

PHISH-STOP: PHISHING DETECTION PIPELINE USING EMAIL HEADERS AND LOGS

Chandrakala. S¹, Sathishkumar. P², Balamurugan. S³, Jaysankar. P⁴, Kishore. M⁵

¹Assistant Professor, Dept. of Cybersecurity, Paavai Engineering College

²Assistant Professor, Dept. of Cybersecurity, Paavai Engineering College

³Student, Dept. of Cybersecurity, Paavai Engineering College

⁴Student, Dept. of Cybersecurity, Paavai Engineering College

⁵Student, Dept. of Cybersecurity, Paavai Engineering College

Abstract - The Phishing Detection Pipeline Using Email Headers and Logs is a lightweight, browser-based extension that detects phishing attempts in real time. It analyzes metadata such as sender information, domain reputation, message routing, and suspicious patterns in email headers. This system identifies malicious activity with high accuracy. To keep up with changing phishing tactics, the extension uses a real-time dataset that updates daily from trusted sources like PhishTank, OpenPhish, Abuse.ch, Google Safe Browsing, and SpamAssassin rules. The data is stored locally using IndexedDB and refreshes automatically, which ensures offline access and quick response times. With Firebase integration for dataset updates and user testing, the pipeline provides an efficient, scalable, and user-friendly way to prevent phishing.

Key Words: Phishing Detection, Email Headers, Logs, Real-Time Detection, Browser Extension, Cybersecurity, IndexedDB, Firebase, Threat Intelligence, PhishTank, OpenPhish

1. INTRODUCTION

In the modern digital ecosystem, the connections that define communication and commerce have also increased challenges related to privacy and identity protection. Phishing attacks, in particular, remain one of the most common and damaging cybersecurity threats. They often deceive users through fake emails. Every online interaction, from social networking to e-commerce, adds to a digital footprint that bad actors can exploit. These scattered digital footprints lead to data misuse, identity theft, and profiling.

Traditional spam filters try to offer some protection, but they often depend on static content analysis or blacklists. Attackers are always changing their tactics, using tricks like URL shortening or Unicode domains to get around these defenses. This creates a serious gap in security, leaving users exposed.

The proposed Phishing Detection Pipeline, "Phish-Stop," improves protection by focusing on a different and more reliable set of indicators: email headers and logs. By analyzing technical metadata, like suspicious sender addresses, domain spoofing, unusual routing paths, and

harmful links, the system can automatically flag threats. This approach includes real-time dataset integration, using Firebase for dataset distribution and IndexedDB for local caching. This ensures daily updates without slowing down performance. This hybrid model provides an effective, scalable, and user-friendly solution for modern phishing detection.

2. PRIMARY OBJECTIVES OF THE STUDY

- Automation constitutes the foundational pillar of the system's architecture. The primary focus is to develop a fully autonomous detection mechanism capable of operating efficiently with minimal human supervision. This is realized through the seamless integration of continuously updated phishing intelligence datasets aggregated from globally recognized threat intelligence sources. Such dynamic incorporation ensures that the system not only scales effectively in handling extensive data streams but also evolves in real time to counter emerging tactics, techniques, and procedures employed by cyber adversaries.

- Achieving equilibrium between analytical accuracy and computational efficiency is critical to the system's success. The detection pipeline is rigorously optimized to reduce false positive occurrences, thereby ensuring that alerts maintain high fidelity and operational reliability. Excessive false alerts often contribute to alert fatigue, diminishing user responsiveness and trust; hence, precision in detection is a paramount design criterion. Concurrently, the system is engineered to maintain a minimal computational footprint, ensuring seamless operation as a browser-based extension without introducing perceptible latency, excessive memory utilization, or a degraded user experience.

- Resilience is an intrinsic element of the system's design philosophy, ensuring continuous protection regardless of network stability. By leveraging IndexedDB for client-side data management, the system offers persistent, secure, and offline operational capabilities. This architectural choice enables the pipeline to sustain its analytical capacity and threat mitigation functions by utilizing cached intelligence—such as known phishing signatures, behavioral heuristics, and

previously acquired models—even in the absence of an active internet connection.

- The practical implementation of the proposed system defines its ultimate research value. The project seeks to consolidate intelligent automation, high-precision detection, and offline resilience within a unified, user-oriented browser extension. Unlike theoretical constructs, this framework aspires to deliver a tangible, deployable cybersecurity solution that fortifies individuals and organizations against evolving phishing threats. By embedding the defensive mechanism directly within the user's browsing environment, the system ensures immediate, continuous, and adaptive protection against the ever-expanding landscape of phishing campaigns.:

3. LITERATURE SURVEY

A comprehensive review of existing academic and technical literature reveals a diverse landscape of methodologies developed for phishing detection. These approaches range from statistical analysis to decentralized verification systems, each presenting a unique set of advantages and inherent limitations. Understanding this landscape is crucial for identifying gaps in current research and effectively positioning the contributions of the present project. The most prominent strategies are detailed below.

3.1 Machine Learning and Header Analysis

A significant body of research, exemplified by the work of A. Jain, P. Sharma, and K. Mehta (2022), has focused on applying machine learning (ML) models to email header feature extraction. This technique treats the email header—the metadata that documents an email's path from sender to recipient—as a digital fingerprint. By training algorithms on features such as the sender's IP address, the results of authentication protocols like SPF (Sender Policy Framework) and DKIM (DomainKeys Identified Mail), and various header anomalies (e.g., mismatched "From" and "Reply-To" fields), these models learn to distinguish between legitimate and malicious correspondence. The primary strength of this approach lies in its ability to identify sophisticated spoofing attempts that might otherwise appear convincing. However, its principal disadvantage is its critical dependency on the size and quality of the labeled training dataset. An improperly balanced or insufficiently large dataset can cause the model to develop biases, resulting in poor generalization to new, unseen threats and diminishing its real-world accuracy.

3.2 Log-Based Behavioral Analysis

Another prevalent approach, explored by M. Lee and S. Kim (2023), involves real-time log analysis combined with heuristic rules and pattern recognition. Instead of analyzing the content of a message, this method focuses on the behavioral context surrounding it. It scrutinizes system and server logs to identify anomalous patterns, such as multiple

failed login attempts followed by a successful one from a new geographic location, or an unusual spike in email volume from a single account. By establishing a baseline of normal user behavior, the system can flag deviations that suggest an account compromise or an internal phishing attack. While this technique is powerful for detecting threats in real time, its main drawback is a tendency to suffer from high false-positive rates, particularly during its initial training and calibration phase. Benign but unusual activities can be mistakenly flagged as malicious until the system has gathered enough data to refine its understanding of a user's unique behavior.

3.3 Threat Intelligence Integration

The strategy proposed by R. Patel and D. Singh (2021) leverages the power of collective defense through API-based threat intelligence and blacklist matching. This method integrates with continuously updated, real-time databases from reputable services like PhishTank and OpenPhish. When an email containing a link is received, the system queries these external services to check if the domain or URL is a known malicious entity. This approach is highly effective and computationally efficient for blocking known threats. Its key limitation, however, is its reactive nature; it is fundamentally incapable of detecting zero-day or newly emerging phishing campaigns. A phishing link must first be identified, reported, and added to the blacklist before this method can offer protection, leaving a critical window of vulnerability for novel attacks.

3.4 Blockchain-Based Verification

Pushing the boundaries of innovation, L. Zhou and T. Wang (2024) introduced a novel framework that utilizes blockchain technology and smart contracts for sender verification. This decentralized approach aims to solve the core problem of email spoofing by creating a public, immutable ledger of sender identities. In this system, an organization's official email servers are registered on the blockchain. Smart contracts can then automatically verify whether an incoming email originates from a legitimate, registered source. This provides a nearly foolproof method for preventing sender impersonation. Despite its theoretical robustness, this approach faces formidable practical barriers, including significant implementation costs, high energy consumption, and profound scalability challenges. Moreover, its effectiveness is contingent on widespread, universal adoption, which remains a distant prospect.

3.5 Hybrid Models (ML and NLP)

To overcome the limitations of single-method approaches, researchers such as N. Gupta and R. Verma (2023) have developed hybrid models that combine machine learning (ML) with natural language processing (NLP). This sophisticated technique performs a two-pronged analysis: it examines header data for technical red flags (using ML) while

simultaneously scrutinizing the email body's linguistic content for indicators of phishing, such as urgent language, grammatical errors, or suspicious calls to action (using NLP). By correlating findings from both the header and the body, these models can achieve significantly higher detection accuracy. The primary trade-off for this enhanced performance is the high computational overhead and the need for large, well-curated training datasets that include both header and content information, making such systems resource-intensive to build and maintain.

4. ANALYSIS OF EXISTING SYSTEM LIMITATIONS

The comprehensive literature survey reveals a critical gap: existing phishing detection systems—particularly static, client-side tools and services widely available to end users—are burdened with significant functional limitations. These shortcomings are not minor flaws; they represent fundamental vulnerabilities in the current security paradigm that attackers actively exploit.

4.1 Reliance on Static, Rule-Based Filters

A primary issue is the overreliance of major email service providers on built-in spam and phishing filters. These systems predominantly operate on static, signature-based rulesets. They are engineered to detect known attack patterns, specific malicious keywords, or senders who are already blacklisted. However, the cyberthreat landscape is hyper-dynamic. Attackers continuously devise new vectors of attack, such as novel email templates, registrar-hopping techniques, and the use of newly registered domains (NRDs). These static filters are inherently reactive and cumbersome, making them slow to adapt. Consequently, there is a persistent and dangerous lag between the emergence of a new attack methodology and the development and deployment of a filter rule to counter it—leaving a wide window of opportunity for compromise.

4.2 Systemic Failures in Content-Based Analysis

- Traditional content-based analysis, which scans the body of an email for suspicious elements, is now routinely defeated by modern obfuscation techniques. Malicious actors no longer rely on simple, easily detectable text-based links. Instead, they employ a variety of methods designed to deceive both human users and automated filters:
- URL Shortening and Redirection: Services such as Bitly or TinyURL are used to mask the true, malicious destination of a link. The filter perceives a legitimate, high-reputation domain (e.g., bit.ly), while the user is redirected to a compromised server upon clicking.
- Homograph Attacks (Unicode Domains): This technique exploits visually similar Unicode characters from different alphabets to create deceptive domain names (e.g., using the Cyrillic “a” instead of the Latin “a” in example.com). To both

the user and a basic filter, the domain appears legitimate but resolves to a completely different, malicious IP address.

- Image-Based Obfuscation: Attackers embed malicious links or “call-to-action” text within images. This method effectively bypasses text-based scanners, as optical character recognition (OCR) is rarely, if ever, applied in real time at the scale required for email filtering due to its high computational cost.

4.3 The Inherent Delay of Blacklist-Reliant Systems

Many detection tools, including some browser extensions and antivirus programs, depend heavily on standalone blacklists of known phishing sites. This model is fundamentally flawed by latency. For a site to be blacklisted, it must first be discovered, reported by a user or honeypot, verified by a security organization, and then have its signature propagated through an update. This entire process can take hours or even days. Meanwhile, modern phishing campaigns are agile; attackers use “burn” domains that remain active for only a few hours or even minutes, achieving their objectives long before they appear on any blacklist. This creates a critical and unavoidable delay in identifying and blocking new phishing sites, rendering such tools ineffective against the most immediate threats.

These limitations collectively create a deeply flawed security ecosystem—one characterized by high false positives (when static rules are overly aggressive) and, more dangerously, critically high false negatives (missed detections) when confronted with novel attacks. The systemic overreliance on centralized, reactive systems is the root problem, leaving users defenseless against zero-day phishing campaigns. These campaigns are specifically designed to exploit the vulnerability window—the crucial lag time between the launch of a new attack and its eventual detection and inclusion in global blacklist updates. The primary motivation of this project is to close this window.

5. PROPOSED SYSTEM ARCHITECTURE: THE PHISH-STOP PIPELINE

To proactively address the identified limitations of existing systems—namely their reactive nature, latency in updates, and inability to detect zero-day exploits—we propose a novel phishing detection pipeline, “Phish-Stop”. This system is architected as a lightweight, client-side, browser-based solution. This architectural choice is deliberate: by operating directly within the browser, Phish-Stop places its defensive capabilities at the final point of vulnerability, the end-user's client. It is designed as a hybrid, multi-layered solution that integrates several key features to create a robust, resilient, and intelligent defense against a wide spectrum of phishing threats.

5.1 The Phish-Stop Pipeline

The system's design is not a single, monolithic block but rather a synergistic integration of four key pillars. These pillars work in concert to provide a defense-in-depth model that is both fast and intelligent, balancing known-threat detection with heuristic-based anomaly-finding.

- **Email Header & Log Analysis Engine:** This is the proactive heart of the Phish-Stop pipeline. This core engine moves beyond simple content scanning to perform a deep, technical analysis of email metadata. It is engineered to automatically detect sender identity anomalies (e.g., display name spoofing, 'From'/'Reply-To' mismatches) and suspicious IP routing (e.g., an email from a trusted domestic brand originating from an uncharacteristic foreign IP block). Most critically, it programmatically validates email authentication, checking the hard results of SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), and DMARC (Domain-based Message Authentication, Reporting & Conformance). Failures in these protocols are exceptionally strong indicators of a sophisticated spoofing attempt.

- **Real-Time Threat Intelligence Feeds:** To counter the critical "blacklist delay" problem, the system incorporates a continuous, automated update mechanism. It aggregates phishing indicators (malicious URLs, domains, and IP addresses) from multiple trusted, live threat intelligence sources, including PhishTank, OpenPhish, and Abuse.ch. This component ensures the system's knowledge base is not static; it is refreshed daily, arming the user with the most current data on active phishing campaigns, drastically reducing the window of vulnerability.

- **Local Storage via IndexedDB:** For the system to be both resilient and high-performance, it cannot rely on constant, real-time cloud API calls for every check. We leverage IndexedDB, a modern, high-capacity, asynchronous database built into the browser. This allows the entire aggregated phishing dataset to be stored and indexed locally. This design provides two crucial benefits:

1. **Offline Resilience:** The tool remains fully functional even without an internet connection, able to detect threats using its last-synced dataset.

2. **Fast Lookup Times:** Checking a URL against the local database is nearly instantaneous, adding no discernible latency to the user's browsing experience.

- **Firestore Integration for Synchronization:** While the detection logic is local, the dataset management is cloud-based for efficiency and scalability. The system uses Firestore as its backend for robust cloud connectivity. A cloud function in Firestore is responsible for fetching, de-duplicating, and normalizing the indicators from all external threat feeds into a single, optimized dataset. The browser-based client then efficiently synchronizes with this clean dataset from Firestore,

rather than managing multiple, complex API connections itself. This centralized management allows for easy scalability, such as adding new threat feeds in the future.

This hybrid design—combining local heuristic analysis with a frequently updated local blacklist—ensures continuous updates and scalability. It is built to minimize false positives (a blacklist match is a high-confidence event) while simultaneously providing a strong, heuristic-based defense against novel phishing attacks.

5.2 System Flow and Decision Engine

The operational flow of the Phish-Stop pipeline is a logical, multi-stage process designed for maximum efficiency and accuracy. The entire process, from ingestion to user alert, is executed in milliseconds.

- i. **Ingestion:** The process begins the moment a user interacts with a webmail client. The system's listeners, running in the browser, intercept and feed incoming email headers or system logs into the pipeline for analysis.

- ii. **Preprocessing and Feature Extraction:** The raw, unstructured header text is immediately cleaned, normalized, and parsed. This critical step uses targeted regular expressions and parsers to extract all relevant data points. These extracted features include not only visible URLs and domain names but also crucial metadata: the Return-Path, all IP addresses from the Received hops, and the explicit Authentication-Results (SPF, DKIM, DMARC status).

- iii. **Phase 1: Dataset Matching (Known-Threat Check):** The extracted URLs and domains are first checked against the locally stored, real-time phishing dataset in IndexedDB. This is the pipeline's "fast pass." A match here is a high-confidence indicator of a known threat, allowing the system to make an immediate "block" decision.

- iv. **Phase 2: Suspicious Pattern Analysis (Zero-Day Check):** Concurrently, the pipeline's heuristic engine analyzes the other extracted features for suspicious patterns that would not be in a dataset. This includes flagging hard failures in SPF/DMARC, identifying mismatches between the From domain and the domain that actually sent the email, or detecting the use of URL shorteners and other obfuscation techniques.

- v. **Decision Engine & Risk Scoring:** The results from both Phase 1 (Dataset Matching) and Phase 2 (Pattern Analysis) are fed into a weighted decision engine. This engine aggregates the findings to make a final, intelligent determination.

1. A match from Phase 1 (e.g., PhishTank Match) might automatically assign a maximum threat score.

2. A combination of red flags from Phase 2 (e.g., DMARC=fail + IP_Mismatch) would cross a pre-defined threshold, also classifying the email as a high-level threat.

vi. Output & User Alert: The final result—a simple, clear classification like "Safe," "Suspicious," or "Dangerous"—is sent to the browser extension's frontend. This triggers a non-intrusive "safe" message for benign emails or a clear, high-visibility alert banner for threats, actively warning the user before they can interact with the malicious content.

6. IMPLEMENTATION MODULES

The Phish-Stop browser extension is architected as a modular, multi-layered system comprising four interdependent components. This modular structure ensures maintainability, scalability, and security while promoting clear separation of concerns between the user interface, background processing, data management, and configuration logic. Each module performs a distinct functional role, contributing collectively to the robustness and efficiency of the phishing detection pipeline.

6.1 Module 1: User Interface (UI) and Interaction

The User Interface (UI) module serves as the primary interaction point between the user and the system. Implemented through the `popup.html` file and its corresponding JavaScript controller `popup.js`, this module delivers the graphical interface that appears when the user activates the extension. The UI is designed for intuitive operation, presenting the "PhishDetect" interface for streamlined usability and immediate visual clarity.

Core Interactive Components: The interface features a text input area that enables users to manually paste email headers for analysis. The "Analyze Header" button serves as the main execution trigger, transmitting the collected text to the background service worker through an `analyzeHeader` message.

Automated Extraction Functionality: An additional feature, "Extract From Gmail", enables automated retrieval of email headers from an open Gmail message. Utilizing the scripting and `activeTab` permissions, the module dynamically injects a content script into the active Gmail tab. This script parses the Document Object Model (DOM) to locate and extract header data, returning it to the popup for further analysis by the background processing module. This automation provides a seamless "one-click" experience, minimizing user effort.

Feedback and Visualization: A dedicated results display area communicates the analytical outcome to the user. Following background processing, a response—such as "SAFE" or "PHISHING_DETECTED"—is returned and displayed, ensuring immediate and unambiguous feedback.

Styling and Accessibility: A linked CSS file defines the visual aesthetics of the interface. The design adopts a dark-mode color scheme with high-contrast text (primarily orange and white), optimizing readability, reducing eye strain, and ensuring critical warnings remain visually prominent.

6.2 Module 2: Background Processing and Core Logic

The second module, implemented as `background.js`, serves as the central processing core of the extension. Operating as a non-persistent service worker (as mandated by Chrome's Manifest V3 specifications), it functions as the extension's background controller—executing computational tasks only when triggered by relevant events, thus conserving system resources.

Event-Driven Architecture: The core of this module revolves around an event listener, `chrome.runtime.onMessage.addListener`, which remains idle until it receives a command from the UI module—specifically, an `analyzeHeader` request.

Data Retrieval and Analysis Workflow: Upon activation, the listener asynchronously queries the local database using `chrome.storage.local.get('phishingDomains')` to obtain the most recent list of flagged or known malicious domains.

Regex-Based Pattern Extraction: A sophisticated Regular Expression (Regex) pattern, defined as `urlRegex`, is then employed to parse the provided email header text. This expression systematically extracts all hostnames, embedded URLs, and IP addresses from the header structure.

Detection and Matching Loop: Each extracted entity is iteratively compared against the retrieved dataset of phishing domains using an efficient set-based lookup mechanism. If any match is detected, the process is immediately terminated, and a "PHISHING_DETECTED" response is dispatched to the UI. Conversely, if no matches are found, the module returns a "SAFE" result. This design ensures high detection speed and optimized system responsiveness.

6.3 Module 3: Phishing Intelligence and Data Management

The third module, defined in `updateDataset.js`, is dedicated to the acquisition, normalization, and management of phishing intelligence data. Its sole responsibility is to ensure that the extension maintains a continuously updated repository of known malicious domains, thereby enhancing detection precision.

Automated Data Acquisition: The module's primary function, `updatePhishingDataset`, is automatically executed via system events such as `chrome.runtime.onStartup` (triggered when the browser launches) or periodic `chrome.alarms` events.

This ensures the local dataset is refreshed regularly—typically on a daily basis—without requiring user intervention.

Threat Feed Integration: Through the `fetch()` API, the module retrieves raw threat intelligence data from trusted public feeds, such as OpenPhish (<https://openphish.com/feed.txt>).

Data Cleaning and Normalization: The retrieved dataset undergoes preprocessing through the `extractHostnameFromUrl` function, which sanitizes and extracts domain-level information by stripping protocol identifiers (e.g., `http://`, `https://`) and directory paths. This normalization step ensures that only root-level domain names are stored for efficient comparison.

Local Database Update: After cleaning, the refined dataset is stored in the local database using `chrome.storage.local.set({ phishingDomains: ... })`. This operation overwrites outdated records, ensuring Module 2 always references the latest, most accurate intelligence feed during phishing analysis.

6.4 Module 3: Module 4: Extension Framework and Configuration

The fourth module, encapsulated in the `manifest.json` file, serves as the blueprint and security configuration of the extension. While it does not execute any runtime logic, it defines the extension's structure, permissions, and compliance with browser security policies.

Manifest Declaration: The file specifies `manifest_version: 3`, aligning the project with Chrome's latest extension architecture, emphasizing improved performance and security through the use of service workers in place of persistent background pages.

Core Metadata: It defines the extension's metadata, including the name ("PhishDetect") and description, both of which are displayed in the browser's extensions interface and the Chrome Web Store.

Service Worker Registration: The manifest registers `background.js` as the designated service worker responsible for handling core detection and background operations.

Permission Configuration: The manifest explicitly requests permissions essential to the extension's operation:

- `storage` — Allows Modules 2 and 3 to read from and write to the local database.
- `scripting` and `activeTab` — Enable the UI module to inject scripts into the user's active browser tab, particularly for automated Gmail header extraction.
- `host_permissions` — Grants controlled access to external domains, including data fetches from

<https://openphish.com/> and scripted interactions with <https://mail.google.com/>.

7. KEY ADVANTAGES OF THE PHISH-STOP SYSTEM

The proposed Phish-Stop pipeline is designed to overcome the fundamental flaws of traditional security systems. By leveraging a modern, browser-based, and data-driven approach, it delivers distinct and measurable advantages.

Real-Time, Instantaneous Detection : A primary advantage of the Phish-Stop system is its ability to perform instantaneous, on-arrival detection. This is achieved through its client-side architecture, which completely eliminates the "blacklist lag" that plagues conventional, reactive systems.

Instead of waiting for an external server update or user report, Phish-Stop begins analysis the moment an email is accessed. This detection process operates on two levels:

- **Zero-Day Heuristics:** The system's core engine immediately parses and analyzes the email header for intrinsic, telltale signs of spoofing. This includes checking for SPF/DKIM/DMARC authentication failures, anomalous message routing, and deceptive mismatches in sender information. This enables Phish-Stop to identify and flag sophisticated, novel attacks that have never been seen before and do not yet exist in any threat intelligence feed.
- **Instant Local Lookup:** For known threats, the system performs a high-speed lookup against its comprehensive, locally stored dataset in IndexedDB. This query is nearly instantaneous and does not require a round-trip API call to a cloud service.

This proactive, dual-analysis model ensures the user receives a verdict before interacting with malicious content—effectively closing the window of vulnerability that most phishing campaigns exploit.

Adaptive and Continuous Updates : Phish-Stop is not a static tool; it functions as a living, adaptive defense mechanism. It is built to remain resilient against the rapid evolution of phishing tactics by ensuring its threat intelligence is perpetually current.

This capability is driven by an automated data-management module, which operates silently in the background. The module connects to multiple trusted live threat feeds (such as PhishTank and OpenPhish) and programmatically fetches, aggregates, and normalizes this data on a daily basis, ensuring the local database remains continuously refreshed with the latest known malicious domains.

This continuous update cycle directly enhances the system's accuracy and adaptability. As new phishing campaigns are discovered and reported by the global security community, their signatures are automatically integrated into the Phish-

Stop network. This ensures the tool's effectiveness does not degrade over time and that it remains a formidable defense against emerging threats.

High Resilience and Performance : A key architectural strength of Phish-Stop lies in its seamless combination of high performance and robust resilience.

- **Lightweight Footprint:** By design, the extension is exceptionally lightweight. The detection logic—comprising header parsing and database lookup—is highly optimized and executed locally, introducing no discernible latency or overhead to the user's browsing experience.
- **Full Offline Functionality:** The strategic use of IndexedDB to store the entire threat database locally is a critical differentiator. Unlike cloud-reliant scanners that fail without an active internet connection, Phish-Stop remains fully operational offline. It can continue to analyze and block threats based on its last-synced dataset, providing uninterrupted protection for users on unstable networks or in travel scenarios.

The combination of instantaneous detection, self-updating intelligence, and offline resilience makes Phish-Stop a practical and powerful solution for real-world deployment. It effectively bridges the gap between reactive and proactive defense mechanisms, establishing a new standard for client-side phishing protection.

8. IDENTIFIED LIMITATIONS AND DISADVANTAGES

Despite the robust design of the Phish-Stop pipeline, a realistic assessment identifies two primary potential limitations that must be acknowledged. These challenges are inherent in the trade-off between passive security and the kind of active, real-time analysis this system proposes.

Potential for Increased Resource Consumption

A significant consideration is the system's impact on local resource consumption (CPU and memory). Unlike a simple, static filter that only performs a basic keyword or blacklist check, the Phish-Stop pipeline is a far more active analysis engine.

- **Real-time Analysis:** The continuous, real-time analysis of all incoming email headers and logs is, by nature, more computationally intensive. When a user is processing a large volume of email—such as during an initial inbox load or when archiving hundreds of messages—the pipeline's engine must work to parse and analyze each item.
- **Database and Update Overhead:** Furthermore, the daily background fetching, processing, and writing of the updated threat intelligence dataset to IndexedDB, while efficient, will periodically consume network bandwidth and system resources.

While the system is engineered to be lightweight, this active analysis and data management footprint will invariably be higher than that of a passive, server-side filter, which could potentially impact performance on older or resource-constrained machines.

Inherent Risk of False Positives : The second significant challenge, common to all advanced detection systems, is the risk of false positives. As with any security tool that relies on heuristic analysis or complex pattern-based detection, there is a non-zero possibility that legitimate, benign emails may be incorrectly flagged as phishing.

This risk stems from the system's reliance on detecting anomalies. For instance:

- A legitimate, small business might use a third-party email marketing service that is not perfectly configured, causing it to fail a DMARC or SPF check.
- A system administrator sending a valid notification might use a script that generates an email with an unusual but technically valid header configuration.

In such cases, the heuristic engine, designed to be suspicious, might correctly identify the anomaly but incorrectly attribute it to malicious intent. While the goal is to minimize this rate through careful weighting in the decision engine, an overly aggressive heuristic model could lead to user friction, causing them to mistrust or eventually ignore the system's warnings.

9. FUTURE SCOPE: INTEGRATING NEXT-GENERATION INTELLIGENCE

While Phish-Stop represents a robust solution, the conclusion of this research points toward a clear and exciting path forward. The project's framework is an ideal foundation for integrating more advanced, next-generation technologies.

AI-Powered Anomaly Detection: The most immediate and high-impact future iteration would involve incorporating machine learning (ML) models. Instead of relying solely on rule-based heuristics and known blacklists, an ML model could be trained on millions of email headers (both benign and malicious). This model would learn to identify subtle, complex, and previously unknown patterns of malicious behavior, effectively enabling behavior-based risk prediction. For example, it could flag a novel combination of header anomalies (e.g., an unusual server hop, a specific type of DMARC failure, and a newly registered domain) as a high-risk "phishing-like" pattern, even if no component is on a blacklist.

Blockchain-Based Identity Validation: Looking further ahead, the system could explore the blockchain-based identity validation frameworks suggested in the literature review. This represents a paradigm shift from detection to

prevention. By integrating with a decentralized ledger, a future version of the tool could verify a sender's identity against an immutable, cryptographically secure record. This would, as suggested by researchers, create an immutable verification system that makes it computationally infeasible for attackers to spoof a verified corporate or individual identity, effectively solving the root problem of sender impersonation.

Enhanced User-Centric Privacy: Future development could also focus on user-centric privacy solutions. This might involve implementing on-device ML models (via TensorFlow.js) that allow all heuristic and behavioral analysis to occur entirely within the user's browser. This would ensure that potentially sensitive header data never has to be transmitted to an external server, offering a best-in-class solution for the most privacy-conscious users and organizations.

10. CONCLUSIONS

This project demonstrates a critical and intelligent shift in the paradigm of email security, moving decisively away from the industry's traditional, reactive approaches. Conventional systems, reliant on static content-based filters, are increasingly outpaced by the sophistication of modern phishing threats. In contrast, the Phish-Stop system embodies a proactive, metadata-driven methodology. By concentrating on the technical and evidential content of email headers and system logs, the pipeline analyzes the structural and infrastructural "digital DNA" of attacks rather than merely their superficial, user-facing manifestations.

At the core of this advancement is the full automation of deep header and log analysis. This automated framework enables rapid, scalable processing of large volumes of email data while minimizing human fatigue. Crucially, it enhances detection accuracy by identifying subtle technical signatures of spoofing, including DMARC/SPF authentication failures, anomalous IP routing, and discrepancies between header and sender information—features that are typically invisible to end-users and often missed by content-focused filters.

Beyond improving precision, this evidence-based automation delivers significant secondary benefits. It reduces reliance on manual user intervention, liberating both end-users and IT personnel from the burdensome task of manually evaluating suspicious messages. Simultaneously, it mitigates false-positive rates common in heuristic-only systems, providing a more reliable and binary assessment of malicious intent than subjective content analysis alone.

The integration of real-time threat intelligence from globally recognized, crowd-sourced repositories, such as PhishTank and OpenPhish, further enhances the system's effectiveness. This feature transforms the extension from a standalone utility into an active participant within a continuously updated, community-driven cybersecurity ecosystem. The

system dynamically incorporates insights from newly identified phishing attacks worldwide, ensuring near real-time adaptation to evolving threats.

Technically, the project showcases innovation by combining three distinct technologies into a unified framework. Firebase enables cloud-based, centralized data management; IndexedDB provides high-performance, offline-capable local storage; and the real-time analysis engine delivers immediate, on-the-fly detection. Together, these components form an adaptive, future-ready phishing detection framework capable of operating reliably even without internet connectivity, updating dynamically with emerging threats, and remaining extensible for future enhancements.

Overall, the Phish-Stop architecture aligns closely with emerging trends in digital safety, emphasizing proactive identity verification rather than passive trust in content. By providing users with a transparent, client-side analytical tool, the system empowers individuals with clear, data-driven insights, replacing uncertainty with actionable intelligence and establishing a new standard for effective, proactive email security.

11. ACKNOWLEDGMENTS

The authors would like to express their sincere gratitude to their project guide, Mrs. S. Chandrakala, AP/CYS and Mr. P. Sathishkumar, AP/CYS for her invaluable guidance, continuous support, and insightful feedback throughout the duration of this research. Her expertise was instrumental in the successful development and completion of the Phish-Stop pipeline.

The authors also extend their heartfelt thanks to the Department of Cyber Security Engineering for providing the necessary facilities and resources to carry out this research successfully. Special thanks are due to all faculty members and classmates for their helpful suggestions, motivation, and support during the implementation, and testing phases of the project.

REFERENCES

- [1] A. Jain, P. Sharma, and K. Mehta, "Email header analysis for phishing detection," *IEEE Access*, vol. 10, pp. 12894–12907, 2022.
- [2] M. Lee and S. Kim, "Real-time phishing detection using log-based behavioral analysis," *Computers & Security*, vol. 125, pp. 102947–102959, 2023.
- [3] N. Gupta and R. Verma, "A hybrid model for phishing email detection using machine learning and natural language processing," *Expert Systems with Applications*, vol. 229, pp. 120984–120998, 2023.
- [4] S. Sundar and N. Yousuf, "Phishing detection pipeline using email headers and logs," *International Journal of*

Information Security and Smart Systems, vol. 2, no. 1, pp. 101–112, 2025. (Proposed system — mini project work)

- [5] R. Patel and D. Singh, “PhishTank and OpenPhish integration for browser security enhancement,” *Journal of Cybersecurity and Privacy*, vol. 5, no. 2, pp. 65–79, 2021.
- [6] L. Zhou and T. Wang, “Blockchain-based secure email verification framework,” *IEEE Transactions on Information Forensics and Security*, vol. 19, no. 4, pp. 451–463, 2024.
- [7] A. K. Jain and B. B. Gupta, “A Survey of Phishing Detection Techniques: 2015-2021,” *Security and Privacy*, vol. 5, no. 1, e189, 2022.
- [8] S. O. Fatayol, F. A. Ghaleb, S. N. M. K. S. O. Al-Rimy, and M. A. Al-Ahmad, “A Review on Phishing Detection Methods: Taxonomy, Datasets, and Future Directions,” *IEEE Access*, vol. 11, pp. 91705–91728, 2023.
- [9] S. Kitterman, “Sender Policy Framework (SPF) for Authorizing Use of Domains in Email, Version 1,” RFC 7208, Apr. 2014. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7208>
- [10] M. Kucherawy, Ed., and E. Zwicky, Ed., “DomainKeys Identified Mail (DKIM) Signatures,” RFC 6376, Sept. 2011. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc6376>
- [11] M. Kucherawy and E. Zwicky, “Domain-based Message Authentication, Reporting, and Conformance (DMARC),” RFC 7489, Mar. 2015. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc7489>
- [12] P. Resnick, Ed., “Internet Message Format,” RFC 5322, Oct. 2008. [Online]. Available: <https://www.rfc-editor.org/rfc/rfc5322>
- [13] Cisco, “PhishTank: Join the fight against phishing.” [Online]. Available: <http://www.phishtank.com/>. [Accessed: Oct. 31, 2025].
- [14] OpenPhish, “Real-time Phishing Data Feeds.” [Online]. Available: <https://openphish.com/>. [Accessed: Oct. 31, 2025].
- [15] Google, “Google Safe Browsing.” [Online]. Available: <https://safebrowsing.google.com/>. [Accessed: Oct. 31, 2025].