insufficient for detection. To address this limitation and recognizing the growing role of social media, we investigate it as a source of phishing propagation and explore its potential for alternative detection. These platforms are not only channels where phishing links spread, but also rich sources of data – ranging from message content to metadata and propagation patterns – that can be used to identify phishing activity at an early stage. By focusing on these signals, we use our *TelePhish* dataset to develop a lightweight approach that detects phishing at the message level without relying on infrastructure-based indicators.

Finally, we compare this lightweight, propagation-based model against Large Language Model (LLM)s to assess whether advanced text-driven approaches outperform metadata-driven approach in social media. Our findings suggest that our lightweight, propagation-based models perform better for near real-time phishing detection, and it offers a balance between effectiveness and efficiency.

Acknowledgements

I would like to sincerely thank my supervisors for their continuous guidance, patience, and support throughout this work. I am also grateful to my lab colleagues and friends who shared their insights and offered support along the way. Finally, I want to thank my family and friends for their endless understanding and belief in me during this journey.

Dedication

Dedicated to my family, who believed in the finish line long before I could see it.

Table of Contents

List of Tables

List of Figures

Acronyms

API Application programming interface 11, 34, 69, 71, 72, 90, 110

APWG Anti-Phishing Working Group 1

Artificial Intelligence Artificial Intelligence 88

ASN Autonomous System Number 11, 24

CA Certificate Authorities 9

CNN Convolutional Neural Network 89

CT Certificate Transparency 2, 7, 9–11, 14, 20, 61, 88

CV Cross Validation xi, xii, xiv, 36, 39, 48, 50–52, 77–79, 85, 93, 94, 99–101, 125

DDNS Dynamic Domain Name System 28, 31, 45

DNS Domain Name System 2, 7–15, 17–20, 30, 45, 47, 61

DOM Document Object Model 30

ELM Extreme Learning Machines 65

G-Mean Geometric Mean xiii, 36–42, 50, 51, 60, 75–80, 125

HTML HyperText Markup Language 9, 16, 87, 88

HTTP Hypertext Transfer Protocol 33

HTTPS Hypertext Transfer Protocol Secure 7, 30

IP Internet Protocol 8, 9, 11, 24, 30

IQR Interquartile Range 57

LLM Large Language Model iv, ix, xii, 4, 5, 64–66, 84, 86–90, 92–95, 99–104, 107, 109–112

LPA Label Propagation Algorithm 66

MAU Monthly Active Users xiv, 69

MCC Matthews Correlation Coefficient xiii, 36, 38–42, 48, 50–52, 60, 75, 125

ML Machine Learning viii, xi, 22, 23, 36, 43, 47, 49, 59, 60, 65, 80, 92, 107

NCL Neighbourhood Cleaning Rule 81, 82, 110

OCR Optical Character Recognition 88

RFECV Recursive Feature Elimination with Cross-Validation 75

ROS Random Oversampling 81, 82, 110

SHAP Shapley Additive Explanations xiii, 43–47

SLD Second-Level Domain 8

SMOTE Synthetic Minority Over-sampling Technique 52, 81

SMS Short Message Service 65, 86, 87, 89

SSL Secure Sockets Layer 2, 6–11, 14, 17, 18, 20, 45–47, 64

SVM Support Vector Machine 48, 51

t-SNE t-distributed Stochastic Neighbor Embedding xiv, 46, 47

TF–IDF Term Frequency-Inverse Document Frequency 65

TLD Top Level Domain 8, 11, 13

TLS Transport Layer Security 2, 6–11, 14, 17–20, 45, 47

URL Uniform Resource Locator xi, xiii, 9–11, 13–16, 23–26, 28–31, 33, 34, 45, 53, 57, 59, 64–67, 69, 70, 72, 73, 82, 87–90, 92, 94, 101–103, 109, 110

Chapter 1

Introduction

1.1 Phishing Attacks

Phishing remains one of the most persistent and adaptable threats in today's cybersecurity landscape. Recent data from the Anti-Phishing Working Group (APWG) highlights the escalating scale of this threat, reporting that 1,130,292 phishing attacks were detected in the second quarter of 2025, a notable increase from the 1,003,924 attacks identified in the first quarter of the same year [10]. This trend of high volume is also reported by the FBI Internet Crime Complaint Center [48], which identified phishing and spoofing as the most frequent form of cybercrime. With 193,407 complaints recorded in the United States, phishing accounted for nearly 31% of all complaints regarding cybercrimes across major cybercrime categories.

By pretending to be a trusted source and exploiting human trust, attackers aim to deceive people into sharing sensitive information, such as passwords, financial details, or personal data, or into taking actions that benefit the attacker, like transferring money or installing malicious software. The impact of these attacks goes far beyond immediate financial loss; they can lead to identity theft, widespread account takeovers, and a gradual loss of trust in online platforms.

In recent years, phishing has transformed from easy-to-spot attacks into highly sophisticated operations. Attackers now craft convincing messages and build professional-looking websites, making it difficult, even for more cautious users, to distinguish between real and fake content. At the same time, the infrastructure behind phishing has become more diverse. Although previously attacks typically used domains owned and controlled by the attacker, today's campaigns often compromise legitimate websites or take advantage of

free or commercial shared services provided by trusted companies to bypass traditional security measures. Delivery methods have also expanded far beyond email, with phishing now spreading through messaging apps, social media, and other online channels. These campaigns quickly adapt to countermeasures, making early detection and mitigation increasingly challenging.

1.2 Proactive Phishing Detection

A key aspect of phishing defense is proactively detecting malicious activity, ideally before it reaches potential victims. Proactive detection aims to identify the infrastructure or content used in phishing campaigns during the early stages of the attack lifecycle, enabling intervention before the phishing site becomes widely accessible. Many proactive detection systems focus on signals extracted from the infrastructure supporting phishing domains. These include domain registration records [73], the issuance of Secure Sockets Layer (SSL)/Transport Layer Security (TLS) certificates [32, 43, 72, 86], and patterns in Domain Name System (DNS) resolution [8, 31, 61]. When attackers register new domains for a phishing campaign, these infrastructure-level signals can act as early warning signs. By monitoring newly registered domains, detecting suspicious certificate issuances in public Certificate Transparency (CT) logs, and identifying sudden changes in DNS hosting, defenders can flag and act on malicious resources.

Although such methods are effective for detecting phishing sites hosted on infrastructure controlled by attackers, they are less effective when phishing content is deployed on infrastructure outside the attacker's direct administrative control [5, 27, 82, 84]. For example, when attackers compromise a legitimate website and repurpose it to host phishing pages, the domain may have been registered years earlier, the previous SSL/TLS certificate may still be valid, and the DNS configuration may be unchanged. Similarly, the use of trusted third-party hosting platforms and content delivery networks allows attackers to deploy phishing content without triggering the usual red flags that infrastructure-based detection systems look for. In these cases, proactive detection methods that depend solely on infrastructure visibility often fail to provide timely or effective protection.

Therefore, relying solely on features associated with attacker-controlled domains creates a detection gap. Since attackers increasingly exploit this gap by shifting to hosting strategies that leave minimal infrastructure traces, it becomes clear that relying solely on traditional proactive detection methods is insufficient to counter the current phishing threat landscape.

1.3 Motivation and Research Questions

The evolution of phishing strategies and the limitations of current proactive detection raise a significant challenge for the security community. Although infrastructure-level monitoring remains a valuable tool for detecting certain categories of phishing attacks, it fails to cover phishing campaigns hosted on compromised or trusted third-party platforms, resulting in a significant portion of phishing activity remaining undetected in its early stages. This limitation allows such campaigns to remain active longer, weakening the overall effectiveness of such defenses.

Given these challenges, this thesis is guided by two fundamental research questions:

RQ1: Can we gain a comprehensive understanding of the infrastructure supporting phishing attacks, including identifying the range of domain ownership types that phishing campaigns use and determining which ones the current detection methods can or cannot detect?

RQ2: When traditional infrastructure information is not enough, how can we detect phishing as early as possible? What are alternative detection strategies that consider other aspects of phishing activities, such as their propagation behavior on online platforms?

Together, these questions frame the focus of this thesis, guiding the research from analyzing the phishing ecosystem to developing a proactive detection method.

1.4 Research Approach Overview

To address the research questions, this thesis follows a staged research process. The first research question is addressed by examining the types of domain ownership present in the phishing landscape. Given that many proactive detection methods are designed to capture signals associated with domains controlled or owned by attackers, it is essential to determine what proportion of phishing domains fall into this category. To address this, the thesis develops a taxonomy for categorizing domain ownership and applies it to a large-scale phishing dataset collected over a full year from multiple reporting sources. This analysis reveals the distribution of phishing attacks across ownership types and provides insight into who controls the domains used in these campaigns.

The second research question focuses on proactively detecting phishing without limiting the detection method to the phishing infrastructure. It addresses the challenge of detecting phishing in cases where infrastructure-level indicators are absent or insufficient, such as in campaigns hosted on compromised or otherwise legitimate domains. One promising

direction is to study how phishing propagates and to focus on social media as a source of phishing propagation. Social media platforms are not only a channel through which phishing campaigns spread but also a valuable source of data artifacts, such as message content, metadata, and propagation patterns, that can be used to detect malicious activity in its earliest stages. This thesis uses phishing campaigns observed in Telegram to develop and evaluate detection models that leverage these artifacts. By doing so, it explores how phishing can be flagged during its early propagation phase, even in the absence of infrastructure-level indicators.

Figure 1.1 summarizes the research process, illustrating the progression from analyzing the composition of the phishing landscape to developing a proactive detection method that works at the point where potential victims first encounter phishing attempts. The first research question is addressed in Chapters 2 and 3 by reviewing proactive detection methods, introducing a domain ownership taxonomy, and applying it to a large phishing dataset to quantify ownership distributions. The second research question is examined in Chapters 4 and 5 by developing a detection technique based on phishing propagation in social media and comparing it with LLMs for early-stage detection.

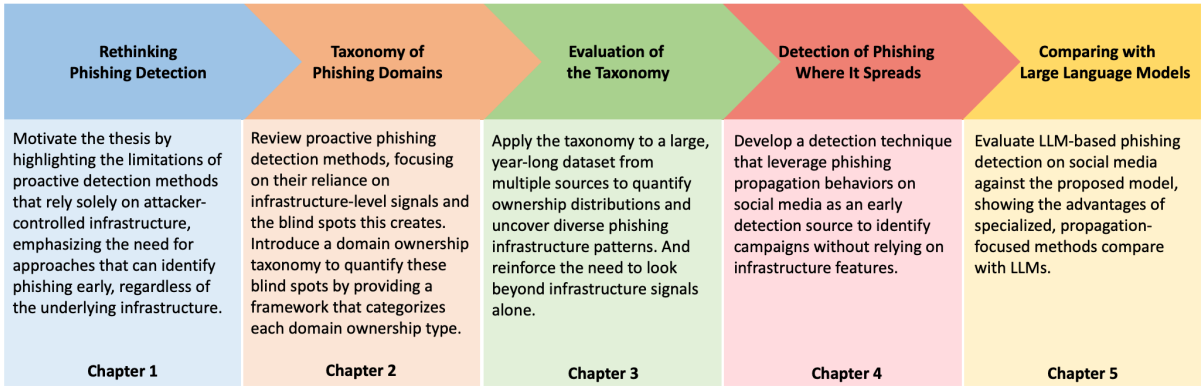


Figure 1.1: Thesis Line Process

1.5 Thesis Structure

The remainder of this thesis is organized as follows. Chapter 2 reviews proactive phishing detection methods, with particular attention to their reliance on infrastructure-level signals and the blind spots this creates. The chapter introduces a taxonomy that categorizes phishing attacks by domain ownership, offering a systematic approach for analyzing the

diverse infrastructures exploited in phishing campaigns. Chapter 3 presents an empirical analysis based on a year-long phishing dataset collected from multiple reporting sources. The chapter applies the proposed taxonomy to the dataset to evaluate its effectiveness and to quantify the distribution of phishing attacks across ownership types. The analysis highlights the proportion of phishing attacks in each domain ownership category and reveals trends in the strategies used by attackers. Chapter 4 proposes and evaluates a proactive detection strategy that uses phishing propagation patterns on social media, with Telegram used as a case study. It analyses phishing campaigns within high-risk group chats and introduces a dataset specifically constructed to capture phishing propagation behaviors. Building on this, the chapter develops and evaluates detection methods at multiple levels of granularity, aiming to identify phishing activities early in their spread, even in the absence of traditional infrastructure-level indicators. Chapter 5 compares the detection methods developed in Chapter 4 with LLMs for phishing detection in social media data, comparing their performance to specialized, domain-specific approaches and assessing their strengths and limitations in this context. Finally, Chapter 6 concludes the thesis by summarizing the key findings, discussing the limitations of the work, and outlining potential directions for future research.

Chapter 2

Taxonomy of Phishing Domains

Phishing campaigns exploit a wide range of infrastructures. They utilize not only domains created for malicious purposes but also legitimate platforms and compromised websites. Although many early-warning detection systems rely on signals related to attacker-controlled infrastructure, such as newly registered domains or new SSL/TLS certificates, these indicators may be absent when phishing content is hosted elsewhere. This creates a critical detection gap in these defences. In response, this chapter introduces a taxonomy that classifies phishing domains based on ownership and level of attacker control. The taxonomy provides a structured foundation to prepare for answering the first research question introduced in Chapter 1. It offers a systematic way to understand the diverse infrastructural strategies used in phishing campaigns and to help identify areas where current proactive detection techniques may fail.

The content of this chapter is based on our paper published in the proceedings of the 2024 APWG Symposium on Electronic Crime Research (eCrime).

Erfan, M., Branco, P., & Jourdan, G. V. (2024). Owned, Pwned or Rented: Whose Domain Is It?. In 2024 APWG Symposium on Electronic Crime Research (eCrime) (pp. 14-26). [34]

2.1 Introduction

The most effective phishing defences are those that detect malicious activities before potential victims engage with phishing content. Many state-of-the-art proactive systems rely on signals that are typically produced only by attacker-owned infrastructure. These signals

include newly registered WHOIS records, recently issued SSL/TLS certificates, and new DNS resolutions (e.g., [8, 14, 32, 36, 43]). Such systems are designed around the assumption that attackers operate domains they have explicitly registered for phishing purposes, making these observable events useful for timely detection. For example, when a phishing operator registers a new domain, that action generates a WHOIS entry; if the domain is configured with Hypertext Transfer Protocol Secure (HTTPS), a CT log entry is produced; and the domain's DNS information becomes resolvable. These infrastructure-level events can be monitored and correlated to identify suspicious activity at an early stage. Detection systems based on this model have proven effective in identifying phishing domains before they are widely circulated.

However, this approach captures only a portion of the phishing ecosystem. Not all phishing campaigns use attacker-owned infrastructure. Phishing actors are increasingly hosting malicious content on compromised legitimate websites, cloud services, and popular third-party platforms such as hosting and file-sharing services. In these cases, attackers leverage pre-existing and often trusted infrastructure, which produces none of the traditional early-warning signals. As a result, detection strategies focused solely on attacker-controlled infrastructure may systematically overlook phishing content that resides elsewhere.

Despite the growing use of such alternative hosting methods, there has been limited effort to quantify or clearly define the various infrastructure types used in phishing attacks. Lack of structured understanding makes it difficult to assess how much current detection strategies are missing or to compare approaches that claim to cover diverse infrastructure types. Without a consistent way to classify phishing infrastructure, detection capabilities cannot be meaningfully evaluated or improved.

To address this gap, this chapter introduces a taxonomy that categorizes phishing domains based on the type of infrastructure used and the degree of control by the attacker. The taxonomy distinguishes among attacker-owned, compromised, and third-party platforms, providing a conceptual framework for reasoning about where phishing attacks occur. This taxonomy provides a foundational understanding of how phishing content is hosted across the Internet and highlights the blind spots that arise when detection strategies are only focused on attacker ownership.

Contributions: Our main contribution here is building a taxonomy of phishing domains based on ownership types. We developed a taxonomy to categorize phishing websites into three distinct groups: Attacker-Owned Domains, Compromised Domains, and Third-Party Platforms. This taxonomy was achieved by designing a set of heuristic features for each category, offering a structured understanding of the diverse infrastructures used in phishing campaigns.

Organization: This chapter is structured as follows. Section 2.2 reviews proactive phishing detection methods that rely on infrastructure-level signals, along with prior research on categorizing phishing domains by ownership and hosting characteristics. Section 2.3 defines the domain ownership categories, introduces a rule-based taxonomy for classifying phishing websites, and presents the features used for this classification. Finally, Section 2.4 summarizes the chapter and outlines its role in guiding future detection efforts.

2.2 Related Work

Previous research on proactive phishing detection has largely focused on identifying malicious activity early in the attack lifecycle by monitoring infrastructure-level signals. These methods are designed to detect phishing domains before they are widely used, often by leveraging observable data artifacts such as domain registration patterns, DNS behavior, or SSL/TLS certificate issuance. In parallel, several efforts have attempted to classify phishing infrastructure based on domain ownership and hosting context. These categorizations offer insight into the infrastructures attackers exploit and can guide detection systems in adapting to different hosting scenarios.

In this section, we first provide a technical background of the key concepts used in these detection methods, we then review infrastructure-based detection techniques that represent the dominant paradigm in proactive phishing defence. Then, we discuss related work on categorizing phishing domains, highlighting the limitations in existing taxonomies and the need for a more comprehensive framework.

2.2.1 Technical Background

To provide a necessary foundation for analyzing phishing infrastructure and the proposed taxonomy, this section outlines the core technical concepts mentioned in this thesis.

DNS and Domain Name Hierarchy: The DNS is the hierarchical and decentralized naming system used to resolve human-readable names into Internet Protocol (IP) addresses. A domain name consists of several levels: the Top Level Domain (TLD) (e.g., .com, .org, .io), the Second-Level Domain (SLD) (e.g., *github*), and subdomains (e.g., *www.github*). A Wildcard DNS record is a specific configuration in a DNS zone file used to match requests for non-existent subdomains. Denoted by an asterisk symbol (*), a wildcard record ensures that any request for a subdomain that has not been explicitly defined (for example, *anything.github.io*) resolves to the same designated IP address. This allows a

domain owner to route traffic for any number of arbitrary subdomains to a single server without needing to create individual records for each one.

CT Logs: Modern web trust relies on SSL/TLS certificates issued by Certificate Authorities (CA)s to encrypt traffic and verify identity. CT is a security standard that requires all such certificates to be recorded in public, append-only ledgers known as CT logs. These logs are critical for proactive detection because they are globally searchable and tamper-proof. For example, if an attacker obtains a certificate for a typosquatted domain like *faceboook.com*, that certificate is published to the CT logs immediately. Proactive monitors can watch these logs to detect the setup of phishing infrastructure before any victims reaches that website.

WHOIS Protocol and Registration Records: The WHOIS protocol is a query and response system used for searching databases that store the registered users or assignees of an Internet resource, such as a domain name or an IP address block. When a domain is registered, specific administrative and technical details are collected by the registrar. A standard WHOIS record contains metadata such as the creation date, the expiration date, and the name of the sponsoring registrar. For instance, a WHOIS query for *uottawa.ca* provides information regarding when the domain was first established and which organization is responsible for its registration. This protocol is a primary method for verifying the ownership history and administrative status of domain names.

HyperText Markup Language (HTML) Structure and Tags: The HTML defines the structure of a webpage through various tags. Beyond visible content, HTML contains behavioral instructions through tags and attributes. For instance, the Anchor tag (`<a>`) utilizes the `href` attribute to specify the URL of the destination page. In the current implementation of the University of Ottawa's website, the visible logo beside the text "uOttawa" acts as an anchor where the tag corresponds to the intended destination URL provided in the `href` attribute (e.g., *https://www.uottawa.ca/en*).

2.2.2 Proactive Detection Using Infrastructure Signals

Proactive phishing detection systems increasingly leverage infrastructure-level signals to identify malicious domains. These approaches are based on the assumption that attackers control or register the domains they use. Therefore, insights derived from certificate issuance (especially through CT logs), DNS resolution behavior, and domain registration metadata have formed the foundation for identifying threats. By capturing these events shortly after a domain is registered or configured, detection systems aim to flag phishing domains before they are widely distributed.

The idea of using SSL/TLS certificates for early phishing detection was originally based on the observation that attackers often rely on readily available certificate authorities, including free services. This motivated the study of CT logs, which offer public records of SSL/TLS certificate issuance. One of the earliest systemic investigations was conducted by Scheitle et al. [91], who studied CT adoption trends and found the logs to be a rich source of timely certificate metadata. Although their main focus was not on phishing detection, they highlighted the viability of using CT data to find phishing domains. Based on this, Fasllija et al. [36] proposed PhishHook, a framework that detects phishing through anomaly analysis in SSL/TLS certificate issuance patterns. Their approach analyzed deviations in certificate field structure and naming conventions to assign phishing likelihood scores. The authors did not evaluate performance on subdomain-level attacks (such as subdomains of legitimate domains), so limitations in those contexts remain open questions. Sakurai et al. [86] introduced an approach that clustered certificates by SSL/TLS Common Name fields and identified phishing. While their method effectively distinguished phishing certificates from benign ones using structural features in CT logs, it did not address the effectiveness of detecting phishing hosted on compromised websites or shared services using legitimate certificates.

Expanding the scope to sector-specific threats, Bijmans et al. [14] examined CT logs for phishing targeting Dutch financial institutions. They demonstrated how attacker behavior patterns—such as mimicking brand names in certificates—can be algorithmically extracted and used for detection. However, their model was tailored to specific sectors and may require retraining for broader applicability. Further generalizing the methodology, Drichel et al. [32] developed a machine learning classifier that used certificate features (e.g., entropy, issuer, and temporal attributes) to distinguish malicious from benign certificates. They acknowledged that their model was not equipped to detect phishing hosted on compromised domains, as these typically reuse legitimate certificates.

To enhance detection without relying on page content, Alrwais et al. [8] introduced a content-agnostic approach combining CT logs and passive DNS data, demonstrating the feasibility of proactive detection without relying on page content. However, they acknowledged their method's limitations in identifying phishing URLs hosted on public infrastructure, such as URL shorteners or widely used hosting services. Brown et al. [43] proposed a system for identifying transaction-based phishing domains by retrieving real-time certificates from CT logs and applying a domain scoring algorithm. While the system achieved high precision, its detection pipeline is fundamentally tailored to newly registered, attacker-controlled domains surfaced through CT monitoring. Notably, the authors did not address the increasingly common scenario in which phishing content is hosted on long-standing, legitimate domains or compromised infrastructure.

Finally, infrastructure visibility is not always enough. Nahapetyan et al. [72] showed that phishing domains often appear in CT logs days before active campaigns begin, suggesting detection potential. Although they did not employ CT data for real-time detection, their findings indicated that monitoring CT logs could enable early identification of phishing domains. However, they also highlighted evasion strategies such as redirection and cloaking URL, which limit the visibility of phishing pages to detection systems.

Beyond certificate analysis, DNS telemetry has been another valuable source. Doğan et al. [31] utilize a limited set of infrastructure signals, such as WHOIS registration data, current DNS lookup results, IP geolocation, and Autonomous System Number (ASN) information, and basic SSL/TLS certificate status—collected via public Application programming interface (API)s. Their dual-layer pipeline is specifically designed to assess newly observed URLs for phishing, malware, or defacement shortly after domain registration and initial resolution. However, the study neither claims nor demonstrates the ability to detect long-established legitimate domains that later become compromised, or malicious content hosted on trustworthy platforms.

A complementary perspective focuses on the economic and procedural factors associated with malicious domain registration. Economic incentives and registrar policies further shape attacker behavior and can be used for some types of detections. Nosyk et al. [73] presented a statistical model trained on features extracted from registrar and TLD metadata. Their findings revealed that domain pricing, availability of automation APIs, and lenient verification strongly correlate with domain abuse. For instance, they found that registrars offering API access had a 401% higher chance of enabling malicious domain registration. The analysis was intentionally scoped to attacker-registered domains, excluding compromised legitimate domains to isolate abuse patterns tied to registration choices.

In summary, while infrastructure-level signals like certificate issuance, DNS behavior, and registration metadata offer valuable early indicators for phishing detection, they are predominantly effective for attacker-owned domains. These approaches often struggle with phishing hosted on compromised sites or public platforms. This limitation highlights the need to better understand and classify the underlying ownership of phishing infrastructure, an aspect that has received limited attention. In the next section, we review efforts toward categorizing phishing domains by ownership type, motivating the development of a more comprehensive taxonomy.

2.2.3 Classification of Phishing Domains

The classification of phishing domains based on ownership types has been explored in various studies, each with its own focus and limitations. The study by APWG in 2016 [9] analyzed phishing domains by dividing them into two categories: maliciously registered domains, which are intentionally created by attackers for phishing, and compromised domains, which are legitimate domains that were hacked and exploited. They reported that 49% of phishing domains were maliciously registered, while 51% were compromised. Maliciously registered domains were identified based on characteristics such as newly registered domains, usage of brand-related keywords, and batch registrations. However, the report does not explicitly address domains hosted on third-party platforms, and it appears that domains not exhibiting characteristics of malicious registration were categorized as compromised. This approach may lead to misclassification of certain domains, limiting its ability to completely capture the diversity of phishing domain infrastructures. A subsequent industry report by PhishLabs [76] showed similar concerns and offered more updated statistics. Unlike earlier research that used registration timing to infer ownership, PhishLabs rejected such heuristics due to the possibility of misclassifying compromised domains. Instead, they adopted a content-based verification process: if the domain hosts legitimate content beyond the phishing path, it is categorized as compromised. Their analysis is performed in real time as part of takedown operations, enabling access to live phishing pages before they are removed. Although this operational process enables accurate domain ownership classification at scale, it relies primarily on page content and does not consider infrastructure-level signals that can support automated and transparent classification. Furthermore, the method's dependence on content limits its robustness against phishing kits that replicate branding across all ownership types, and the proprietary nature of their pipeline limits reproducibility and external validation.

Other works have extended the understanding of compromised domains by focusing on specific tactics used by attackers. Liu et al. [64], for example, studied shadowed domains, a subset of compromised domains where attackers create unauthorized subdomains on legitimate domains. Using passive DNS data, they proposed a system that detects shadowed domains based on deviations in subdomain activity and hosting patterns. While their work offers valuable insights into this particular technique, it was not specifically focused on phishing domains. Instead, it covered a broader range of malicious activities involving shadowed domains, including malware distribution and command-and-control operations. Moreover, the study did not account for other types of phishing domains, such as those that are maliciously registered or hosted on third-party platforms. This limited scope reduces its applicability to the broader landscape of phishing domain detection and classification.

Efforts to classify phishing domains into attacker-owned and compromised categories

have also been explored by Page et al. [56] and Marrofi et al. [69]. Page et al. developed a machine-learning classifier using features like domain historical activity and registration data. While their approach provides insights into attacker-owned and compromised domains, it does not account for third-party platforms, which were likely grouped under the compromised category. Similarly, Marrofi et al. introduced a COMAR, a system focused on distinguishing between maliciously registered and compromised domains, achieving high classification accuracy. However, their work does not clarify how third-party platforms, such as web hosting services or dynamic DNS, were categorized, raising questions about their representation within the framework. Silva et al. [30] introduced a framework that considers hosting contexts, distinguishing between attacker-owned and compromised domains. While their work provides insights into hosting contexts, their analysis relies on VirusTotal data and includes general malicious domains rather than being specific to phishing, which limits its applicability for understanding phishing-specific behaviors.

Bayer et al. [13] advanced COMAR [69] for operational use at country-code TLD registries (.fr and .nl). They first generated an automatic ground truth by correlating registry abuse tickets with suspension events. They filter out "public apex domains" – URL shorteners, dynamic DNS providers, shared hostings – at every stage of the pipeline (label generation, training, and daily scoring). This design means that their reported distribution (approximately 75% maliciously registered vs. 25% compromised) excludes phishing hosted on third-party platforms, leaving this ownership class unmeasured. Zhang et al. [106] added another perspective by examining "dangling domains" on public hosting services -subdomains that attackers can take over once legitimate owners stop using their hosted services. Their hosting checker framework illustrates the prevalence of subdomain takeover, but it does not classify how frequently phishing attacks rely on maliciously registered versus compromised or third-party-hosted infrastructures. More recently, Lim et al. [61] conducted a study of phishing domains and proposed an approach to identify maliciously registered domains using indicators such as DNS behavior, brand targeting, and bulk registration patterns. Their work reinforces the importance of ownership-based classification in phishing detection and builds on techniques such as COMAR [69]. Their classification framework explicitly filters out domains hosted on web platforms, thereby excluding third-party-hosted phishing infrastructure from analysis. However, by excluding third-party-hosted domains, their approach overlooks a significant and growing portion of phishing activity that leverages trusted web platforms, limiting its applicability to categorize the full diversity of phishing infrastructure.

In summary, these studies contribute important insights into phishing domain classification, domain takeovers, and operational modeling. However, they often focus on binary distinctions (e.g., attacker-owned vs. compromised) or do not specifically focus on the rise

of phishing campaigns leveraging trusted third-party environments. This highlights the need for a more comprehensive taxonomy that fully captures the diverse ownership and hosting patterns in phishing attacks, ensuring that future detection methods accurately reflect the evolving phishing landscape.

2.3 Taxonomy of Phishing Domains

Understanding how phishing domains are distributed across different types of infrastructure is critical for evaluating the blind spots of existing detection systems. Our first research question asks whether we can characterize the infrastructures used by phishing campaigns and determine which of them are visible to, or missed by, current proactive methods. A taxonomy of phishing domain ownership types provides the analytical framework required to answer this question. By consistently labeling phishing instances as attacker-owned, compromised, or third-party hosted, the taxonomy enables us to quantify how much of the phishing ecosystem falls within the visibility of infrastructure-based defences – primarily attacker-controlled domains – and how much operates outside it.

For this purpose, we define a taxonomy that categorizes phishing domains based on ownership and the attacker's level of control. The taxonomy distinguishes between three primary categories – attacker-owned, compromised, and third-party platforms – Each reflects a distinct method that attackers use to deliver phishing content. This classification establishes a structured basis for analyzing phishing infrastructures and provides the foundation for the empirical evaluation in the next chapter.

- **Attacker-owned domains** are those registered and controlled directly by the attacker. This ownership grants full administrative access, allowing phishing infrastructure to be built from scratch. Detection systems often target such domains by monitoring signals associated with new domain registrations, anomalous DNS activity, or the issuance of SSL/TLS certificates visible in CT logs.
- **Compromised domains**, by contrast, belong to legitimate organizations or individuals whose websites have been breached. Attackers exploit these sites by injecting phishing content into subdirectories or subdomains. Because these domains often maintain their original reputation and are not newly registered, detection is more difficult, especially for systems that rely on infrastructure signals indicative of new domain ownership.
- **Third-party hosted domains** involve the abuse of legitimate platforms and services that allow user-generated content or custom subdomains. These include web hosting providers, dynamic DNS services, document-sharing platforms, cloud infrastructure, and URL shorteners. Attackers leverage these trusted environments to host phishing content

without the need to register domains or compromise existing ones. This model enables fast deployment and frequent relocation of phishing pages, further complicating detection and takedown efforts.

This taxonomy highlights the diversity of strategies employed by phishing actors and emphasizes the importance of considering infrastructure control when evaluating detection techniques. By distinguishing between different ownership models, we gain a more nuanced understanding of where current detection systems are most effective—and where they are likely to fall short.

2.3.1 Feature Set for Domain Ownership Taxonomy

To categorize phishing domains according to the taxonomy defined above, we rely on a set of infrastructure-level indicators that provide insight into domain ownership and control. These are selected to distinguish between attacker-controlled infrastructure and abuse of compromised and legitimate platforms.

1) *Domain Registration Date* (`new_domain`): This feature refers to the date when the domain hosting the phishing website was initially registered. A recently registered domain may indicate a higher likelihood of being attacker-owned, as malicious actors often register new domains for phishing campaigns. In contrast, older domains might be more associated with legitimate or compromised websites. For this study, a domain is considered new if it was registered less than one year before the phishing URL being reported.
2) *Google Index Status* (`domain_indexed`): This feature checks whether the domain hosting the phishing content shows up in Google search results. Search engine indexing can give clues about a domain's presence and reputation. Domains that aren't indexed might have little online presence or could be deliberately hidden to avoid detection. On the other hand, legitimate websites are usually indexed to enhance visibility and build trust.
3) *Wayback Machine Archive Status* (`is_archived`): This feature evaluates whether the domain or URL has historical snapshots stored in the Wayback Machine. A lack of historical archives could point to a recently created domain, potentially used for malicious activities like phishing. On the other hand, archived content may indicate a legitimate domain.
4) *Known Third-party Platforms* (`known_hosting`): We compiled an expanded list of known hosting platforms, cloud services, URL shorteners, document sharing, and Dynamic DNS services, building on previous research. In our earlier study, 85% of the URLs in the dataset were manually reviewed to identify their hosting providers. Based

on those findings, we identified additional third-party platforms and added them to the previous list, enhancing our ability to detect third-party domains.

5) *Similarity in Domain Archives* (`between_archives_distance`): For archived domains, we analyzed the similarity between the archives of their subdomains or different content paths. Hosting or third-party platforms often exhibit low similarity between their archived subdomains or content paths. In contrast, compromised domains, which are likely owned by individuals or organizations, tend to maintain a more consistent structure between their subdomains and content paths. To measure the similarity between the HTML structures of two web pages, we employed a technique used in a previous work by Lavoie et al. [55] for nearest neighbor clone detection, using Manhattan distance. Inspired by this method, we calculate the frequency distribution of HTML tags in the archived page and represent each page as a vector of these tag frequencies. The two pages are considered structurally similar if the Manhattan distance between the two HTML tag distribution frequency vectors is less than a predefined threshold. We set the threshold to be a fixed percentage of the smaller sum of each vector. After examining various multiplier values, we found that 10% of the sum of the smaller vector provided the most suitable threshold for our case study (the results of this analysis are available in the Appendix B). The formula for this structural similarity metric is presented in Equation 2.1, with a and b representing HTML tag frequency vectors of two pages, and n the length of the vectors.

$$\sum_{i=1}^{n} |a_i - b_i| < 0.1 \cdot \min\left(\sum_{i=1}^{n} |a_i|, \sum_{i=1}^{n} |b_i|\right) \tag{2.1}$$

6) *Similarity between Archives and Phishing Page* (`phish_archives_distance`): We compared the layout of the observed phishing page to the historical archives of the domain. Dissimilarity between the phishing page and the domain's previous layouts may indicate a benign domain rather than an attacker domain. We used the same approach introduced above to examine the similarity of archives and phishing pages.

7) *URL Hosting Location*: This feature is divided into two distinct components, each providing insights into the hosting environment of the phishing website:
   (a) *Subdomain Hosting* (`is_subdomain`): This component evaluates whether the phishing website was hosted on a subdomain of the primary domain (e.g., 'phishing.example.com').
   (b) *Root Path Hosting* (`is_on_root`): This component determines whether the phishing page was hosted on the root path of the main domain (e.g., 'https://example.com/') rather than in a subdirectory or subdomain. Hosting phishing content on the root path is more likely associated with attacker-owned domains, as this scenario is un-

common for compromised servers or third-party platforms. When phishing occurs on the root path, it often points to an attacker-controlled domain rather than a defacement or exploitation of an existing legitimate site.

8) *Wildcard DNS Resolution* (`control_over_dns`): For phishing websites hosted on subdomains, this feature examines whether the subdomain resolves to a wildcard DNS record for the main domain. This feature evaluates the level of control by the subdomain owner over its configuration, such as DNS settings.
9) *Wildcard SSL/TLS Certificate* (`control_over_ssl`): If the phishing website was hosted on a subdomain, we checked if there was an issued SSL/TLS certificate for the subdomain or if a wildcard certificate from the main domain was used. This feature evaluates the level of control by the subdomain owner over its SSL/TLS certificate.

The reason for including the last two features (*Wildcard DNS Resolution* and *Wildcard SSL/TLS Certificate*) was our observation that many websites hosted as subdomains on legitimate third-party platforms have limited control over their DNS and SSL/TLS certificates. These hosting platforms often manage the DNS and SSL/TLS certificates configurations centrally, rather than allowing the individual website owners to have full control. As a result, wildcard DNS and SSL/TLS certificate records can indicate that a website is hosted on a legitimate platform, rather than being directly owned or controlled by an attacker.

2.3.2 Taxonomy Rule Set

To transform the conceptual ownership taxonomy into a functional classification system, we developed a rule-based approach that can be systematically applied to real phishing data based on observable features introduced in Section 2.3.1. This method infers the ownership category of phishing websites – attacker-controlled, compromised, or hosted on a third-party platform – by applying a structured set of heuristics derived from collected features. Rule-based systems offer interpretability, simplicity, and control, making them particularly suitable for operational environments or as an initial labeling layer for downstream learning models. Defining an explicit rule set provides a simple, practical way to translate the theoretical taxonomy into a measurable form. This approach provides a transparent classification that is reproducible and easy to interpret. There are two motivations for using a rule-based method in this context. First, it provides a mechanism for labeling phishing infrastructure at scale. Second, it enables researchers and defenders to understand the logic behind classification decisions and to adapt the system over time as attacker behavior evolves.

The rules are constructed using features such as registration metadata (e.g., domain

age), platform knowledge (e.g., lists of known hosting services), indexing and archival data (e.g., presence in Google Search and the Wayback Machine), and structural similarity metrics that compare the current phishing page with historical content. Each of these features captures behavioral signs that help distinguish between the three ownership categories. For example, domains owned by attackers typically exhibit minimal online history. They are often newly registered, lack archival snapshots, and are not indexed by major search engines. These domains often host phishing content from the root path and are hosted on an infrastructure controlled by the attacker. In contrast, compromised domains tend to be older, indexed, and archived, with visible divergence between historical site layouts and the observed phishing content. Third-party platforms, such as GitHub Pages, Google Docs, or dynamic DNS services, are characterized by shared infrastructure and standardized service patterns. These domains often display structural inconsistency across subdomains, and may rely on wildcard DNS records or SSL certificates.

Rather than relying on any single feature as a binary decision point, our rule-based system evaluates the presence or absence of multiple signals to produce a classification procedure. Each ownership category is associated with a rule profile that combines relevant indicators. A sample might exhibit signals associated with more than one category, but the final classification is assigned based on the cumulative weight. This approach enables transparency in ownership inference.

To translate this logic into an operational system, we implemented a weighted scoring-based mechanism in which each ownership category is defined by a set of heuristic rules. When evaluating a phishing website, each satisfied rule contributes a score to the corresponding category. While most rules are weighted equally, certain indicators – such as the presence of wildcard SSL/TLS certificates or wildcard DNS records – receive higher weights, particularly when the phishing page is hosted on a subdomain, as these signals help distinguish third-party hosting platforms. To ensure fairness across categories with differing rule counts, we normalize the accumulated scores by dividing them by the total number of rules defined for that category. The ownership label is then assigned to the category with the highest normalized score. To validate the suitability of this weighting scheme and the assumptions behind it, we conducted a series of preliminary experiments using a subset of the labeled samples. These exploratory tests allowed us to refine the specific weights assigned to high-impact indicators and ensured that the scoring logic remained robust across all three domain ownership categories. This scoring-based approach offers an interpretable and scalable method for classifying domain ownership, while remaining flexible enough to adapt as attacker behaviors evolve. The complete set of rules is presented in Table 2.1.

The heuristics used in our rule sets are a combination of established indicators and novel

Table 2.1: Categorization rule sets

Category	**Rules**
Attacker Domain	1. A phishing website is placed in this category if: • it is not hosted as a sub-domain, and • it is located in the root directory. 2. Otherwise, a weighted scoring system determines the category. The scoring rules for this category include: • main domain not indexed; • main domain $<$ 1 year old; • main domain not archived, or its archive resembles the phishing page.
Third-Party Platform (without sub-domain)	1. A phishing website is placed in this category if: • it is hosted on a recognized third-party platform. 2. Otherwise, a weighted scoring system determines the category. The scoring rules for this category include: • main domain indexed; • main domain $\geq$ 1 year old; • domain archived; • there is no similarity between the archives; • there is no similarity between the phishing page and the archives.
Third-Party Platform (with sub-domain)	1. A phishing website is placed in this category if: • it is hosted on a recognized third-party platform. 2. Otherwise, a weighted scoring system determines the category. The scoring rules for this category include: • main domain indexed; • main domain $\geq$ 1 year old; • main domain archived; • no similarity between the archives; • no similarity between the phishing page and the archives; • wildcard TLS certificate for the sub-domain; • sub-domain resolves via wildcard DNS from the main domain.
Compromised Domain	A weighted scoring system determines the category. The scoring rules for this category include: • main domain indexed; • main domain $\geq$ 1 year old; • main archived; • similarity between the archives; • no similarity between the phishing page and the archives.

signals developed for this study. Indicators such as *Domain Registration Date* and *Known Third-party Platforms* are well-established in cybersecurity literature for identifying domain reputation. We have adapted more common signals, such as *Google Index Status* and *Wayback Machine Archive Status*, by integrating them into a weighted scoring framework rather than using them as absolute binary filters.

Notably, this taxonomy introduces original contributions through structural and administrative control signals. Specifically, the use of *the HTML tag frequency distribution* (Equation 2.1) to perform structural similarity checks between domain archives and live phishing pages is a novel application of similarity check techniques for domain ownership inference. Additionally, we introduce *Wildcard DNS Resolution* and *Wildcard SSL/TLS Certificate* analysis as original heuristics to distinguish between administrative control at the domain level versus limited control in shared third-party environments.

2.4 Conclusion

In this chapter, we introduced a taxonomy for classifying phishing websites by domain ownership, distinguishing between attacker-controlled domains, compromised legitimate websites, and third-party platforms. This framework captures the infrastructural diversity of phishing campaigns. To motivate the need for such a taxonomy, we first reviewed proactive detection techniques that rely on infrastructure-level signals, such as newly registered domains, DNS resolution behavior, and CT logs. Although these methods are effective in identifying attacker-controlled infrastructure, they often fail to detect when phishing campaigns exploit compromised domains or third-party platforms. To understand and quantify this blind spot, a systematic classification framework is required. Therefore, we surveyed prior research on phishing domain categorization, which revealed the lack of a comprehensive taxonomy to capture ownership diversity. To address this, we proposed a rule-based taxonomy based on interpretable heuristics and a weighted scoring mechanism. This system leverages features such as registration history, hosting behavior, archival visibility, and structural consistency to determine ownership categories.

While this chapter focused on the conceptual and methodological groundwork, the next chapter moves toward empirical validation and answering the first research question defined in Chapter 1. We apply the taxonomy to a real-world phishing dataset to evaluate its effectiveness in capturing ownership diversity. By scaling the inferred models, we uncover broader attacker strategies and demonstrate how domain ownership signals can improve our understanding of phishing infrastructure and support more proactive detection efforts.

Chapter 3

Evaluating Domain Ownership Taxonomy and Uncovering Trends

In the previous chapter, we reviewed proactive phishing detection methods based on infrastructure-level signals. We introduced a taxonomy of phishing domain ownership types to capture the diverse infrastructures used in phishing campaigns. This chapter evaluates the taxonomy's performance and applies it to a year-long phishing dataset to uncover trends in attacker infrastructure over time. Therefore, it addresses the first research question posed in Chapter 1 – whether we can characterize the infrastructures used by phishing campaigns and determine which ones are visible to current proactive detection methods – and provides insight into which domain types are most frequently exploited by phishing campaigns, as well as where current detection techniques are insufficient.

The content of this chapter is based on two papers: a paper published in the 2024 APWG Symposium on Electronic Crime Research (eCrime), and a paper under review in the Computers & Security journal.

Erfan, M., Branco, P., & Jourdan, G. V. (2024). Owned, Pwned or Rented: Whose Domain Is It?. In 2024 APWG Symposium on Electronic Crime Research (eCrime) (pp. 14-26). [34]

Erfan, M., Branco, P., & Jourdan, G. V. (2025). Domains of Deception: Phishing Through the Lens of Ownership. Computers & Security (Revised and under review).

3.1 Introduction

Understanding how phishing domains are distributed across ownership types requires more than conceptual frameworks; it requires empirical validation. In this chapter, we evaluate the domain ownership taxonomy by applying it to a real-world phishing dataset. In this chapter, we have three goals: first, to evaluate the performance of the rule-based classification system developed earlier in Chapter 2; second, to determine whether ML models can offer improved performance using the same set of features; and third, to apply the taxonomy at scale to a large phishing corpus to uncover longitudinal patterns in attacker infrastructure.

To achieve this, we begin by applying the taxonomy's rule set to a labeled phishing dataset, allowing us to measure how well its heuristics align with real-world infrastructure. We then train supervised and unsupervised ML models on the same ownership features and compare their effectiveness against the defined rule-based taxonomy. This evaluation provides insight into the effectiveness of the features and also reveals whether human-defined logic or data-driven inference better captures domain ownership signals.

In addition to evaluating these approaches, we apply the best-performing method to a year-long phishing dataset to analyze how attackers structure their infrastructure at scale. This enables us to measure how frequently different ownership types appear across the phishing landscape and to identify where existing detection methods fail. For instance, if most phishing campaigns rely on infrastructure types that evade existing detection systems (e.g., compromised or third-party hosted domains), then current research and mitigation strategies may be overlooking a significant portion of the threat landscape.

In this chapter, we test how well the rule-based taxonomy works on real phishing data, compare its performance with ML-based approaches, and then use it to explore large-scale trends in attacker infrastructure. The insights derived here inform not only the reliability of ownership classification but also guide the development of more comprehensive detection mechanisms.

Contributions: Our main contributions in this chapter are as follows:

1. Detailed Experimental Evaluation of the Taxonomy: We designed and comparatively evaluated three different categorization methods to assess the performance of our proposed taxonomy: rule-based, supervised ML, and unsupervised ML.
2. Extensive Data Collection and Analysis: We conducted a year-long data collection effort from October 2023 to October 2024, aggregating phishing reports from multiple independent sources. By including multiple feeds, we reduced source-specific biases and captured a broader view of phishing activities. The dataset's scale and year-long duration make it well-suited for longitudinal analysis of phishing infrastructure.

3. Public Dataset Release of two datasets: We released the PhishXtract and PhishXtract-Class datasets to the research community, providing both the raw phishing corpus and a preprocessed version tailored for taxonomy evaluation.
4. Reporting Feed Dynamics: We examined the dynamics of phishing reporting feeds by measuring their overlap and uniqueness, identifying first reporters through pairwise comparisons, and analyzing which third-party platforms were reported most frequently within each feed.
5. Ownership Distribution and Trends: Ownership Distribution and Trend Analysis: We applied the taxonomy to the PhishXtract-Class dataset to quantify attacker-owned, compromised, and third-party-hosted domains. In addition to estimating their overall proportions, we tracked how these ownership types evolved over the year-long period to reveal shifts in attacker strategies and emerging infrastructure trends.

Organization: This chapter is structured as follows. Section 3.2 introduces the collected phishing corpus, analyzes the reporting feed dynamics, and describes all the data preprocessing needed. Section 3.3 presents the experimental evaluation of the taxonomy, comparing rule-based, supervised, and unsupervised ML approaches. Section 3.4 applies the taxonomy at scale to examine ownership distributions and infrastructure trends over the year-long dataset. Finally section 3.5 concludes the chapter with a summary of findings and their implications for phishing detection.

3.2 PhishXtract: A Multi-Feed Phishing Dataset

In this section, we first present the PhishXtract dataset. We describe our methodology for collecting phishing URLs from multiple reporting sources and present a detailed evaluation of the comprehensiveness of the data collected from these sources. By examining the characteristics of the phishing corpus, we assess whether the data is reliable and representative enough to offer a clear picture of the phishing landscape for further analysis. We also describe the preprocessing steps applied to the PhishXtract dataset, which produced the PhishXtract-Class dataset used for the evaluation task presented in Section 3.3.

3.2.1 Phishing Feed Sources and Collection Methodology

PhishXtract is a dataset containing phishing URLs collected from three major feeds: PhishTank [3], OpenPhish [4], and APWG [2]. PhishTank is a community-driven platform where users submit and verify phishing URLs. For PhishTank, we only included

verified entries, which at least two users have confirmed to be phishing. The number of required votes is not fixed but depends on the credibility of the users who vote, as determined by PhishTank's system. OpenPhish is an automated threat intelligence feed that detects phishing and other online threats. APWG compiles phishing intelligence from member companies and global research partners.

Phishing reports were collected over one year, from October 1, 2023, to October 31, 2024. Each report included the phishing URL and a timestamp indicating when the website was first discovered or submitted to the feed. To ensure the relevance of our analysis, we only retained phishing URLs that were active at the time of data collection, allowing for further analysis. Additionally, for each reported URL, we contacted the corresponding servers to gather further details using the Tor network to anonymize our requests. This was done immediately after each report was received to ensure that phishing websites were still accessible and had not yet been changed or taken down. This step included checking the reachability of the server, detecting redirections to other domains, and collecting information on all involved domains in cases where redirections occurred. We also obtained domain-related details, including associated IP addresses, WHOIS records, IP geographical locations, and ASN organization information. Table 3.1 summarizes the number of URLs obtained from each source. This dataset serves as the foundation for our analysis.

Table 3.1: Overview of Phishing URLs Collected from Each Feed (October 2023–October 2024)

Feed	**Unique URLs**	**Overlapping URLs** *(with other feeds)*
PhishTank	186,430	47,282
OpenPhish	208,679	41,574
APWG	283,430	32,604
Total	**678,539**	**121,460**

Note: Overlapping URLs refer to phishing websites reported by more than one feed within a seven-day window. To ensure consistency, URLs were normalized by removing query parameters, fragments, and trailing slashes, treating logically equivalent URLs as identical.

3.2.2 Analysis of Collected Phishing Data

Coverage and Temporal Overlap Across Phishing Feeds: With the PhishXtract dataset collected, one objective was to evaluate the coverage and overlap among the

phishing reporting feeds. Understanding the frequency with which these sources report the same phishing URLs provides insight into their comprehensiveness, uniqueness, and potential redundant information. To achieve this, we examined the overlap of reported URLs across the feeds. To ensure consistency, we used normalized URLs, where variations such as query parameters, fragments, URL-encoded characters, and trailing slashes (/) were standardized to treat logically equivalent URLs as identical. For example, the URLs `https://112x.cn/#%2F/` and `https://112x.cn/` were treated as equivalent (`https://112x.cn`) because they lead to the same destination website. This normalization process allowed us to identify overlaps more accurately, regardless of minor URL variations. Furthermore, we considered a time frame of seven days to account for potential reporting delays across feeds, ensuring a more reliable comparison of URL overlaps.

The results of this overlap analysis are illustrated in Figure 3.1, which shows the number of phishing URLs reported by one, two, or all three feeds. We observe that the majority of URLs reported by each feed were unique, indicating minimal overlap. This suggests that the feeds rely on distinct sources for detecting phishing activity or that different entities report URLs to specific feeds. This reinforces the importance of aggregating multiple feeds, which allows capturing a broader and more representative view of phishing attacks.

To further explore the overlaps identified in the previous step, we conducted pairwise comparisons of the feeds. For each pair of overlapping feeds, we analyzed which feed reported a phishing URL earlier. Figure 3.2 illustrates these comparisons for the following feed pairs: APWG-PhishTank (Figure 3.2(a)), APWG-OpenPhish (Figure 3.2(b)), and OpenPhish-PhishTank (Figure 3.2(c)). In each plot, the blue and orange bars represent the number of overlapping URLs reported earlier by each feed. A third category, shown in green, corresponds to overlapping URLs that were reported by both feeds at the same time (since they were relatively rare, the green bars are not visible in the stacked plots). From these comparisons, we observed that while some feeds usually reported overlapping phishing URLs earlier, the reporting times varied significantly depending on the feed pair. For example, APWG reported earlier than PhishTank and OpenPhish for most overlapped URLs in most months, while OpenPhish had a slight lead over APWG for certain periods.

Additionally, the tables embedded in the figure present the average time delay between the reporting times of overlapped URLs for each pair of feeds. Specifically, the table shows the average amount of time it took for the second feed to report a phishing URL after the first feed had already reported it. For instance, in the APWG-PhishTank pair (Figure 3.2(a)), the second column of the table indicates that when APWG reported a URL first, PhishTank typically followed with an average delay of 1 to 20 hours per month. On the other hand, when PhishTank was the initial reporter, APWG took a monthly average of 12 to 40 hours to report the same URL, as shown in the third column of the

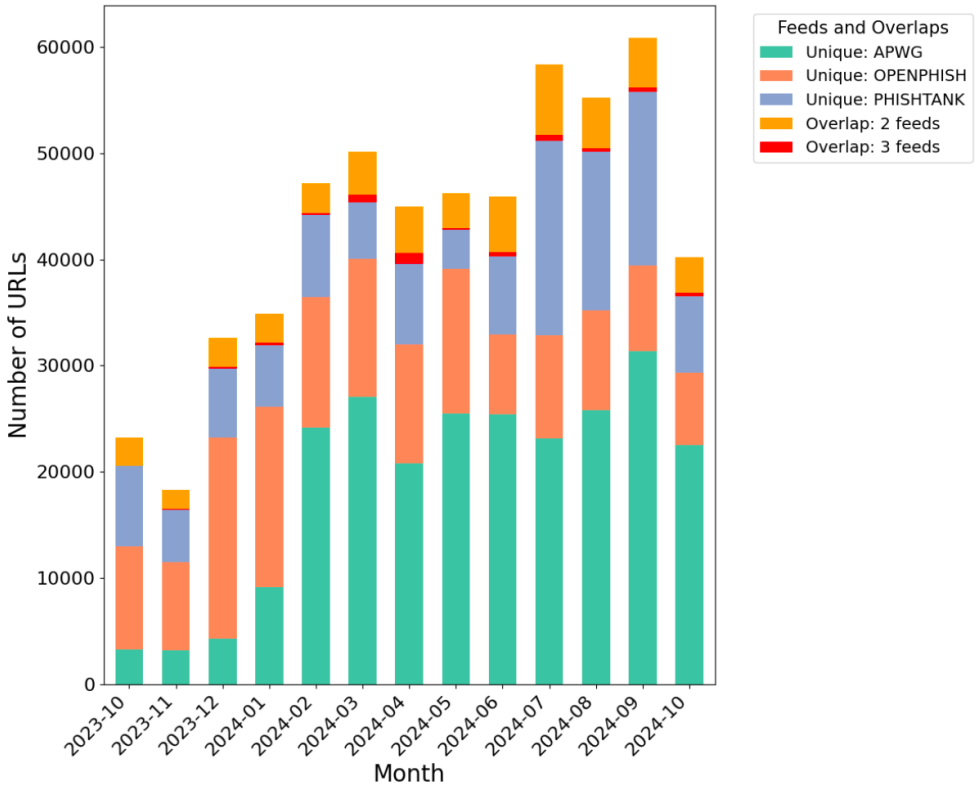
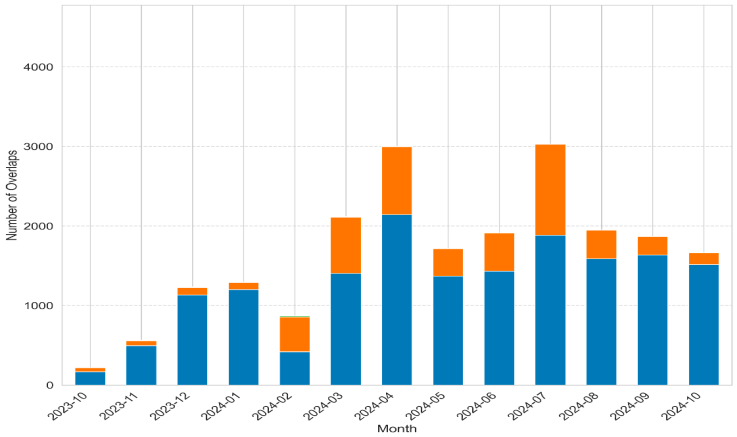


Figure 3.1: Stacked Bar Chart, Illustrating Unique and Overlapping Active Phishing URLs Among Feeds.

table. Similarly, for the APWG-OpenPhish pair (Figure 3.2(b)), APWG led by a monthly average between 6 to 14 hours when it reported first, while it had an average delay of 9 to 39 hours when OpenPhish was the initial reporter. In the OpenPhish-PhishTank pair (Figure 3.2(c)), PhishTank reported first with OpenPhish following after an average delay of 4 to 10 hours, and when OpenPhish reported first, OpenPhish had an average delay of 14 to 34 hours.

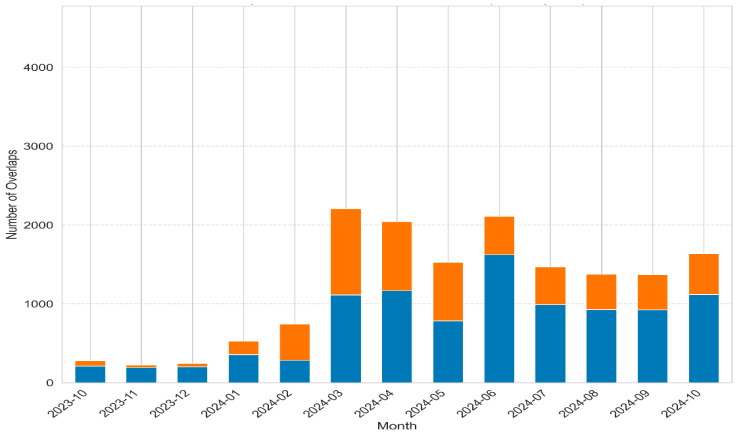
These average delays illustrate the relative reporting speed of each feed compared to the others. Specifically, OpenPhish demonstrates shorter delays when other feeds report first, indicating its capability to quickly follow up on phishing URLs identified by others. On the other hand, when OpenPhish or PhishTank reports first, APWG exhibits longer delays in reporting the same URLs. However, the pairwise plots reveal that APWG is frequently the first to report overlapping URLs across most periods. This dual behavior suggests that APWG may prioritize certain phishing URLs or possess a faster detection



Monthly Average Time Delays (hours)

Month	APWG first	PHISHTANK first
2023-10	16.37	23.45
2023-11	2.20	13.24
2023-12	1.62	27.98
2024-01	9.90	20.87
2024-02	6.04	39.17
2024-03	19.13	21.04
2024-04	10.22	14.12
2024-05	11.79	13.50
2024-06	13.71	12.50
2024-07	7.63	14.90
2024-08	5.32	34.51
2024-09	2.78	32.32
2024-10	3.69	26.50

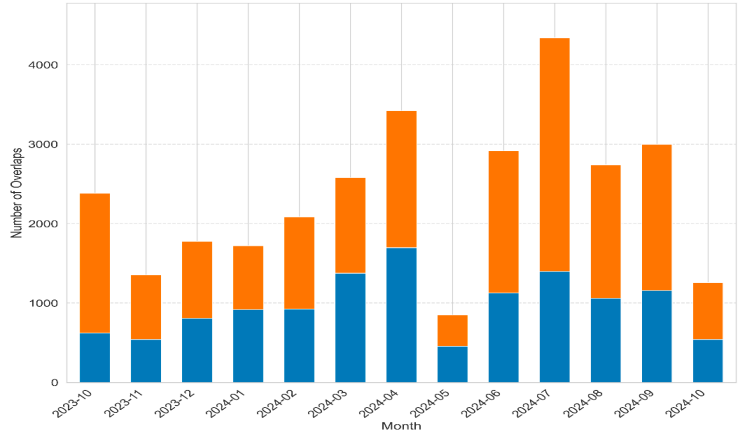
(a) Overlap and First-Reporter Analysis: APWG vs. PhishTank



Monthly Average Time Delays (hours)

Month	APWG first	OPENPHISH first
2023-10	10.08	38.09
2023-11	6.01	24.20
2023-12	6.47	35.89
2024-01	11.65	25.44
2024-02	7.92	44.37
2024-03	11.40	21.15
2024-04	11.86	11.24
2024-05	13.45	9.46
2024-06	13.39	15.06
2024-07	12.28	24.41
2024-08	13.03	23.38
2024-09	11.99	29.48
2024-10	12.71	24.53

(b) Overlap and First-Reporter Analysis: APWG vs. OpenPhish



Monthly Average Time Delays (hours)

Month	OPENPHISH first	PHISHTANK first
2023-10	16.32	5.00
2023-11	18.41	6.17
2023-12	14.08	4.98
2024-01	33.53	5.54
2024-02	16.62	4.38
2024-03	37.98	6.63
2024-04	26.04	4.80
2024-05	29.45	5.38
2024-06	26.64	5.95
2024-07	20.84	5.90
2024-08	19.19	5.65
2024-09	17.91	9.18
2024-10	30.51	5.60

(c) Overlap and First-Reporter Analysis: OpenPhish vs. PhishTank

Figure 3.2: Pairwise Overlap Comparison Among the Feeds

mechanism for identifying specific types of phishing threats. This means that while APWG leads in the early detection of many phishing URLs, its response time increases when it follows other feeds, highlighting its strategic focus on particular phishing threats. This temporal dynamic indicates the complementary strengths of each feed and highlights the potential benefits of integrating multiple sources to enhance the overall effectiveness of the phishing domain classification system we explore in this study.

Identifying Frequently Abused Third-Party Platforms: In addition to overlaps, we investigated the infrastructures and domains most frequently used for phishing within the reported URLs. This analysis was performed both at the individual feed level and across the combined reports. The results, visualized in Figure 3.3, highlight the most commonly used domains for hosting phishing content. As shown in the figure, third-party platforms dominate the list of domains used for phishing. These include popular services such as Dynamic Domain Name System (DDNS) services, Cloudflare workers, Google's document sharing services, and GitHub pages, which provide attackers with readily available and trusted infrastructures to host their phishing websites or share links to them. The consistent use of these platforms across feeds reveals the attackers' preference to use legitimate services to increase the credibility of their campaigns.

Overall, our analysis reveals key insights into the behavior and effectiveness of the phishing reports collected:

1. Minimal Overlap: The majority of URLs reported by each feed were unique, demonstrating that each feed captures distinct segments of the phishing landscape. This lack of significant overlap highlights the complementary nature of the data and validates that it is comprehensive in representing diverse phishing threats.
2. Feed Reporting Variability: Pairwise comparisons revealed that no single feed consistently leads in reporting times, indicating that each feed detects phishing URLs with different priorities or focuses. This variation shows that the feeds complement each other, ensuring that the phishing reports collected offer a broader and comprehensive view of the phishing landscape.
3. Third-Party Platforms: The widespread use of third-party platforms for phishing campaigns, as revealed in our data analysis, emphasizes the attackers' reliance on trusted and readily available infrastructures to host their malicious content. This finding highlights the importance of analyzing phishing domain ownership types.

These findings provide a deeper understanding of the dynamics of the phishing feeds and emphasize the importance of feed aggregation for monitoring phishing activities. Moreover, the significant reliance on third-party platforms for hosting phishing campaigns underscores the necessity to categorize and analyze phishing domains based on ownership types. In the next subsection, we describe the preprocessing steps applied to the PhishXtract dataset to

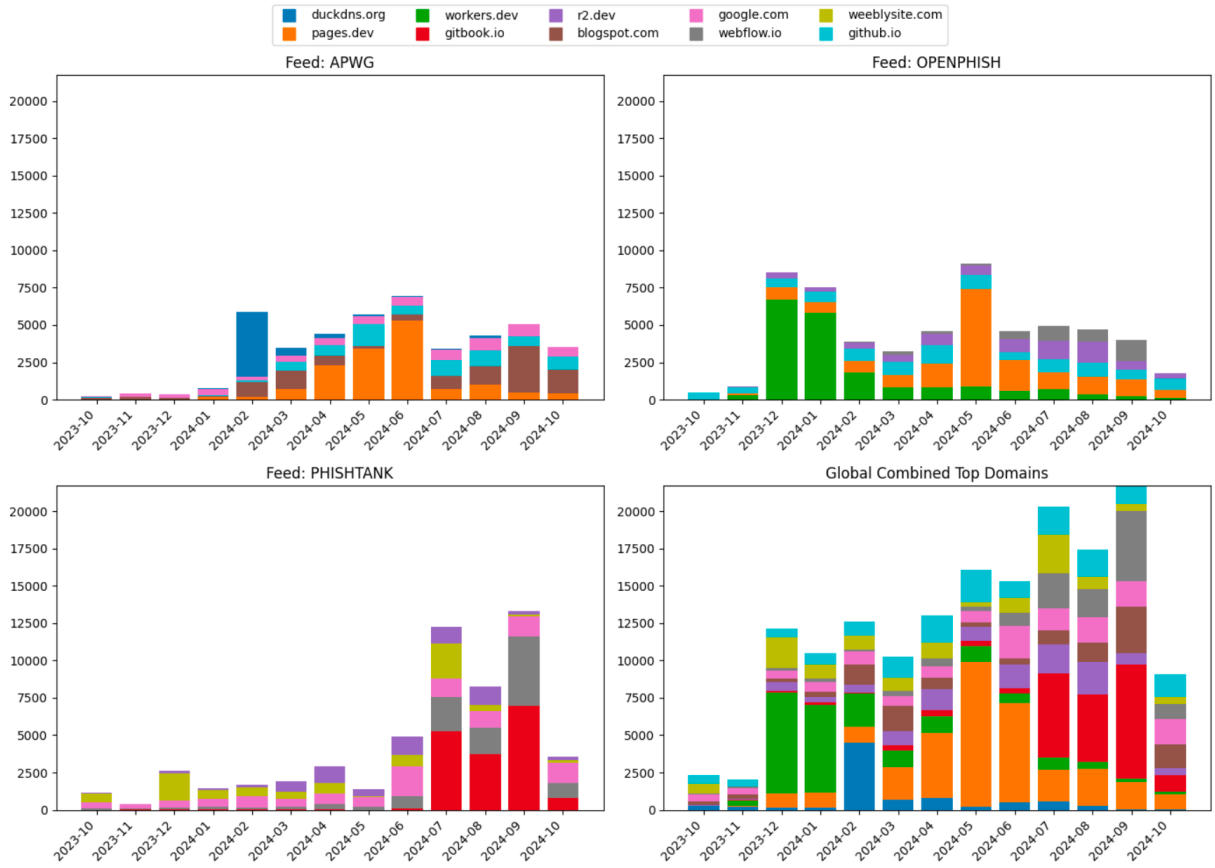


Figure 3.3: Commonly Used Domains in the Phishing Attacks Reported by the Feeds

enable our subsequent categorization analysis.

3.2.3 Preprocessing PhishXtract for Taxonomy Evaluation

To evaluate the taxonomy of phishing websites based on the domain ownership types, we derived a new dataset, PhishXtract-Class, from the original PhishXtract collection. This involved a series of preprocessing steps to improve data quality and extract relevant features for analysis. We started by extracting key characteristics associated with each phishing URL to support the development of a domain ownership taxonomy. This was followed by data cleaning steps, including handling missing values and removing duplicates. Additionally, to enable a supervised classification taxonomy, we relied on a ground truth labeling process, building on manually verified phishing URLs categorized based on domain

ownership. The following subsections detail each applied preprocessing step, which forms the foundation for the taxonomy analysis of the phishing websites presented in Section 3.3.

Feature Extraction: To align with our research goal of categorizing the domain ownership of HTTPS-based phishing websites, additional information was collected to analyze the structural, hosting, and historical attributes of these sites. The Document Object Model (DOM) of each phishing website was retrieved to examine its structural characteristics. WHOIS data and IP addresses were collected to understand domain registration and hosting. This preprocessing also included analyzing DNS resolution details, retrieving archived snapshots of domains from the Wayback Machine [1], and checking the index status of domains in Google's search engine. These processed data sources were used to extract a set of features, previously introduced in Chapter 2, Section 2.3.1, where we detailed each feature along with the rationale for its inclusion and its potential association with a specific domain ownership type.

Handling Missing Values and Duplicates To ensure the reliability of the dataset, samples with missing values were removed from this study. Missing values occurred primarily in WHOIS information, DNS resolution, Wayback Machine archive status, and Google index status. These data points were obtained from external services out of our control, and unreachability or unavailability sometimes led to incomplete records. Since these features were essential for the subsequent analysis and classification, we excluded samples with missing values to maintain the integrity of the dataset. This preprocessing step ensured that all retained data points contained complete information across the extracted features, supporting an accurate analysis in the following sections.

To ensure the dataset was clean and representative, we also addressed duplicate URLs within each feed by applying a seven-day window. During this process, URLs were normalized to treat those with the same path but different query parameters, URL fragments, trailing slashes (/), default ports, case sensitivity in domain names, and URL-encoded characters as identical. This normalization allowed us to remove redundancies by keeping only the first reported instance of each URL. This step ensured that duplicate entries did not skew the analysis, providing a clearer view of unique phishing activities.

Labeling and Ground Truth: PhishXtract-Class includes a manually labeled subset of phishing URLs, referred to as PhishXtract-Class-Labeled. This subset categorizes URLs into three domain ownership types: *attacker-owned*, *compromised*, and *third-party platforms*, as defined in Chapter 2, Section 2.3. The PhishXtract-Class-Labeled includes phishing URLs reported between April 20 and April 24, 2024, which we labeled based on a systematic manual verification process. The URLs in this subset were manually verified through a detailed analysis of multiple data sources and indicators, as outlined in Chapter 2, Section 2.3.1.

The development of the taxonomy rule set and the subsequent labeling of the ground truth subset represented a significant manual research effort. The taxonomy's rules were derived through an investigation of 512 preliminary phishing samples to capture recurring infrastructure patterns and ownership indicators. Following the establishment of these rules, the labeling of the 5,493 URLs in PhishXtract-Class-Labeled required manual expert analysis. Each entry was manually verified through a multi-step process, involving the joint analysis of domain registration records, historical snapshots, structural features of phishing pages, search engine indexing status, and hosting context. We distinguished between the three domain ownership types using the following criteria:

- Attacker-Owned Domains: Indicators included recently registered domains with no prior legitimate content, absence from Google search engine index, lack of historical usage (based on archive snapshots), and in some cases, registrant information linked to malicious campaigns. Hosting on the root path of the domain was also considered as one potential indicator, as this scenario is less common for the other two domain ownership types.
- Compromised Domains: Evidence included the existence of unrelated legitimate content on other parts of the site (e.g., homepages), archived benign snapshots before the phishing event, older registration dates, and inconsistencies in structure or branding between the phishing path and the rest of the site, but consistencies between other parts of the site.
- Third-Party Platforms: Domains associated with known infrastructure providers (e.g., cloud services, file-sharing platforms, or DDNS) were matched against a verified list of third-party platforms. For domains not in this list, we performed manual verification by inspecting the domain's content, reviewing archived snapshots, and searching for platform-specific documentation or offered services. This included checking whether the domain offered services such as document sharing, hosting, form creation, URL shortening, or any other multi-tenant functionalities that allow users to publish or store content under shared infrastructure. This approach helped us identify and accurately label less-known or newly observed third-party platforms.

To ensure consistent application of these labeling criteria, we followed a multi-step process detailed below:

1. Cross-referencing Data Sources: Each URL was evaluated by jointly analyzing features collected to determine the domain's ownership type. The assessment did not rely on any single indicator but considered the combined evidence.
2. Assessing Consistency Over Time: We compared the structure of phishing pages with archived content from the same domain, subdomain, or path to detect deviations or consistency.

3. Evaluating Domain Characteristics: Registration date, registrant identity, and visibility in public search engines were used to assess the legitimacy and control context of the domain.

We must highlight that, although phishing pages can appear in a variety of languages, we did not use the language of the webpage as a factor in determining domain ownership type, particularly for identifying third-party platforms. Our classification relied on infrastructure-based signals such as domain age, historical archives, and search engine indexing. We also investigated the services offered by the domain to assess whether it functions as a hosting provider, document-sharing service, form builder, or similar platform. Recognizable branding elements often helped confirm third-party domains. To further mitigate the risk of misclassification, especially when dealing with regional or non-English platforms, we used browser translation tools during manual review to understand the purpose of the domain. Together, these infrastructure and service-based indicators, independent of webpage language, offered a reliable basis for classifying domain ownership.

Semi-Automated Labeling of Compromised Domains: A critical challenge encountered during the manual labeling process was the low prevalence of the *compromised* domain class, which created a severe class imbalance relative to third-party and attacker-owned domains. To address this and ensure that the compromised category was better represented for robust evaluation, we adopted a semi-automated labeling mechanism inspired by [13]. This approach was designed to identify legitimate domains that had been exploited for phishing by leveraging a longitudinal analysis. By comparing WHOIS data collected at the time of the attack with follow-up WHOIS queries and browser-based checks conducted 15 months later, we were able to isolate domains that remained active and stable under their original registrants while the specific phishing paths returned 404 errors. This high-confidence indicator of content removal on a stable domain allowed us to expand the compromised class beyond the limitations of manual inspection, providing a more balanced dataset for our experiments.

1. Initial WHOIS Snapshot: During the April 2024 data collection period, we obtained WHOIS data for each domain, including registration dates and status codes.
2. Follow-Up WHOIS Retrieval: 15 months later, we re-acquired WHOIS data for these domains to examine any status changes, particularly signs of suspension.
3. Hold Status Filtering: Domains that had `clientHold` or `serverHold` status were assumed to be attacker-controlled, as such statuses typically indicate suspension by registrars in response to abuse. These were excluded from the set of potential compromised domains
4. Registration Date Consistency: For domains not in a hold state, we checked whether the registration date had remained unchanged between the two WHOIS snapshots.

Domains with consistent registration dates were considered stable and likely still under the control of their original (benign) registrant.

5. Hypertext Transfer Protocol (HTTP) Status Check: We browsed the original phishing URLs using an automated browser. URLs returning an HTTP 404 (Not Found) status were considered taken down, suggesting that only the malicious content was removed while the domain remained active, consistent with compromise scenarios.
6. Manual Verification: All flagged domains were manually inspected to confirm that they were not part of the hosting infrastructure or third-party platforms. This final step ensured that misclassification was avoided.

By checking domain status over time and verifying the consistency of the registration data, this mechanism helped identify compromised domains without relying solely on manual inspection. It enabled us to marginally expand the set of compromised domains in PhishXtract-Class-Labeled, while preserving label quality and minimizing false classification.

Final Labeled Dataset and Label Distribution: URLs for which the available data was insufficient, conflicting, or inconclusive were excluded. This process resulted in a reliable, ownership-aware ground truth suitable for evaluating phishing taxonomy approaches. A summary of the label distribution in PhishXtract-Class-Labeled is presented in Table 3.2. These class counts form the basis of the taxonomy evaluation described in Section 3.3.

Table 3.2: Label Distribution in PhishXtract-Class-Labeled

Ownership Category	**Number of URLs**
Attacker-Owned Domains	1,376
Third-Party Platforms	3,954
Compromised Domains	163
Total	**5,493**

Dataset Structure Overview. The overall structure of the proposed dataset is illustrated in Figure 3.4. As shown in the top panel, the main collection, *PhishXtract*, aggregates phishing reports from multiple threat intelligence feeds, including PhishTank, OpenPhish, and APWG. Following preprocessing, deduplication (both between and within feeds), and feature extraction, the refined dataset—referred to as *PhishXtract-Class* – is obtained, as illustrated in the middle panel of the figure. Finally, *PhishXtract-Class* is divided into labeled and unlabeled subsets used for taxonomy evaluation, as shown in the bottom panel.

- *PhishXtract-Class-Labeled:* A manually verified subset of phishing URLs categorized according to ownership type – *attacker-owned*, *compromised*, or *third-party platform*. This labeled subset serves as the ground truth for evaluating classification methods. The *PhishXtract-Class-Labeled* data correspond to phishing URLs reported between April 20 and April 24, 2024.
- *PhishXtract-Class-Unlabeled:* The remaining phishing URLs whose ownership types were inferred automatically using the most reliable classification method identified during taxonomy evaluation.

3.2.4 Limitations of the Dataset

The created dataset has certain limitations. First, the data was collected from three primary sources: APWG, PhishTank, and OpenPhish. Although these are major feeds in the phishing domain, incorporating additional data sources could further enhance the comprehensiveness of the study.

Second, we could only label a subset of data. Due to the short-lived nature of phishing websites and the limited annotation capacity, this subset was limited to a 5-day window (April 20–24, 2024) selected to ensure timely access to live phishing content. However, any labeling errors in the training data may propagate through the model to the unlabeled examples. Because the true ownership labels for these automatically classified domains are unknown, we cannot directly quantify the extent of such error propagation. Furthermore, labeling a larger sample was infeasible given resource constraints, and expanding the labeled set would have introduced a higher risk of human error. These labeling errors, when used to train machine learning models, can propagate to the automatically labeled portion of the dataset, especially in the absence of ground truth for validation. Although this 5-day labeled subset was chosen to balance practical constraints with timely data access, its ability to represent the broader year-long dataset is an important consideration. To address this, we conducted a dedicated analysis to assess the representativeness of the labeled period relative to the full dataset, which is discussed in Section 3.4.1.

Third, some feature extraction depended on APIs and external services beyond our control, which occasionally resulted in missing values. To ensure the integrity of our analysis, we excluded samples with incomplete data, but this approach may have introduced a selection bias. Finally, the relatively small size of the labeled dataset poses a challenge for training a more robust model, especially for underrepresented ownership categories.

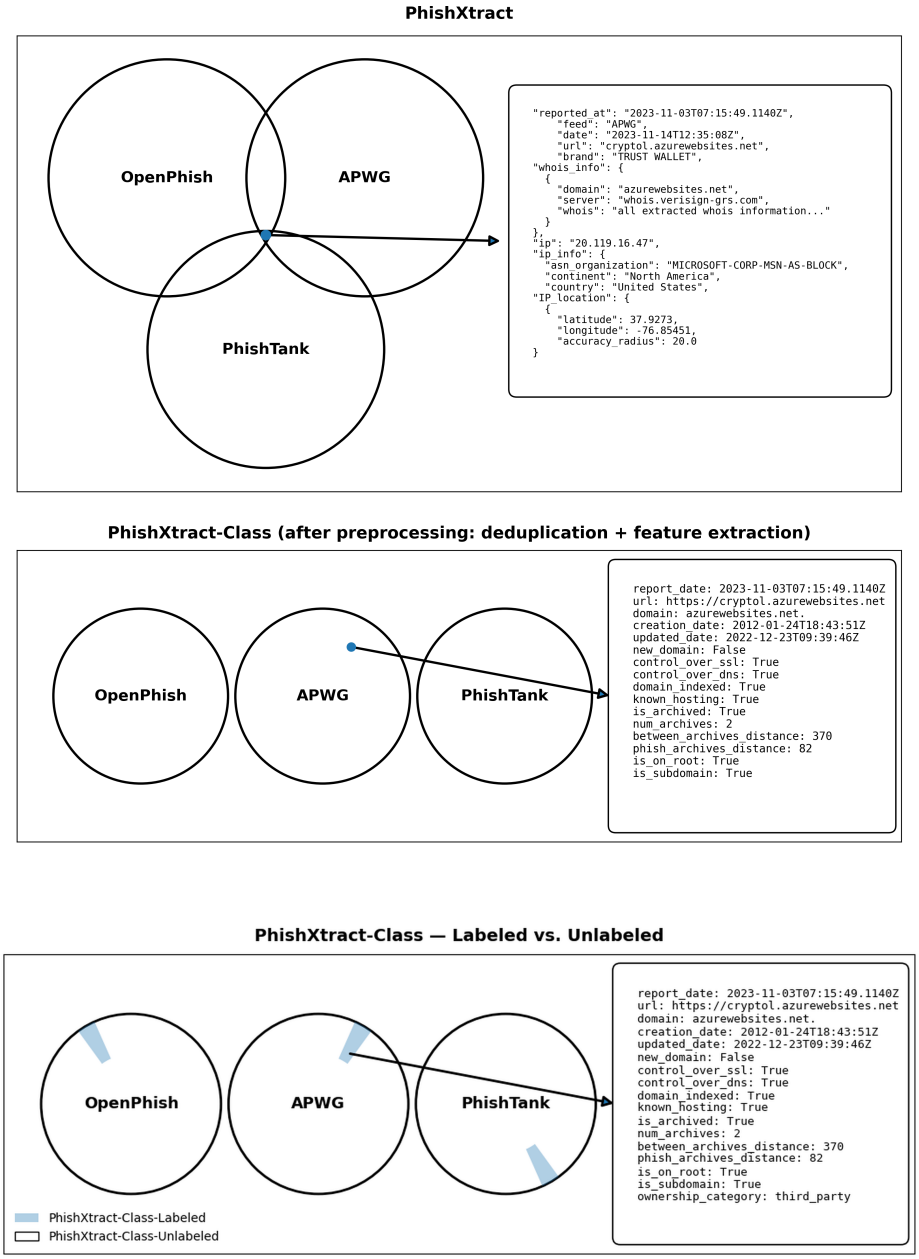


Figure 3.4: Overview of the dataset construction and decomposition.

3.3 Evaluation of the Taxonomy

This section evaluates how effectively the proposed taxonomy distinguishes phishing domain ownership types. Using the manually verified *PhishXtract-Class-Labeled* dataset as ground truth, we first measure the performance of the rule-based classification introduced in Chapter 2.

For comparison, we also evaluate ML alternatives using the same ownership features (with the exception that the archive-similarity measures – `between_archives_distance` and `phish_archives_distance` – are used as normalized continuous distances rather than thresholded indicators so that the models can leverage their full variation). Supervised models are trained on the labeled data to learn classification boundaries, while an unsupervised clustering method is applied to explore whether meaningful ownership groups can be discovered without using any labels. Together, these approaches provide different perspectives: one based on human-defined rules, the others on data-driven inference. Comparing rule-based and learned strategies highlights which method best separates attacker-owned, compromised, and third-party domains. The strongest method is then applied to the full *PhishXtract-Class* dataset. This large-scale annotation enables a comprehensive analysis of ownership distributions across the phishing landscape, providing insights into the attackers' infrastructure.

3.3.1 Preliminary Experimental Setup

Evaluation Method: All three approaches – rule-based, supervised, and unsupervised – are evaluated against the *PhishXtract-Class-Labeled* dataset, each using an evaluation strategy suited to its nature. For the supervised models, we perform 5-fold CV to assess generalization across different data splits. The unsupervised approach is evaluated by comparing its cluster assignments against the ground truth labels. The rule-based system, being deterministic, is directly evaluated through its agreement with the labeled dataset. Each method's performance is measured using standard classification metrics to ensure a fair comparison of their effectiveness. After identifying the top-performing method, we extend it to the remaining unlabeled portion of the dataset, producing a complete ownership label set suitable for subsequent large-scale analysis of attacker infrastructure.

Evaluation Metrics: To assess the performance of each approach, we used standard evaluation metrics: Precision, Recall, Specificity, F1-score, G-Mean, and MCC. These metrics provide insights into the accuracy and reliability of our classification approach, particularly in distinguishing between attacker-owned, compromised, and third-party-hosted phishing domains.

Precision (Equation 3.1) measures the proportion of correctly classified phishing domains of a given category out of all instances predicted as belonging to that category. It quantifies how many of the classifications were actually correct, reducing the likelihood of false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3.1}$$

A high precision score indicates that the model minimizes misclassifications when identifying phishing domain ownership types.

Recall or Sensitivity (Equation 3.2) measures the proportion of correctly classified phishing domains in a given category out of all actual instances of that category in the dataset. It reflects the model's ability to detect all relevant cases, reducing false negatives.

$$\text{Recall} = \text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3.2}$$

A high Recall score indicates that the model effectively identifies most phishing domains within a specific category.

The F1-score (Equation 3.3) is the harmonic mean of Precision and Recall. In the F1-score, the same weight is given to precision and recall, indicating that both are relevant.

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

A higher F1-score signifies a well-balanced model that achieves both high Precision and Recall, making it an essential metric for evaluating classification performance.

Specificity (Equation 3.4) measures the proportion of true negatives among all actual negative instances. It is useful to assess how well the model avoids false positives for each class.

$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3.4}$$

G-Mean (Equation 3.5) is a metric that reflects the model's ability to balance classification performance across all classes. In the binary case, G-Mean combines sensitivity (recall) and specificity to penalize classifiers that perform poorly on either the positive or negative class:

$$\text{G-Mean} = \sqrt{\text{Sensitivity} \times \text{Specificity}} \tag{3.5}$$

Equation 3.5 presents the binary form to illustrate the intuition behind the metric. However, since our taxonomy involves three ownership categories (attacker-owned, compromised, and third-party platforms), we use the multi-class generalization of G-Mean for

evaluation. In multi-class settings, G-Mean is generalized by computing the geometric mean of the per-class recall values (Equation 3.6). This is especially valuable under class imbalance, as it emphasizes the need for consistent performance across all classes rather than being biased toward the majority ones:

$$\text{G-Mean (multi-class)} = \sqrt[n]{\prod_{i=1}^{n} \text{Recall}_i} \tag{3.6}$$

MCC (Equation 3.7) is a correlation-based metric that consider all four elements of the confusion matrix (TP, TN, FP, FN). It is particularly suitable for imbalanced datasets, as it is informative even when class distributions are skewed. MCC ranges from –1 (total disagreement) to +1 (perfect prediction), with 0 representing random guessing.

$$\text{MCC} = \frac{(\text{TP} \cdot \text{TN}) - (\text{FP} \cdot \text{FN})}{\sqrt{(\text{TP} + \text{FP})(\text{TP} + \text{FN})(\text{TN} + \text{FP})(\text{TN} + \text{FN})}} \tag{3.7}$$

3.3.2 Preliminary Experimental Results and Discussion

Rule-Based Taxonomy: The results (provided in Table 3.3) show that this approach performs very well when identifying attacker-owned domains and third-party platforms. Both classes show F1-scores close to 1 (0.961 and 0.977), with high precision and recall. The strong balance between true-positive and true-negative rates is reflected in the high G-Mean values (0.966 and 0.940) and MCC scores (0.949 and 0.916). This indicates that the heuristic rules were successful in capturing the distinguishing infrastructure patterns of these two categories. The challenge appeared with compromised domains. Recall drops to 0.300, pulling the F1-score down to 0.445 despite good precision (0.860). The lower G-Mean (0.548) and MCC (0.501) confirm that the model misses many true compromised cases. The lower performance suggests that this category was more difficult to define and identify precisely using the rule-based approach. Websites hosted on compromised domains may exhibit more ambiguous or overlapping characteristics compared to the other two categories.

Supervised Learning: The supervised learning approach is designed to train a predictive model that predicts the ownership category of a phishing domain based on labeled data. This process involves representing each phishing sample as a feature vector and using this representation to train a classifier on the known labels. The trained classifier can assign ownership categories to previously unseen phishing websites. Unlike rule-based logic, which relies on rule matches, supervised learning models can learn nonlinear relationships. We

Table 3.3: Performance metrics for the rule-based taxonomy

Class	Precision	Recall	Specificity	F1-Score	G-Mean	MCC
Attacker Domain	0.985	0.938	0.995	0.961	0.966	0.949
Third-Party Platform	0.957	0.998	0.884	0.977	0.940	0.916
Compromised Domain	0.860	0.300	0.998	0.445	0.548	0.501

trained supervised classifiers using the same features defined in Chapter 2, Section 2.3.1. Among the candidate models, we started with Random Forest [16] for an initial evaluation. This choice was motivated by three factors: (i) ensemble tree methods are well known for their strong and stable performance across a wide range of classification tasks, (ii) Random Forests are particularly effective when working with mixed feature types, as in our dataset, and (iii) their inherent majority voting process helps mitigate overfitting.

As shown in Table 3.4, the classifier maintains very strong performance for attacker-owned and third-party domains, with F1-scores of 0.983 and 0.992. Precision, recall, and G-Mean stay very high (≥ 0.98), and the MCC scores (0.977 and 0.971) reflect the model's strong balance between correct and incorrect predictions. For compromised domains, the model achieves a noticeable improvement over the rule-based system. Recall rises to 0.754 (from 0.300), raising the F1-score to 0.760. The G-Mean (0.864) and MCC (0.758) confirm more balanced, reliable detection, suggesting the Random Forest could better capture subtle patterns of compromise that rule-based heuristics missed. Overall, the Random Forest results suggest that supervised learning provides a stronger basis compared to rule-based taxonomy, particularly in reducing false negatives in the compromised class, where the rule-based approach struggled the most.

Table 3.4: Random Forest results with 5-Fold CV (average across folds).

Class	Precision	Recall	Specificity	F1-Score	G-Mean	MCC
Attacker Domain	0.986	0.979	0.995	0.983	0.987	0.977
Third-Party Platform	0.990	0.993	0.975	0.992	0.984	0.971
Compromised Domain	0.786	0.754	0.993	0.760	0.864	0.758

Unsupervised Clustering. After evaluating the performance of the rule-based taxonomy and its supervised counterpart, we further examined whether domain ownership patterns could be uncovered without relying on any labels. This exploration was motivated by two goals: first, to determine whether the extracted features could provide sufficient inherent structure to distinguish ownership categories, and second, to compare unsupervised

learning against both rule-based and supervised approaches.

For this analysis, we applied the K-means [42] clustering algorithm. K-means was selected because it is one of the most widely used clustering techniques, offering a straightforward and interpretable baseline. However, it also relies on several strong assumptions, notably that clusters are roughly spherical, equally sized, and have equal variance. Although these assumptions may not hold perfectly in all feature spaces, K-means provides a useful reference for evaluating how well unsupervised methods can capture domain ownership patterns. We experimented with different values for the number of clusters, using the Elbow method [96] to identify the optimal configuration. The Elbow method involves plotting the within-cluster sum of squares against the number of clusters and identifying the 'elbow' or point where adding more clusters provides less improvement. The analysis suggested that five clusters provided the best fit for the data. We then applied majority voting, assigning each cluster the ownership label (attacker-owned, compromised, or third-party platforms) that appeared most frequently among the samples within it. This resulted in two clusters being identified as attacker-owned domains and the remaining three as third-party platforms. By adopting this scheme, we were able to measure the performance metrics.

The clustering results are summarized in Table 3.5. K-means formed clear groups for the attacker-owned and third-party classes, resulting in strong scores: attacker domains reached an F1-score of 0.960, G-Mean of 0.964, and MCC of 0.948, while third-party platforms achieved an F1-score of 0.968, G-Mean of 0.914, and MCC of 0.882. This indicates that the clusters aligned well with their true ownership categories. In contrast, the model failed to capture any meaningful structure for compromised domains, returning zeros across all metrics. This outcome highlights both the potential and the limitations of clustering. On the one hand, the results confirm that attacker-owned and third-party platforms have sufficiently distinct characteristics to shape distinct clusters. However, compromised domains lack clear, homogeneous patterns. In practice, this means that while clustering offers useful insights into the structure of the dataset, it is not suitable as a standalone replacement for the rule-based or supervised methods.

Table 3.5: Performance metrics for K-means clustering.

Class	Precision	Recall	Specificity	F1-Score	G-Mean	MCC
Attacker Domain	0.989	0.933	0.997	0.960	0.964	0.948
Third-Party Platform	0.940	0.997	0.837	0.968	0.914	0.882
Compromised Domain	0.000	0.000	0.000	0.000	0.000	0.000

Comparative Analysis: We implemented three approaches to categorize phishing web-

sites by domain ownership types: a rule-based taxonomy, an unsupervised clustering method, and a supervised classification model. The rule-based taxonomy, while straightforward to implement, struggled to capture the distinctions between the "Compromised Domain" and "Third-Party Platform" categories. This was evidenced by relatively high misclassification rates observed in its confusion matrix provided in Table 3.6 and also the low F1-score, G-Mean, and MCC rates demonstrated in Figure 3.5. The unsupervised clustering analysis captured underlying patterns, specifically for the "Attacker Domain" and "Third-Party Platform" categories. However, it completely failed to identify compromised domains, with zero rate across all metrics as shown in Figure 3.5 and the confusion matrix presented in Table 3.6, highlighting the limitations of clustering when categories lack consistent structural patterns. The supervised classification model, on the other hand, demonstrated the strongest overall performance, achieving the highest F1-score, G-Mean, and MCC across the different domain ownership categories as presented in Figure 3.5. Its confusion matrix, provided in Table 3.6, showed the most distinct separation between the target classes, indicating a stronger ability to differentiate between various types of phishing domains.

Actual Values	Predicted Values		
	Attacker	Third-Party	Compromised
Attacker	1292	77	8
Third-Party	7	3947	0
Compromised	13	101	49

(a) Rule-based Taxonomy

Actual Values	Predicted Values		
	Attacker	Third-Party	Compromised
Attacker	1348	9	20
Third-Party	8	3928	18
Compromised	11	29	123

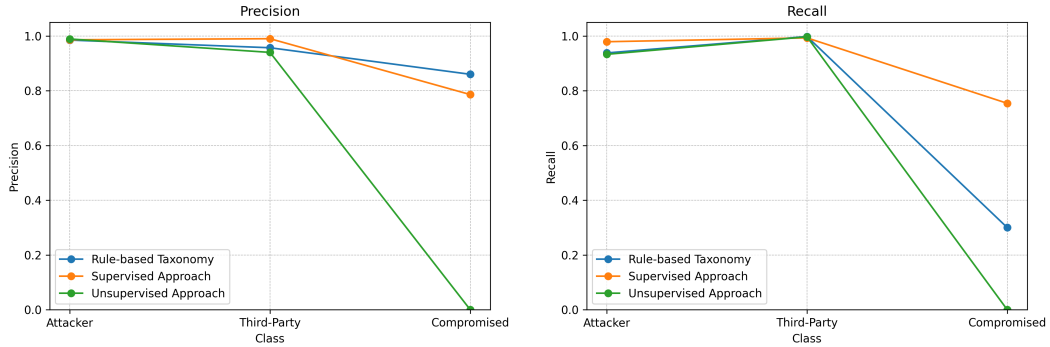
(b) Supervised Approach

Actual Values	Predicted Values		
	Attacker	Third-Party	Compromised
Attacker	1285	92	0
Third-Party	10	3944	0
Compromised	4	159	0

(c) Unsupervised Approach

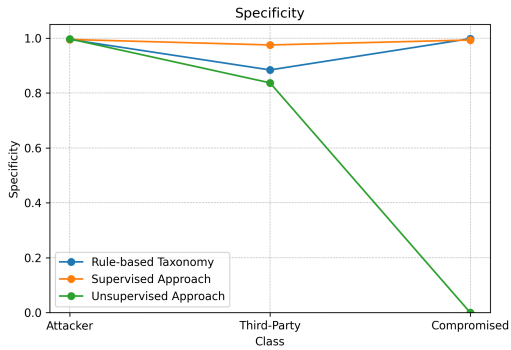
Table 3.6: Confusion matrices for the rule-based, supervised, and unsupervised approaches evaluated on the PhishXtract-Class-Labeled.

Based on these comparative results carried out on the manually verified dataset, the supervised classification approach appears as the most effective method for categorizing phishing websites according to their domain ownership characteristics. While the other techniques provided valuable complementary insights, the classification model's superior predictive capability makes it the recommended choice for this task. As a result, we have

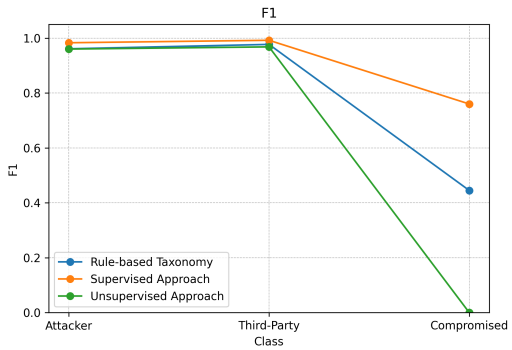


(a) Precision

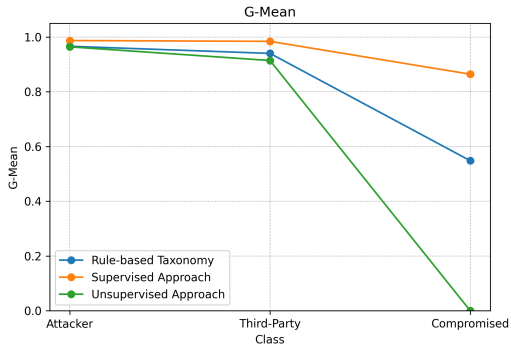
(b) Recall



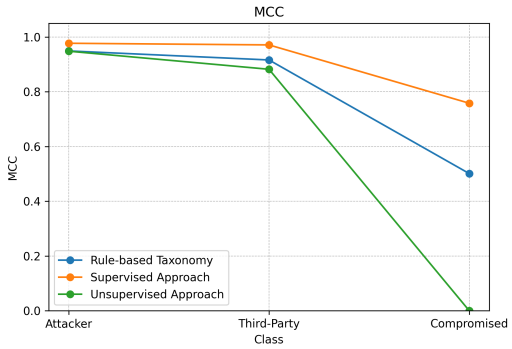
(c) Specificity



(d) F1-score



(e) G-Mean



(f) MCC

Figure 3.5: Comparison of rule-based, supervised, and unsupervised approaches across domain ownership types for Precision (a), Recall (b), Specificity (c), F1-score (d), and G-Mean (e), MCC (f)

chosen to use this supervised classification approach to train a model that can categorize the entire dataset of phishing websites based on their domain ownership types.

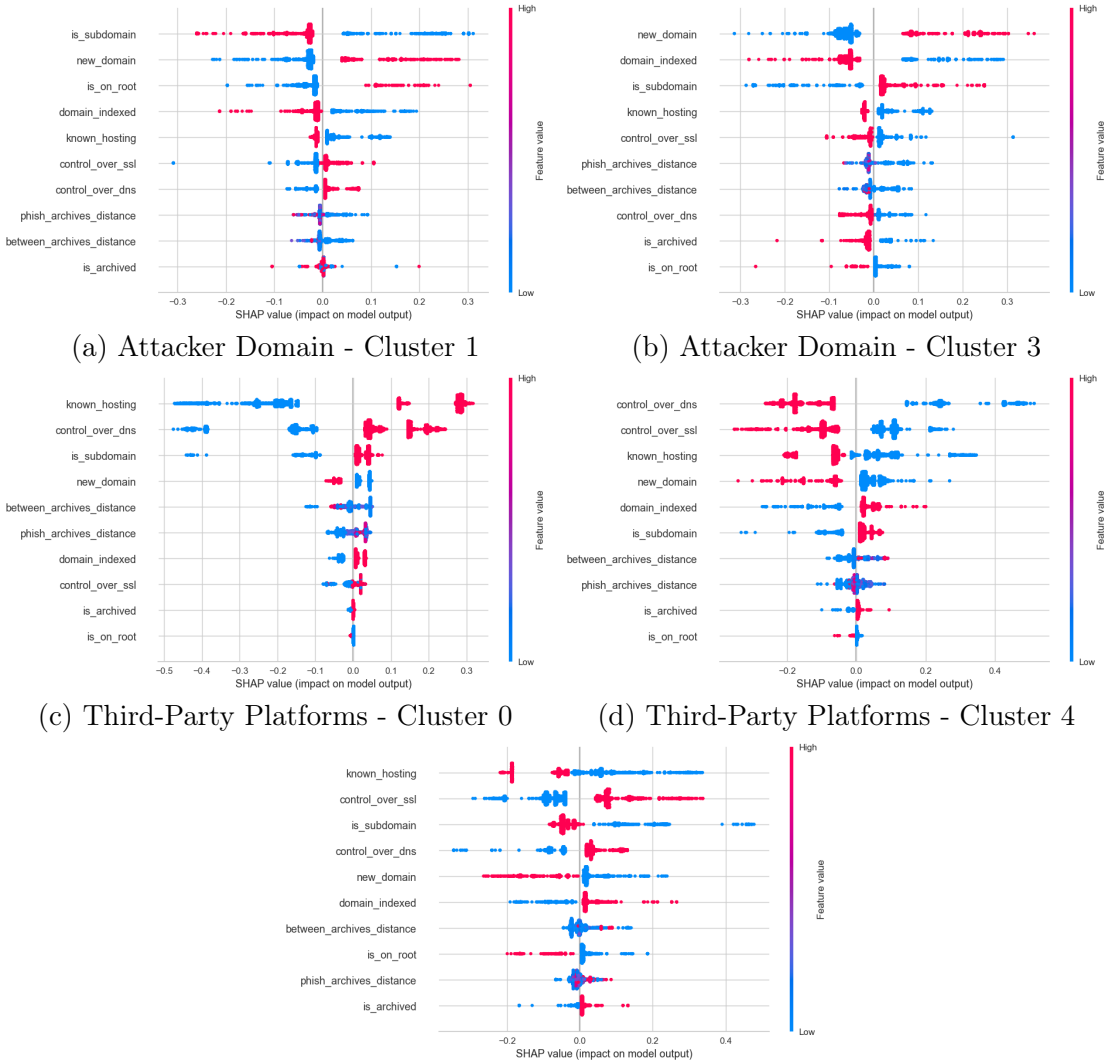
Interpretability: To better understand the characteristics that distinguish the identified clusters, we trained a Random Forest classifier using the extracted features as input and the cluster assignments as target labels. This model reproduces the clustering structure, allowing us to analyze which features contribute the most to separating the clusters.

For this analysis, we applied the SHAP framework [66]. SHAP is a game-theoretic approach to explain the output of any ML model using the Shapley values [92] from game theory. SHAP quantifies the contribution of each feature to individual predictions. Specifically, we used the TreeExplainer [65] module within the SHAP framework, which is designed to explain the output of tree-based ML models such as Random Forest. The TreeExplainer calculates the SHAP value for each feature by considering how the model's prediction changes when that feature is included or excluded, averaging these effects across all possible subsets of features. By computing the SHAP values for each feature in each cluster, we could quantify their contribution to the model predictions. This enabled us to identify the distinguishing characteristics and underlying structure of the three phishing domain ownership types.

As shown in Figures 3.6(a) and 3.6(b), two of the clusters contain samples that exhibit characteristics of domains created and controlled by attackers for phishing. Cluster 1 exhibits a stereotype attacker profile. A recent registration (`new_domain`) is one of the strongest right-pushing features, consistent with the fact that some attackers register domains shortly before starting the campaigns. The model's predicted likelihood increases when pages are hosted at the root path (`is_on_root`) rather than on subdomains, which aligns with the intuition that attackers often control the entire domain and host phishing content directly at the root. Some features show the opposite effect; samples that are not indexed tend to push right, showing the limited search presence of attacker-owned domains. Finally, being a known hosting platform (`known_hosting`) pushes left, indicating attackers' domains not hosted on third-party platforms. Together, these effects show new, standalone infrastructure with little historical archives, precisely the indicators of attacker-owned domains.

Cluster 3, as demonstrated in 3.6(b), presents a second attacker-like grouping with a slightly different pattern. The `new_domain` feature again contributes strongly and positively, while a lack of search indexing and sparse archival history reinforce the impression of minimal historical presence. However, unlike Cluster 1, many samples in this cluster are hosted on subdomains rather than at the root. Despite this variation, the aggregate SHAP signal still points to attacker-owned domains.

The SHAP pattern for Cluster 0, as shown in 3.6(c) is dominated by the third-party



(a) Attacker Domain - Cluster 1

(b) Attacker Domain - Cluster 3

(c) Third-Party Platforms - Cluster 0

(d) Third-Party Platforms - Cluster 4

(e) Third-Party Platforms - Cluster 2

Figure 3.6: SHAP summary-plot for each of the clusters (cluster labels are assigned based on majority voting scheme).

platforms' signature. High values of `known_hosting` push to the right, indicating that URLs hosted on recognizable platforms (such as web hosting, file sharing, URL shorteners, DDNS providers) are highly characteristic of this cluster. Hosting on subdomains further increases cluster membership, exactly what we expect when phishing content appears as a tenant page existing under a larger provider's platform. Having `control_over_dns` (indicating no wildcard records at the parent domain) also contributes positively, implying that tenants are resolved to explicit subdomain records rather than wildcard DNS. Archival and indexing features (`is_archived` and `domain_indexed`) contribute on the positive side, consistent with platforms that are generally visible to search engines and have a well-archived history.

Similarly, the SHAP pattern for Cluster 4 demonstrated in 3.6(d) reflects a third-party platform signature, but this time dominated by indicators of limited operational control. Absence of `control_over_dns` and `control_over_ssl` (i.e., absence of wildcard DNS or subdomain-specific certificates) pushes predictions strongly to the right, consistent with tenant websites that lack provider-level control of DNS or SSL/TLS certificates. As in Cluster 0, subdomain placement contributes positively while root-path hosting pushes against it. Overall, Cluster 4 aligns with third-party platforms, highlighted by the evidence that tenant websites have little administrative control.

Finally, we examined the SHAP analysis for Cluster 2 that contained the majority of samples verified as compromised domains during manual inspection, along with some samples from the other two ownership categories. As shown in Figure 3.6(e), the SHAP plot for this cluster reveals the most important features contributing to the model's predictions. The analysis indicated that the phishing samples in this cluster were hosted on domains that were not subdomains, were not included in our list of known third-party platforms, and were relatively aged, indexed, and archived. These last three attributes are often associated with legitimate, benign domains rather than malicious phishing domains, however, we could find some malicious domains in this cluster that share the same characteristics. Upon further investigation of the samples assigned to this cluster, we found that it included nearly all the phishing websites hosted on compromised domains. However, the majority of the samples in this cluster belong to the third-party platform category, specifically those for which phishing samples were not set up as subdomains.

Looking across the five plots, the SHAP attributions align well with both the taxonomy evaluation and the way classes are distributed across clusters. Clusters 1 and 3 show the familiar attacker pattern: recently registered domains, little or no search or archival history, and not hosted on known major hosting platforms. Clusters 0 and 4 have the third-party platforms' signature, with domains sitting under well-known third-party providers, typically on subdomains rather than the root; Cluster 4 shows especially low control over DNS

and SSL. Cluster 2 holds most of the compromised samples, domains that look legitimate in many ways (indexed, often archived) but are not hosted under known platforms and usually have phishing content on subdomains rather than the root. This matches the evaluation results: attacker and third-party categories are separated cleanly, while compromised domains remain harder to isolate because their footprint partly overlaps with third-party platforms.

Projection of the Dataset: To further explore the structure of the discovered clusters, we performed dimensionality reduction using t-distributed Stochastic Neighbor Embedding (t-SNE) [100]. The t-SNE algorithm allows us to project the high-dimensional feature space onto a two-dimensional plane, preserving the local proximity of data points as much as possible. Figure 3.7 shows the resulting t-SNE visualization of the dataset, where each data point represents an individual phishing website, and the colors correspond to the cluster assignments from the unsupervised clustering analysis.

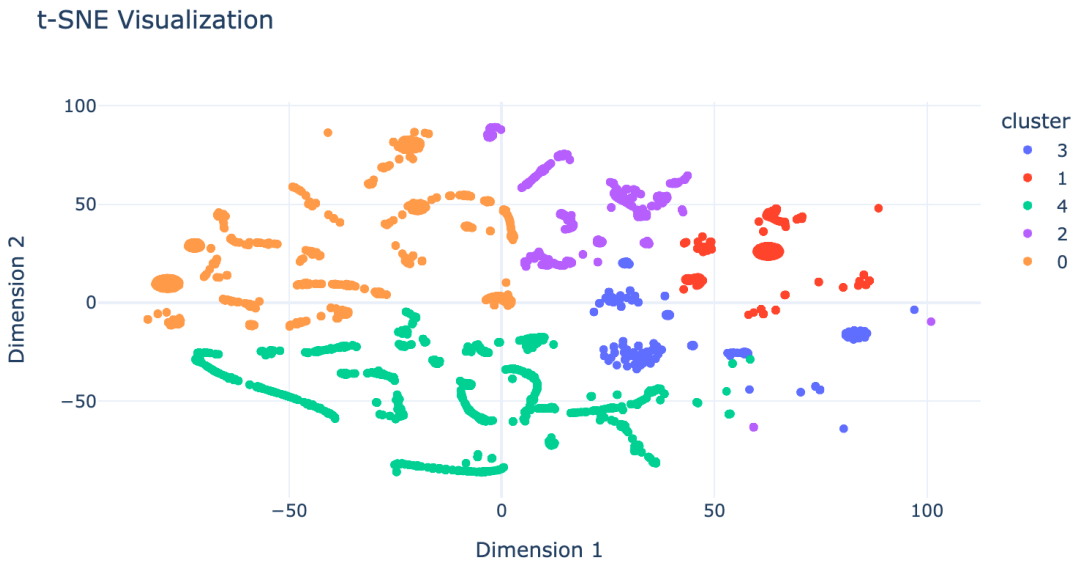


Figure 3.7: t-SNE visualization of the dataset after clustering (cluster 0 and cluster 2 are labeled as "Attacker Domain" and the other clusters are labeled as "Third-Party Platform" by the majority voting).

The t-SNE plot shows clear differences in degrees of cohesion among the clusters. The two attacker-owned clusters (Clusters 1 in red and 3 in blue) occupy adjacent regions on the right-hand side of the plot, with several compact regions that reflect internally consistent neighborhoods, a pattern that matches their shared SHAP profile (new domains, limited

indexing, and no archiving history). By contrast, the third-party platforms clusters spread more broadly; Cluster 0 (orange) stretches across the upper left, indicating a mix of tenant websites hosted across well-known platforms, while Cluster 4 (green) stretches across the lower left, showing other platforms characterized by limited DNS and SSL control.

Cluster 2 (purple), which contains most of the verified compromised samples, is between the shared and attacker regions and shows the greatest overlap with its neighbors. This position explains the confusion analysis; compromised pages inherit legitimate-like patterns (indexing/archiving) from their hosts but are not hosted on known third-party platforms, so their neighborhoods partially overlap with unknown third-party platforms. It is also clear that this cluster has some overlaps with attacker infrastructure (those that are relatively old and have some level of archive history). The t-SNE primarily preserves local neighborhoods, not global distances, but the consistent neighborhood patterns reinforce the SHAP interpretations and the observed evaluation: attackers and third-party platforms are well separated, but compromised domains fall into an overlapping zone (in between the two other ownership categories), which is harder to isolate cleanly.

3.3.3 In-Depth Experiments on the Supervised Classification Approach

The initial evaluation in Section 3.3.2 showed that supervised learning models outperform both the rule-based taxonomy and unsupervised clustering when categorizing phishing domains by ownership type. Building on this result, we improved the classification process in three stages: (i) feature refinement (ii) Exploring more supervised algorithms and (iii) hyperparameter tuning.

Feature Refinement: We first revisited the feature set introduced in Chapter 2, Section 2.3.1 to assess which attributes were most useful for classification. In the original feature set, we included *Wildcard SSL/TLS Certificate (control_over_ssl)* to measure the level of control over subdomains. This feature distinguished between phishing sites on subdomains with their own certificates and those covered by a wildcard certificate issued to the parent domain. By re-evaluation, we removed this attribute due to its high computational cost and minimal contribution to classification performance. This adjustment simplified the feature set while maintaining model performance.

Learning Algorithms and Hyperparameter Tuning: Next, we extended our evaluation beyond the Random Forest model used in the initial analysis. A range of supervised classifiers was compared. This comparison ensured that the chosen model aligns well with the characteristics of the ownership classification task. We evaluated seven different ML

classifiers from three major categories of models: linear models, probabilistic models, and tree-based methods. These algorithms were chosen for their diverse strengths in handling structured data and their effectiveness in classification tasks. Logistic Regression [46] uses the logistic function to estimate probabilities, making it suitable for both binary and multi-class classification. Support Vector Machine (SVM) [28] identify optimal hyperplanes for data separation in high-dimensional spaces. Naive Bayes [59], a probabilistic model based on Bayes' Theorem, assumes feature independence to ensure simplicity and computational efficiency. Decision Tree [17] uses a hierarchical, tree-like structure to split data based on feature values, offering interpretability and adaptability. Random Forest [16], an ensemble method, combines multiple independent decision trees to enhance prediction accuracy and prevent overfitting by averaging their outputs. XGBoost [25], a gradient-boosting technique, iteratively refines weak learners to produce an optimized model for speed and performance. Lastly, Extremely Randomized Trees [38] add randomness to split-point selection, further reducing variance while retaining the advantages of ensemble methods.

We used a nested stratified CV strategy [20] for performance evaluation to ensure both accurate hyperparameter selection and unbiased performance assessment. In this approach, stratification maintains the same class distribution across all folds, while the nested structure separates the model tuning and evaluation phases. As illustrated in Figure 3.8, the inner loop performed hyperparameter tuning by systematically testing different configurations (e.g., kernel type in SVM, number of estimators in Random Forest, learning rate in XGBoost). This helped identify the best configuration for each classifier. Then, in the outer loop, we evaluated these tuned models on separate data folds to measure their precision, recall, and F1-score. This nested strategy prevents overly optimistic results by ensuring that the data used for tuning hyperparameters is kept separate from the data used to evaluate generalization performance. By clearly dividing the tuning (inner loop) and evaluation (outer loop) phases, we obtained more reliable performance comparisons across classifiers. The model with the highest average performance in the outer loop was ultimately chosen as the best-performing classifier. Table 3.7 presents the explored hyperparameter search space along with the best-performing values selected for each algorithm in bold.

Table 3.8 presents the average performance of all classifiers based on precision, recall, F1-score, G-Mean, specificity, and MCC, computed via nested 5-fold CV. For each metric, we report the *macro-average*, which is calculated as the unweighted mean of the metric computed for each class (attacker-owned, compromised, and third-party hosted). Macro-average gives equal importance to all classes, regardless of their frequency in the dataset.

Macro-Level Comparison: Among all classifiers, XGBoost achieved the highest macro-average precision (0.923), while SVM attained the best macro-average recall (0.924) and

Table 3.7: Hyperparameter Search Space for Evaluated ML Algorithms. The best-performing hyperparameter values for each model are highlighted in **bold**

(a) Tree-Based Models

Hyperparameters	DT	RF	XGB	Extra Trees
n_estimators	-	{100, **200**, 500}	{**100**, 200, 500}	{**100**, 200, 500}
max_depth	{None, 5, 10, **20**}	{**None**, 5, 10, 20}	{3, **5**, 7}	{None, 5, 10, **20**, 50}
min_samples_split	{**2**, 5, 10}	{**2**, 5, 10}	-	{**2**, 5, 10}
criterion	{gini, **entropy**}	{**gini**, entropy}	-	{gini, **entropy**}
class_weight	{**balanced**, None}	{**balanced**, balanced_subsample}	-	{**balanced**, balanced_subsample}
learning_rate	-	-	{0.01, **0.1**, 0.2}	-
subsample	-	-	{0.6, 0.8, **1.0**}	-
colsample_bytree	-	-	{0.6, 0.8, **1.0**}	-
gamma	-	-	{**0**, 0.1, 0.2}	-

(b) Linear Models

Hyperparameters	LR	SVM
class_weight	{**None**}	{**balanced**}
C	{0.1, **1**, 10}	{0.1, **1**, 10}
gamma	{auto, **scale**}	{auto, **scale**}
kernel	-	{linear, **rbf**, poly, sigmoid}
penalty	{l1, **l2**, elasticnet, None}	-
solver	{liblinear, newton-cg, **lbfgs**, sag, saga}	-
max_iter	{**1000**, 2000}	-
l1_ratio	{0.1, **0.5**} (if penalty is elasticnet)	-

(c) Probabilistic Models

Hyperparameters	NB
var_smoothing	$\{10^{-9}, 10^{-8}, \ldots, \mathbf{10^{-1}}\}$

Abbreviations: LR = Logistic Regression, SVM = Support Vector Machine, NB = Naive Bayes, DT = Decision Tree, RF = Random Forest, XGB = XGBoost, Extra Trees = Extremely Randomized Trees. All algorithms were implemented using the scikit-learn library, except XGBoost, which was implemented using the xgboost package.

Table 3.8: Performance Comparison of Classifiers Across Classes. The best results for each model are highlighted in **bold**.

(a) Average Precision and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.977 ± 0.014	0.984 ± 0.013	0.979 ± 0.014	0.976 ± 0.012	**0.984 ± 0.009**	0.980 ± 0.010	0.979 ± 0.012
Compromised	0.718 ± 0.088	0.275 ± 0.012	0.398 ± 0.050	0.703 ± 0.069	0.761 ± 0.062	**0.801 ± 0.058**	0.789 ± 0.019
Third-Party	0.956 ± 0.003	**0.996 ± 0.001**	0.977 ± 0.003	0.991 ± 0.005	0.991 ± 0.005	0.989 ± 0.005	0.990 ± 0.006
Macro-Average	0.884 ± 0.035	0.752 ± 0.009	0.785 ± 0.022	0.890 ± 0.029	0.912 ± 0.025	**0.923 ± 0.024**	0.919 ± 0.014

(b) Average Recall (Sensitivity) and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.962 ± 0.009	0.943 ± 0.014	0.960 ± 0.008	0.975 ± 0.010	0.979 ± 0.008	0.985 ± 0.004	**0.987 ± 0.004**
Compromised	0.129 ± 0.064	**0.914 ± 0.029**	0.614 ± 0.078	0.754 ± 0.104	0.754 ± 0.085	0.700 ± 0.103	0.712 ± 0.127
Third-Party	**0.994 ± 0.004**	0.915 ± 0.006	0.962 ± 0.008	0.988 ± 0.003	0.992 ± 0.003	0.993 ± 0.004	0.991 ± 0.003
Macro-Average	0.695 ± 0.026	**0.924 ± 0.016**	0.845 ± 0.031	0.906 ± 0.039	0.909 ± 0.032	0.892 ± 0.037	0.897 ± 0.040

(c) Average Specificity and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.992 ± 0.005	0.995 ± 0.004	0.993 ± 0.005	0.992 ± 0.004	**0.995 ± 0.003**	0.993 ± 0.003	0.993 ± 0.004
Compromised	**0.998 ± 0.001**	0.926 ± 0.004	0.971 ± 0.004	0.990 ± 0.004	0.992 ± 0.003	0.995 ± 0.002	0.994 ± 0.001
Third-Party	0.884 ± 0.008	**0.990 ± 0.003**	0.942 ± 0.008	0.977 ± 0.012	0.977 ± 0.014	0.973 ± 0.012	0.974 ± 0.016
Macro-Average	0.958 ± 0.005	0.970 ± 0.004	0.969 ± 0.006	0.986 ± 0.007	**0.988 ± 0.007**	0.987 ± 0.006	0.987 ± 0.007

(d) Average F1-score and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.970 ± 0.009	0.963 ± 0.010	0.970 ± 0.007	0.976 ± 0.006	0.981 ± 0.005	0.982 ± 0.006	**0.983 ± 0.007**
Compromised	0.214 ± 0.090	0.423 ± 0.016	0.482 ± 0.059	0.721 ± 0.053	**0.751 ± 0.028**	0.742 ± 0.065	0.743 ± 0.070
Third-Party	0.975 ± 0.002	0.953 ± 0.003	0.969 ± 0.004	0.989 ± 0.003	**0.992 ± 0.002**	0.991 ± 0.003	0.991 ± 0.003
Macro-Average	0.719 ± 0.034	0.780 ± 0.010	0.807 ± 0.023	0.895 ± 0.021	**0.908 ± 0.018**	0.905 ± 0.025	0.905 ± 0.025

(e) Average G-Mean and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.977 ± 0.005	0.968 ± 0.008	0.976 ± 0.004	0.984 ± 0.004	0.987 ± 0.004	0.989 ± 0.003	**0.990 ± 0.003**
Compromised	0.349 ± 0.084	**0.920 ± 0.015**	0.771 ± 0.049	0.862 ± 0.059	0.864 ± 0.047	0.832 ± 0.062	0.838 ± 0.074
Third-Party	0.937 ± 0.004	0.951 ± 0.003	0.951 ± 0.004	0.982 ± 0.007	**0.984 ± 0.006**	0.983 ± 0.007	0.982 ± 0.008
Macro-Average	0.754 ± 0.031	**0.947 ± 0.009**	0.899 ± 0.019	0.943 ± 0.023	0.945 ± 0.019	0.935 ± 0.024	0.937 ± 0.025

(f) Average MCC and Standard Deviation Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	Extra Trees
Attacker	0.960 ± 0.012	0.951 ± 0.014	0.960 ± 0.010	0.968 ± 0.007	0.975 ± 0.006	0.976 ± 0.009	**0.977 ± 0.009**
Compromised	0.291 ± 0.082	0.478 ± 0.019	0.475 ± 0.063	0.716 ± 0.056	**0.747 ± 0.028**	0.739 ± 0.064	0.739 ± 0.069
Third-Party	0.908 ± 0.008	0.858 ± 0.009	0.893 ± 0.012	0.962 ± 0.012	**0.970 ± 0.006**	0.968 ± 0.012	0.966 ± 0.011
Macro-Average	0.719 ± 0.034	0.762 ± 0.014	0.776 ± 0.031	0.882 ± 0.025	**0.898 ± 0.013**	0.894 ± 0.028	0.894 ± 0.028

Abbreviations: LR = Logistic Regression, SVM = Support Vector Machine, NB = Naive Bayes, DT = Decision Tree, RF = Random Forest, XGB = XGBoost, Extra Trees = Extremely Randomized Trees.

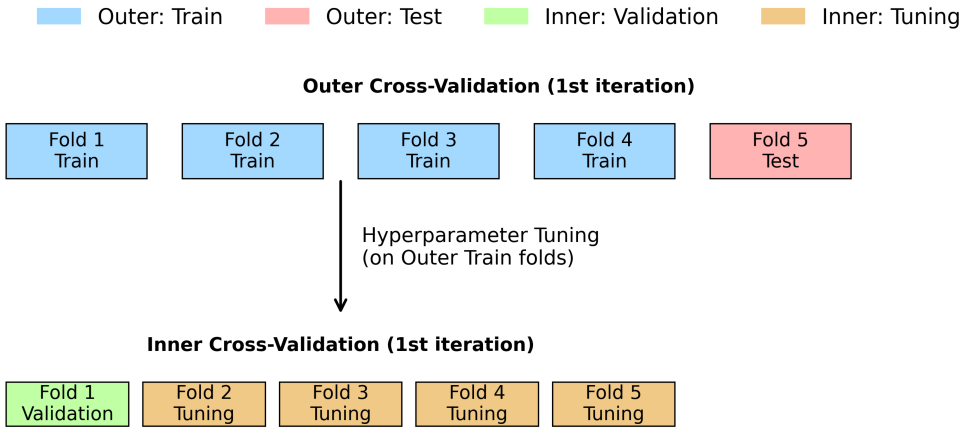


Figure 3.8: Nested 5-Folds CV: One iteration of the outer loop is shown, with four folds used for training and one for testing. The inner loop performs hyperparameter tuning using a separate 5-fold split of the outer training data.

G-Mean (0.947). However, Random Forest exhibited the most balanced and consistent performance: it placed in the top two across macro-F1 (0.908), G-Mean (0.943), and MCC (0.897), and also achieved the highest macro-specificity (0.988). This indicates that RF maintains a strong balance between true positives and false positives across all classes.

Attacker and Third-Party Classes: All classifiers performed well on attacker-owned and third-party domains, with precision, recall, and F1-scores exceeding 0.95 in most cases. Tree-based methods, including Random Forest, XGBoost, and Extra Trees, dominated in F1-score and MCC for these classes, reflecting their strong ability to distinguish clear-cut phishing infrastructure. Extra Trees slightly outperformed others in F1-score for the attacker class (0.983), while Random Forest achieved the top result for third-party domains (0.992).

Compromised Class: Identifying compromised domains is more challenging. Some models, such as SVM, achieve high recall in detecting these cases but often have low precision, leading to false positives. In contrast, Decision Trees, Random Forests, XGBoost, and Extremely Randomized Trees offer a better balance between precision and recall for compromised domains, resulting in a higher F1-score compared to other methods.

Best Model: While XGBoost offered the highest precision and SVM achieved the highest recall and G-Mean, Random Forest consistently placed near the top in all metrics and had the best macro-average MCC (0.898), macro F1-score (0.908), and specificity (0.988). Decision Trees, XGBoost and Extremely Randomized Trees also performed strongly, Random

Forest consistently placed near the top in all metrics and delivered the best macro-average MCC (0.898), macro F1-score (0.908), and specificity (0.988). In large-scale or real-time detection settings, even small differences can be critical. Decision Tree is appealing for its simplicity, but it can be more prone to overfitting if not carefully tuned, and one single tree may miss certain subtle distinctions in the data. Extra Trees, while also an ensemble, uses more randomness in picking split points. This can help prevent overfitting, but in our particular tests, we found that Random Forest's balance between randomness and control gave it the most consistent performance. Section 3.4 examines how Random Forest categorizes the phishing corpus collected over the year, thus revealing trends in domain ownership and usage.

Due to the limited number of compromised domain samples in our labeled dataset, we also explored oversampling techniques to mitigate the class imbalance during training. Specifically, we applied both RandomOverSampler [58] and Synthetic Minority Over-sampling Technique (SMOTE) [24] within each training fold of the CV. While these techniques helped slightly improve recall for the minority class, they also introduced noise, reducing precision and failing to improve F1-scores. As a result, we reported the results from the original imbalanced dataset. Detailed performance results for the oversampled versions are included in the Appendix A. Next, to better understand how the classifiers differentiate between ownership categories, we focus on the interpretability of the models. Among the tested classifiers, Decision Tree offers a good balance between performance and interpretability, allowing us to reason about the most influential features in classification outcomes.

Interpretation of the Supervised Classification Model: To better understand how our model distinguishes between ownership categories, we examine the interpretability of the classifiers. Although ensemble models such as Random Forest provided stronger performance, their aggregated architecture makes them less transparent for explaining decision paths in trees. In contrast, a Decision Tree offers a clear, interpretable structure that allows us to trace how specific features influence classification outcomes. As shown in Table 3.8, the Decision Tree demonstrated competitive performance compared to ensemble models, making it a suitable candidate for interpretability analysis. By examining the tree structure and identifying which features are split, we gain insight into how the model distinguishes between classes. To understand the rationale behind these results, we analyzed the tree's feature usage and the top rules at its first few levels.

During training, the Decision Tree model used distinct features, with the *Similarity in Domain Archives* and *Similarity between Archives and Phishing Page* appearing most frequently (42 and 30 times, respectively). Their frequent use suggests that structural consistency across archived versions, or similarity between archived content and the phishing

page, is a strong indicator for classification. Meanwhile, features like *New Domain Registration* and *Known Third-Party Platform* appeared only once, likely helping to resolve specific edge cases rather than guiding primary decision boundaries. An examination of the top three levels of the tree shows how the model prioritizes different cues to distinguish between attacker-owned, compromised, and third-party domains. The root node splits on whether the domain is newly registered, followed by checks for hosting on a known third-party platform or visibility in Google's index. As the tree deepens, the model increasingly relies on the two archival similarity features to differentiate between domains with stable, consistent footprints and those with structural inconsistencies. Other features contribute to finer-grained decisions in specific cases. Other features provide secondary refinement where needed. Overall, these splits highlight the Decision Tree's decision-making process. The model's reliance on archival similarity features underscores the importance of temporal page structure in classifying phishing domain ownership types.

3.4 Scaling the Domain Ownership Model on Year-Long Dataset

After identifying the best-performing classifier using the PhishXtract-Class-Labeled dataset, we applied it to the PhishXtract-Class-Unlabeled subset to predict the domain ownership types for the remaining URLs. As previously mentioned, the PhishXtract-Class dataset spans phishing reports collected between October 2023 and October 2024. We excluded February 2024 due to incomplete data from the APWG feeds, which were not received in time for accurate feature extraction. By omitting February 2024, we avoided the confounding effects of partial data, maintaining valid comparisons among feeds.

Monthly Overview and Feed Comparison: To analyze domain ownership trends over time, we examined the labels across the entire PhishXtract-Class dataset. These labels were produced using the best-performing supervised classifier identified in our earlier evaluation (the Random Forest model). This includes both manually verified labels from the PhishXtract-Class-Labeled subset and predicted labels for the remaining data, PhishXtract-Class-Unlabeled. Figure 3.9 shows the monthly volume of phishing reports from APWG, PhishTank, and OpenPhish, categorized into *attacker-owned*, *compromised*, and *third-party hosted*. Several notable trends emerge:

- Feed-Specific Reporting Patterns: Although PhishTank and OpenPhish consistently identify a large volume of third-party hosted domains, APWG shows a more variable distribution. Notably, starting in March 2024, APWG reports a higher portion of attacker-owned domains than in previous months. This shift could be linked to APWG's

specialized sources—such as organization partners or researchers—who focus on newly registered or suspicious domains used in emerging phishing campaigns.

- Third-Party Hosted Dominance: Throughout the year, third-party hosted domains consistently made up the largest portion of our phishing dataset, as shown when we analyzed all three data feeds together. This trend likely highlights how attackers are increasingly relying on established hosting platforms, shared hosting services, or legitimate third-party domains to execute their phishing campaigns.
- Compromised Domains: Compromised domains represent a relatively small proportion in every feed. One reason could be that detecting compromised domains usually requires more in-depth inspection (e.g., content comparisons, and structural analysis). Another explanation is that compromising a domain for phishing demands considerable attacker effort, so it may happen less frequently than simply registering or reusing a third-party service.

Figure 3.9: Overall class distribution across the Dataset and for each Feed.

Aggregation and Duplicate Treatment: At the feed level, we applied a seven-day deduplication window to prevent counting the same phishing URL multiple times. We took the same strategy in the aggregated plots—where all three feeds' reports are combined. However, to highlight how actively each feed detects phishing campaigns and preserves the relative weight of each source in the combined data, Figure 3.10 represents the total input from each feed.

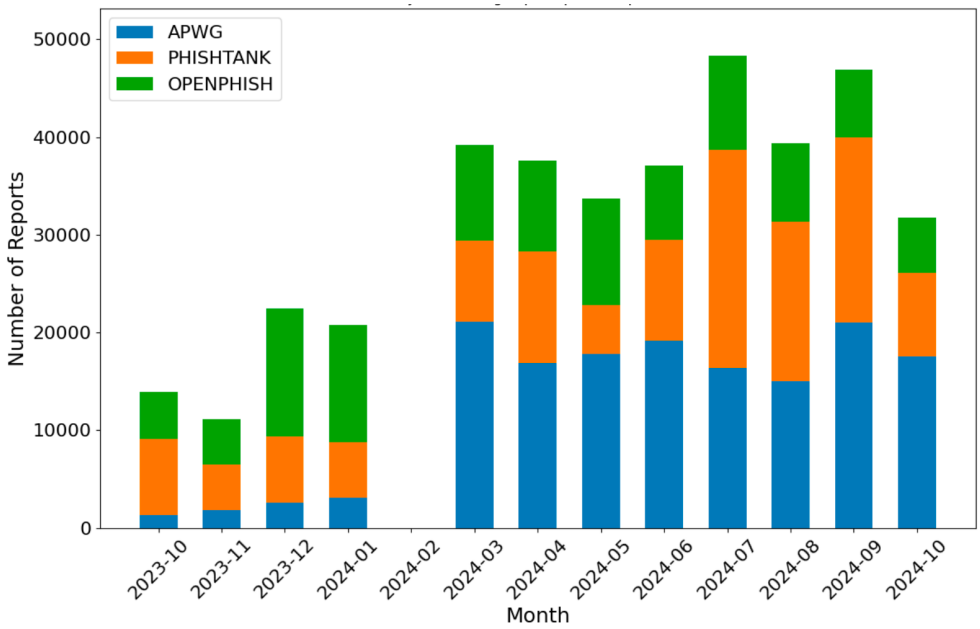


Figure 3.10: Number of Phishing Reports per Feed per Month

Overall, the results highlight the dominance of third-party hosted domains throughout most of the year, with compromised domains consistently underrepresented. Meanwhile, APWG's data display notable shifts in attacker-owned domain detection after March 2024, pointing to feed-specific differences in sources and detection methods. By examining these trends across the past year (excluding February 2024), we have gained clearer insights into how attackers are adapting their methods. They are either exploiting legitimate third-party platforms or registering new domains specifically for malicious purposes.

3.4.1 Representativeness Analysis of the Labeled Dataset for Descriptive Analysis:

Manual labeling of phishing domains is a resource-intensive process that requires expert analysis and quick access to live phishing websites. During our study, we faced significant constraints on both labeling capacity and data accessibility. Because phishing infrastructure is short-lived and often goes offline within hours or days, labeling must occur soon after collection to ensure that webpage content and hosting indicators remain available. Distributing the labeling effort across the full year was also infeasible, as older phishing sites were no longer accessible, and our manual annotation resources were limited. We therefore selected a 5-day window (April 20–24, 2024) that maximized access to live pages and aligned with our annotation capacity.

Although the labeled period spans only five consecutive days, it was chosen to balance practical constraints with the need for reliable labeling. Capturing a sample that accurately reflects a dynamic, year-long dataset is inherently challenging. Moreover, selecting representative time windows based on fixed intervals risks introducing bias, inconsistent annotation quality, or overlooking important variation. To avoid these pitfalls, we focused labeling on a short window and then evaluated its representativeness using feature distribution comparisons and statistical testing, as described below. In this context, representativeness refers to the statistical alignment of features between the 5-day labeled subset and the broader one-year-long dataset.

We began our representativeness analysis by comparing the 5-day labeled subset to the full year-long dataset. For each feature, we calculated the proportion of `True` values for Boolean indicators and the mean for numeric features. As this comparison involves only one aggregate value per group, formal statistical testing was not applicable. Table 3.9 presents this side-by-side comparison. Across most features, the values in the 5-day labeled subset closely align with those of the full dataset. For instance, features like `domain_indexed`, `is_subdomain`, and `is_archived` exhibit consistent proportions, while numeric indicators such as `between_archives_distance` and `phish_archives_distance` follow the same trend, with slightly lower values in the labeled data. Although some variation exists – such as lower `control_over_dns` and `known_hosting` rates in the labeled subset – the overall patterns remain consistent. These results offer additional evidence that the labeled 5-day window provides a structurally representative snapshot of the year-long phishing activity and is suitable for use in model training and evaluation.

To further evaluate the representativeness of the labeled subset at a more granular level, we compared the labeled dataset to multiple 5-day samples randomly selected from across the year. Specifically, we selected two 5-day periods per month –one consisting

Table 3.9: Comparison of feature values between the full year-long dataset and the 5-day labeled subset. Boolean features are shown as the percentage of `True` values; numeric features are shown as means.

Feature Name	**Full Year**	**5-day Labeled**
Boolean Features (True %)		
`new_domain`	30.23	24.43
`control_over_dns`	79.11	69.15
`domain_indexed`	72.08	77.68
`known_hosting`	59.08	45.29
`is_archived`	77.92	82.31
`is_on_root`	4.68	6.79
`is_subdomain`	84.33	89.80
Numeric Features (Mean)		
`between_archives_distance`	459.85	301.24
`phish_archives_distance`	257.94	178.67

of consecutive days and one of non-consecutive days – excluding February 2024 due to incomplete feed data. This resulted in 24 sampled periods across the year. For each of these periods, we calculated the distributions of the features, including both Boolean and numeric types. Boolean features (e.g., the presence of indexed domains) were summarized using the percentage of `True` values, while numeric features (e.g., similarity in domain archives) were summarized using their mean values. We then compared these statistics with those of the labeled dataset. As shown in Figures 3.11, 3.12, and 3.13, the labeled subset exhibits feature distributions that fall well within the Interquartile Range (IQR) of the sampled periods. This suggests that the labeled dataset reflects the overall diversity and structure of the phishing URLs encountered throughout the year.

To confirm the visual insights from the previous analysis, we performed statistical testing using the Mann–Whitney U test [68] for each feature. This test is commonly used to compare whether two independent groups come from the same distribution, without assuming that the data follows a normal distribution. In our case, the test evaluates whether the distribution of a given feature in the labeled subset is significantly different from that of the sampled periods. The null hypothesis for each test is that there is no difference in distributions between the two groups. A p-value greater than or equal to 0.05 indicates insufficient evidence to reject this hypothesis; that is, we cannot conclude that a difference between the distributions exists with %95 confidence. Table 3.10 reports the p-

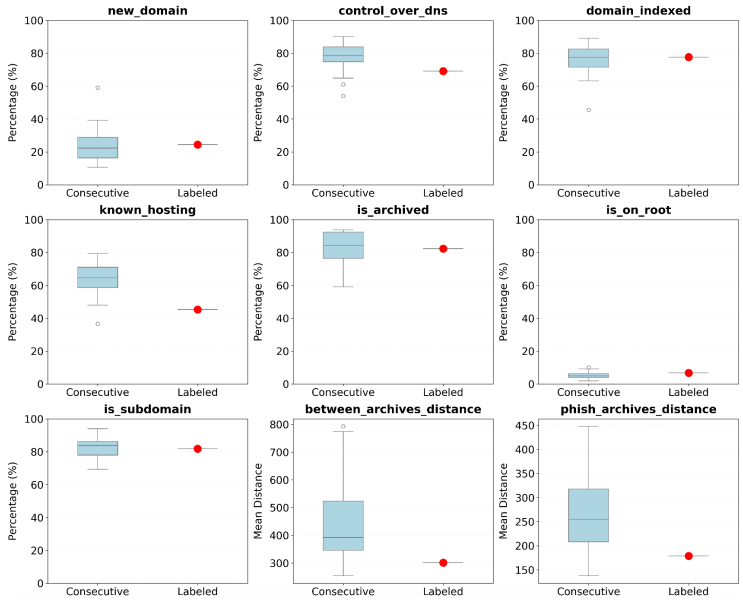


Figure 3.11: Feature Distribution Comparison: 5-day Consecutive Periods vs. 5-day Labeled Subset

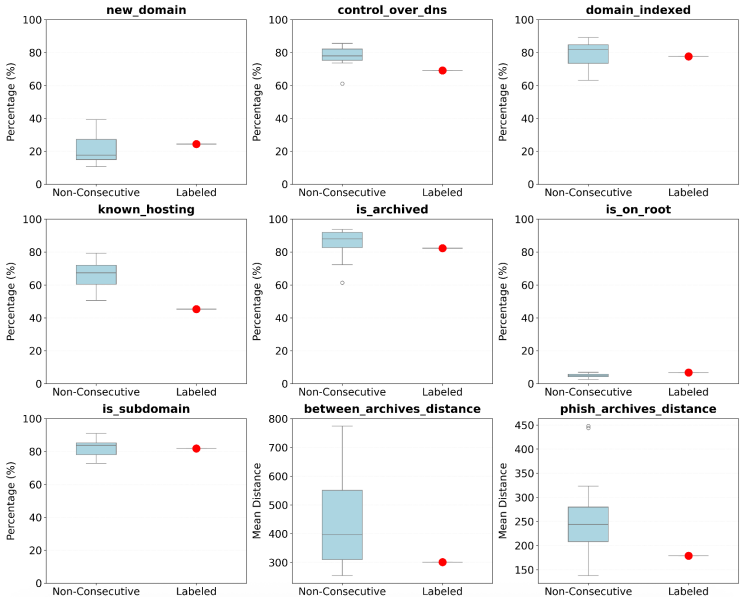


Figure 3.12: Feature Distribution Comparison: 5-day Non-consecutive Periods vs. 5-day Labeled Subset

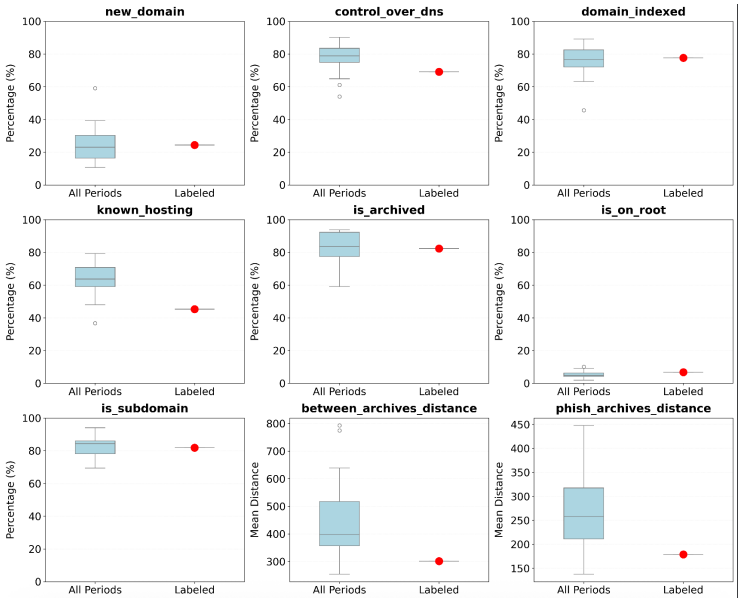


Figure 3.13: Feature Distribution Comparison: All 5-day Periods vs. 5-day Labeled Subset

values obtained for each feature, comparing the labeled subset against both consecutive and non-consecutive 5-day samples, as well as all samples combined. In all cases, the p-values were above the 0.05 threshold, indicating no statistically significant difference between the labeled subset and the sampled periods across the features evaluated.

It is also worth mentioning that the features used in our model, as described in Chapter 2, Section 2.3.1, are not time-dependent. That is, they represent static characteristics of the domain at the time the phishing URL was detected rather than capturing any chronological sequence or historical trend. As a result, there is no temporal dependency in the data, and applying a model trained on URLs from April to classify URLs from other time periods does not violate any assumptions or introduce prediction bias.

We acknowledge that representativeness here is defined and assessed at the level of feature distributions across 5-day groups. Although the labeled data set reflects the feature distribution of the full dataset, it is not optimized for proportion-based inference of class distributions or trend analysis. Those applications require a more temporally distributed ground truth, which was infeasible to obtain under our constraints. Thus, to address this limitation and uncover trends in domain ownership and usage, we rely on a ML model. This strategy, which has also been adopted in prior security research [13,56,69], offers a scalable and reliable alternative to manual annotation or simple proportion-based inference.

Table 3.10: Mann–Whitney U test results comparing the labeled dataset against 5-day sampled periods. All features show no significant (ns) difference ($p \geq 0.05$), confirming distributional similarity.

Feature	Cons.		Non-Cons.		All Periods	
	p-value	sig.	p-value	sig.	p-value	sig.
new_domain	0.96	ns	0.92	ns	1.00	ns
control_over_dns	0.40	ns	0.31	ns	0.38	ns
domain_indexed	1.00	ns	0.92	ns	1.00	ns
known_hosting	0.16	ns	0.15	ns	0.15	ns
is_archived	0.80	ns	0.46	ns	0.84	ns
is_on_root	0.48	ns	0.46	ns	0.46	ns
is_subdomain	0.96	ns	0.92	ns	0.92	ns
between_archives_distance	0.40	ns	0.46	ns	0.38	ns
phish_archives_distance	0.40	ns	0.46	ns	0.38	ns

3.4.2 ML for Descriptive Analysis

To perform descriptive analysis at scale, we extended ownership labels from a manually verified subset to the entire PhishXtract-Class dataset using a supervised classifier. This approach was selected because of practical constraints; manually annotating over 100,000 domains is infeasible due to the intensive labor involved, and fast annotation is essential to capture volatile infrastructure indicators before phishing sites go offline. As discussed in Section 3.4.1, we focused our labeling efforts on a recent 5-day window, selected because it was recent relative to the time of experimentation, ensuring that phishing websites were still live and infrastructure indicators accessible. We then validated the structural representativeness of this subset through feature distribution analysis and statistical testing.

Alternative approaches proved inadequate. Rule-based heuristics were previously explored in our prior work [34], but these lacked the flexibility to detect subtle patterns across phishing infrastructures. We also evaluated unsupervised clustering techniques, which failed in distinguishing compromised domains, one of the most challenging ownership types. While proportion-based inference might appear simpler, it is not optimized for identifying evolving trends in attacker behavior or infrastructure usage. Accurately applying such inference would require multiple labeled samples distributed over time, which was not feasible given labeling constraints and the short lifespan of phishing content.

Despite slightly weaker performance on the compromised class, our supervised model achieved over %90 macro-averaged F1-score, G-Mean, and MCC, demonstrating strong

Table 3.11: Prior work using a small ground-truth subset to label a much larger corpus.

Study	Labelled Subset	Auto-labelled	Descriptive Analysis
Page et al. [56]	5,744 domains	32k domains	Compromised vs. malicious phishing domains
Marrofi et al. [69]	2,329 domains	38k domains	Compromised vs. malicious domains, role of TLS-price, keyword abuse
Bayer et al. [13]	15,811 domains	219k domains	Compromised vs. malicious domains

overall reliability. Random Forest was selected for its consistent performance across classes and folds. While some classification is inevitable, particularly in the minority class, our pipeline minimizes risk through careful model selection, macro-averaged evaluation, and reliance on a carefully manually verified subset. This methodology aligns with prior research that has scaled descriptive analyses in this research domain using small labeled subsets (Table 3.11).

3.5 Conclusion

In this chapter, we conducted a detailed analysis of phishing domain ownership types using a year-long dataset aggregated from multiple reporting sources. We began by comparing three major phishing feeds – PhishTank, OpenPhish, and APWG – finding limited overlap among them. This confirmed the value of aggregation for building a diverse and representative corpus of phishing activity.

We then applied our domain ownership taxonomy, categorizing phishing domains into three groups: attacker-owned, compromised, and third-party hosted. The analysis revealed distinct attacker strategies, including a notable shift toward exploiting third-party platforms. This observation highlights a potential limitation of detection approaches that rely primarily on infrastructure-level signals—such as newly registered domains, DNS behavior, or CT logs, which are generally effective for attacker-owned domains but may miss phishing activity hosted on trusted platforms. Consequently, a considerable portion of phishing campaigns could remain invisible to infrastructure-focused monitoring.

By addressing the first research question – quantifying phishing website distributions across domain ownership types – this analysis identifies areas where current infrastructure-based methods may be less effective. These findings motivate the next stage of the thesis: exploring detection strategies that do not rely solely on infrastructure features. The following chapter investigates this direction, focusing on phishing propagation as an alternative source of signals for early detection.

Chapter 4

Beyond Infrastructure: Detecting Phishing at the Source of Spread

In the previous chapter, we analyzed the distribution of phishing domains and showed a growing reliance on third-party platforms, which are less visible to infrastructure-based detection. Because these platforms evade traditional signals, identifying phishing requires looking beyond domain-level indicators and approaching the problem from a different perspective. Building on this insight, Chapter 4 investigates the detection at the source of spread by examining phishing propagation on social media. To address our research question of detecting phishing beyond domain-level indicators, this chapter focuses on a strategy grounded in social propagation signals.

The content of this chapter has been accepted and presented in the 22nd International Conference on Privacy, Security, and Trust (PST), 2025.

M. Erfan, P. Branco, G.-V. Jourdan, Comparing Macro and Micro Approaches for Detecting Phishing Where It Spreads, in 2025 22nd Annual International Conference on Privacy, Security and Trust (PST).

4.1 Introduction

Based on the limitations identified in Chapter 3, where we showed a growing part of phishing activity was hosted on legitimate third-party platforms, not on attackers' domains, this chapter moves the detection point from malicious domain-level indicators to the platforms attackers use to distribute phishing content. Instant messaging and social media

applications, such as Telegram and WhatsApp, have become popular channels for attackers due to their widespread use and the ease of quickly sharing content [6, 87]. Our case study focuses on Telegram because, unlike more closed platforms such as WhatsApp, it provides extensive public groups and channels that can be accessed for large-scale data collection, making it well-suited for empirical analysis of phishing propagation. It is a widely used messaging platform that hosts numerous large public groups, including those notably prone to phishing, such as Cryptocurrency groups [26,102], Games groups [67], and Darknet discussions [11]. We examine how phishing threats can be detected proactively by investigating how they spread rather than just focusing on the website (or domain) involved. Specifically, we track group interactions, messaging frequencies, user engagement, and other social cues that attackers exploit to distribute phishing links at scale.

Notably, our approach is proactive because it can identify malicious intent at the point of transmission. While reactive systems depend on updated blocklists or user reports – often arriving after a campaign has already claimed its victims – our method leverages signals that are available when a message is posted. By capturing these early propagational indicators, we can flag threats, effectively shifting detection to the very beginning of the attack lifecycle.

In addition to focusing on these data, we explore the impact of model granularity on detection performance, ranging from a more general (macro) approach to a more specialized (micro) approach. Namely, we test four modeling schemes for phishing detection, each reflecting a different level of granularity: (i) a general model that trains a single classifier on all aggregated data from Telegram, (ii) general-cluster models that apply unsupervised clustering before classification, (iii) category-specific models tailored to each high-risk group type, and (iv) category-cluster models that combine category specialization with intra-category clustering. This progression from broad, platform-wide to targeted, category-specific modeling helps identify the best detection strategy.

Contributions: Our main contributions in this chapter include:

1. TelePhish Dataset: We introduce one of the first large-scale datasets of Telegram messages explicitly designed for phishing detection. Compiled from public groups in three high-risk categories (Cryptocurrency, Games, and Darknet), the dataset captures real-world conversations where phishing links frequently appear.
2. Systematic Evaluation of Modeling Strategies: We empirically investigate four modeling strategies, ranging from a generalized classifier trained on all data to category-specialized models, to assess how granularity impacts phishing detection effectiveness.
3. Key Insights into Phishing Behavior: Our experimental results show that the performance of phishing detection varies significantly between categories. Category-specialized models outperform generalized approaches by better capturing category-specific phish-

ing propagation and user behavior patterns.

Organization: This chapter is structured as follows. Section 4.2 surveys current phishing detection methods in social media. Section 4.3 describes the creation of the *TelePhish* dataset. Section 4.4 outlines the details of the proposed detection methodology. Section 4.5 presents our experimental setup as well as results and key findings, and limitations. Finally, Section 4.6 concludes the chapter.

4.2 Related Work

Phishing detection on social media has evolved as attackers increasingly exploit these platforms to distribute malicious links. Recent research has explored various approaches to detecting phishing on social media platforms, which we review in this section. High-quality datasets are essential to address the problem of phishing detection on social media. Given that datasets are a key component of our research, we also review the main datasets previously developed for phishing detection on social media.

4.2.1 Phishing Detection on Social Platforms

Phishing detection on social media has been explored from several perspectives, focusing on URL structures, textual content, and user accounts, or graph-level patterns. Early research relied heavily on URLs, while more recent studies have applied transformers and LLMs to textual content. In addition to these, a few metadata-driven detection and community-based anomaly analysis have offered complementary perspectives. Overall, the literature shows a broad set of strategies, but also has gaps in how phishing propagation is analyzed, particularly in social messaging apps like Telegram.

URL-Based Detection Approaches: Assuming that attackers rely on malicious domains to host phishing content, many detection approaches focus on URL analysis. Kustanto et al. [52] argued that many phishing URLs have formulaic patterns that enable transformers to detect them, so they proposed a transformer-based approach using only URL tokens. Their dataset contained phishing and legitimate URLs, pre-processed to keep only the scheme and domain. Their method could capture structural and semantic regularities in URLs by tokenizing and inputting them into a DistilBERT [90] transformer.

Building on the same assumption, Rashid et al. [80] developed a detection method for Twitter. They collected a dataset of tweets containing URLs and extracted URL features, including length, number of dots, digits in the host, path-to-URL ratio, SSL status,

and WHOIS-derived attributes like domain age, registrar, and Alexa rank. While both Rashid et al. and Kustanto et al. achieved high accuracy, their reliance on URLs makes their approach weak to URL-shortener, third-party platforms, compromised domains, and redirection chains commonly used by attackers.

Text-Based Detection Approaches: Beyond URLs, many works have emphasized the role of textual content. Modadugu et al. [45] applied Extreme Learning Machines (ELM) classifier for spammer detection on Twitter. They included linguistic patterns (tweet text, presence of URLs) alongside user-specific attributes (account age, follower-following ratio, activity rate) to detect spammers. Similarly, Podorozhniak et al. [78] worked on textual contents. They focused on comment filtering on Telegram, targeting not just spam, but also propaganda and misinformation. They preprocessed text through stop-word removal and vectorization and applied ensemble learning with diverse classifiers to filter unwanted messages in Telegram.

More recently, Oh et al. [74] addressed the data scarcity of messenger phishing detection, particularly for languages like Korean. They built a new messenger phishing dataset by collecting and annotating real cases, and then augmented it with smishing (Short Message Service (SMS) phishing) and voice phishing data. Their experiments showed that leveraging heterogeneous phishing sources improved performance, but they still focused on textual cues (e.g., suspicious words).

Transformer-based models have also been explored for phishing detection using text. Al Daoud et al. [7] evaluated phishing across Arabic social media posts, using datasets of Arabic social media posts. They compared four strategies: zero-shot LLMs (GPT-4o, Claude-3, Gemini-1.5), transformers as feature extractors with Random Forest, fine-tuned small language models, and ensembles. Results showed that zero-shot LLMs performed best. Chang and Aimeur [21] extended the application of LLMs to messaging applications such as WhatsApp, Telegram, Messenger, and Discord. They gathered a dataset of real-world scam and non-scam chat screenshots shared on X, Reddit, and Facebook tested five LLMs against human participants. Their results again showed that LLMs can operate effectively in chat contexts.

Ponnuchamy and Subbulakshmi [77] focused on spam detection in online social networks, using a primarily text-based approach. After preprocessing messages through tokenization, stop-word removal, and stemming, they extracted Bag-of-Words and Term Frequency-Inverse Document Frequency (TF–IDF) representations, along with word and character frequency features. These text-derived vectors were then classified by examining multiple ML classifiers. The study illustrated that content-driven features can provide high detection performance in online social networks, but such reliance on message text leaves models vulnerable to evasion through paraphrasing or text manipulation.

Behavior-Based Detection Approaches In contrast to URL and text-centric methods, a few works highlighted metadata and structural patterns as signals for phishing detection. Sun et al. [94] proposed a near real-time spam detection framework for Twitter. Instead of relying on keywords or textual analysis, they relied on account-level features (e.g., follower/following counts) and statistical message indicators (hashtags, mentions, message length). Evaluating multiple classifiers, their system demonstrated the practicality of a lightweight detection method for Twitter.

Gong and Karabiyik [40] also highlighted the value of platform metadata in their forensic analysis of the Tencent QQ messaging application. They showed how artifacts such as payment logs, file transfers, group chat records, and message withdrawals could be extracted to reveal fraud activity. Although not a proactive phishing detection system, their study underscores how metadata can uncover malicious operations without relying on textual features.

Other research has expanded from individual metadata to community and graph-level modeling. Soo et al. [93] approached phishing as a community-level spreading process rather than a message-level or URL-level classification problem. Using a Facebook graph dataset, they applied the Label Propagation Algorithm (LPA) [37] to detect communities and then used eigenvector centrality [39] to identify the most influential node in each anomalous community. A trapping model was then used to simulate how phishing influence expands as these communities lure new members over time. Their method improved the detection of phishing growth—that is, the expansion of simulated phishing communities. While this work highlights the structural dynamics of phishing growth at the community level, it does not perform phishing detection of individual messages or accounts, limiting its applicability for real-time defense in social platforms like Telegram.

Prior research on phishing detection has progressed in three main directions: URL-based analysis, text and content modeling, and behavior-based detection. URL-focused systems use URL and registration features to flag malicious domains, but they are weak against common evasion tactics such as shorteners, legitimate domains, and redirection chains. Text-based methods, including transformer and LLM-driven approaches may achieve high performance by identifying linguistic and social engineering patterns; however, they are computationally costly and remain vulnerable to paraphrasing and adversarial manipulation. Behavior-based approaches move away from content by analyzing account metadata, user activity patterns, or network communities, but these efforts either remain at the account level (e.g., detecting spammers) or are offline studies of phishing community growth, rather than supporting real-time detection at the message level. In messaging platforms such as Telegram, phishing campaigns often do not rely on sophisticated text or URLs but on propagation dynamics: repeated forwards, identical reposts across users, and coordin-

ated link distribution in groups. These patterns are not well captured by prior work. To address this gap, our study introduces a lightweight message-level framework that combines user metadata, message presentation cues, and engagement features reflecting propagation behavior. By focusing on how phishing messages spread rather than their textual or domain content, our approach offers greater resilience to manipulation while remaining scalable for high-volume, dynamic environments like Telegram.

4.2.2 Public Datasets for Social Media-Based Detection

Since one of our key goals is to release a phishing-focused dataset from Telegram, we highlight existing Telegram datasets that, while valuable, do not fully meet the needs of phishing detection and propagation analysis.

Pushshift Telegram Dataset [12] contains 317 million messages. It was compiled for large-scale studies of misinformation. Although substantial, it does not include specific phishing annotations. TGDataset [70] is even larger, with over 400 million messages. However, TGDataset similarly does not incorporate labels for phishing messages and does not provide insight into which messages specifically represent phishing attempts. This absence prevents training or benchmarking of phishing-focused detection models. DarkGram [83] offers a smaller but more specialized snapshot of 339 Telegram "cybercriminal-activity" channels. It includes posts that distribute compromised credentials, pirated software, and other blackhat resources, with some fraction of links containing phishing campaigns. Despite labeling certain malicious posts, DarkGram's emphasis is on hacking-themed channels rather than analyzing phishing propagation across Telegram communities.

Our work closes these gaps by providing a fully annotated dataset – capturing both raw Telegram messages and phishing labels – to support detection and propagation analysis in a social-media environment. We focus specifically on phishing link distribution in Telegram communities, enabling the first study of how phishing content spreads in a large-scale messaging platform.

4.3 TelePhish Dataset

In this section, we present the methodology used to construct *TelePhish*, a dataset designed to facilitate phishing detection within Telegram. Unlike prior approaches that focus on message content or URLs, we emphasize propagation-based patterns, as attackers increasingly use obfuscated text and legitimate domains to evade text or content-based filters.

Our goal was to collect, process, and label messages from Telegram, focusing on high-risk public groups where phishing activity is more likely to occur. The dataset captured real-world phishing propagation patterns. Below, we detail the steps in data collection, feature extraction, labeling, and handling missing data.

4.3.1 Data Collection

To construct the *TelePhish* dataset, we collected messages from public Telegram groups over four months, from October 2024 to January 2025. To select which groups to crawl, we identified three categories (Cryptocurrency, Games, and Darknet) as high-risk based on prior research [11, 26, 67], known scamming tactics, and the vulnerabilities described below. Then, we leveraged TGStat [95], a service that categorizes and ranks groups based on engagement metrics, to locate and rank the top active groups in these chosen categories. Specifically, we selected the following three high-risk categories from TGStat's taxonomy:

- Cryptocurrency: Telegram is widely used for cryptocurrency discussions, including investment groups, token launches, and exchange support communities. Attackers exploit these spaces by impersonating official accounts, promoting fraudulent airdrops, and distributing phishing links to fake wallet services. Phishing scams frequently target cryptocurrency users through fake embedded wallet services and fraudulent transactions [26, 102].
- Games: Attackers frequently promote game scams (such as fake giveaways, free in-game currency, or cheat tools) that prompt users to provide credentials or install malicious software, often resulting in stolen accounts or financial loss. Games communities are a major target for phishing, exploiting users looking for in-game benefits [67].
- Darknet: Telegram hosts various communities linked to underground markets and illegal services, where attackers distribute phishing links disguised as access points to hidden marketplaces. Phishing is widespread even within darknet ecosystems [11], suggesting that Telegram, as a hub for darknet discussions, can also facilitate the spread of phishing links.

After identifying the three target categories, we compiled a list of the top 100 public groups from each category based on active user participation metrics provided by TGStat. We ranked groups by active participants rather than total membership to ensure the dataset reflected live and ongoing discussions. However, because there was a short delay between retrieving the list of groups and the actual data collection, and given Telegram's highly dynamic ecosystem, some groups had changed status during that time – becoming private, deleted, or renamed – making them inaccessible. As a result, we were able to collect data successfully from 204 public groups, forming the foundation of the *TelePhish* dataset.

Figure 4.1 summarizes the distributions of participants, messages, and MAU across these categories. In this context, participants refer to all members of a group, while MAU denotes the number of unique members who posted at least one message within a month. Cryptocurrency groups show broader participation and engagement, while Games and Darknet groups are generally smaller and more concentrated around lower activity levels.

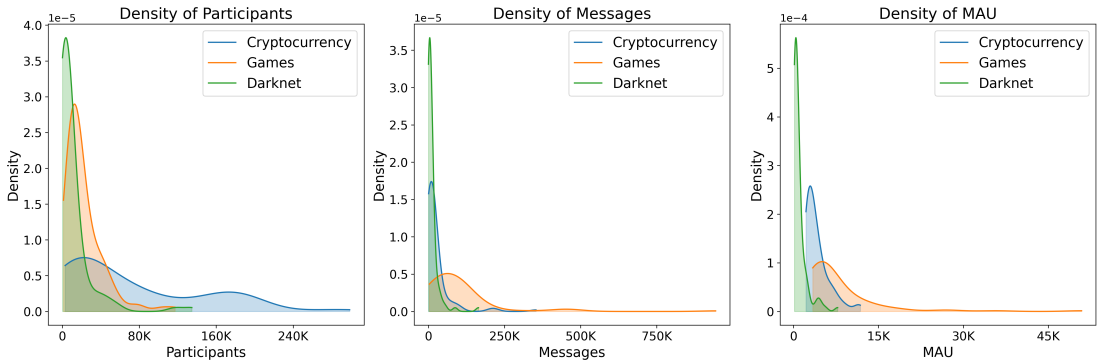


Figure 4.1: Density plots of the number of participants, messages, and MAU across three group categories: Cryptocurrency, Games, and Darknet.

4.3.2 Feature Extraction

Beyond storing raw messages, we enriched the dataset by extracting features that capture social dynamics and message characteristics. These features were designed to focus more on propagation patterns rather than content analysis, ensuring robustness against content-based evasion tactics. The extracted features fall into three main categories: *user-level*, *message-level*, and *engagement* attributes. These features were extracted for each of the messages included in the dataset.

We obtained raw group messages using Telegram's *Export Chat History* covering the period from October 2024 to January 2025. To focus on phishing attempts involving URLs, we filtered out messages that did not contain links, resulting in 143,597 messages. To supplement these exported data, we used *Telethon* [35], an open-source Python wrapper of the official Telegram APIs, which provides additional metadata – such as bot indicators, the earliest messages in a group, and user profile details – allowing us to build a more complete view of each message.

User-Level Features: User-level features describe the sender's account attributes, offering insights into suspicious behavior:

(1) *Bot Account Indicator:* We check whether Telegram designates the user as a bot. Automated bots can rapidly distribute phishing links, though some bots serve legitimate functions.
(2) *Username-Group Language Consistency:* To detect suspicious accounts, we analyze whether a user's username aligns linguistically with the dominant language of the group. For users with a username, we compare it against dictionaries in the group's dominant language (e.g., Russian) and English to determine if it forms a semantically meaningful word or phrase. Usernames containing only random character strings (e.g., x7q3z) or those mismatched with the group's language or the English language are flagged as potentially automated or temporary and are assigned a value of *False*.
Notably, we do not remove or exclude messages from users who do not have a registered username. In these cases, the absence of a username is also assigned a value of *False* within this boolean feature. It is important to emphasize that this feature, like all others in our methodology, is not used in isolation to label a message as phishing. Instead, it serves as one of several behavioral signals provided to the machine learning model.

Message-Level Features: Message-level features reflect how links and text are presented, highlighting potential phishing patterns:

(1) *Message Length:* Measures the number of characters in the message. Extremely short or overly long posts may indicate phishing attempts.
(2) *Number of Unique Links:* Phishers sometimes include multiple URLs to maximize clicks or switch between different landing pages.
(3) *Forwarded Message Indicator:* Telegram flags forwarded content. Attackers often forward the same phishing link to several groups, suggesting coordinated spam activity.
(4) *Reply Message Indicator:* Whether the message replies to an existing thread. Phishers may insert links into ongoing discussions to appear more credible.
(5) *Formatted Text Count:* Counts the number of bold, italic, or emoji markers, as phishers commonly use attention-grabbing formats to lure clicks.
(6) *Language Consistency:* We infer the primary language of each group by sampling recent posts and then comparing the language detected in each message with that baseline. Mismatches can signal automated or spam-like posting.
(7) *Text Content Presence:* Indicates whether a message contains additional text beyond the URL(s). Some campaigns rely on repeated link-only messages.
(8) *Media Attachments:* Flags whether images, videos, or documents are included. Malicious actors may embed phishing links within media descriptions or file previews.

Engagement Features: Engagement features capture a user's activity level and history within each group:

(1) *Account Activity Duration:* Telegram does not provide direct information on when a user joined a group. Instead, we approximate the group's creation time by its earliest archived message and then record the user's first post. The difference (normalized by the group's age) provides a measure of how long the user has been active. Sudden, high-volume posting from a recently active account can indicate opportunistic phishing.
(2) *Message Repetition:* Counts how many times a user reposts the same message within the data collection window. Repeated identical posts can signify spam-like or automated phishing tactics.
(3) *Cross-User Message Repetition:* This metric quantifies how many distinct users shared the exact same message within the same group and date window. A high value suggests coordinated distribution of content, a sign of phishing campaigns where attackers use multiple accounts to amplify phishing links propagation.
(4) *Total Messages:* We record the total number of messages a user posted in the group on the same date they shared a message containing a link. This feature measures the user's engagement level in group discussions on that specific date, helping to distinguish active participants from users who primarily post links with little interaction.

By emphasizing these structural and behavioral indicators, *TelePhish* focuses on how malicious links propagate and how users engage with groups, rather than relying solely on text or domain-based heuristics. This design choice offers increased resilience against phishing campaigns that exploit legitimate domains or employ sophisticated content obfuscation.

4.3.3 Handling Missing Values and Duplicates

To ensure the reliability of the *TelePhish* dataset, we implemented a two-stage preprocessing pipeline to address missing values and duplicates. The primary source of missing data came from Telegram's API limitations, particularly for features such as account activity duration. Since Telegram does not provide direct access to group creation timestamps or user join dates, we inferred group creation dates from the earliest archived message and derived user activity duration from their first observed post. In cases where these approximations were infeasible (e.g., deleted messages or API restrictions), samples with incomplete essential features were excluded. This step removed 1,017 messages from further processing. The secondary source of missing values came from the optional usernames in Telegram. Rather than treating missing usernames as incomplete data, we explicitly encoded their absence as a distinct category in the username-group language consistency feature. This design choice reflects our assumption that phishing operators frequently omit or randomize usernames to evade detection, making anonymity itself a meaningful signal

of suspicious accounts.

In the next step, to mitigate redundancy, we addressed duplicate messages originating from repeated posting of identical content. A duplicate was defined as any instance where the same user shared the exact message within the same group. We kept only the first occurrence of such duplicates to avoid overcounting spam-like behavior. However, repetition metrics – such as message repetition (the count of identical messages shared by a user in a group on the same date) and cross-user message repetition (the number of distinct users sharing identical messages in a group-date window) – were preserved as features. After computing these metrics, duplicate messages were removed. This process eliminated 84,371 messages. This strategy balances two objectives: (1) ensuring uniqueness in the dataset to prevent skewed statistical analyzes and (2) keeping behavioral signals that capture coordinated or automated phishing activity. Following these preprocessing steps, the final dataset contains 58,209 messages.

4.3.4 Labeling and Ground Truth

To establish ground truth labels for phishing detection, we leveraged VirusTotal's URL analysis API [101], a widely trusted threat intelligence platform. VirusTotal aggregates scans from 70+ commercial antivirus engines, providing a multi-source assessment on URL maliciousness. VirusTotal does not distinguish scams from phishing URLs; therefore, in this study, any message containing phishing or scam links is labeled as *phishing*. This choice is justified by our focus on propagation behavior; from a defensive standpoint, the mechanism of delivery – a deceptive message designed to elicit a click – is the primary signal our model seeks to detect. Whether the ultimate goal of the URL is to harvest credentials or facilitate a crypto scam, the observable behavioral patterns on the platform remain consistent. While a technical distinction can be made between phishing (typically credential harvesting) and scams (fraudulent schemes designed to deceive victims into financial loss, the disclosure of personal information, or the performance of unwanted tasks), in the context of high-risk Telegram communities, these threats are operationally identical as both rely on deceptive social engineering lures to direct users toward malicious URLs. Consequently, we treat them as a single threat category for the purposes of this behavioral study.

The URLs are only used to verify whether messages are phishing (or scam) or benign for the dataset ground truth, not as a standalone feature for our detection methods. This ensures consistent labeling while acknowledging that true phishing detection must include more than just URL analysis. For each message containing links, we submitted all extracted links to VirusTotal's API. A message was labeled as *phishing* if at least one vendor

flagged any of the URLs in the message as phishing. This conservative threshold prioritizes detection sensitivity, as phishing campaigns often target victims before security tools update their blocklists. Table 4.1 presents the dataset summary.

Table 4.1: Overview of messages collected from each Telegram Category (October 2024 - January 2025)

Group Category	**Messages**	**Phishing**	**Benign**
Cryptocurrency	32,584	216	32,368
Games	22,350	1,184	21,166
Darknet	3,275	84	3,191
Total	58,209	1,484	56,725

4.4 Phishing Detection Methodology: From Macro to Micro

With the *TelePhish* dataset—designed to capture phishing propagation patterns—we investigate which modeling approach proves more effective: general models that operate across all contexts, or specialized models tailored to specific Telegram group categories. To explore this, we adopt a "macro-to-micro" experimental design, gradually narrowing the modeling scope from the most general to the most localized. Figure 4.2 illustrates the four strategies we evaluate. We begin with a general model (Model 1 in the figure) trained on all available data across categories. Next, we apply clustering to the full dataset and train a separate model for each cluster (Model 2 in the figure). We then build category models, training separate classifiers for each group category to capture distinct behavioral patterns (Model 3 in the figure). Finally, we apply clustering within each category to develop category-cluster models (Model 4 in the figure). This allows us to systematically assess how varying granularity and levels of specialization affect phishing detection performance. The four modeling strategies are detailed as follows:

General Model: In the first stage, we develop a general model, training a single classifier on an aggregated dataset that combines messages from all target Telegram groups (Cryptocurrency, Games, and Darknet). This general model aims to capture broad phishing characteristics potentially common across diverse Telegram environments.

General-Cluster Model: To further investigate the classifier trained on all Telegram messages, we introduced a cluster-based extension to the general model. While the standard

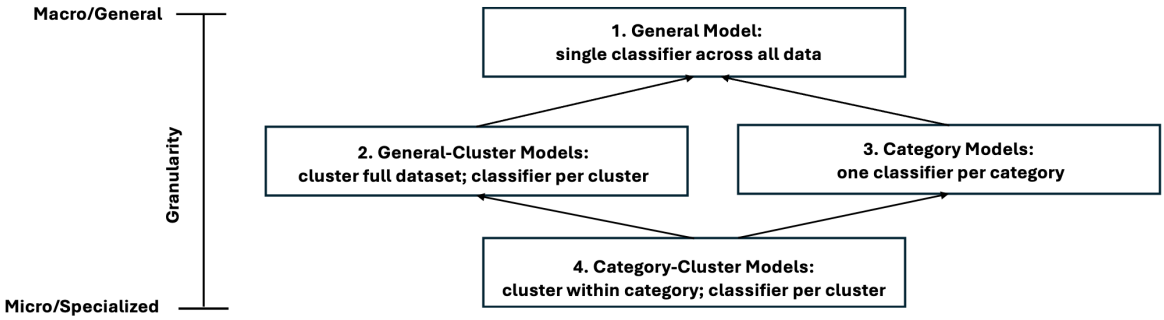


Figure 4.2: Macro-to-Micro phishing detection strategies: including general, general-cluster, category, and category-cluster models.

general model aggregates all messages from high-risk categories and trains one classifier, this alternative approach applies clustering to the entire dataset before training any model. The intuition is that phishing behaviors – even across different group categories – may form naturally separable subgroups, each requiring a specialized model. We used the K-means (with $k \in \{2, 3\}$) to cluster the entire dataset, then within each cluster, a separate classifier was trained.

Category Models: The third stage introduces category models to account for variations in phishing behaviors across different group categories. We train separate classifiers using data from each targeted high-risk category, allowing the models to specialize by learning the distinct patterns and behaviors characteristic of each category.

Category-Cluster Models: In the final stage, we apply K-means clustering to segment messages within each category into distinct clusters (using $k \in \{2, 3\}$) and then train dedicated classifiers for each cluster. This category-cluster approach allows us to identify sub-patterns, capturing finer-grained phishing behaviors within categories.

It is worth mentioning that clustering, which is used in two of the proposed modeling strategies, provides a finer-grained understanding of phishing behaviors, but also has certain assumptions and limitations. The use of K-means clustering assumes roughly spherical, equally sized clusters with similar variance, which may not reflect the true structure of Telegram messages. As a result, some patterns might be oversimplified or split across clusters. Moreover, the three Telegram categories in the *TelePhish* dataset vary significantly in size – Cryptocurrency and Games groups are substantially larger and more active than Darknet groups – which can affect clustering and model generalization. This means that clusters derived from the smaller category may be less representative or more sensitive to noise, resulting in less reliable models for that category compared to the larger ones.

4.5 Experimental Evaluation

4.5.1 Experimental Settings

Evaluation Metrics: To assess the performance of each modeling strategy defined in the previous section, we used standard evaluation metrics: Precision, Recall, Specificity, F1-score, G-Mean, and MCC. These metrics were described earlier in Chapter 3, Section 3.3.1.

Learning Algorithms: To classify Telegram messages in the *TelePhish* dataset into phishing and benign, we evaluated several classifiers representing different machine learning paradigms: Logistic Regression (linear method), Support Vector Machines (nonlinear method), Naive Bayes (probabilistic method), Decision Trees, Random Forest, and XGBoost (tree-based methods), and K-Nearest Neighbors [29] (distance-based method). These algorithms were chosen for their varied classification mechanisms, allowing us to assess their suitability for phishing detection.

Hyperparameter Tuning and Performance Evaluation: To ensure accurate hyperparameter selection and unbiased performance assessment, we employed a nested stratified 5-fold cross-validation strategy as introduced in Chapter 3, Section 3.3.3. This approach prevents overly optimistic evaluations by ensuring that hyperparameter tuning and model assessment are performed on distinct data partitions. All performance results reported in this chapter were obtained using this methodology.

Feature Selection: To focus on the most effective features for phishing detection, we conducted a feature selection step from the extracted features. We used a Recursive Feature Elimination with Cross-Validation (RFECV) approach. We started with all the features, we trained a Random Forest classifier and iteratively removed the least informative ones. At each step, the feature subset was evaluated using classification metrics, ensuring that only features that contributed meaningfully to classification performance were retained. We finalized a set of 12 features when further removal no longer improved classification performance. The selected feature set includes two user-level features (*Bot Account Indicator*, *Username-Group Language Consistency*), seven message-level features (*Message Length*, *Reply Message Indicator*, *Number of Unique Links*, *Formatted Text Count*, *Language Consistency*, *Media Attachments*, *Unique Users Reposting Identical Message*), and three engagement features (*Account Activity Duration*, *Message Repetition*, *Total Messages*). We consistently used this final subset in all subsequent experiments to ensure that any performance differences resulted solely from modeling strategies rather than input variations.

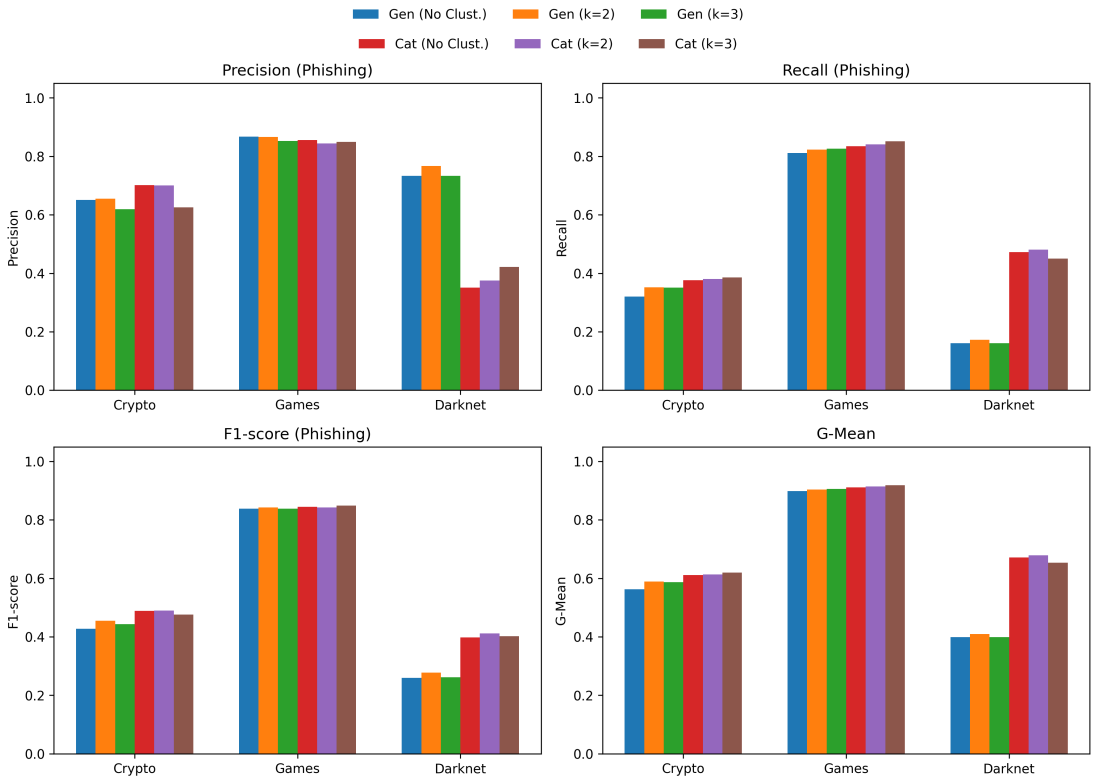


Figure 4.3: Comparison of General Model vs. General-Cluster Model (k=2, k=3) vs. Category Model vs. Category-Cluster Model (k=2, k=3) across Crypto, Games, and Darknet categories for the Phishing class.

4.5.2 Results and Discussion

Figure 4.3 and Table 4.2 present the detailed results of our macro-to-micro experiments using the Random Forest classifier. We focus on RF because it consistently outperformed other classifiers across all four modeling strategies (general, general-cluster, category, and category-cluster), as introduced in Section 4.4. The visual summary in Figure 4.3 shows model performance across four evaluation metrics – Precision, Recall, F1-score, and G-Mean – for each category: Cryptocurrency, Games, and Darknet. We identify three key patterns:

(1) Category matters. In the Crypto and Darknet groups, the red bars (category models) consistently outperform the blue bars (general models) across Recall, F1-score, and G-Mean, indicating that phishing behavior is highly specific to each of these categories.

The most notable improvement belongs to the Darknet category.

(2) The Games category is easier to model. All six model variants perform strongly on Games, with F1-scores near 0.85 and G-Mean nearly above 0.9, suggesting more consistent phishing behaviors in this category.

(3) Clustering provides minimal improvement. The orange bar (general-cluster model, $k = 2$) shows only a slight gain over the blue bar (general model), and within-category clustering (purple vs. red bars) offers partial benefit.

Table 4.2: Performance of the four macro-to-micro modeling strategies (mean ± std across outer CV folds).

Model	Category	Precision		Recall		F1-score		G-Mean
		Benign	Phish	Benign	Phish	Benign	Phish	
1. General Model	Crypto	$0.995_{\pm 0.000}$	$0.651_{\pm 0.106}$	$0.999_{\pm 0.000}$	$0.320_{\pm 0.063}$	$0.997_{\pm 0.000}$	$0.427_{\pm 0.072}$	$0.563_{\pm 0.055}$
	Games	$0.992_{\pm 0.002}$	$0.867_{\pm 0.026}$	$0.995_{\pm 0.000}$	$0.811_{\pm 0.037}$	$0.994_{\pm 0.001}$	$0.838_{\pm 0.024}$	$0.898_{\pm 0.021}$
	Darknet	$0.978_{\pm 0.005}$	$0.733_{\pm 0.226}$	$0.998_{\pm 0.003}$	$0.161_{\pm 0.036}$	$0.988_{\pm 0.004}$	$0.260_{\pm 0.050}$	$0.399_{\pm 0.045}$
2. General-Cluster Models	Crypto	$0.995_{\pm 0.001}$	$0.655_{\pm 0.099}$	$0.999_{\pm 0.000}$	$0.352_{\pm 0.085}$	$0.997_{\pm 0.000}$	$0.455_{\pm 0.089}$	$0.589_{\pm 0.072}$
	Games	$0.993_{\pm 0.002}$	$0.866_{\pm 0.031}$	$0.995_{\pm 0.001}$	$0.823_{\pm 0.041}$	$0.994_{\pm 0.001}$	$0.843_{\pm 0.026}$	$0.904_{\pm 0.023}$
	Darknet	$0.978_{\pm 0.005}$	$0.767_{\pm 0.226}$	$0.998_{\pm 0.002}$	$0.172_{\pm 0.052}$	$0.988_{\pm 0.003}$	$0.278_{\pm 0.077}$	$0.410_{\pm 0.062}$
3. Category Models	Crypto	$0.995_{\pm 0.000}$	$0.701_{\pm 0.066}$	$0.999_{\pm 0.000}$	$0.376_{\pm 0.063}$	$0.997_{\pm 0.000}$	$0.489_{\pm 0.069}$	$0.611_{\pm 0.052}$
	Games	$0.993_{\pm 0.002}$	$0.856_{\pm 0.030}$	$0.994_{\pm 0.001}$	$0.835_{\pm 0.046}$	$0.994_{\pm 0.001}$	$0.845_{\pm 0.028}$	$0.911_{\pm 0.025}$
	Darknet	$0.986_{\pm 0.003}$	$0.351_{\pm 0.079}$	$0.976_{\pm 0.011}$	$0.472_{\pm 0.138}$	$0.981_{\pm 0.006}$	$0.398_{\pm 0.096}$	$0.671_{\pm 0.099}$
4. Category-Cluster Models	Crypto	$0.995_{\pm 0.000}$	$0.700_{\pm 0.073}$	$0.999_{\pm 0.000}$	$0.380_{\pm 0.070}$	$0.997_{\pm 0.000}$	$0.490_{\pm 0.073}$	$0.613_{\pm 0.058}$
	Games	$0.994_{\pm 0.001}$	$0.844_{\pm 0.037}$	$0.994_{\pm 0.002}$	$0.841_{\pm 0.023}$	$0.994_{\pm 0.001}$	$0.842_{\pm 0.025}$	$0.914_{\pm 0.013}$
	Darknet	$0.986_{\pm 0.005}$	$0.375_{\pm 0.096}$	$0.978_{\pm 0.007}$	$0.481_{\pm 0.132}$	$0.982_{\pm 0.004}$	$0.412_{\pm 0.087}$	$0.679_{\pm 0.095}$

Note: Cluster-based models were evaluated with $k = 2$ and $k = 3$, but only results for $k = 2$ are shown here to simplify presentation.

Table 4.2 complements the plot by providing exact values for each metric. Together, the figure and table show that specialized models, particularly category models, are more effective than the general model in detecting phishing across Telegram groups.

The remainder of this section presents the results of each modeling strategy, highlighting their performance strengths and trade-offs across categories.

General Model Results: We evaluated the seven classifiers introduced in Section 4.5.1 using the entire *TelePhish* dataset. Table 4.3 shows that Random Forest outperformed other approaches in both F1-score and G-Mean, indicating a stronger ability to minimize false positives and false negatives[1]. We, therefore, focused on Random Forest as our general model and observed its performance separately on each category (Cryptocurrency, Games, and Darknet). As summarized in Table 4.2, the general model achieved strong phishing

[1]We also observed the same performance for the other model granularities, i.e., Random Forest was the best performing model across the considered granularities.

detection in the Games category but struggled with Cryptocurrency and Darknet. The lower Recall in those latter categories (only around 0.32 for Cryptocurrency and 0.16 for Darknet) suggests that the general model missed many phishing instances in these domains.

Table 4.3: Performance results of general classifiers. (Positive class for metrics calculation is indicated in the first column.)

(a) Average Precision with Standard Deviation of Classifiers Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	KNN
Benign	0.98 ± 0.000	**0.993 ± 0.000**	0.982 ± 0.001	0.992 ± 0.001	**0.993 ± 0.001**	0.992 ± 0.000	**0.993 ± 0.000**
Phishing	0.506 ± 0.156	0.056 ± 0.003	0.163 ± 0.040	0.699 ± 0.016	0.833 ± 0.018	**0.869 ± 0.013**	0.803 ± 0.012

(b) Average Recall with Standard Deviation of Classifiers Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	KNN
Benign	**1.00 ± 0.000**	0.741 ± 0.013	0.989 ± 0.002	0.995 ± 0.000	0.997 ± 0.000	0.998 ± 0.000	0.997 ± 0.000
Phishing	0.013 ± 0.006	**0.747 ± 0.008**	0.108 ± 0.025	0.597 ± 0.046	0.659 ± 0.026	0.621 ± 0.014	0.632 ± 0.020

(c) Average F1-score with Standard Deviation of Classifiers Across 5-Fold CV

Class	LR	SVM	NB	DT	RF	XGB	KNN
Benign	0.990 ± 0.000	0.849 ± 0.009	0.985 ± 0.001	0.993 ± 0.000	**0.995 ± 0.000**	**0.995 ± 0.000**	**0.995 ± 0.000**
Phishing	0.025 ± 0.011	0.104 ± 0.006	0.130 ± 0.030	0.643 ± 0.031	**0.735 ± 0.019**	0.724 ± 0.013	0.707 ± 0.011

(d) G-Mean Across 5-Fold CV

	LR	SVM	NB	DT	RF	XGB	KNN
G-Mean	0.111 ± 0.027	0.744 ± 0.010	0.325 ± 0.039	0.770 ± 0.029	**0.810 ± 0.016**	0.787 ± 0.009	0.794 ± 0.013

Abbreviations: LR = Logistic Regression, SVM = Support Vector Machine, NB = Naive Bayes, DT = Decision Tree, RF = Random Forest, XGB = XGBoost, and KNN = K-Nearest Neighbors. The best values for each class are highlighted in **bold**.

General-Cluster Models Results: To investigate whether clustering improves the detection in the general approach, we used K-means $k = \{2, 3\}$ to segment the training fold during each outer fold of our nested cross-validation. Specifically, we fit K-means on the training data, assign each training sample to a cluster, and then train a separate classifier (Random Forest) per cluster using the same tuning/evaluation strategy from Section 4.5.1. At test time, each sample was first assigned to one of the previously fitted clusters and then classified by that cluster's model. We aggregated these fold-level predictions to derive overall metrics for each category. The results of clustering with two different values of k are shown in Table 4.4. While clustering led to slight performance gains, the improvements were not substantial enough to indicate a meaningful advantage over the baseline general model. This suggests that, in a general setting, unsupervised segmentation may not uncover distinct enough phishing patterns to significantly boost detection performance.

Table 4.4: Comparison of cluster-based models performance using $k = 2$ and $k = 3$ (mean $\pm$ std over 5-fold CV).

Model	Category (k)	Precision		Recall		F1-score		G-Mean
		Benign	Phish	Benign	Phish	Benign	Phish	
2. General-Cluster Models	Crypto (k=2)	0.995 $\pm$ 0.001	0.655 $\pm$ 0.099	0.999 $\pm$ 0.000	0.352 $\pm$ 0.085	0.997 $\pm$ 0.000	0.455 $\pm$ 0.089	0.589 $\pm$ 0.072
	Crypto (k=3)	0.995 $\pm$ 0.001	0.619 $\pm$ 0.064	0.998 $\pm$ 0.000	0.351 $\pm$ 0.095	0.997 $\pm$ 0.000	0.443 $\pm$ 0.086	0.587 $\pm$ 0.080
	Games (k=2)	0.993 $\pm$ 0.002	0.866 $\pm$ 0.031	0.995 $\pm$ 0.001	0.823 $\pm$ 0.041	0.994 $\pm$ 0.001	0.843 $\pm$ 0.026	0.904 $\pm$ 0.023
	Games (k=3)	0.993 $\pm$ 0.002	0.853 $\pm$ 0.032	0.994 $\pm$ 0.001	0.826 $\pm$ 0.039	0.994 $\pm$ 0.001	0.838 $\pm$ 0.026	0.906 $\pm$ 0.021
	Darknet (k=2)	0.978 $\pm$ 0.005	0.767 $\pm$ 0.226	0.998 $\pm$ 0.002	0.172 $\pm$ 0.052	0.988 $\pm$ 0.003	0.278 $\pm$ 0.077	0.410 $\pm$ 0.062
	Darknet (k=3)	0.978 $\pm$ 0.005	0.733 $\pm$ 0.133	0.998 $\pm$ 0.001	0.161 $\pm$ 0.036	0.988 $\pm$ 0.003	0.262 $\pm$ 0.050	0.399 $\pm$ 0.046
4. Category-Cluster Models	Crypto (k=2)	0.995 $\pm$ 0.000	0.700 $\pm$ 0.073	0.999 $\pm$ 0.000	0.380 $\pm$ 0.070	0.997 $\pm$ 0.000	0.490 $\pm$ 0.073	0.613 $\pm$ 0.058
	Crypto (k=3)	0.996 $\pm$ 0.000	0.626 $\pm$ 0.098	0.998 $\pm$ 0.001	0.386 $\pm$ 0.050	0.997 $\pm$ 0.000	0.476 $\pm$ 0.062	0.620 $\pm$ 0.040
	Games (k=2)	0.994 $\pm$ 0.001	0.844 $\pm$ 0.037	0.994 $\pm$ 0.002	0.841 $\pm$ 0.023	0.994 $\pm$ 0.001	0.842 $\pm$ 0.025	0.914 $\pm$ 0.013
	Games (k=3)	0.994 $\pm$ 0.001	0.849 $\pm$ 0.024	0.994 $\pm$ 0.001	0.851 $\pm$ 0.030	0.994 $\pm$ 0.001	0.849 $\pm$ 0.022	0.919 $\pm$ 0.016
	Darknet (k=2)	0.986 $\pm$ 0.005	0.375 $\pm$ 0.096	0.978 $\pm$ 0.007	0.481 $\pm$ 0.132	0.982 $\pm$ 0.004	0.412 $\pm$ 0.087	0.679 $\pm$ 0.095
	Darknet (k=3)	0.985 $\pm$ 0.005	0.422 $\pm$ 0.171	0.979 $\pm$ 0.013	0.450 $\pm$ 0.148	0.982 $\pm$ 0.006	0.402 $\pm$ 0.101	0.654 $\pm$ 0.105

Category Models Results: Although the general model trained across all categories performed well overall, it showed limitations in categories with sparse phishing samples, primarily due to significant class imbalance. Such an imbalance often prevents a generalized classifier from adequately capturing subtle, category-specific phishing behaviors. To explore whether category-specialized classifiers (Random Forest) handle the task better, we trained a separate classifier for each category. For each new message to be classified, we first determined its category and then assigned it to the corresponding classifier to obtain a prediction. Detailed performance results for this approach are provided in Table 4.2. Comparing the general model with the category models, the advantages of specialized classifiers can be noted as follows:

(1) Cryptocurrency: The specific model's Recall (Phishing) and Precision (Phishing) both improved comparably (Recall: from 0.32 to 0.376, Precision: from 0.651 to 0.701. As a result, the F1-score for phishing increased due to the balanced improvement in both metrics.

(2) Games: The general model already had a strong performance as a foundation. Table 4.2 confirms that category-specific training has further improvements, raising both F1-score (Phishing) and G-Mean for this category.

(3) Darknet: The largest difference is in Recall, jumping from 0.16 with the general model to 0.47 in the category model. Although Precision did not improve, the F1-score (Phishing) and G-Mean are higher, confirming that focusing on Darknet-only data helps the model specialize in the unique phishing patterns found there.

These findings suggest that the general model's performance limitations in some categories were likely due to an insufficient number of phishing examples. Training category models helps mitigate this by allowing the classifier to better focus on the unique charac-

teristics of phishing messages in each category.

Category-Cluster Models Results: The category-cluster models, which apply finer segmentation through K-means clustering, show improvements in phishing detection compared to the category models. By clustering the messages within each category, we can capture more subtle phishing patterns. Table 4.4 shows the category-cluster model performance with $k = \{2, 3\}$. The results can be summarized as follows:

(1) Cryptocurrency: Clustering slightly improves Recall compared to the category models (in Table 4.2), but at the cost of Precision. Specifically, F1-score for phishing is highest at $k = 2$, while G-Mean improves more at $k = 3$. The drop in Precision at higher values of k indicates that clustering introduces sub-groups where some benign messages are misclassified as phishing, likely due to phishing-like patterns within certain clusters.
(2) Games: Unlike Cryptocurrency and Darknet, the category model for Games already performed well (Recall ≈ 0.83, F1-score ≈ 0.84), leaving little room for improvement. Clustering still resulted in marginal improvements in G-Mean, but the phishing detection rate remained largely unchanged. This suggests that phishing in Games groups follows a more uniform pattern, making sub-cluster segmentation unnecessary.
(3) Darknet: Clustering had a slight improvement here as well, increasing F1-score at $k = 2$ from ≈ 0.39 in category model to ≈ 0.41 (based on the comparison in Table 4.2). This mild improvement implies that while Darknet phishing exhibits a more complex pattern, the advantage of finer segmentation is less significant than initially expected.

In real-world deployment, clustering may offer incremental benefits in certain categories, but it adds complexity in the model-building pipeline. If maximizing the recall of rare phishing attacks is the core objective, segmenting the data into clusters may be worthwhile. Conversely, if the goal is maintaining precision or a simpler architecture, category models may suffice, particularly for categories with uniform phishing behaviors, as observed in Games.

4.5.3 Improving Detection by Addressing Class Imbalance

A major challenge in our dataset is the imbalance between phishing and benign messages, which appeared in every category we examined. The problem was more severe in the Cryptocurrency and Darknet groups, where phishing attempts were only a small share of the total messages. Even in the Games category, which had relatively more phishing cases, benign messages still outnumbered phishing samples. Such an imbalance is expected in real-world social messaging platforms, where phishing is rare compared to the high volume of regular conversations. Such an imbalance poses a major challenge for detection. ML

classifiers usually focus on the majority class, which can result in models that miss minority phishing instances.

To address this, we experimented with a wide range of resampling techniques. Our approach included oversampling methods that generate synthetic phishing instances. We also tested hybrid strategies, where the data is oversampled to increase the representation of phishing cases and also undersampled to remove noise and borderline instances. The idea is that oversampling alone risks amplifying noise to the minority class. By combining it with undersampling, we aimed to increase the representation of phishing messages while also reducing noise among benign examples.

All imbalance handling was conducted in the category-specific setting, which we had identified as the most effective baseline in Section 4.5.2. In this setup, a separate classifier was trained for each category, and resampling was applied independently within the training folds of each category. This section presents the results of these experiments and highlights which resampling strategies most effectively improved phishing detection for category-specific setting.

Resampling Algorithms: For both the oversampling and undersampling sides, we applied multiple approaches to ensure that our analysis was not based on a single technique. For oversampling, we used Random Oversampling (ROS) [58] as a basic baseline, an extended version of ROS where Gaussian noise was added to duplicated samples to increase variability, and more advanced methods such as SMOTE [24] and its variants [51] (Cluster-SMOTE, G-SMOTE, NT-SMOTE, etc.). These methods differ in how they create new minority samples. For instance, ROS duplicates existing samples, Gaussian Distribution modifies them with noise, SMOTE interpolates between nearest neighbors, and the variants extend this idea by using clustering, geometric regions, or neighborhood constraints to generate more diverse synthetic examples. In addition, we explored hybrid methods that combine oversampling with undersampling methods such as Tomek Links [97] or Neighbourhood Cleaning Rule (NCL) [54] to remove noise. This set of algorithms allowed us to compare a wide range of resampling strategies.

Parameter Tuning for Resampling Strategies: To ensure fair comparison across methods, we did not select resampling parameters arbitrarily. Instead, we built a pipeline that systematically explored combinations of parameter values for each algorithm. Specifically, for oversampling, we experimented with different ratios of minority-to-majority classes (0.1, 0.3, 0.5, 1.0). Among these, the ratio of 0.1 consistently provided the best trade-off for the F1-scores of both classes. We also adjusted the number of nearest neighbors for SMOTE-based methods and NCL, and experimented with different cleaning parameters in hybrid approaches. This allowed us to evaluate the sensitivity of each method to its configuration and to identify stable settings that balanced minority recall with overall per-

formance. Additionally, for hybrid methods, we also varied the sequence of operations. In most cases, oversampling was applied first, and then undersampling was used to clean noisy or borderline examples. However, for methods involving ROS, we tested the reverse sequence, because applying cleaning techniques like Tomek Links after duplication can remove only one of a Tomek pair and leave its duplicate, which preserves borderline noise.

Results of Resampling on Category-Specific Detection: Table 4.5 summarizes the category-level performance of different resampling strategies across both benign and phishing classes. The results show that while several methods modestly improved phishing recalls, the NCL-ROS hybrid provided the most consistent improvements. Specifically, it reached a phishing recall of 0.523 in Cryptocurrency and 0.827 in Games, outperforming other methods that often struggled to improve recall beyond 0.4–0.5. NCL first removes borderline or mislabeled benign samples that obscure decision boundaries, and after that ROS increases the representation of phishing instances in a cleaner space. This sequence ensures that oversampling does not simply replicate noisy examples, but instead amplifies informative phishing patterns.

The benign class performance remained stable across all resampling methods. As seen in Table 4.5, precision, recall, and F1-scores for benign messages stayed consistently high (often more than 0.98), showing that resampling improved sensitivity to phishing without decreasing the classifier's ability to identify benign messages. This finding emphasizes the utility of hybrid resampling: it enhances minority-class detection while preserving performance on the majority class.

4.5.4 Limitations

Our analysis focused specifically on three high-risk Telegram categories – Cryptocurrency, Darknet, and Games – which were identified for frequent phishing activities. While this limited scope enabled us to do an in-depth investigation into these domains, it excludes other potentially relevant categories such as investment and trading. As a result, the findings may not fully generalize across the diverse range of phishing tactics present in other groups.

To handle the strong class imbalance in our dataset, we relied on hybrid resampling methods. Although resampling strategies such as NCL-ROS improved recall for phishing messages, these methods introduce synthetic samples into the training data. There is always a risk that classifiers learn these data rather than generalizable phishing behaviors, which could limit robustness in real-world deployments.

Another limitation comes from how we created the ground truth labels. We used the VirusTotal service to check the URLs, but these services may not flag new phishing links

immediately. This means that some links in our dataset might have been incorrectly labeled as benign at the time of collection. While this labeling approach is practical and scalable, it introduces a small amount of uncertainty.

Finally, the use of K-means clustering in two of our modeling strategies introduces additional assumptions. K-means assumes clusters to be roughly spherical, equally sized, and of similar variance, conditions that may not hold in our dataset. Furthermore, the imbalance between categories – particularly the smaller size of Darknet groups compared to Cryptocurrency and Games – could impact clustering so that clusters derived from the smaller category might be less representative or more sensitive to noise.

4.6 Conclusion

In this chapter, we presented *TelePhish*, a new dataset of annotated Telegram messages collected from public groups in high-risk categories such as Cryptocurrency, Games, and Darknet. *TelePhish* focuses on the propagational behavior of phishing by capturing Telegram group interactions, message patterns, and user engagement, enabling future research in phishing detection. We also introduced and evaluated four classification strategies for detecting phishing in Telegram groups: a general model trained on the entire dataset, a general-cluster variation that segments the full dataset and trains per-cluster models, and two category modeling approaches (with and without clustering) that focus on category-tailored phishing patterns.

Our results showed that category models outperform the general model, particularly in more complex categories such as Cryptocurrency and Darknet. Clustering within categories further improves recall by isolating phishing subgroups, although it can increase complexity. The greatest improvement resulted from category-specific models, highlighting their importance for phishing detection in heterogeneous platforms such as Telegram.

An important challenge in phishing detection in Telegram was the class imbalance between phishing and benign messages, especially in the Cryptocurrency and Darknet categories, where phishing was only a small fraction of the data. To address this, we systematically tested a range of resampling methods in this chapter. The results showed that resampling can improve the recall of phishing detection without decreasing performance on benign messages.

Taken together, these findings show that propagation-aware features, category-specific modeling, and carefully designed resampling strategies make phishing detection in Telegram effective. At the same time, they highlight open challenges, such as ensuring that models

generalize beyond synthetic resampling artifacts and extending the analysis to a broader range of categories.

In the next chapter, we directly compare LLM with the resampled, category-specific models developed here to observe whether LLMs can offer an advantage for phishing detection in messaging platforms like Telegram, or if lightweight, propagation-based models remain the stronger option.

Table 4.5: Category-Specific models performance per resampling method (](mean ± std over 5-fold CV). Best phishing results are highlighted in **bold**.

Resampling	Category	Benign			Phishing		
		Precision	Recall	F1	Precision	Recall	F1
KMeans-SMOTE [53]	Crypto	0.996±0.000	0.999±0.000	0.997±0.000	0.781±0.098	0.394±0.082	0.521±0.089
	Darknet	0.983±0.004	0.994±0.003	0.989±0.003	0.631±0.123	0.361±0.079	0.454±0.087
	Games	0.993±0.001	0.993±0.001	0.993±0.001	0.848±0.019	0.824±0.053	0.835±0.021
ISMOTE [60]	Crypto	0.996±0.000	0.996±0.001	0.996±0.000	0.466±0.050	0.469±0.095	0.465±0.068
	Darknet	0.986±0.005	0.983±0.002	0.985±0.002	0.415±0.095	0.480±0.159	0.438±0.107
	Games	0.995±0.002	0.990±0.001	0.993±0.000	0.788±0.007	0.886±0.039	0.833±0.014
G-SMOTE [89]	Crypto	0.996±0.000	0.998±0.001	0.997±0.000	0.587±0.114	0.418±0.073	0.485±0.081
	Darknet	0.983±0.004	0.989±0.004	0.986±0.002	0.454±0.095	0.361±0.054	0.397±0.059
	Games	0.994±0.002	0.994±0.002	0.994±0.001	0.846±0.033	0.843±0.050	0.844±0.031
NT-SMOTE [104]	Crypto	0.996±0.001	0.998±0.001	0.997±0.000	0.570±0.076	0.396±0.082	0.465±0.076
	Darknet	0.982±0.006	0.990±0.004	0.986±0.004	0.458±0.123	0.321±0.121	0.365±0.123
	Games	0.994±0.001	0.994±0.001	0.994±0.001	0.854±0.027	0.846±0.028	**0.849±0.014**
SL-SMOTE [19]	Crypto	0.996±0.001	0.999±0.000	0.997±0.000	0.707±0.085	0.390±0.073	0.501±0.080
	Darknet	0.983±0.004	0.989±0.003	0.986±0.002	0.480±0.087	0.379±0.041	0.418±0.043
	Games	0.995±0.002	0.991±0.001	0.993±0.001	0.808±0.019	0.879±0.040	0.842±0.024
SMOTE–IPF [49]	Crypto	0.996±0.000	0.997±0.001	0.996±0.000	0.511±0.043	0.461±0.067	0.482±0.047
	Darknet	0.985±0.004	0.987±0.004	0.986±0.002	0.480±0.096	0.462±0.062	0.462±0.044
	Games	0.995±0.001	0.992±0.001	0.993±0.001	0.808±0.018	0.866±0.029	0.836±0.018
SMOTE+NCL	Crypto	0.996±0.000	0.997±0.001	0.997±0.000	0.509±0.036	0.460±0.073	0.481±0.051
	Darknet	0.985±0.003	0.983±0.004	0.984±0.003	0.404±0.072	0.425±0.062	0.412±0.061
	Games	0.994±0.001	0.993±0.001	0.993±0.001	0.831±0.027	0.858±0.027	0.844±0.024
SMOTE+Tomek	Crypto	0.996±0.000	0.997±0.001	0.997±0.000	0.560±0.030	0.456±0.074	0.500±0.050
	Darknet	0.983±0.005	0.990±0.004	0.986±0.002	0.498±0.117	0.367±0.073	0.410±0.046
	Games	0.993±0.001	0.994±0.001	0.994±0.001	0.805±0.014	0.830±0.035	0.839±0.021
ROS	Crypto	0.996±0.001	0.999±0.000	0.997±0.000	0.771±0.103	0.389±0.094	0.513±0.100
	Darknet	0.983±0.005	0.991±0.003	0.987±0.003	0.545±0.102	0.377±0.096	0.437±0.080
	Games	0.993±0.002	0.994±0.001	0.994±0.001	0.844±0.028	0.836±0.039	0.840±0.024
NCL-ROS	Crypto	0.996±0.000	0.999±0.001	0.997±0.000	0.679±0.056	0.428±0.073	**0.523±0.069**
	Darknet	0.988±0.005	0.984±0.004	0.986±0.002	0.484±0.100	0.563±0.126	**0.510±0.077**
	Games	0.995±0.002	0.990±0.001	0.992±0.001	0.777±0.018	0.886±0.038	0.827±0.019
Tomek-ROS	Crypto	0.995±0.001	0.999±0.000	0.997±0.000	0.838±0.119	0.361±0.071	0.501±0.083
	Darknet	0.981±0.004	0.997±0.002	0.989±0.002	0.723±0.162	0.267±0.068	0.377±0.068
	Games	0.993±0.002	0.994±0.001	0.994±0.001	0.856±0.020	0.828±0.041	0.841±0.026
GaussNoise-ROS	Crypto	0.995±0.001	1.000±0.000	0.997±0.000	0.829±0.071	0.339±0.081	0.478±0.087
	Darknet	0.982±0.004	0.992±0.005	0.987±0.003	0.543±0.123	0.305±0.070	0.397±0.068
	Games	0.992±0.001	0.995±0.001	0.994±0.001	0.875±0.028	0.812±0.027	0.841±0.018

Chapter 5

Can LLMs Detect Phishing in Telegram using User and Message Signals?

In the previous chapter, we explored the potential of a social messaging platform, Telegram, for phishing detection as a source of phishing propagation. We showed that category-specific models using message metadata and propagational behavior are effective in detecting phishing messages in high-risk categories. In this chapter, we evaluate the LLM-based approach for phishing detection in Telegram and compare it to our method.

The study presented in this chapter is currently under review for publication in a journal.

Erfan, M., Branco, P., & Jourdan, G. V., 2025. Proactive Phishing Detection in Telegram: Can Large Language Models Keep Up with Traditional Models? Digital Threats: Research and Practice. (under review)

5.1 Introduction

LLMs have recently become powerful tools for security tasks by achieving strong performance in domains such as phishing email classification, smishing detection, and website fraud detection. Their ability to reason over text, adapt to few-shot examples, and generalize across different data sources opens the question of whether they can also be effective in more dynamic environments, such as social media. While emails, SMS, and web pages each have their own challenges – some of which overlap with those of social platforms – platforms such as Telegram further introduce issues such as informal language, multilingual

content, and platform-specific behaviors. These characteristics make phishing detection on such platforms especially challenging, and the suitability of LLMs in this context is less explored.

Building on the previous chapter, which showed the effectiveness of category-specific models tailored to Telegram's high-risk groups, this chapter analyzes whether LLMs can be a practical alternative. Specifically, we systematically evaluate multiple LLMs, prompting strategies, and input configurations to identify the best-performing setup. We then apply the strongest configuration on the entire dataset and compare the results directly against the category-specific models developed in Chapter 4. Moreover, we also carried out experiments using previously published phishing datasets to ensure that our results are not simply affected by the model and setup we chose, but instead reflect the unique challenges posed by Telegram data.

Contributions: The main contributions of this chapter are:

- Systematic evaluation of three representative LLMs (GPT-4o [47], LLaMA-3.1 [33], and DeepSeek v3 [63]) across multiple prompting strategies to select the best-performing for phishing detection in Telegram.
- Ablation study on the impact of the number of shots in few-shot configurations.
- Ablation study on the impact of different input representations on phishing detection.
- Comparative analysis between LLMs and category-specific models from Chapter 4.
- Comparison through reproducing prior LLM-based phishing detection studies, clarifying whether observed limitations arise from the model or from the unique characteristics of Telegram phishing content.

Organization. The remainder of this chapter is structured as follows. Section 5.2 reviews related work on LLM-based phishing detection. Section 5.3 describes the dataset, evaluated models, prompting strategies, also setups for our experiments. Section 5.4 presents the experimental results, including identified best-performing LLMs, comparative analysis of LLM-based detection with category-specific models and a discussion of input variations. Section 5.5 presents a cross-domain comparison by reproducing prior work. Finally, Section 5.6 summarizes the findings.

5.2 Related Work

LLMs have been applied to phishing detection in several ways, showing the different forms of phishing attacks. Researchers have investigated their use on email content, URLs, HTML webpages, SMS, and chat messages. The following review highlights the state of the art in these areas.

Many of the LLM-based phishing detection research has focused on emails, one of the most common phishing attacks. Zhang et al. [105] examined GPT-4, Claude, PaLM, and LLaMA in classifying both human-generated and Artificial Intelligence (Artificial Intelligence)-generated phishing and legitimate English emails, finding that LLMs usually matched or outperformed human judgment. Similarly, Heiding et al. [44] evaluated the same set of models for raw email text and highlighted Claude's reliable performance. Koide et al. [50] proposed ChatSpamDetector, a system that analyzes both headers and bodies of multilingual emails, showing that GPT-4 achieved the strongest results. Chataut et al. [23] extended this work by introducing CyberGPT, a customized ChatGPT model that outperformed GPT-3.5 and even GPT-4. Other comparative work by Ptel et al. [75] benchmarked 15 LLMs for phishing detection in emails, showing that the models of the GPT family (ChatGPT-4, GPT-3.5, Turbo-Instruct) were the most accurate in flagging phishing attempts.

Besides emails, researchers have also investigated URLs and websites as phishing indicators. Rashid et al. [79] tested five LLMs in multiple phishing URL datasets. Their results showed that while the few-shot setting had marginal improvements, one-shot prompting was the most cost-effective strategy. Trad and Chehab [99] further investigated ensemble prompting and modeling strategies for phishing URL edetction. They found that the ensemble worked better in the cases where individual components – whether prompts or LLMs – showed equivalent performance levels. In another study [98], the same authors compared prompt engineering with fine-tuning, showing that while prompt engineering achieved good results, fine-tuned models could outperform, specifically for imbalanced conditions.

Lee et al. [57] extended LLM use to phishing webpage detection, combining webpage screenshot and HTML analysis in a two-phase framework: brand identification followed by domain verification. Ling et al. [62] presented the Meta Phishing Detector, which integrated email headers and bodies with chain-of-thought prompting, using spoofed headers and social engineering patterns. In a related domain, Muzammil et al. [71] used CT logs to flag crypto-related domains and then applied LLM-based classification to determine if a site is a scam. Specifically, they combined LLaMA-3 and GPT-4, using Optical Character Recognition (OCR)-extracted site text (and in GPT-4 Vision's case, screenshots) as input. In another recent work, Bittab et al. [15] explored fraudulent shopping sites, where phishing-like deception targets consumers. They developed SCAMNET using fine-tuned LLaMA-3, and integrating URLs, WHOIS, and website content to classify sites while providing human-readable explanations. They showed explainability and robustness can be achieved using LLMs, with careful fine-tuning.

Some other research focused on spam detection using LLMs. Salman et al. [88] introduced SpaLLM-Guard, a hybrid two-stage pipeline: first, open-source LLMs (such as

LLaMA-2, Falcon, and MPT) were used for lightweight initial classification to filter out obvious spams; then, commercial LLMs (such as GPT-4 and Claude) for more complex cases. However, they acknowledged relying on short, template-like benchmark spam messages, which raises concerns about the application of their method for real-world generalizability. Roumeliotis et al. [81] confirmed similar challenges and investigated the generalizability of some approaches. They evaluated GPT against BERT, RoBERTa, and Convolutional Neural Network (CNN) for spam filtering. They found that while fine-tuned BERT achieved very high accuracy on Kaggle datasets, but GPT-4 generalized better across datasets.

Finally, LLM-based phishing detection has started to be applied in social media and messaging platforms, which are increasingly exploited by attackers and align most directly with our focus on Telegram. Chang and Aimur [21] built a dataset of scam chats from messaging platforms such as WhatsApp, Telegram, Messenger, and Discord, but collected indirectly from screenshots shared on X, Reddit, and Facebook, and then they evaluated five LLMs. Based on their results, Llama-3 achieved the best F1-score among all. In another research, Al Daoud et al. [7] evaluated phishing across Arabic social media posts, using datasets of Arabic social media posts. They compared four strategies: zero-shot LLMs (GPT-4o, Claude-3, Gemini-1.5), transformers as feature extractors with Random Forest, fine-tuned small language models, and ensembles. They found that zero-shot LLMs had the best performance.

While previous work shows that LLMare highly effective for detecting phishing across emails, URLs, websites, and even SMS messages, research on social media and chat platforms are rare. Although Chang and Aimur [21] contributed to this area by building a dataset of scam-related chat screenshots, their collection strategy – taking 541 scam chats and 200 legitimate ones from public posts on X, Reddit, and Facebook – has key limitations. The dataset is limited to English conversations, relies on self-reported or selectively shared cases, and lacks large-scale coverage, which may have the risk of making the dataset biased toward more obvious scams. These limitations open an important question about the generalizability of LLMs when applied to real-world Telegram interactions. In this work, we focus on detecting phishing in Telegram chats, going beyond just English or obvious scam cases, and use LLMs to capture subtle patterns, and the platform-specific phishing strategies.

5.3 Methodology

5.3.1 Dataset

We used the *TelePhish* dataset introduced in the previous chapter (Chapter 4, Section 4.3). *TelePhish* contains over 58,000 Telegram messages collected from 204 public groups across three high-risk categories: Cryptocurrency, Games, and Darknet. Each message includes user-level, message-level, and engagement features that capture propagation behaviors, with phishing ground-truth labels inferred using VirusTotal URL analysis.

In this chapter, we used the same dataset to evaluate the performance of LLMs in phishing detection. However, unlike in the previous chapter, where only the features extracted from metadata and Telegram API were used for model development, here we used all available inputs – including the raw message content, the original Telegram-exported metadata, and the features extracted – to enable a more comprehensive evaluation of LLM performance.

5.3.2 Evaluated Models and Prompting Strategies

Models: We investigated three representative LLMs to evaluate their ability to detect phishing in Telegram messages. We examined these models to cover both state-of-the-art commercial systems and open-source alternatives that are more flexible for research and development. Thus, we selected two commercial models, GPT-4o and DeepSeek v3, and an open-source model LLaMA-3.1 (8B), which was the only multilingual option from open-source models. GPT-4o is one of the most advanced commercially available models, widely recognized for its strong reasoning. LLaMA-3.1, represents an open-source model, designed for scalability and efficiency while supporting multilingual capabilities. Finally, DeepSeek V3 is a recently released commercial model that emphasizes efficiency.

These models capture a range of design trade-offs: commercial vs. open-source, large-scale vs. efficient architectures. We first compare their performance, and then use the best-performing model for further experiments.

Prompting Strategies: Since prompt design plays a critical role in determining LLM performance for classification tasks, we explored several strategies that differed in three key dimensions: (i) the use of role, (ii) the number of labeled examples (*shots*) provided for in-context learning [18], and (iii) the requirement for chain-of-thought reasoning [103]. These variations allowed us to evaluate how prompting could impact model performance.

- Task/Role Orientation: Task-oriented prompts simply ask the model to decide whether a message is phishing or benign. Role-oriented prompts instruct the model to take a persona (e.g., a helpful assistant or a security analyst) before providing its answer. The idea was to see whether assigning a role encourages the model to solve the task more carefully, compared to the task-only prompting.
- Shot Configuration: We compared zero-shot and few-shot prompting. In the zero-shot setting, the model was asked to classify a message as phishing or benign, without seeing any labeled examples. In the few-shot setting, we provided a number of labeled phishing and benign messages as examples to guide the model.
- Reasoning Requirement: We tested the impact of requiring chain-of-thought reasoning. In these prompts, the model was instructed to think step-by-step before giving its final classification. Although only the final label (phishing or benign) was used for evaluation, this setup allowed us to investigate whether reasoning helps the model reach more accurate conclusions.

By combining these three dimensions, we designed 12 prompting strategies in total: six in the zero-shot setting and six in the few-shot setting (Table 5.1). For example, the *ZS-Analyst-CoT* configuration corresponds to a zero-shot prompt where the model is asked to adopt the role of a security analyst, reason step by step, and then provide its final classification. This systematic design allowed us to evaluate the models as well as the prompting style to maximize phishing detection performance in Telegram chats.

Table 5.1: Overview of prompting strategies combining shot configuration, role conditioning, and reasoning style.

Shot Configuration	**Role Conditioning**	**Reasoning Style**	**Strategy ID**
Zero-shot	None	None	ZS-Task
	None	Chain-of-Thought	ZS-Task-CoT
	Helpful Assistant	None	ZS-Assistant
	Helpful Assistant	Chain-of-Thought	ZS-Assistant-CoT
	Security Analyst	None	ZS-Analyst
	Security Analyst	Chain-of-Thought	ZS-Analyst-CoT
Few-shot	None	None	FS-Task
	None	Chain-of-Thought	FS-Task-CoT
	Helpful Assistant	None	FS-Assistant
	Helpful Assistant	Chain-of-Thought	FS-Assistant-CoT
	Security Analyst	None	FS-Analyst
	Security Analyst	Chain-of-Thought	FS-Analyst-CoT

5.3.3 Evaluation Method and Experimental Stages

To ensure a structured evaluation, the experiments were conducted in multiple stages, each building upon the findings of the previous one. Specifically, stage A identifies the most effective models and prompt configurations, stage B optimizes the few-shot learning setup by adjusting the number of examples, stage C evaluates the selected configuration on the entire dataset, Stage D compares the performance of the selected LLMs with our previously developed ML models from Chapter 4, and stage E investigates the effect of different input representations (message text, URLs, metadata, and their combinations). This systematic process enables a fair comparison across models, prompting strategies, and data representations.

Stage A – Model and Prompt Selection: This first stage was aimed at identifying the most effective combination of model and prompt strategy. We used a hold-out evaluation subset of 100 messages, consisting of 50 phishing and 50 benign samples randomly drawn from all categories in the *TelePhish* dataset. This allowed us to perform a systematic comparison across all combinations of the three models and the 12 prompting strategies listed in Table 5.1. For few-shot settings, we used an equal number of phishing and benign examples (one phishing and one benign per category). Each model–prompt configuration was evaluated on the same hold-out evaluation set, and performance was measured using class-specific Precision, Recall, and F1-score.

In addition to the balanced evaluation subset, we also created a second hold-out test set of 100 messages that was stratified by class distribution. Because phishing messages represented less than 3% of the overall samples in *TelePhish*, we ensured that each Telegram category included at least one phishing sample, while preserving the original category distribution for benign samples. This stratified subset was designed to better reflect the natural imbalance present in the dataset and to examine model performance under realistic conditions. For few-shot configurations, we applied the same shot setup as in the balanced evaluation (one phishing and one benign example per category).

The outcome of this stage was two best-performing model–prompt configurations: one selected based on the balanced test evaluation and another based on the stratified evaluation. These two configurations were then carried forward for further analysis in subsequent stages.

Stage B – Ablation Study on Shots (for the few-shot configuration): Once the most effective model–prompt configurations were identified in stage A, we used them for the next stages of experimentation. In stage B, we conducted a systematic study to optimize the number of shots for the few-shot settings. Using the selected model–prompt configuration from stage A, we varied the number of shots. We had two objectives: (i) to

determine the optimal total number of shots, and (ii) to assess whether shots should be balanced between phishing and benign classes or not. To do this, we built five independent few-shot subsets, each containing 24 phishing and 24 benign samples (48 in total). For each subset, phishing and benign samples were stratified across the three Telegram message categories in the dataset – Cryptocurrency, Darknet, and Games – by selecting eight samples per category.

The optimization was performed in two phases. In the first phase, we evaluated balanced shot configurations by varying the number of phishing and benign examples equally. Specifically, we tested 1:1, 3:3, 6:6, 9:9, 12:12, and 24:24 shot ratios. For performance evaluation, the selected subsets were excluded from the dataset, and a validation set was created by sampling 10% of the dataset. We wanted to test the hypothesis that a larger number of shots would improve classification performance. In the second stage, after identifying the best-performing total number of shots from the balanced setting, we fixed that total and explored imbalanced shot distributions. For instance, having 24 as the best-performing total shots, we tested phishing-to-benign ratios of 3:21, 6:18, 9:15, 15:9, 18:6, and 21:3. With these settings, we could investigate whether equal class representation for shots is performing better, or whether performance could be improved by biasing the shots towards either phishing or benign samples.

Stage C – Full-Dataset Evaluation: After optimizing the shot configuration, both best-performing model–prompt setups identified in Stage A (the balanced-based and the stratified-based configurations) were evaluated on the remaining dataset. The remainder data was defined as the portion of the data left after removing the few-shot examples and the 10% validation split used for shot optimization. To ensure robust performance estimation, we used five-fold CV, reporting performance metrics for both phishing and benign classes, overall and across the three Telegram message categories: Cryptocurrency, Darknet, and Games. This stage provided a large-scale performance comparison between the two top configurations under consistent experimental settings.

Stage D – Comparative Evaluation with Category-Specific Models: To evaluate LLM performance against our model, we compared the results obtained from stage C with the category-specific Random Forest classifiers developed in Chapter 4. The comparison used the same dataset remainder and five-fold cross-validation setup. Performance was reported both overall and by category. For the Random Forest models, per-category results were already available (Cryptocurrency, Darknet, and Games), and overall performance was obtained by aggregating predictions across categories. This stage enabled us to assess whether LLM-based phishing detection can outperform the performance of traditional machine-learning models trained specifically for each category.

Stage E – Ablation Study on Input Representations Initially, the LLMs performance

were tested using message text (containing URL) together with metadata. Phishing signals can appear in URLs, message wording, or behavioral patterns. This opens up the question of whether some of these inputs could provide stronger signals to the model than others. To examine how different input sources affect detection, we conducted an additional ablation study focusing on input representation.

This analysis was performed using the best-performing LLM configuration on the full-dataset evaluation from Stage C–D, allowing us to isolate the contribution of input composition while keeping the model fixed. The evaluated input configurations, summarized in Table 5.2, combine various elements of the Telegram data, including message text, extracted URLs, and metadata. By designing these experiments, we had two objectives. First, we could test whether the LLM model can detect phishing content directly from suspicious links, which are often the core element of phishing attacks. Second, by comparing combinations such as metadata-only or message + URL + metadata, we could analyze whether providing more information improves classification, or whether extra input confuses the model. Through this extended evaluation, our goal was to better understand the role of the input representation LLMs for detecting phishing in social messaging apps such as Telegram. All configurations were evaluated using five-fold CV on the dataset remainder (defined as the data left after excluding shots and validation samples).

Table 5.2: Input variations tested for DeepSeek. Each configuration combines different sources of information extracted from Telegram.

Input Setting	**Description**
Message	Raw Telegram message text only
URL	Extracted URL(s) from the message body
Metadata	Telegram metadata and features extracted via API
Message + URL	Raw message text with extracted URL(s)
URL + Metadata	Extracted URL(s) combined with metadata features
Message + URL + Metadata	Full input including message text, URLs, and metadata

5.3.4 Evaluation Metrics

In all stages, standard evaluation metrics were used for performance evaluation: Precision, Recall, and F1-score. These metrics were described earlier in Chapter 3, Section 3.3.1.

5.4 Experimental Results

5.4.1 Stage A – Model and Prompt Selection Results

Stage A focused on comparing multiple LLMs and prompting strategies to identify the most effective configuration for Telegram phishing detection. The evaluation was performed using two hold-out subsets: a balanced test set with equal numbers of phishing and benign samples, and a stratified test set reflecting the natural class imbalance of the *TelePhish* dataset. Tables 5.3 and 5.4 present the results for these two settings.

Table 5.3: Performance of GPT-4o, DeepSeek v3, and LLaMA-3.1 (8B) across prompting strategies on balanced test-set. Values show Precision/Recall/F1-score. Best results (used for further experiments) are in **bold**.

Shot	Prompt Strategy	Class	GPT-4o	DeepSeek v3	LLaMA-3.1(8B)
Zero-shot	ZS-Task	Benign	0.489/0.440/0.463	0.429/0.360/0.391	**0.476/0.800/0.597**
		Phishing	0.490/0.540/0.541	0.448/0.520/0.481	0.375/0.120/0.182
	ZS-Assistant	Benign	0.487/0.380/0.427	0.447/0.420/0.433	0.481/0.760/0.589
		Phishing	0.492/0.600/0.540	0.453/0.480/0.466	0.429/0.180/0.254
	ZS-Analyst	Benign	0.486/0.340/0.400	0.452/0.380/0.413	0.481/0.760/0.589
		Phishing	0.492/0.640/0.556	0.465/0.540/0.500	0.429/0.180/0.254
	ZS-Task-CoT	Benign	0.534/0.620/0.574	0.500/0.620/0.553	0.475/0.760/0.585
		Phishing	0.548/0.460/0.500	0.500/0.380/0.432	0.400/0.160/0.229
	ZS-Assistant-CoT	Benign	0.518/0.560/0.538	0.508/0.660/0.574	0.457/0.640/0.533
		Phishing	0.522/0.480/0.500	0.514/0.360/0.423	0.400/0.240/0.310
	ZS-Analyst-CoT	Benign	0.527/0.580/0.552	0.492/0.640/0.556	0.456/0.620/0.525
		Phishing	0.533/0.480/0.505	0.585/0.340/0.400	0.406/0.260/0.317
Few-shot	FS-Task	Benign	0.628/0.440/0.518	0.650/0.520/0.578	0.500/0.140/0.219
		Phishing	0.569/0.740/0.643	**0.600/0.720/0.654**	0.500/0.860/0.632
	FS-Assistant	Benign	0.618/0.420/0.500	0.619/0.520/0.565	0.500/0.140/0.219
		Phishing	0.561/0.740/0.638	0.586/0.680/0.630	0.500/0.860/0.632
	FS-Analyst	Benign	0.628/0.440/0.518	0.595/0.500/0.543	0.500/0.140/0.219
		Phishing	0.570/0.740/0.643	0.569/0.660/0.611	0.500/0.860/0.632
	FS-Task-CoT	Benign	0.641/0.500/0.562	0.551/0.540/0.545	0.462/0.120/0.190
		Phishing	0.590/0.720/0.649	0.550/0.560/0.554	0.494/0.860/0.628
	FS-Assistant-CoT	Benign	0.622/0.460/0.529	0.541/0.520/0.530	0.462/0.120/0.190
		Phishing	0.571/0.720/0.637	0.538/0.560/0.549	0.494/0.860/0.628
	FS-Analyst-CoT	Benign	0.619/0.520/0.565	0.520/0.520/0.520	0.500/0.120/0.194
		Phishing	0.586/0.680/0.630	0.520/0.520/0.520	0.500/0.880/0.638

Results on Balanced Test Set: Several patterns are clear in Table 5.3. First, few-shot prompting consistently outperformed zero-shot across all three models. Few-shot context helps LLMs adapt their decisions to relevant examples. However, the degree of improvement varied by model: DeepSeek and GPT-4o showed the largest relative improvements, while LLaMA-3.1 improved less.

Second, role conditioning did not provide a consistent benefit. The best DeepSeek scores occurred with task-only few-shot prompting (FS-Task), not with "Helpful Assistant" or "Security Analyst" roles. Also, adding chain-of-thought reasoning showed mixed effects across models. For DeepSeek, it consistently reduced detection performance, suggesting that step-by-step reasoning introduced unnecessary complexity. In contrast, GPT-4o and LLaMA-3.1 were largely unaffected. This suggests that step-by-step reasoning may not be effective for phishing detection in Telegram discussions, where messages are typically brief and lack the contextual depth that structured reasoning uses.

Third, LLaMA-3.1 consistently underperformed compared to GPT-4o and DeepSeek, particularly for the phishing class, where recall was often high but precision remained low, dragging down the F1-score. Third, while GPT-4o produced competitive results, DeepSeek's performance showed better trade-offs between classes. Beyond its performance advantage, DeepSeek is considerably more cost-effective than GPT-4o, making it the more suitable choice for large-scale evaluation.

Results on Stratified Test Set: Under the stratified test setting, where the class distribution represented the actual imbalance of the data, the models showed different performance patterns compared to the balanced configuration. Here, the overall performance declined for all models, reflecting the difficulty under realistic class imbalance. Across all model-prompt strategies, LLaMA-3.1(8B) achieved the highest F1-scores for both benign and phishing classes. Incorporating Chain-of-Thought reasoning led to some improvements, specifically for GPT-4o and DeepSeek, but phishing detection remained challenging.

For few-shot prompting, all three models followed a similar pattern: phishing class performance declined overall. GPT-4o showed results comparable to its zero-shot CoT variant, while LLaMA's few-shot performance dropped substantially. Overall, the stratified results highlight the impact of class imbalance on the performance results.

Based on the experimental results across both evaluations, two model–prompt configurations were selected for further experimentation. DeepSeek with few-shot prompting (from the balanced setup) was chosen due to its strong trade-off phishing and benign detection performance, while LLaMA-3.1(8B) with zero-shot prompting (from the stratified setup) was selected as the best alternative under realistic imbalance. These selected configurations form the basis for the subsequent evaluation stage.

5.4.2 Stage B – Few-Shot Configuration Optimization Results

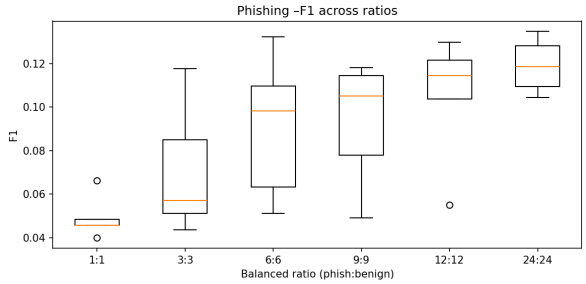
By identifying DeepSeek with few-shot prompting as one of the best-performing configurations, we evaluated how the number of provided shots affects classification performance.

Table 5.4: Performance of GPT-4o, DeepSeek v3, and LLaMA-3.1 (8B) across prompting strategies on stratified test-set. Values show Precision/Recall/F1-score. Best results (used for further experiments) are in **bold**.

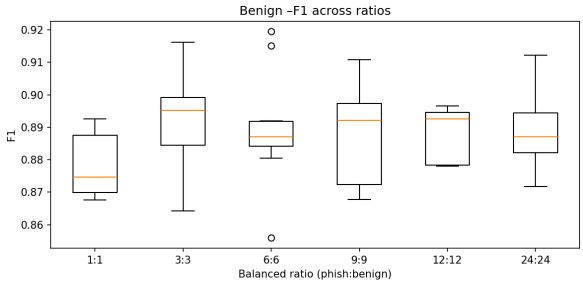
Shot	Prompt Strategy	Class	GPT-4o	DeepSeek v3	LLaMA-3.1(8B)
Zero-shot	ZS-Task	Benign	1.00/0.680/0.810	1.00/0.618/0.764	**0.989/0.887/0.935**
		Phishing	0.088/1.00/0.162	0.075/1.00/0.139	**0.154/0.667/0.250**
	ZS-Assistant	Benign	1.00/0.618/0.764	1.00/0.670/0.802	0.988/0.845/0.911
		Phishing	0.075/1.00/0.139	0.086/1.00/0.158	0.118/0.667/0.200
	ZS-Analyst	Benign	1.00/0.547/0.707	1.00/0.608/0.756	0.988/0.825/0.899
		Phishing	0.063/1.00/0.120	0.073/1.00/0.136	0.106/0.667/0.182
	ZS-Task-CoT	Benign	0.987/0.814/0.893	0.987/0.783/0.873	0.966/0.866/0.913
		Phishing	0.100/0.667/0.174	0.087/0.667/0.154	0.000/0.000/0.000
	ZS-Assistant-CoT	Benign	1.00/0.763/0.865	0.987/0.814/0.893	0.988/0.814/0.893
		Phishing	0.115/1.00/0.207	0.100/0.667/0.174	0.100/0.667/0.174
	ZS-Analyst-CoT	Benign	1.00/0.732/0.845	0.987/0.804/0.886	0.987/0.804/0.886
		Phishing	0.103/1.00/0.187	0.095/0.667/0.167	0.095/0.667/0.167
Few-shot	FS-Task	Benign	0.985/0.660/0.790	0.986/0.732/0.840	1.00/0.268/0.423
		Phishing	0.057/0.667/0.105	0.071/0.667/0.129	0.041/1.00/0.078
	FS-Assistant	Benign	1.00/0.618/0.764	0.986/0.711/0.826	1.00/0.289/0.448
		Phishing	0.075/1.00/0.139	0.067/0.667/0.121	0.042/1.00/0.080
	FS-Analyst	Benign	1.00/0.618/0.764	0.985/0.691/0.812	1.00/0.258/0.410
		Phishing	0.075/1.00/0.139	0.062/0.667/0.114	0.040/1.00/0.077
	FS-Task-CoT	Benign	1.00/0.722/0.838	0.986/0.753/0.853	1.00/0.299/0.460
		Phishing	0.100/1.00/0.182	0.077/0.667/0.138	0.042/1.00/0.081
	FS-Assistant-CoT	Benign	1.00/0.722/0.838	0.986/0.753/0.853	1.00/0.320/0.484
		Phishing	0.100/1.00/0.182	0.077/0.667/0.138	0.043/1.00/0.083
	FS-Analyst-CoT	Benign	1.00/0.732/0.845	0.986/0.753/0.853	1.00/0.289/0.448
		Phishing	0.103/1.00/0.187	0.077/0.667/0.138	0.042/1.00/0.080

Figure 5.1 presents the boxplots of F1-scores for both phishing and benign classes for these balanced settings. The results show an increase in phishing F1-score as the number of shots increases, but benign F1 remains relatively stable. Notably, the 12:12 configuration resulted in the best overall performance, achieving consistently high phishing F1 without degrading benign F1-score. Increasing to 24:24 offered no substantial improvement. Based on this finding, we selected 12:12 as the optimal balanced configuration.

Having 24 as the best-performing total shots, we tested phishing-to-benign ratios of 3:21, 6:18, 9:15, 15:9, 18:6, and 21:3. Figure 5.2 demonstrates the boxplots of F1-scores for these imbalanced settings. The results show that highly skewed ratios generally reduced phishing detection performance. For example, configurations with a majority of benign shots (e.g., 3:21 or 6:18) resulted in lower phishing F1-score. In contrast, phishing-skewed ratios (e.g., 15:9 or 21:3) improved phishing F1-score marginally, but in these cases, F1-scores for the benign class showed greater variability. The 12:12 balanced configura-

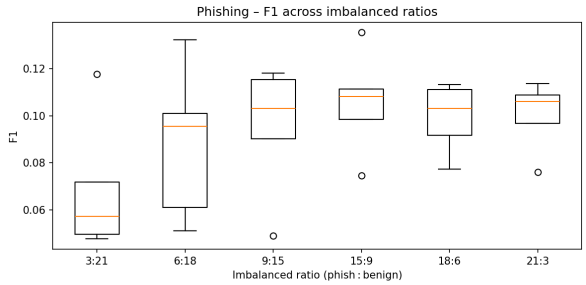


(a) Phishing F1-score across balanced shot ratios.

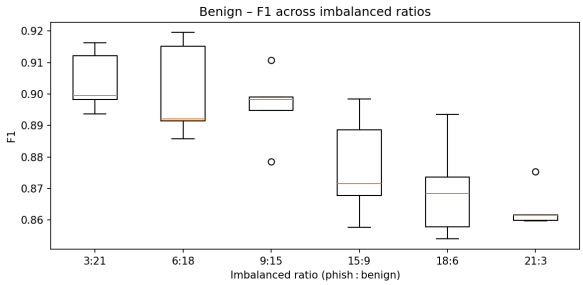


(b) Benign F1-score across balanced shot ratios.

Figure 5.1: Performance of phishing vs. benign detection for different balanced shot configurations.



(a) Phishing F1-score across imbalanced shot ratios.



(b) Benign F1-score across imbalanced shot ratios.

Figure 5.2: Performance of phishing vs. benign detection for different imbalanced shot configurations.

tion consistently outperformed these imbalanced alternatives, providing the best trade-off between the two classes. This confirms that balanced shots are fair and also effective for our experiments.

These experiments let us measure the trade-offs between the number and distribution of shots, and gave us a clear basis for choosing the best few-shot setup for the full-dataset experiments.

5.4.3 Stage C – Full-Dataset Evaluation Results

Using the selected DeepSeek model with the 12:12 balanced few-shot configuration (FS-Task prompting strategy according to table 5.1), and also LLaMA-3.1(8B) with zero-shot

configuration (ZS-Task prompting strategy according to table 5.1), we evaluated performance on the full dataset through five-fold CV. Table 5.5 reports precision, recall, and F1-scores for both phishing and benign classes, overall and across the three categories (Cryptocurrency, Darknet, and Games) of Telegram messages.

Table 5.5: DeepSeek and LLaMA classification performance across stratified 5-fold CV (mean ± std).

Model	Category	Benign			Phishing		
		Precision	Recall	F1	Precision	Recall	F1-score
DeepSeek	Overall	0.989±0.002	0.833±0.021	0.904±0.013	0.065±0.016	0.554±0.094	0.116±0.028
	Crypto	0.997±0.001	0.810±0.013	0.893±0.008	0.023±0.004	0.618±0.090	0.045±0.008
	Darknet	0.997±0.002	0.778±0.010	0.874±0.006	0.097±0.026	0.891±0.070	0.175±0.043
	Games	0.977±0.006	0.881±0.042	0.926±0.025	0.163±0.076	0.496±0.135	0.241±0.097
LLaMA	Overall	0.981±0.001	0.848±0.003	0.910±0.002	0.028±0.003	0.214±0.027	0.049±0.006
	Crypto	0.994±0.000	0.823±0.004	0.901±0.002	0.013±0.002	0.314±0.043	0.024±0.004
	Darknet	0.983±0.006	0.818±0.012	0.893±0.009	0.070±0.012	0.503±0.068	0.123±0.018
	Games	0.963±0.001	0.893±0.003	0.926±0.001	0.056±0.011	0.154±0.032	0.082±0.016

Overall, DeepSeek achieved better phishing detection performance compared to LLaMA, with a phishing F1-score of 0.116 versus 0.049. While both models demonstrated high performance for the benign class (DeepSeek F1-score: 0.90; LLaMA F1-score: 0.91), their effectiveness in detecting phishing messages was very different. DeepSeek showed better recall (0.55 vs. 0.21). However, both models had very low precision. This suggests that while the models identified a portion of phishing messages, they frequently misclassified benign messages as phishing, resulting in a high false positive rate.

At the category level, DeepSeek performed best on the Games category (F1 = 0.24), followed by Darknet (0.17) and Crypto (0.05). This suggests that phishing messages in gaming groups contained clearer patterns that the model could learn effectively from few-shot examples. In contrast, LLaMA's strongest performance appeared in the Darknet category (F1 = 0.12), with lower F1-scores in Games (0.08) and Crypto (0.02).

Overall, the results highlight the limitations of LLM-based detection in Telegram. Based on the results, phishing detection in Telegram was a challenge. Although both models were able to distinguish benign messages, they struggled to capture characteristics of phishing content.

5.4.4 Stage D – Comparative Evaluation Results with Category-Specific Models

In this section, we compare the performance of DeepSeek and LLaMA with the category-specific models developed in Chapter 4. Looking at the overall results in Table 5.6, the category-specific models perform much better in detecting phishing messages, with F1-score of 0.749 compared to 0.116 for DeepSeek and 0.049 for LLaMA. While all models achieved strong results on the benign class, the performance gap for phishing messages was significant.

Table 5.6: Overall classification performance of DeepSeek, LLaMA, and category-specific models across 5-fold CV (mean ± std).

Model	Benign			Phishing		
	Precision	Recall	F1	Precision	Recall	F1-score
DeepSeek	0.989±0.002	0.833±0.021	0.904±0.013	0.065±0.016	0.554±0.094	0.116±0.028
LLaMA	0.981±0.001	0.848±0.003	0.910±0.002	0.028±0.003	0.214±0.027	0.049±0.006
Category-Specific RF	0.995 ± 0.000	0.994 ± 0.000	**0.995 ± 0.000**	0.738 ± 0.019	0.760 ± 0.021	**0.749 ± 0.016**

We can see the same pattern when breaking overall results across categories, as shown in Table 5.7. Category-specific Random Forest models consistently achieved high phishing F1-scores across all three categories – 0.507 in Crypto, 0.449 in Darknet, and 0.834 in Games – indicating their ability to adapt to the phishing patterns specific to each Telegram domain. DeepSeek performed quite well at recall for phishing messages across all three categories, but its precision was extremely low. The highest phishing F1-score it could achieve was in the Games category (0.24). LLaMA F1-scores were also low across all three categories, with the highest observed in the Darknet category (0.12).

These results highlight that generic prompting with LLMs, while sensitive to certain malicious signals, fails to achieve the precision required for practical deployment on Telegram. While fine-tuning these models might potentially narrow this performance gap, such approaches were not the primary focus of our evaluation. In a real-world, high-velocity environment like Telegram, the high computational costs of fine-tuning at scale can be a prohibitive factor. Consequently, traditional tailored models remain the more efficient and effective choice for real-time monitoring, as they provide balanced performance without the operational complexity inherent in large-scale language models.

Table 5.7: Category-level classification performance of DeepSeek, LLaMA, and category-specific models across 5-fold CV (mean ± std).

Model	Category	Benign			Phishing		
		Precision	Recall	F1	Precision	Recall	F1-score
DeepSeek	Crypto	0.997±0.001	0.810±0.013	0.893±0.008	0.023±0.004	0.618±0.090	0.045±0.008
	Darknet	0.997±0.002	0.778±0.010	0.874±0.006	0.097±0.026	0.891±0.070	0.175±0.043
	Games	0.977±0.006	0.881±0.042	0.926±0.025	0.163±0.076	0.496±0.135	0.241±0.097
LLaMA	Crypto	0.994±0.000	0.823±0.004	0.901±0.002	0.013±0.002	0.314±0.043	0.024±0.004
	Darknet	0.983±0.006	0.818±0.012	0.893±0.009	0.070±0.012	0.503±0.068	0.123±0.018
	Games	0.963±0.001	0.893±0.003	0.926±0.001	0.056±0.011	0.154±0.032	0.082±0.016
Category-Specific	Crypto	0.996±0.000	0.998±0.000	**0.997±0.000**	0.647±0.045	0.420±0.051	**0.507±0.043**
	Darknet	0.985±0.006	0.985±0.005	**0.985±0.004**	0.451±0.085	0.465±0.128	**0.449±0.094**
	Games	0.996±0.001	0.990±0.001	**0.993±0.001**	0.781±0.016	0.894±0.028	**0.834±0.017**

5.4.5 Stage E – Input Representation Ablation Results

Using the DeepSeek configuration (as the best performing LLM configuration), we compared model performance across the different input representations defined earlier in Table 5.2 (message text, URLs, metadata, and their combinations). The summarized results are presented in Table 5.8. Although combined input (such as message + URL + metadata), improved benign performance, they decreased phishing detection performance, and URL-only was the best among all other input types. This indicates that in the context of LLM-based phishing detection in Telegram, URL-focused representation is the most effective input choice. It is also worth mentioning that the message-only setting could not be evaluated meaningfully, as many Telegram posts in the dataset consisted only of URLs without any accompanying text. Because excluding these samples could bias the dataset and reduce comparability across input variations, we report this setting as unavailable in Table 5.8.

Table 5.8: Classification performance of DeepSeek across different input variations using stratified 5-fold CV (mean ± std). Best scores per metric are highlighted in **bold**.

Input	**Benign**			**Phishing**		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Message	-	-	-	-	-	-
URL	**0.998 ± 0.002**	0.749 ± 0.041	0.855 ± 0.027	**0.070 ± 0.009**	**0.918 ± 0.070**	**0.131 ± 0.016**
Metadata	0.990 ± 0.005	0.724 ± 0.044	0.839 ± 0.035	0.056 ± 0.005	0.684 ± 0.047	0.104 ± 0.004
Message + URL	0.993 ± 0.001	0.761 ± 0.035	0.861 ± 0.022	0.059 ± 0.006	0.724 ± 0.056	0.109 ± 0.010
URL + Metadata	0.994 ± 0.001	0.789 ± 0.020	0.880 ± 0.013	0.070 ± 0.006	0.776 ± 0.044	0.129 ± 0.010
Message + URL + Metadata	0.989 ± 0.002	**0.833 ± 0.021**	**0.904 ± 0.013**	0.065 ± 0.016	0.554 ± 0.094	0.116 ± 0.028

5.4.6 Limitations of the Experiments

In addition to dataset-specific considerations discussed in the previous chapter, our experiments have some methodological limitations. First, our study is limited to prompting-based use of LLMs, including zero and few-shot settings. Although this approach is resource-efficient, it inherently limits performance because the model may not be fine-tuned for the phishing context. Second, the shot optimization process relied on stratified subsets of phishing and benign messages. Although we built multiple subsets to reduce sampling bias, different random selections could produce slightly different outcomes. Similarly, our choice of models – DeepSeek, GPT-4o, and LLaMA-3.1 (8B) – was motivated by considerations of cost, availability, and multilingual support. Larger or more recent variants, such as GPT-5 or LLaMA (70B) might show different performance. Finally, LLM performance is highly dependent on prompting design. While we explored a variety of prompting strategies, including role/task-oriented, shot configuration, and reasoning requirements, we could not explore the full design space. Some other strategies, such as multi-turn prompting, may improve the detection performance.

Furthermore, while we could have given the models more chances by implementing recursive prompting or extensive fine-tuning, these methods were intentionally not pursued due to their incompatibility with real-time operational requirements. In messaging environments, where detection must occur nearly instantaneously to prevent victims from engaging with malicious links, the increase in token consumption and inference latency associated with advanced LLM techniques outweighs marginal gains in accuracy. Our evaluation thus focuses on the models in their most accessible and low-latency configurations to determine their baseline viability for platform-wide protection.

5.5 Comparison with Prior LLM Studies

To better understand whether the limitations we observed in Telegram phishing detection using LLM come from the context or from the choice of LLM and prompting strategy, we evaluated several prior studies that reported strong performance of LLMs for phishing detection. Our objective was to evaluate DeepSeek (The best performing LLM from our experiments) on datasets used in these works, following their experimental setup as closely as possible.

For this purpose, we selected prior works that not only reported strong results but also provided available datasets and sufficient detail about the experimental setup. One of the most common benchmarks for LLM-based phishing detection is URL classification. Among

those, the study by Trad and Chehab [98] reported an F1-score of 0.90 using Claude-2 on a balanced test set of 500 phishing and 500 benign URLs. Since their exact test set was not available, we instead created five test sets of 1000 URLs each (balanced between phishing and benign), randomly sampled from the original dataset, and evaluated DeepSeek. As shown in Table 5.9, DeepSeek achieved an average F1-score of 0.88 across these test sets. The small difference compared to the reported result can be explained by differences in the test set and the models. However, the high performance confirms that our chosen model and prompting approach are effective, and suggests that the lower results observed on Telegram data are more likely because of the characteristics of the context rather than experimental limitations.

Table 5.9: Comparison of reported results in prior LLM-based phishing detection studies and our replication with DeepSeek.

Domain/Study/Model	Precision	Recall	F1-score	Setup / Notes
URL-based phishing detection				
Trad & Chehab [98]				
Claude-2 (authors' reported)	0.91	0.89	0.90	Hannousse–Yahiouche dataset [41], balanced 500 phishing / 500 benign links
DeepSeek	0.885 ± 0.021	0.884 ± 0.027	0.884 ± 0.005	avg. across 5 random balanced test sets (1000 links each) drawn from the same dataset
Rashid et al. [79]				
GPT-4 (authors' reported)	NA	NA	0.92	EBBU [85], one-shot setting, balanced 500 phishing / 500 benign links
DeepSeek	0.894 ± 0.035	0.891 ± 0.045	0.891 ± 0.011	avg. across 5 random balanced test sets (1000 links each) drawn from the same dataset
Chat-based phishing detection				
Chang & Aimur [21]				
LLaMA-3 (authors' reported)	0.982	0.962	0.973	541 scam chats + 200 legitimate chats
DeepSeek	0.783 ± 0.145	0.747 ± 0.210	0.727 ± 0.050	avg. across 5 random test sets from the extended dataset [22]
LLaMA-3	0.654 ± 0.062	0.653 ± 0.082	0.647 ± 0.028	avg. across 5 random test sets from the extended dataset [22]

Similarly, Rashid et al. [79] showed that one-shot prompting with GPT-4 and Claude achieved strong performance on multiple phishing URL datasets. In their experiments, one-shot prompting was found to be the most cost-effective configuration, reaching F1-scores around 0.92 with GPT-4. To replicate their setup, we evaluated DeepSeek on the EBBU [85] dataset, which was publicly available, applying the same one-shot strategy. Since their exact test splits were not provided, we created five balanced subsets of 1000 URLs each (500 phishing, 500 benign) and measured performance across them. As shown in Table 5.9, DeepSeek achieved an average F1-score of 0.89. These results, while slightly lower than those reported by the authors, again confirm that DeepSeek is competitive when applied to benchmark URL datasets. The small differences are likely related to variations in test splits, shot sample, and the differences between the LLMs.

We also examined a chat-based phishing dataset introduced by Chang and Aimur [21]. They evaluated several LLMs on 541 scam and 200 legitimate chat segments collected from screenshots shared on X, Reddit, and Facebook. Their reported results showed that LLaMA-3 achieved an F1-score of 0.973. To replicate this experiment, we applied DeepSeek

using their extended dataset (since the original one was no longer available). Their exact test splits were not specified, so we built five test subsets that preserved the original phishing-to-benign ratio (541 scam vs. 200 benign) and reported averaged results across these subsets. As shown in Table 5.9, DeepSeek reached an average F1-score of 0.727, while our evaluation using LLaMA-3 resulted in a F1-score of 0.647. Although both results are notably lower than the 0.973 reported by the authors, they still show that DeepSeek performed competitively on this benchmark. These differences may be related to the usage of the dataset and the absence of the exact test set. However, these results also suggest that DeepSeek is capable of achieving strong performance on external phishing datasets, supporting our conclusion that the challenges we observed in Telegram data came from the nature of Telegram phishing messages rather than from the model.

5.6 Conclusion

This chapter examined the effectiveness of LLMs for detecting phishing messages in Telegram and compared their performance with the category-specific models developed in the previous chapter. The motivation for this analysis came from the growing use of LLMs in various security applications, raising the question of whether these models could be also a viable solution for phishing detection on Telegram.

First, We systematically explored different LLMs, and prompting strategies. On hold-out evaluation sets, DeepSeek with few-shot prompting and LLaMA with zero-shot prompting achieved the strongest results among all tested combinations. When evaluated on the full dataset, however, DeepSeek demonstrated comparatively better phishing detection performance than LLaMA, but its overall precision remained very low, resulting in a weak F1-score despite high recall for phishing messages.

In the comparative stage, we reused the DeepSeek results to evaluate how LLMs compares against the category-specific Random Forest models introduced in Chapter 4. The results showed that category-specific models achieved higher and more balanced precision–recall trade-offs across all Telegram categories. They achieved better performance, particularly because they were tailored to the distinct phishing strategies of each public group category in Telegram. In contrast, DeepSeek's prompting-based approach struggled to capture these differences, highlighting the difficulty of relying on generic LLMs without further adaptation. Finally, additional experiments with DeepSeek and replications of prior studies confirmed that the weak results were not caused by the selected LLM itself, but by the challenges of phishing detection usingLLMs in Telegram environment.

Overall, these findings suggest that while LLMs can offer useful insights for phishing

detection, their effectiveness on Telegram is limited. In contrast, feature-based methods tailored to the platform were more reliable. This chapter underscores the importance of aligning detection strategies with the specific characteristics of the data and environment, rather than relying solely on general-purpose language models.

In the next chapter, we provide an overview of the thesis, discuss the limitations and future directions of this work, and conclude the study.

Chapter 6

Conclusion

6.1 Overview of the Thesis

This thesis explored the problem of proactive phishing detection from two complementary perspectives. The first focused on understanding the types of infrastructure used to host phishing campaigns and quantifying how much of the phishing landscape is visible to existing infrastructure-based detection methods. Since such systems primarily detect attacker-owned domains, this phase aimed to determine what proportion of phishing activity relies on those domains versus on compromised or legitimate third-party infrastructures that evade proactive monitoring. The second perspective investigated how phishing can be detected proactively when infrastructure signals are absent or insufficient, by exploring alternative data sources that capture phishing propagation behaviors on social platforms. Together, these two perspectives provide a comprehensive view of proactive phishing detection — from measuring what current infrastructure-based defences can see, to identifying how phishing can be detected early through behavioral and propagation-based analysis.

Chapter 2 provided the conceptual foundation for the first research question by first reviewing existing proactive detection methods and their dependence on infrastructure-level signals. The chapter highlighted the limitations of these methods, particularly when attackers use legitimate services instead of using their own registered domains. Recognizing this limitation, the chapter proposed a rule-based taxonomy that used interpretable heuristics to classify phishing websites by domain ownership types, distinguishing between attacker-controlled domains, compromised websites, and third-party platforms. This taxonomy was motivated by the need to understand the infrastructure diversity of phishing campaigns better.

Chapter 3 applied the taxonomy to real-world phishing data to validate its effectiveness. The chapter first evaluated how well the rule-based taxonomy performed by comparing it against supervised and unsupervised ML models. This comparison revealed that the supervised ML model was more effective in categorizing phishing websites based on the domain ownership types. Based on these findings, the supervised approach was then applied to a year-long phishing dataset *PhisXtract*, to identify phishing website ownerships at scale and capture the current trends. The analysis demonstrated that a significant number of attacks are hosted on legitimate third-party infrastructures, phishing setups that traditional proactive detection methods struggle to detect. These results confirmed the value of the taxonomy in highlighting the detection blind spots that motivate the next phase of this research.

Building on these findings, Chapter 4 addressed the second research question, examining whether phishing can be detected when traditional infrastructure signals are insufficient. This chapter investigated whether other observable artifacts could support early detection. In particular, we explored phishing propagation behaviors in social communication platforms, where phishing messages often reach potential victims before the underlying infrastructure is flagged. Using Telegram as a case study, we introduced the *TelePhish* dataset, capturing phishing and benign messages from public group discussions across high-risk categories such as Cryptocurrency, Games, and Darknet. The chapter developed category-specific models that used propagation patterns to identify phishing attempts. The findings confirmed that proactive phishing detection is feasible even without using traditional infrastructure indicators, using only the observable patterns of message propagation.

Finally, Chapter 5 evaluated whether LLMss – which have shown promise in various cybersecurity applications – can work as general-purpose detectors of phishing in messaging platforms such as Telegram. Through a systematic evaluation of multiple LLMs, prompting strategies, and input configurations, the study compared their performance with the category-specific models from 4. The results revealed that specialized category-specific models trained on the propagation behavior of high-risk categories on Telegram remain more effective in identifying phishing in real-world Telegram data.

Together, these studies provide a comprehensive view of proactive phishing detection: from understanding what today's infrastructure-based defences can detect, to exploring how many phishing attacks they miss, and exploring alternative detection strategies that operate independently of infrastructure-level assumptions.

6.2 Key Contributions

This thesis advances proactive phishing detection by first quantifying the visibility gap in current infrastructure-based defences and then introducing alternative detection methods that identify phishing at the propagation level, independent of infrastructure assumptions.

- **A Taxonomy of Phishing Domain Ownership:** This thesis introduced a taxonomy for classifying phishing websites by domain ownership, distinguishing among attacker-controlled, compromised, and third-party platforms. Using features such as registration history, search visibility, archival consistency, and hosting control, the taxonomy provides a framework to analyze the infrastructural diversity of phishing campaigns and quantify how much of the phishing landscape falls outside the visibility of existing infrastructure-based detection systems.
- **Large-Scale Analysis of Phishing Infrastructure using PhishXtract Dataset:** To evaluate the proposed taxonomy, we introduced *PhishXtract*, a large-scale dataset of phishing websites collected over a full year from multiple reporting sources. PhishXtract combines domain-level metadata, hosting details, search visibility information, and archival visibility, enabling the analysis of phishing infrastructures. Using this dataset, the thesis tracked how the distribution of phishing ownership changed over time across attacker-controlled, compromised, and third-party platforms. The results showed that a significant proportion of phishing attacks are deployed on legitimate third-party platforms, revealing that traditional proactive detection approaches fail to capture a large and growing segment of phishing campaigns. The dataset and accompanying analysis provide valuable insight for understanding how phishing campaigns are distributed across different hosting and ownership environments.
- **The TelePhish Dataset: Capturing Phishing Propagation on Telegram:** To extend phishing detection beyond infrastructure traces, we introduced *TelePhish*, a dataset of Telegram messages collected from public groups across three high-risk categories: Cryptocurrency, Games, and Darknet. *TelePhish* enabled us to analyze the propagation of phishing messages through social interactions, using behavioral metadata extracted directly from Telegram. It represents one of the first datasets for analyzing phishing propagation, enabling proactive phishing detection in real-time communication platforms.
- **Propagation-Aware Phishing Detection Models for Telegram:** Using the *TelePhish* dataset, this thesis developed and evaluated propagation-aware models for phishing detection in Telegram. By analyzing detection performance at different levels of granularity – from generalized models trained across all data to category-specific models targeting distinct phishing categories, including Cryptocurrency, Games, and Darknet – it demonstrated that finer-grained, category-specific modeling improves accuracy. These

models used propagation and engagement features, such as message frequency, user activity, and user participation, showing that phishing can be detected proactively at the platform level, even without relying on message text or URL analysis.

- **Systematic Evaluation of LLMs for Phishing Detection:** The thesis presented a systematic evaluation of LLMs for phishing detection in Telegram. It compares several representative LLMs (GPT-4o, DeepSeek v3, and LLaMA-3) across multiple prompting strategies. The experiments revealed that while LLMs may capture some linguistic patterns and URL cues of phishing in some cases, their detection ability is limited in real-time environments such as Telegram.
- **Comparative Analysis Between LLMs and Category-Specific Models:** Finally, the thesis compared the LLM-based approach to the category-specific propagation models, showing that specialized, feature-driven models outperform general-purpose LLMs in detecting phishing within Telegram's dynamic environment. This comparison showed that task-specific models tailored to the propagation behavior of phishing messages are more effective for platform-level detection.

6.3 Publications and Awards

6.3.1 Publications

The material presented in this thesis has resulted in multiple publications and submissions. Two papers have been published in international conferences, and the studies derived from Chapters 3 and 5 are under review for journal publications.

- Erfan, M., Branco, P. and Jourdan, G.V., 2024. Owned, Pwned or Rented: Whose Domain Is It?. In 2024 APWG Symposium on Electronic Crime Research (eCrime) (pp. 14-26). IEEE.https://doi.org/10.1109/eCrime66200.2024.00008 [34]
- Erfan, M., Branco, P., & Jourdan, G. V., 2025. Comparing Macro and Micro Approaches for Detecting Phishing Where It Spreads. In 2025 22nd Annual International Conference on Privacy, Security and Trust (PST). (accepted and to be published)
- Erfan, M., Branco, P., & Jourdan, G. V., 2025. Domains of Deception: Phishing Through the Lens of Ownership. Computers & Security. (revised and under review)
- Erfan, M., Branco, P., & Jourdan, G. V., 2025. Proactive Phishing Detection in Telegram: Can Large Language Models Keep Up?. Digital Threats: Research and Practice. (under review)

6.3.2 Awards

- Silver Medal for Best Paper (Second Place) at the 2024 APWG Symposium on Electronic Crime Research (eCrime)

6.4 Limitations

Although this thesis provides a comprehensive investigation of proactive phishing detection, several limitations should be acknowledged. These limitations come from the scope of data and methodological designs.

First, at the infrastructure level, the analysis relied on data aggregated from three major reporting sources: APWG, PhishTank, and OpenPhish. Although these feeds are reliable and widely used in phishing research, they reflect phishing attacks that have already been detected and reported, potentially under-representing stealthier phishing attacks that have not been found. Additionally, the extraction of several ownership-related features depended on external services and APIs, which could return incomplete responses. Missing data were excluded to preserve analytic integrity, but this filtering step may have introduced a selection bias.

Second, the ownership classification experiments were also limited by the availability of ground-truth labels. Manual labeling was conducted on a small, time-bounded subset of phishing websites because of the short lifespan of phishing websites and the limited annotation resources. The small training size and labeling inaccuracies could propagate to the automatically classified data.

Third, in the propagation level analysis, the Telegram-based analysis focused specifically on three public group categories – Cryptocurrency, Darknet, and Games – identified as high-risk categories for phishing activity. This focus enabled an in-depth investigation of phishing behavior within these communities, but may not fully generalize to other categories, such as investment or trading groups, where phishing strategies may differ. Additionally, the class imbalance between phishing and benign messages was addressed using a resampling method (NCL-ROS), which improved the phishing detection results but introduced synthetic data that might not perfectly reflect real-world messages. Moreover, ground-truth labeling relied on VirusTotal for URL verification; while scalable, this approach can produce occasional false negatives for new phishing links that were not yet detected by the service at the time of collection.

Lastly, in the LLM evaluation, the experiments were limited to prompting-based methods, including zero-shot and few-shot configurations, without fine-tuning. This may limit

performance since the models were not explicitly optimized for phishing detection. Fine-tuning could improve results, but these approaches were outside the computational and cost scope of this study. In addition, the study was limited to three models – DeepSeek v3, GPT-4o, and LLaMA-3.1 (8B) – selected because of their accessibility and multilingual support. Larger or newer variants, such as GPT-5 or LLaMA-70B, might show different performance patterns.

6.5 Future Work

Several directions come from this research that can further advance proactive phishing detection and address the limitations identified throughout this thesis.

First, future work should focus on expanding the diversity and scale of the datasets used for both infrastructure-level and propagation-level analysis. The infrastructure study relied on three major open phishing feeds. Incorporating additional reporting sources and expanding the *PhishXtract* dataset would provide a more representative view of attackers' behavior. Similarly, at the propagation level, expanding the *TelePhish* dataset to include more categories beyond Cryptocurrency, Games, and Darknet (e.g., investment, trading, and sale) would improve generalizability.

Second, there is strong potential to refine model design and training for phishing detection in Telegram. The category-specific models in Chapter 4 demonstrated that using propagation features can be highly effective for proactive detection. However, future research could explore extracting richer representations using deep learning architectures. These models could capture dependencies that traditional feature engineering might miss.

Third, fine-tuning LLMs for phishing detection represents another promising direction. While this thesis relied on prompting-based analysis to maintain efficiency, domain-specific fine-tuning could improve model performance. A hybrid architecture that integrates fine-tuned LLMs with feature-based classifiers could also be investigated. This approach would enable systems that can decide about both the linguistic and behavioral aspects of phishing activities, making them effective against evolving attack strategies.

Finally, this work opens the possibility of extending the propagation-based detection framework beyond Telegram to other social and communication platforms, such as Discord, Reddit, or WhatsApp. Each platform has distinct message formats, metadata, and dynamics, which could reveal platform-specific phishing patterns. A comparative study across these environments would help determine whether phishing propagation behaviors are consistent across social media platforms or platform-dependent.

Together, all these study directions can help identify and mitigate phishing attacks at the earliest possible stage.

6.6 Final Remarks

Phishing continues to evolve, exploiting trusted infrastructures that challenge traditional detection methods. A comprehensive understanding of phishing infrastructure can be achieved through a domain ownership taxonomy. By creating a domain ownership taxonomy and applying that to the *PhishXtract* dataset, we addressed our first research question and proved that the phishing landscape is not dominated by attacker-owned domains, as is commonly targeted by current proactive detection methods. Instead, the majority of current phishing activities leverage third-party platforms, which largely evade today's proactive systems because they do not exhibit the typical "malicious-at-birth" registration signals. For cybersecurity practitioners, this finding is a critical warning: current proactive detection methods mostly target malicious domains that do not align with the majority of hosting realities, creating a significant and exploitable visibility gap.

Furthermore, our research addressed the second research question by demonstrating that even without using traditional infrastructure clues, phishing can be identified early through the analysis of propagation behaviors. By focusing on behavioral metadata, we showed that malicious activity can be detected proactively at the platform level. This strategy provides a viable alternative for detection that operates independently of the hosting environment. For cybersecurity tool builders, this underscores the usefulness of propagation-aware systems that can analyze behavioral signals from social platforms like Telegram.

Finally, while LLMss offer linguistic analysis, our evaluation suggests they are not yet ideal for real-time detection tasks in social platforms such as Telegram. Considering the operational costs and the computational complexity associated with the fine-tuning process, specialized propagation-aware machine learning models remain the most efficient and effective choice for real-time platform defence. Ultimately, this work highlights the effectiveness of proactive phishing defence in detecting phishing where it spreads, not just where it is hosted.

References

[1] Wayback machine, 2001. https://wayback-api.archive.org/.

[2] Anti-phishing working group, 2003. https://apwg.org/.

[3] Phishtank - join the fight against phishing, 2006. https://www.phishtank.com/.

[4] Openphish - phishing intelligence, n.d. https://openphish.com/.

[5] Alsharif Abuadbba, Shuo Wang, Mahathir Almashor, Muhammed Ejaz Ahmed, Raj Gaire, Seyit Camtepe, and Surya Nepal. Towards web phishing detection limitations and mitigation. *arXiv preprint arXiv:2204.00985*, 2022.

[6] Rufai Ahmad and Sotirios Terzis. Understanding phishing in mobile instant messaging: a study into user behaviour toward shared links. In *International Symposium on Human Aspects of Information Security and Assurance*, pages 197–206. Springer, 2022.

[7] Essam Al Daoud, Laith Al Daoud, Mahmoud Asassfeh, Ala'a Al-Shaikh, Ala'a Saeb Al-Sherideh, and Suha Afaneh. Enhancing cybersecurity with transformers: Preventing phishing emails and social media scams. In *2024 IEEE Conference on Dependable and Secure Computing (DSC)*, pages 31–36. IEEE, 2024.

[8] Mashael AlSabah, Mohamed Nabeel, Yazan Boshmaf, and Euijin Choo. Content-agnostic detection of phishing domains using certificate transparency and passive dns. In *Proceedings of the 25th International Symposium on Research in Attacks, Intrusions and Defenses*, pages 446–459, 2022.

[9] Anti-Phishing Working Group (APWG). Global phishing survey 2015-2016: Trends and domain name use. Technical report, 2016. Accessed: 2024-12-15.

[10] Anti-Phishing Working Group (APWG). Phishing activity trends report, 2025, 2025.

[11] Frederick Barr-Smith and Joss Wright. Phishing with a darknet: Imitation of onion services. In *2020 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–13. IEEE, 2020.

[12] Jason Baumgartner, Savvas Zannettou, Megan Squire, and Jeremy Blackburn. The pushshift telegram dataset. In *Proceedings of the international AAAI conference on web and social media*, volume 14, pages 840–847, 2020.

[13] Jan Bayer, Ben Chukwuemeka Benjamin, Sourena Maroofi, Thymen Wabeke, Cristian Hesselman, Andrzej Duda, and Maciej Korczyński. Operational domain name classification: from automatic ground truth generation to adaptation to missing values. In *International Conference on Passive and Active Network Measurement*, pages 564–591. Springer, 2023.

[14] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection. In *30th USENIX security symposium (USENIX security 21)*, pages 3757–3774, 2021.

[15] Marzieh Bitaab, Alireza Karimi, Zhuoer Lyu, Ahmadreza Mosallanezhad, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, and Adam Doupé. Scamnet: Toward explainable large language model-based fraudulent shopping website detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 27841–27848, 2025.

[16] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[17] Leo Breiman. *Classification and regression trees*. Routledge, 2017.

[18] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[19] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling

the class imbalanced problem. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 475–482. Springer, 2009.

[20] Gavin C Cawley and Nicola LC Talbot. On over-fitting in model selection and subsequent selection bias in performance evaluation. *The Journal of Machine Learning Research*, 11:2079–2107, 2010.

[21] Yuan-Chen Chang and Esma Aïmeur. Chat or trap? detecting scams in messaging applications with large language models. In *2024 8th Cyber Security in Networking Conference (CSNet)*, pages 92–99. IEEE, 2024.

[22] Yuan-Chen Chang and Esma Aïmeur. Decept-chat. https://github.com/kibiddd/DECEPT-Chat, 2025. Accessed: June, 6 2025.

[23] Robin Chataut, Prashnna Kumar Gyawali, and Yusuf Usman. Can ai keep you safe? a study of large language models for phishing detection. In *2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0548–0554. IEEE, 2024.

[24] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002.

[25] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.

[26] Zhuo Chen, Yufeng Hu, Bowen He, Dong Luo, Lei Wu, and Yajin Zhou. Dissecting payload-based transaction phishing on ethereum. *arXiv preprint arXiv:2409.02386*, 2024.

[27] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli. Deltaphish: Detecting phishing webpages in compromised websites. In *European Symposium on Research in Computer Security*, pages 370–388. Springer, 2017.

[28] Corinna Cortes. Support-vector networks. *Machine Learning*, 1995.

[29] Thomas Cover and Peter Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[30] Ravindu De Silva, Mohamed Nabeel, Charith Elvitigala, Issa Khalil, Ting Yu, and Chamath Keppitiyagama. Compromised or {Attacker-Owned}: A large scale classification and study of hosting domains of malicious {URLs}. In *30th USENIX security symposium (USENIX security 21)*, pages 3721–3738, 2021.

[31] Nadide Bilge Doğan, Alp Barış Beydemir, Şerif Bahtiyar, and Umutcan Doğan. Dual-layered approach for malicious domain detection. In *2024 9th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE, 2024.

[32] Arthur Drichel, Vincent Drury, Justus von Brandt, and Ulrike Meyer. Finding phish in a haystack: A pipeline for phishing classification on certificate transparency logs. In *Proceedings of the 16th International Conference on Availability, Reliability and Security*, pages 1–12, 2021.

[33] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

[34] Mina Erfan, Paula Branco, and Guy-Vincent Jourdan. Owned, pwned or rented: Whose domain is it? In *2024 APWG Symposium on Electronic Crime Research (eCrime)*, pages 14–26. IEEE, 2024.

[35] Lonami Exo. Telethon: Telegram client library for python. https://docs.telethon.dev/en/stable/modules/client.html.

[36] Edona Fasllija, Hasan Ferit Enişer, and Bernd Prünster. Phish-hook: Detecting phishing certificates using certificate transparency logs. In *Security and Privacy in Communication Networks: 15th EAI International Conference, SecureComm 2019, Orlando, FL, USA, October 23–25, 2019, Proceedings, Part II 15*, pages 320–334. Springer, 2019.

[37] Sara E Garza and Satu Elisa Schaeffer. Community detection with the label propagation algorithm: a survey. *Physica A: Statistical Mechanics and its Applications*, 534:122058, 2019.

[38] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63:3–42, 2006.

[39] Sergio Gómez. Centrality in networks: finding the most important nodes. In *Business and consumer analytics: New ideas*, pages 401–433. Springer, 2019.

[40] Yufeng Gong and Umit Karabiyik. Forensics analysis of android social networking application: Tencent qq revisited. In *2024 International Symposium on Networks, Computers and Communications (ISNCC)*, pages 1–8. IEEE, 2024.

[41] Abdelhakim Hannousse and Salima Yahiouche. Web page phishing detection. Mendeley Data, V3, 2021.

[42] John A Hartigan and Manchek A Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979.

[43] Bowen He, Yuan Chen, Zhuo Chen, Xiaohui Hu, Yufeng Hu, Lei Wu, Rui Chang, Haoyu Wang, and Yajin Zhou. Txphishscope: Towards detecting and understanding transaction-based phishing on ethereum. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, pages 120–134, 2023.

[44] Fredrik Heiding, Bruce Schneier, Arun Vishwanath, Jeremy Bernstein, and Peter S Park. Devising and detecting phishing emails using large language models. *IEEE Access*, 12:42131–42146, 2024.

[45] P Hemalatha, N Ragapriya, V Sanjana, and K Shiva Bhavani. Extreme learning machine for spammer detection and fake user identification from twitter. *J. Crit. Rev*, 10(03):184–192, 2023.

[46] David W Hosmer Jr, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression*. John Wiley & Sons, 2013.

[47] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[48] FBI Internet Crime Complaint Center (IC3). Ic3 annual report, 2024, 2025.

[49] Liangxiao Jiang, Chen Qiu, and Chaoqun Li. A novel minority cloning technique for cost-sensitive learning. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(04):1551004, 2015.

[50] Takashi Koide, Naoki Fukushi, Hiroki Nakano, and Daiki Chiba. Chatspamdetector: Leveraging large language models for effective phishing email detection. In *International Conference on Security and Privacy in Communication Systems*, pages 297–319. Springer, 2024.

[51] György Kovács. smote-variants: a python implementation of 85 minority oversampling techniques. *Neurocomputing*, 366:352–354, 2019. (IF-2019=4.07).

[52] Matthieau Vic Kustanto, Hafizh Farhan Pratama, Aryaputra Ekalaya Gumay, Galih Dea Pratama, and Silviya Hasana. Detecting potential phishing attacks on twitter using transformers. In *2023 3rd International Conference on Smart Cities, Automation & Intelligent Computing Systems (ICON-SONICS)*, pages 137–141. IEEE, 2023.

[53] F Last, G Douzas, and F Bacao. Oversampling for imbalanced learning based on k-means and smote. arxiv 2017. *arXiv preprint arXiv:1711.00837*, 2, 2017.

[54] Jorma Laurikkala. Improving identification of difficult small classes by balancing class distribution. In *Conference on artificial intelligence in medicine in Europe*, pages 63–66. Springer, 2001.

[55] Thierry Lavoie, Foutse Khomh, Ettore Merlo, and Ying Zou. Inferring repository file structure modifications using nearest-neighbor clone detection. In *2012 19th Working Conference on Reverse Engineering*, pages 325–334. IEEE, 2012.

[56] Sophie Le Page and Guy-Vincent Jourdan. Victim or attacker? a multi-dataset domain classification of phishing attacks. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE, 2019.

[57] Jehyun Lee, Peiyuan Lim, Bryan Hooi, and Dinil Mon Divakaran. Multimodal large language models for phishing webpage detection and identification. In *2024 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–13. IEEE, 2024.

[58] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18(17):1–5, 2017.

[59] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[60] Hu Li, Peng Zou, Xiang Wang, and Rongze Xia. A new combination sampling method for imbalanced data. In *Proceedings of 2013 Chinese Intelligent Automation Conference: Intelligent Information Processing*, pages 547–554. Springer, 2013.

[61] Kyungchan Lim, Raffaele Sommese, Mattis Jonker, Ricky Mok, Doowon Kim, et al. Registration, detection, and deregistration: Analyzing dns abuse for phishing attacks. *arXiv preprint arXiv:2502.09549*, 2025.

[62] Feng Ling, Hao Yang, Yongcai Xiao, and Lei Hu. Meta gpt-based agent for enhanced phishing email detection. In *Proceedings of the 2024 14th International Conference on Communication and Network Security*, pages 78–84, 2024.

[63] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

[64] Daiping Liu, Zhou Li, Kun Du, Haining Wang, Baojun Liu, and Haixin Duan. Don't let one rotten apple spoil the whole barrel: Towards automated detection of shadowed domains. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 537–552, 2017.

[65] Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):2522–5839, 2020.

[66] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30, 2017.

[67] Durgasuany Maniraja and Asmaa Mahfoud Al-Hakimi. Hero cyber quest: Cybersecurity awareness training mobile application. In *2024 IEEE 15th Control and System Graduate Research Colloquium (ICSGRC)*, pages 114–119. IEEE, 2024.

[68] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[69] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoit Ampeau, and Andrzej Duda. Comar: classification of compromised versus maliciously registered domains. In *2020 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 607–623. IEEE, 2020.

[70] Massimo La Morgia, Alessandro Mei, and Alberto Maria Mongardini. TGDataset. https://github.com/SystemsLab-Sapienza/TGDataset, 2023.

[71] Muhammad Muzammil, Abisheka Pitumpe, Xigao Li, Amir Rahmati, and Nick Nikiforakis. The poorest man in babylon: A longitudinal study of cryptocurrency investment scams. In *Proceedings of the ACM on Web Conference 2025*, pages 1034–1045, 2025.

[72] Aleksandr Nahapetyan, Sathvik Prasad, Kevin Childs, Adam Oest, Yeganeh Ladwig, Alexandros Kapravelos, and Bradley Reaves. On sms phishing tactics and infrastructure. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 1–16. IEEE, 2024.

[73] Yevheniya Nosyk, Maciej Korczynski, Sourena Maroofi, Jan Bayer, Zul Odgerel, Andrzej Duda, Samaneh Tajalizadehkhoob, and Carlos Gañán. Infermal: Inferential analysis of maliciously registered domains.

[74] Seokjin Oh, Keonwoong Noh, Sunyub Kim, Dohyung An, and Woohwan Jung. Improving messenger phishing detection using heterogeneous phishing data. In *2025 International Conference on Electronics, Information, and Communication (ICEIC)*, pages 1–3. IEEE, 2025.

[75] Het Patel, Umair Rehman, and Farkhund Iqbal. Evaluating the efficacy of large language models in identifying phishing attempts. In *2024 16th International Conference on Human System Interaction (HSI)*, pages 1–7. IEEE, 2024.

[76] PhishLabs. Most phishing attacks use compromised domains and free hosting. Technical report, 2021. accessed: 2025-07-29.

[77] Andrii Podorozhniak, Vasyl Oliinyk, and Nataliia Liubchenko. Strategies for filtering unwanted comments in social media. In *2024 IEEE 5th KhPI Week on Advanced Technology (KhPIWeek)*, pages 1–4. IEEE, 2024.

[78] Kardeepa Ponnuchamy and N Subbulakshmi. Comparative analysis of eminent algorithms for detecting ham and spam contents in osn. In *2024 4th International Conference on Sustainable Expert Systems (ICSES)*, pages 702–709. IEEE, 2024.

[79] Fariza Rashid, Nishavi Ranaweera, Ben Doyle, and Suranga Seneviratne. Llms are one-shot url classifiers and explainers. *Computer Networks*, 258:111004, 2025.

[80] Hina Rashid, Hannan Bin Liaqat, Muhammad Usman Sana, Tayybah Kiren, Hanen Karamti, and Imran Ashraf. Framework for detecting phishing crimes on twitter using selective features and machine learning. *Computers and Electrical Engineering*, 124:110363, 2025.

[81] Konstantinos I Roumeliotis, Nikolaos D Tselikas, and Dimitrios K Nasiopoulos. Next-generation spam filtering: Comparative fine-tuning of llms, nlps, and cnn models for email spam classification. *Electronics*, 13(11):2034, 2024.

[82] Sayak Saha Roy, Unique Karanjit, and Shirin Nilizadeh. A large-scale analysis of phishing websites hosted on free web hosting domains. *arXiv preprint arXiv:2212.02563*, 2022.

[83] Sayak Saha Roy, Kiarash Khanmohammadi, and Others. DarkGram: A Large-Scale Analysis of Cybercriminal Activity Channels on Telegram. https://github.com/UTA-SPRLab/DarkGram, 2025.

[84] Asadullah Safi and Satwinder Singh. A systematic literature review on phishing website detection techniques. *Journal of King Saud University-Computer and Information Sciences*, 35(2):590–611, 2023.

[85] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine learning based phishing detection from urls. *Expert Systems with Applications*, 117:345–357, 2019.

[86] Yuji Sakurai, Takuya Watanabe, Tetsuya Okuda, Mitsuaki Akiyama, and Tatsuya Mori. Discovering httpsified phishing websites using the tls certificates footprints. In *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, pages 522–531. IEEE, 2020.

[87] Fai Ben Salamah, Marco A Palomino, Maria Papadaki, Matthew J Craven, and Steven Furnell. Evaluating the risks of human factors associated with social media cybersecurity threats. In *International Symposium on Human Aspects of Information Security and Assurance*, pages 349–363. Springer, 2023.

[88] Muhammad Salman, Muhammad Ikram, Nardine Basta, and Mohamed Ali Kaafar. Spallm-guard: Pairing sms spam detection using open-source and commercial llms. *arXiv preprint arXiv:2501.04985*, 2025.

[89] Tushar Sandhan and Jin Young Choi. Handling imbalanced datasets by partially guided hybrid sampling for pattern recognition. In *2014 22nd international conference on pattern recognition*, pages 1449–1453. IEEE, 2014.

[90] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[91] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C Schmidt, and Matthias Wählisch. The rise of certificate transparency and its implications on the internet ecosystem. In *Proceedings of the Internet Measurement Conference 2018*, pages 343–349, 2018.

[92] Lloyd S Shapley et al. A value for n-person games. 1953.

[93] Sarjita Soo, Nilanjana Saha, Amrita Namtirtha, and Animesh Dutta. Identification of phishing mechanism of the mastermind of anomaly community in the social networks. In *2025 17th International Conference on COMmunication Systems and NETworks (COMSNETS)*, pages 926–930. IEEE, 2025.

[94] Nan Sun, Guanjun Lin, Junyang Qiu, and Paul Rimba. Near real-time twitter spam detection with machine learning techniques. *International Journal of Computers and Applications*, 44(4):338–348, 2022.

[95] TGStat. Tgstat – telegram analytics service. https://tgstat.com/.

[96] Robert L Thorndike. Who belongs in the family? *Psychometrika*, 18(4):267–276, 1953.

[97] Ivan Tomek. Two modifications of cnn. *IEEE Transactions on Systems, Man, and Cybernetics*, 1976.

[98] Fouad Trad and Ali Chehab. Prompt engineering or fine-tuning? a case study on phishing detection with large language models. *Machine Learning and Knowledge Extraction*, 6(1):367–384, 2024.

[99] Fouad Trad and Ali Chehab. To ensemble or not: Assessing majority voting strategies for phishing detection with large language models. In *International Conference on Intelligent Systems and Pattern Recognition*, pages 158–173. Springer, 2024.

[100] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[101] VirusTotal. Virustotal. https://www.virustotal.com/. Accessed: 2024-12-15.

[102] Hiroki Watanabe, Kohei Ichihara, and Takumi Aita. Vellet: Verifiable embedded wallet for securing authenticity and integrity. In *2024 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 254–258. IEEE, 2024.

[103] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[104] Yu Hui Xu, Hui Li, Lu Ping Le, and Xiao Yun Tian. Neighborhood triangular synthetic minority over-sampling technique for imbalanced prediction on small samples of chinese tourism and hospitality firms. In *2014 seventh international joint conference on computational sciences and optimization*, pages 534–538. IEEE, 2014.

[105] Jun Zhang, Peiqiao Wu, Jeffrey London, and Dan Tenney. Benchmarking and evaluating large language models in phishing detection for small and midsize enterprises: A comprehensive analysis. *IEEE Access*, 2025.

[106] Mingming Zhang, Xiang Li, Baojun Liu, Jianyu Lu, Yiming Zhang, Jianjun Chen, Haixin Duan, Shuang Hao, and Xiaofeng Zheng. Detecting and measuring security risks of hosting-based dangling domains. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 7(1):1–28, 2023.

APPENDICES

Appendix A

Oversampling Results for Domain Ownership Classification Approach

Table A.1: Comparison of classification performance under different oversampling strategies. Results are shown as mean ± standard deviation over 5 folds CV.

Method	Class	Precision	Recall	F1-score	Specificity	G-Mean	MCC
Random Oversampling	Attacker	0.984 ± 0.009	0.977 ± 0.005	0.980 ± 0.005	0.995 ± 0.003	0.986 ± 0.003	0.974 ± 0.007
	Compromised	0.735 ± 0.054	0.785 ± 0.056	0.755 ± 0.008	0.991 ± 0.003	0.881 ± 0.030	0.750 ± 0.008
	Third-Party	0.992 ± 0.004	0.993 ± 0.003	0.992 ± 0.001	0.980 ± 0.010	0.986 ± 0.004	0.971 ± 0.004
SMOTE	Attacker	0.983 ± 0.010	0.978 ± 0.008	0.981 ± 0.005	0.995 ± 0.003	0.986 ± 0.002	0.975 ± 0.006
	Compromised	0.702 ± 0.055	0.791 ± 0.077	0.740 ± 0.039	0.988 ± 0.003	0.887 ± 0.028	0.726 ± 0.026
	Third-Party	0.993 ± 0.005	0.989 ± 0.004	0.991 ± 0.002	0.981 ± 0.010	0.984 ± 0.004	0.964 ± 0.006
No Oversampling	Attacker	0.985 ± 0.009	0.979 ± 0.008	0.981 ± 0.005	0.995 ± 0.003	0.987 ± 0.004	0.975 ± 0.006
	Compromised	0.761 ± 0.062	0.754 ± 0.085	0.751 ± 0.028	0.992 ± 0.003	0.864 ± 0.047	0.747 ± 0.028
	Third-Party	0.991 ± 0.005	0.992 ± 0.003	0.992 ± 0.002	0.977 ± 0.014	0.984 ± 0.006	0.970 ± 0.006

Class imbalance posed a challenge in our phishing domain classification task, with compromised domains being underrepresented compared to attacker-owned and third-party-hosted domains. To explore whether oversampling could mitigate this imbalance and improve detection performance – especially for the compromised class – we conducted a series of experiments using the Random Forest classifier (the best-performing model identified in Chapter 3, Section 3.3.2). Specifically, we applied two widely used oversampling techniques: Random Oversampling and SMOTE for this purpose.

We evaluated both methods across a range of resampling configurations. These included (1) the "not majority" setting, which balances all classes except the majority class, and (2) fixed target counts for the minority class (compromised domains), with resampling

levels of 500, 1000, 1500, and 2000 instances. Our goal was to assess whether increasing the representation of the compromised class in the training data could improve its classification results without sacrificing overall model performance. The best configuration for Random Oversampling was observed when the number of compromised domains was increased to 1000, while SMOTE achieved its best performance at the 500-sample level. However, as shown in Table A.1, despite minor improvements in recall for the compromised class, oversampling generally led to reduced precision. Ultimately, despite slight recall improvements for compromised domains, no oversampling method significantly outperformed the no-oversampling baseline. These results suggest that the classifier had already captured most of the meaningful patterns in the data, and that oversampling offered limited additional benefit.

Appendix B

Performance Metrics Across Different Thresholds for the Rule-Based Taxonomy

Table B.1 presents the performance metrics for the rule-based taxonomy that we have developed. The table shows the precision, recall, and f1-score for three different categories: Attacker Domain, Third-Party Platform, and Compromised Domain. These metrics are calculated at five threshold levels: 10%, 20%, 40%, 60%, and 80% for HTML structure similarity checks between two web pages. Based on the results, As the threshold value increases from 10% to 80%, we can observe some interesting trends in the performance metrics:

- For the Attacker Domain class, the precision remains consistently high (around 0.986) across all the thresholds, indicating that the rule-based approach is very effective at identifying true attacker domains. The recall, on the other hand, increases slightly from 0.939 at 10% to 0.945 at 80%, suggesting that the higher thresholds can capture a few more true attacker domains, but at the cost of a small decrease in overall precision.
- The Third-Party Platform class shows an opposite trend, where the precision increases from 0.970 at 10% to 0.976 at 80%, while the recall decreases from 0.998 to 0.988. This indicates that the higher thresholds result in fewer false positives but also missing some true third-party platforms.
- The Compromised Domain class exhibits the most significant changes across the thresholds. As the threshold increases, the precision decreases from 0.850 at 10% to 0.586 at 80%, while the recall increases from 0.481 to 0.613. This suggests that the lower thresholds are more effective at identifying compromised domains, but they also tend to include

more false negatives in the classification.

Overall, the 10% threshold appears to provide the best trade-off between precision and recall across the three classes. It maintains high precision for the "Attacker Domains" and "Third-Party Platform" categories while achieving the highest f1-score for the "Compromised Domains" among the tested thresholds. This makes the 10% threshold the most suitable choice for the rule-based taxonomy, as it maximizes the overall performance of this approach.

Table B.1: Results across different thresholds for the rule-based taxonomy

Class	Threshold	Precision	Recall	F1-Score
Attacker Domain	10%	0.986	0.939	0.962
	20%	0.986	0.939	0.962
	40%	0.986	0.942	0.963
	60%	0.985	0.943	0.964
	80%	0.985	0.945	0.964
Third-Party Platform	10%	0.970	0.998	0.984
	20%	0.971	0.995	0.983
	40%	0.973	0.994	0.984
	60%	0.975	0.991	0.983
	80%	0.976	0.988	0.982
Compromised Domain	10%	0.850	0.481	0.614
	20%	0.746	0.500	0.599
	40%	0.713	0.538	0.613
	60%	0.639	0.585	0.611
	80%	0.586	0.613	0.599