PhishIntel: Toward Practical Deployment of Reference-Based Phishing Detection

Yuexin Li
National University of Singapore
Singapore
yuexinli@u.nus.edu

Hiok Kuek Tan
National University of Singapore
Singapore
e1357001@u.nus.edu

Qiaoran Meng
National University of Singapore
Singapore
qiaoran@nus.edu.sg

Mei Lin Lock
NCS Cyber Special Ops R&D
Singapore
meilin.lock@ncs.com.sg

Tri Cao
National University of Singapore
Singapore
tricao@nus.edu.sg

Shumin Deng
National University of Singapore
Singapore
shumin@nus.edu.sg

Nay Oo
NCS Cyber Special Ops R&D
Singapore
nay.oo@ncs.com.sg

Hoon Wei Lim
NCS Cyber Special Ops R&D
Singapore
hoonwei.lim@ncs.com.sg

Bryan Hooi
National University of Singapore
Singapore
dcsbhk@nus.edu.sg

Abstract

Phishing is a critical cyber threat, exploiting deceptive tactics to compromise victims and cause significant financial losses. While reference-based phishing detectors (RBPDs) have achieved notable advancements in detection accuracy, their real-world deployment is hindered by challenges such as high latency and inefficiency in URL analysis. To address these limitations, we present *PhishIntel*, an end-to-end phishing detection system for real-world deployment. PhishIntel intelligently determines whether a URL can be processed immediately or not, segmenting the detection process into two distinct tasks: a fast task that checks against local blacklists and result cache, and a slow task that conducts online blacklist verification, URL crawling, and webpage analysis using an RBPD. This fast-slow task system architecture ensures low response latency while retaining the robust detection capabilities of RBPDs for zero-day phishing threats. Furthermore, we develop two downstream applications based on PhishIntel: a phishing intelligence platform and a phishing email detection plugin for Microsoft Outlook, demonstrating its practical efficacy and utility.

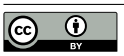
CCS Concepts

• **Security and privacy → Intrusion detection systems**; • **Software and its engineering → Real-time systems software**.

Keywords

Phishing Detection System, Task Queue, Real-Time Systems

ACM Reference Format:
Yuexin Li, Hiok Kuek Tan, Qiaoran Meng, Mei Lin Lock, Tri Cao, Shumin Deng, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2025. PhishIntel: Toward Practical Deployment of Reference-Based Phishing Detection. In *Companion Proceedings of the ACM Web Conference 2025 (WWW Companion '25), April 28-May 2, 2025, Sydney, NSW, Australia.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3701716.3715192



WWW Companion '25, Sydney, NSW, Australia

ACM ISBN 979-8-4007-1331-6/25/04
https://doi.org/10.1145/3701716.3715192

1 Introduction

The vast majority of cyber attacks originate from phishing[3]. For instance, phishing attacks deceive users into giving away their credentials, such as via malicious emails with links to webpages that impersonate known brands. Such illegal campaigns have resulted in tremendous financial loss, establishing phishing as one of the most severe social engineering threats today. Consequently, deploying effective and efficient phishing detection systems is essential to safeguard network users from such harm.

Recent advancements in phishing detection have predominantly focused on enhancing detection effectiveness. Among these, reference-based phishing detectors (RBPDs) have garnered significant research attention [4–8]. These detectors rely on website content analysis, such as URLs, screenshots, and HTML, to determine phishing activity. Specifically, RBPDs classify phishing webpages by first determining the brand that the webpage appears to contain based on its screenshot and HTML, called its 'brand intention'. If a webpage exhibits a certain brand intention but its domain fails to match any legitimate domains of that brand, it is classified as phishing. RBPDs rely on the fundamental invariant that attackers cannot create a webpage whose brand intention matches its actual domain: e.g., while attackers can mimic the PayPal brand, they cannot create a webpage with the actual PayPal domain (www.paypal.com). This enables RBPDs to achieve high precision and robustness while providing clear explanations for their predictions.

Despite their robustness and explainability, several challenges remain when deploying RBPDs in real-world scenarios, significantly affecting the user experience. **(1) Latency**. RBPDs fundamentally rely on webpage content for their analysis, but in an email setting, we only have direct access to URLs (which appear in the emails). While integrating a webpage crawler can address this limitation by fetching the required data, it significantly increases latency. We empirically find that loading a single webpage using a crawler can take

up to ten seconds, creating substantial delays. Such prolonged wait times severely hinder user experience, particularly when handling a large volume of requests. **(2) Efficiency**. Processing all incoming URL requests indiscriminately can lead to unnecessary inference costs, especially for URLs that have already been publicly or locally analyzed. Therefore, a systematic and optimized phishing detection system architecture is needed.

To tackle these practical challenges, we introduce *PhishIntel*, an end-to-end phishing detection system for real-world deployment. The key design principle of PhishIntel is to process the URLs with different tiers. Therefore, we design a fast-slow task system architecture. Specifically, the *fast task* checks the URLs against publicly available blacklists and a result cache that stores the result of an RBPD. If a URL is in the blacklist or the cache, its phishing detection result can be immediately returned; otherwise, it will be sent to the slow task queue pending additional processing. The *slow task*, on the other hand, handles tasks that take much longer processing time. This includes online blacklist verification, webpage crawling, and webpage content analysis through an RBPD. Once a result is obtained, it will be added back to the result cache for future reference, instead of analyzing it again.

The complete PhishIntel system design ensures low response latency while retaining the robust detection capabilities of RBPDs for zero-day phishing threats. To validate its practical utility and effectiveness, we demonstrate PhishIntel with a phishing intelligence platform and a phishing email detection plugin in an enterprise-level email system. We also provide a case study to understand the effect of major system components. In summary, we make the following contribution:

- **Phishing Detection System**. We propose PhishIntel, an end-to-end phishing detection system featuring real-time response while retaining the ability to detect zero-day phishing.
- **Practical Applications**. We develop two applications to demonstrate the practical utility of PhishIntel: (1) a phishing intelligent platform to analyze phishing URLs and track emerging phishing trends, and (2) a phishing email detection plugin integrated into Microsoft Outlook for safeguarding users from email phishing.

2 Methodology

2.1 Problem Statement

Phishing detection is essentially a binary classification task. Given a URL u, a phishing detector $\mathcal{D}$ analyzes the information associated with u, such as its webpage screenshot s and HTML h, and outputs a detection result r, classifying it as either benign or phishing.

In real-world scenarios, a deployed phishing detection system $\mathcal{F}$ typically processes a large volume of URLs. Formally, each incoming batch of URLs can be represented as a list $\mathcal{U} = \{u_1, u_2, ..., u_n\}$, with the corresponding detection results denoted as $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, where n is the number of URLs in this batch. Their associated webpage screenshots $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ and HTML $\mathcal{H} = \{h_1, h_2, ..., h_n\}$ are not initially available and must be fetched using an additional webpage crawler C. However, integrating C introduces significant runtime overhead, leading to increased system latency. This latency is particularly problematic in real-world applications, such as phishing email detection, where high throughput and responsiveness are critical for a positive user experience.

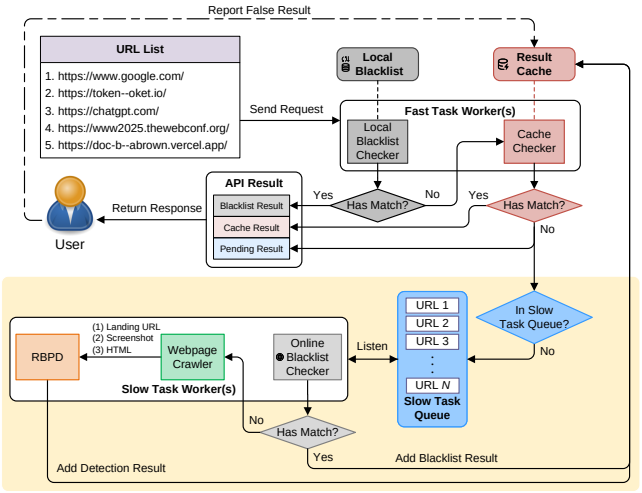


Figure 1: An overview of PhishIntel: URLs are first processed through FTW, with those failing to obtain results forwarding to the slow task queue for pending STW analysis.

To address this challenge, we propose a fast-slow task system architecture. This approach categorizes URLs into those that can be processed instantly, providing immediate results, and those requiring further analysis, thereby balancing efficiency and zero-day phishing detection ability in large-scale URL streams.

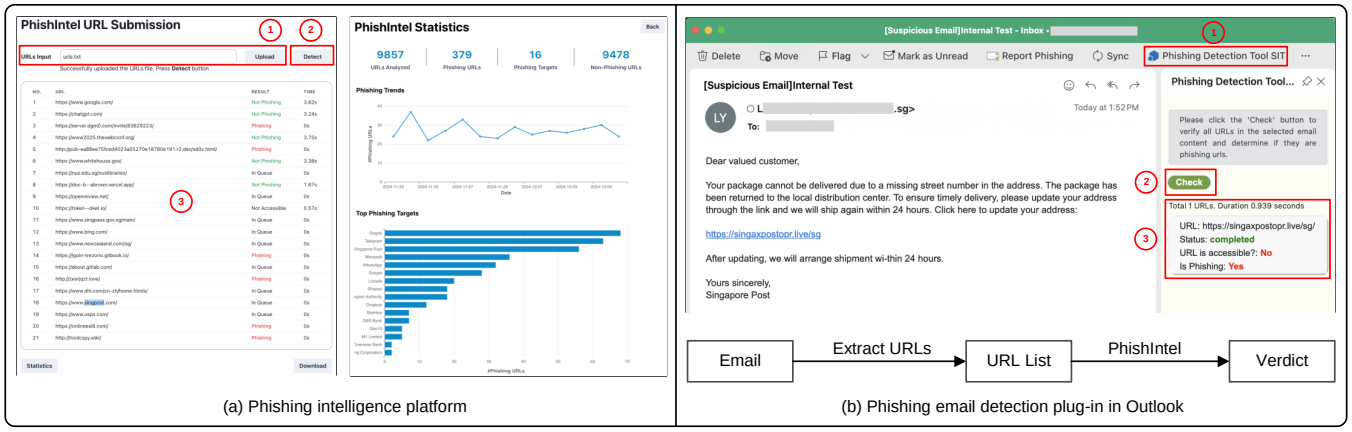
2.2 Overview

Figure 1 offers an overview of *PhishIntel*, our proposed phishing detection system designed with a fast-slow task architecture. The system is composed of three primary components: (1) the **Fast Task Worker (FTW)**, which processes incoming URL lists and promptly returns preliminary analysis results; (2) a **Slow Task Queue**, which temporarily stores URLs that the FTW cannot handle for more comprehensive analysis; and (3) the **Slow Task Worker (STW)**, which retrieves URLs from the Slow Task Queue to conduct in-depth phishing analysis.

2.3 Fast Task Worker

End-to-end phishing URL detection using state-of-the-art RBPDs is often a time-consuming task[4–8], particularly when integrated with a webpage crawler to obtain necessary webpage data for analysis. For instance, we empirically observe that the webpage crawler can take up to 10 seconds to fetch webpage content, and a recent RBPD can spend another 10 seconds analyzing the webpage content[4, 8]. In contrast, users in real-world scenarios generally expect a response within a reasonable timeframe—typically around 2 seconds. Therefore, a straightforward combination of a crawler and an RBPD fails to meet the practical requirements for deployment.

Our FTW is designed to mitigate latency issues by enhancing the system efficiency. An FTW is composed of two components: a local blacklist checker and a result cache checker. These components enable the system to efficiently categorize URLs into those with immediately available results and those requiring further analysis. Specifically, if a URL u is found in the local blacklist, the FTW will



(a) Phishing intelligence platform

(b) Phishing email detection plug-in in Outlook

Figure 2: Illustration of two downstream applications built upon PhishIntel: (a) is a phishing intelligence platform, and (b) is a phishing email detection plugin in Outlook.

classify it as phishing and promptly return the result. If u is not present in the local blacklist but matches an entry in the result cache, the FTW directly retrieves and returns the cached result. Conversely, if u fails both checks, the FTW returns a 'pending' result, indicating that u will be forwarded to the slow task queue for further analysis. All these URL-matching processes are runtime-efficient, ensuring rapid processing and minimizing response latency.

The local blacklist is initialized using publicly available URL blacklists, effectively eliminating redundant analyses of known phishing URLs. To maintain its accuracy and relevance, a dedicated service process periodically updates the local blacklist by synchronizing with the public blacklist provider. The result cache is continuously updated with outputs from the slow task worker, which will be discussed in detail in the subsequent section. We additionally provide a means for users to provide their feedback on any detection results, to correct missing or inaccurate results. User feedback will be manually checked and the corrected entries will be placed in the cache.

2.4 Slow Task Worker

The STW is responsible for performing additional analysis on the URLs in the slow task queue, which often requires significantly more processing time than the FTW does. It operates by continuously listening to the slow task queue. Whenever the STW is available, it will fetch a URL from the slow task queue to perform analysis, if the slow task queue is not empty.

Specifically, an STW consists of three major components, an online blacklist checker, a webpage crawler, and an RBPD. The online blacklist checker initially checks u against proprietary, non-publicly accessible blacklists via their URL matching APIs. This serves as an additional layer of the URL filtering process to complement FTW. If a match is found, the STW records a phishing result in the result cache. Otherwise, the webpage crawler retrieves the webpage content of u, including its screenshot and HTML. This content is subsequently analyzed by the RBPD, which generates a detection result that is also stored in the result cache.

2.5 Implementation Details

We build the FTW with Quart[1] and Gunicorn[2]. The local blacklist is instantiated with the PhishTank database[3], while the result cache is implemented using Redis[4]. The STW and slow task queue are built with Celery[5]. We use Google Web Risk [6] as the online blacklist checker, Playwright[7] as the webpage crawler, and the KnowPhish Detector[4] as the RBPD.

To optimize resource utilization and enhance request throughput, PhishIntel is deployed with 8 FTWs and 4 STWs. The complete system operates on an AWS EC2 instance with an NVIDIA L4 Tensor Core 24GB GPU.

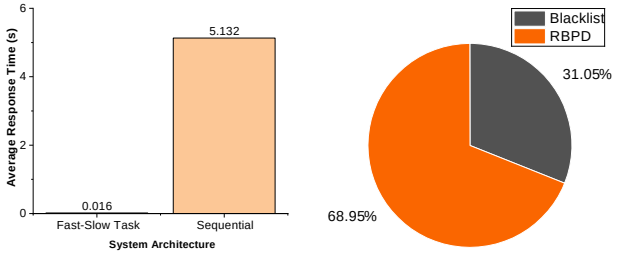
3 Demonstrations

We demonstrate the utility of PhishIntel via two applications: (1) phishing intelligence platform, and (2) phishing email detection, as shown in Figure 2.

3.1 Phishing Intelligence Platform

The first application we developed is a web-based phishing intelligence platform that primarily offers two core functionalities: phishing URL analysis and a phishing statistics dashboard. Users can conduct phishing detection on a list of URLs by uploading a text file containing the URLs through the `Upload` button on the webpage. After uploading, users can click the `Detect` button to retrieve phishing detection results for the provided URLs. Figure 2(a) illustrates the phishing analysis results for 21 URLs. These results are directly returned by the FTWs, leveraging the local blacklist and result cache. URLs requiring further analysis are labeled as `In Queue`, indicating that they are still pending the STW processing.

[1]https://github.com/pallets/quart
[2]https://gunicorn.org/
[3]https://phishtank.org/
[4]https://redis.io/
[5]https://github.com/celery/celery
[6]https://cloud.google.com/security/products/web-risk
[7]https://playwright.dev/python/



(a) Comparison of the average response time with different system architectures.

(b) Distribution of the phishing reports from blacklist and RBPD.

Figure 3: Performance evaluation of PhishIntel.

This platform also features a dashboard page that visualizes phishing detection statistics derived from the result cache in PhishIntel, offering deeper insights into the current phishing attack landscape. For example, users can explore the number of unique phishing URLs, targeted entities, and recent trends, such as the daily detection count of phishing URLs and the most frequently targeted entities. Deployers can utilize this data to notify users about emerging phishing trends, safeguarding them from such attacks through regular reminders and alerts.

3.2 Phishing Email Detection

We also integrate PhishIntel into an enterprise-level email system, as a phishing email detection plugin in Outlook. Specifically, we create a button in the tool menu. When users encounter a suspicious email and would like to report it for analysis, they can click this button to activate the phishing email detection tool. After clicking the `Check` button in the tool, it will automatically extract all the URLs in the selected email as a list, and send it to PhishIntel. For the particular case in Figure 2(b), the only URL in the email is classified as phishing, hence, the email is deemed as a phishing email.

3.3 Performance Evaluation

We further evaluate two key performance indicators of PhishIntel: the system response latency and the impact of blacklist filtering. For response latency, we compare the average response times of two architectural approaches: a fast-slow task design and a simple sequential pipeline, where the latter generates responses only after all components have completed their respective analyses. For blacklist filtering, we measure the proportion of phishing reports derived from blacklists and the RBPD. To conduct this evaluation, we input 1k randomly selected benign URLs from Tranco[8] and 1k randomly selected phishing URLs from OpenPhish[9] to PhishIntel.

The results, presented in Figure 3, demonstrate that the fast-slow task architecture significantly reduces system latency. Additionally, the blacklist successfully filters out a substantial portion of URLs that do not require further analysis. These findings affirm the overall effectiveness of PhishIntel's design.

[8]https://tranco-list.eu/
[9]https://openphish.com/

4 Related Work

The past few decades have witnessed significant advancements in phishing detection methodologies. Early research primarily focused on blacklist-based[9] and heuristics-based[2] approaches, while more recent studies have shifted towards leveraging machine learning models [10]. A prominent framework in this area is RBPD, which leverages computer vision[5, 7, 8] and natural language processing models[4] to analyze webpage information. The booming development of large language models also inspires the utility of LLM agents to extract phishing-related knowledge[6] and understand the webpage content[1], as a complement to existing RBPDs.

Despite these advancement efforts, to the best of our knowledge, no prior work has addressed the practical deployment of RBPDs. This paper thus presents a novel phishing detection system design, serving as a foundational demonstration to inform and advance the development of more deployment-focused phishing detectors.

5 Conclusion

In this paper, we introduce PhishIntel, an end-to-end phishing detection system designed for real-world deployment. PhishIntel integrates RBPDs, blacklists, and user reports collaboratively, enhancing both the effectiveness and efficiency of phishing detection. Its fast-slow task system architecture ensures real-time responsiveness and improves user experience. We demonstrate its practical utility through two applications: a phishing intelligence platform and a phishing email detection plugin in Microsoft Outlook. Looking ahead, we plan to optimize PhishIntel for larger-scale deployments.

Acknowledgments

This work was supported in part by the National Research Foundation Singapore, NCS Pte. Ltd. and National University of Singapore under the NUS-NCS Joint Laboratory (Grant A-0008542-00-00).

References

[1] Tri Cao, Chengyu Huang, Yuexin Li, Huilin Wang, Amy He, Nay Oo, and Bryan Hooi. 2024. PhishAgent: A Robust Multimodal Agent for Phishing Webpage Detection. arXiv:2408.10738 [cs.CR]
[2] Yan Ding, Nurbol Luktarhan, Keqin Li, and Wushour Slamu. 2019. A Keyword-Based Combination Approach for Detecting Phishing Webpages. *Comput. Secur.* 84, C (jul 2019).
[3] Introducing Cloudflare's 2023 phishing threats report [n. d.]. https://blog.cloudflare.com/2023-phishing-report/.
[4] Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2024. KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection. In *USENIX Security '24*.
[5] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A Hybrid Deep Learning Based Approach to Visually Identify Phishing Webpages. In *USENIX Security '21*.
[6] Ruofan Liu, Yun Lin, Xiwen Teoh, Gongshen Liu, Zhiyong Huang, and Jin Song Dong. 2024. Less Defined Knowledge and More True Alarms: Reference-based Phishing Detection without a Pre-defined Reference List. In *USENIX Security '24*.
[7] Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. 2022. Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision Based Approach. In *USENIX Security '22*.
[8] Ruofan Liu, Yun Lin, Yifan Zhang, Penn Han Lee, and Jin Song Dong. 2023. Knowledge Expansion and Counterfactual Interaction for Reference-Based Phishing Detection. In *USENIX Security '23*. Anaheim, CA.
[9] Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. 2007. The Ghost in the Browser: Analysis of Web-based Malware. In *First Workshop on Hot Topics in Understanding Botnets (HotBots 07)*.
[10] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-Scale Automatic Classification of Phishing Pages. In *NDSS '10*.