PhishParrot: LLM-Driven Adaptive Crawling to Unveil Cloaked Phishing Sites

Hiroki Nakano*, Takashi Koide*, Daiki Chiba*
*NTT Security Holdings Corporation & NTT, Inc., Japan
Email: hi.nakano.sec@gmail.com

***Abstract*—Phishing attacks continue to evolve, with cloaking techniques posing a significant challenge to detection efforts. Cloaking allows attackers to display phishing sites only to specific users while presenting legitimate pages to security crawlers, rendering traditional detection systems ineffective. This research proposes PhishParrot, a novel crawling environment optimization system designed to counter cloaking techniques. PhishParrot leverages the contextual analysis capabilities of Large Language Models (LLMs) to identify potential patterns in crawling information, enabling the construction of optimal user profiles capable of bypassing cloaking mechanisms. The system accumulates information on phishing sites collected from diverse environments. It then adapts browser settings and network configurations to match the attacker's target user conditions based on information extracted from similar cases. A 21-day evaluation showed that PhishParrot improved detection accuracy by up to 33.8% over standard analysis systems, yielding 91 distinct crawling environments for diverse conditions targeted by attackers. The findings confirm that the combination of similar-case extraction and LLM-based context analysis is an effective approach for detecting cloaked phishing attacks.**

***Index Terms*—Phishing, Cloaking, Large Language Model.**

I. Introduction

Phishing attacks have become increasingly sophisticated, posing a serious threat to internet users worldwide. Attackers employ "cloaking" techniques to evade detection mechanisms. They display phishing content only to users who meet specific criteria, while presenting legitimate pages or error messages to security crawlers and other unwanted visitors [1], [2].

Conventional phishing detection systems analyze features such as HTML content and screenshots using machine learning models or rule-based algorithms, with some incorporating network data like IP addresses and HTTP transactions [3]–[6]. However, these systems fundamentally depend on successfully retrieving phishing content. When attackers identify security crawlers through cloaking mechanisms, they serve non-phishing content instead, which makes accurate analysis impossible. Research analyzing security crawler profiling has demonstrated that even when simulating "typical user-like" requests, without appropriate environmental adjustments, crawlers cannot satisfy all conditions targeted by attackers [7]. Consequently, content acquisition methods dependent on fixed crawling environments have limited ability to reach phishing content. To address these limitations, a new approach capable of flexibly and accurately capturing the characteristics of phishing attacks is essential.

This research proposes PhishParrot, a novel crawling environment optimization system designed to improve detection rates for cloaked phishing sites. PhishParrot adaptively adjusts browser profiles (such as HTTP headers) and network environments (e.g., country information, residential or mobile networks) based on crawling results information to satisfy attacker-defined conditions and access phishing content (Fig. 1). The system accumulates phishing site information (Domain Information, Network Information, HTML Information) collected from diverse environments and incorporates a mechanism to extract similar cases. Based on this information, a Large Language Model (LLM) suggests optimal user profiles (browser settings and access environments), enabling the construction of environments closely resembling those targeted by attackers, thereby bypassing cloaking to retrieve phishing content. The advanced language comprehension capabilities of LLMs enable the flexible and accurate identification of trending characteristics in the latest phishing attacks. This potentially enhances effectiveness against cloaking-based detection avoidance, a challenge for conventional fixed-environment crawling approaches.

In this study, we evaluated PhishParrot's performance using three types of LLMs on 32,487 suspicious URLs collected over a 21-day period. Results showed that the best-performing model achieved 88.0% accuracy with an average processing time of 16.69 seconds per URL. When combining PhishParrot with three different Phishing Detectors, we confirmed up to a 33.8% improvement in phishing site detection rates compared to standard analysis systems. PhishParrot also demonstrated the ability to suggest 91 optimal user profiles based on attack trends and phishing site similarities, enabling flexible environment construction tailored to attackers' targets. These results indicate that rule-based approaches struggle to comprehensively cover the diverse user access environments targeted by attackers. In contrast, the sophisticated context analysis capabilities of LLMs enable flexible and effective crawling environment construction unachievable by other approaches. The findings of this research demonstrate that incorporating LLMs into systems is an effective strategy for addressing the challenge of detection avoidance through cloaking.

II. Related Work

Numerous studies have been conducted on phishing site detection, with many systems proposed to identify phishing sites based on HTML content and screenshots obtained when

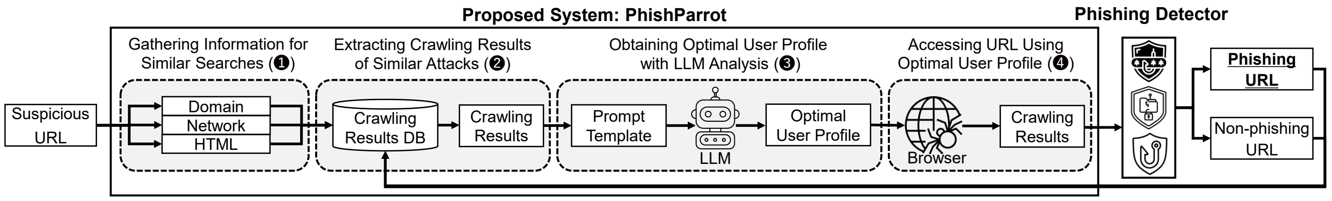
Proposed System: PhishParrot
Phishing Detector
Gathering Information for Similar Searches (❶)
Extracting Crawling Results of Similar Attacks (❷)
Obtaining Optimal User Profile with LLM Analysis (❸)
Accessing URL Using Optimal User Profile (❹)
Suspicious URL
Domain
Network
HTML
Crawling Results DB
Crawling Results
Prompt Template
LLM
Optimal User Profile
Browser
Crawling Results
Phishing URL
Non-phishing URL

Fig. 1: PhishParrot is composed of 4 steps to overcome sophisticated cloaking techniques and improve phishing detection accuracy. The system takes suspicious URLs as input and outputs actual phishing content hidden by cloaking mechanisms. The system performs a preliminary access to gather information for similar attack searches (❶). It extracts crawling results of similar attacks from a database using cloaking pattern similarity-based retrieval (❷). It embeds this information into prompts and requests LLM analysis to determine the optimal user profile matching attacker's targeting criteria (❸). It configures the browser with this profile and accesses the URL to retrieve phishing content hidden from security crawlers (❹). This content is sent to phishing detectors, enabling accurate classification impossible with conventional crawling approaches. Without PhishParrot's adaptive approach, even sophisticated phishing detectors would fail as cloaking mechanisms prevent access to malicious content.

accessing URLs [3], [6], [8], [9]. Liu et al. proposed "PhishIntention," which visually extracts both brand intention and credential-harvesting intention from webpage appearance and dynamic behavior, verifying credential-harvesting intention through interaction with the webpage [5]. Lin et al. introduced "Phishpedia," a system that accurately recognizes brand logos from webpage screenshots and matches logo variations of the same brand, enabling high-precision identification and visual explanation of phishing sites [4]. While these studies demonstrate high detection capabilities in situations where phishing content is accessible, their detection accuracy inevitably decreases when accessing continuously from the same or known analysis environments, as cloaking techniques prevent observation of the actual phishing content.

Cloaking techniques refer to technologies that display different content based on the characteristics of the access source. When a phishing URL is accessed, attackers analyze request characteristics in detail (User-agent, IP address, HTTP headers, etc.). Based on this analysis, they distinguish between "Victim Access" (access from targeted typical users) and "Non-victim Access" (access from security researchers or detection systems). According to Zhang et al. [2], 96.52% (2,831 cases) of the 2,933 phishing kits analyzed contained cloaking techniques. IP address and User-agent based cloaking were found to be the most commonly used, with 1,983 cases (approximately 82%) implementing User-Agent verification and 1,660 cases (approximately 69%) implementing IP address verification among the 2,421 phishing kits analyzed.

To address these cloaking techniques, researchers have proposed analyses of cloaking technologies embedded in phishing kits and techniques to bypass specific types of cloaking [2], [10], [11]. Zhang et al. proposed "CrawlPhish," a framework for automatically detecting and classifying client-side cloaking used in phishing sites [1]. Their analysis of 112,005 phishing sites collected over 14 months from 2018 to 2019 revealed that the use of cloaking techniques increased from 23.32% to 33.70%. Acharya et al. proposed the "PhishPrint" framework, which pre-profiles the characteristics of security crawlers to identify weaknesses and uses them to evade phishing detection through cloaking attacks [7]. Through evaluation of 23 security crawlers, they revealed weaknesses against cloaking, such as a lack of diversity in browser fingerprints and IP address limitations. While existing research implements simple countermeasures against cloaking, our proposed system differs significantly in that it adapts to the latest phishing attack trends and constructs access environments that more effectively bypass cloaking.

III. Proposed System: PhishParrot

A. Design Goals and Objectives

To bypass attackers' cloaking techniques and retrieve phishing content that typical users encounter, we propose "PhishParrot" (Fig. 1). Our system focuses primarily on attributes such as HTTP headers and IP addresses, which constitute the majority of current cloaking techniques, to accurately reproduce the target user's access environment. The system aims to achieve the following two goals:

Goal 1: Identify similar cloaking patterns. Phishing attacks have been reported to share similarities based on toolkits or attacker groups [2]. Cloaking techniques follow similar trends; therefore, we extract similar examples from accumulated crawling results.

Goal 2: Bypass cloaking to access phishing content. Bypassing cloaking techniques requires mimicking "typical user-like" target access. Based on past successful examples, LLM analyzes appropriate crawling environments and retrieves content from target URLs.

B. Gathering Information for Similar Searches (❶)

In this step, we define the data structure of crawling results used for similar case searches and LLM prompt embedding. As shown in Table I, we establish data structures in four categories: Domain Information, Network Information, HTML Information, and Crawling Environment Information, and collect this information.

Domain Information. In some cases, domain names can be analyzed to reveal characteristic strings or subdomain structures reused from past phishing attacks [12]. Additionally, registration timing and certificate reuse can help identify

TABLE I: Defined Structure of Crawling Results

Category	Type	Content
Domain Information	Registration	`{"domain_name": "EXAMPLE.COM", "registrar": "...", ...}`
	DNS	`{"Status": 0, "Answer": [{"name": "example.com", "TTL: 123, "data": 192.0.2.0", ...}, {...}]}`
	TLS/SSL	`{"issuer": {"O": "ExampleCert", ...}, "subject": {"CN": "example.com"}, ...}`
Network Information	Requests	`{"requests": [{"id": "1", "method": "GET", "url": "https://example.com", "headers":{...}, ...}, {"id": "2", "method": "GET",...},...]}`
	Responses	`{"responses": [{"id": "1", "status": "200", "url": "https://example.com", "headers": {...} ,...{"id": "2", "status": "200",..},...]}`
HTML Information	Visible Text	`403 Forbidden nginx/1.26.2 You do not have permission to access this resource.`
	Tag Structure	`<html><head><title></title><style></style></head><body><h1></h1><p></p></body></html>`
Crawling Environment Information	IP Geolocation	`{"country": "US", "city": "Sample City", "region":"Sample"}`
	ASN	`{"asn": "AS1234", "name": "Example Business", "domain": "examplebusiness.example", ...}`
	Language	`en-US`

attacks from the same group. If similar attack cases can be identified, cloaking techniques may also exhibit similar patterns, enabling the construction of optimal user profiles based on past successful cases. Therefore, we extract domain names from target URLs and obtain registration information, DNS resolution, and TLS/SSL certificates.

Network Information. When cloaking techniques exist in a target URL, characteristic responses similar to past cases may occur. For example, security crawler access has been reported to be redirected to legitimate sites of impersonated companies or prominent sites (e.g., Google homepage) [13]. To capture this trend, we perform a simple access to the target URL using Playwright and collect requests and responses.

HTML Information. Phishing sites often mimic prominent companies and copy legitimate sites, resulting in similar HTML structures and appearances [3], [4]. By analyzing the similarity between HTML and past phishing attack crawling results, it becomes possible to identify identical cloaking patterns. From the DOM tree obtained during the simple access mentioned above, we extract visible text and HTML tag structures.

Crawling Environment Information. Attackers have been reported to determine routing destinations based on network information from the access source [2], [10]. For example, cloaking techniques may be applied to IP addresses that appear to be from analysis environments or accesses from countries other than the target, preventing the retrieval of phishing content. Conversely, access from general residential or mobile networks in specific countries may match the attacker's target and reach the phishing site. Therefore, we store network location (IP Geolocation), network provider (ASN), and browser language settings (Language) for each crawling result.

C. *Extracting Crawling Results of Similar Attacks (❷)*

In this step, we extract crawling results of similar attacks using state-of-the-art vector representation techniques and search methods.

Crawling Results DB Integration. We store crawling results (with the defined data structure) in a vector database, labeling them as successful crawling or failed crawling. Vector databases enable comparisons based on semantic similarity, allowing effective retrieval of past crawling results from similar attacks. For string vectorization, we utilize OpenAI's text-embedding-3-small model, which transforms text into a 1,536-dimensional vector space. We selected this model because it offers an optimal balance between analysis speed and performance compared to other prominent models (text-embedding-3-large, text-embedding-ada-002). While the system requires a certain amount of labeled initial data to operate, as shown in Fig. 1, it automatically expands successful crawling and failed crawling examples by combining continuous system operation with detection results from Phishing Detectors.

Similarity-based Crawling Results Extraction. In PhishParrot, we convert information from the three defined categories (Domain, Network, and HTML Information) into string representations and extract similar crawling results based on semantic similarity. For example, when searching Domain Information, we compare similarity using 1,536-dimensional vectors of registration, DNS, and TLS/SSL strings. The procedure for extracting crawling results is as follows:

1) Create embedding vectors for each Domain Information, Network Information, and HTML Information of the target URL
2) Retrieve results with cosine similarity above a certain threshold from successful and failed examples in the database
3) Generate embedding vectors from the entire text of the results and extract representative results using the Maximum Marginal Relevance (MMR) algorithm [14]

Cosine similarity quantifies the structural and semantic similarity of target URL information in multidimensional space, enabling high-precision identification of similar patterns in attack methods and strategies. Cosine similarity ranges from 0 (unrelated) to 1 (identical), and we set a threshold of 0.65 based on preliminary experiments that successfully identified similar attacks. The MMR algorithm balances relevance and diversity by selecting documents that maximize marginal relevance, creating a representative result set that covers diverse environmental conditions useful for detecting similar cloaking patterns. While λ in MMR ranges from 0 (emphasizing diversity) to 1 (emphasizing relevance), we adopt 0.7 to prioritize relevance while still considering diversity.

Prompt-Optimized Crawling Results Filtering. We filter redundant information from the extracted results for LLM prompts. Specifically, we extract only registration from Domain Information, main communications from Network Information, and Crawling Environment Information from each result. While other information is important for vector comparison, it is unnecessary for LLM's user profile decisions. Although the number of results for prompt input is arbitrary, we adopt a total of 10 cases: 5 successful examples and 5 failed examples. From preliminary experiments, we confirmed this number allows the LLM to analyze success (e.g., geolocations) and failure (e.g., desktop restrictions) factors.

D. Obtaining Optimal User Profile with LLM Analysis (❸)

In this step, we derive the optimal user profile using LLM analysis of the extracted crawling results.

Prompt Definition. Table II shows the prompt for LLM interaction. Prompts are information provided by users to help LLMs generate appropriate responses. We define a cybersecurity expert persona specializing in phishing detection and anti-cloaking techniques (Persona). We provide instructions focusing on attacker strategies for cloaking analysis (Instruction). We specify that analysis results should be output in JSON format with five types of information (Output Format). Finally, we embed the target information in the prompt (Analysis Request).

LLM Analysis Request. We embed the target URL information as `url`, successful examples from extracted crawling results as `successful_examples`, and failed examples as `failed_examples` in the prompt. This approach, known as Retrieval-Augmented Generation (RAG), enhances LLM reasoning with external knowledge, enabling responses based on current and specific information [15]. Through RAG, we provide the LLM with phishing site cloaking patterns and past crawling results, facilitating more adaptive crawling environment construction. The analysis results include three types of information for crawling (`http_header`, `ip_location`, `network_provider`) and two types of rationale information (`target_victim`, `reason`). In this step, these five types of information constitute the output.

E. Accessing URL Using Crawling Environment (❹)

We aim to bypass cloaking and reach phishing content using the optimal user profile output by the LLM. Our system employs Playwright for network log acquisition and operational stability. We configure headers with those specified in `http_header`. For networking, we utilize multiple pre-prepared environments, setting country information and network types (e.g., typical residential, mobile networks) to match those specified in `ip_location` and `network_provider`. We access the target URL in this environment and obtain the final page content, screenshots, and network logs after JavaScript interpretation, which constitute the system's final output.

IV. Evaluation

A. Experimental Setup

We evaluate the detection accuracy and execution time of PhishParrot for identifying phishing sites.

Baseline Systems. Existing cloaking analysis frameworks like CrawlPhish [1] and PhishPrint [7] are designed for research purposes to understand cloaking mechanisms, rather than for operational detection. Therefore, we evaluate PhishParrot against practical baseline systems. To enable this comparison, we prepared two baseline systems: the Standard Analysis System and the Typical User System. The Standard Analysis System simulates a typical analysis environment by accessing URLs using Google Chrome in headless mode on Linux, from a cloud environment in the United States. In contrast, the Typical User System randomly selects one combination of `http_header`, `ip_location`, and `network_provider` to emulate a typical user environment. Specifically, we defined `http_headers` corresponding to 17 different access environments (with Safari restricted to macOS and iOS) by combining five OS types (Windows, macOS, Linux, Android, iOS) and four browsers (Microsoft Edge, Google Chrome, Safari, Mozilla Firefox). Additionally, to capture high internet usage and regional diversity, we set up 10 `ip_locations` (United States, India, United Kingdom, Germany, Japan, Brazil, Saudi Arabia, Canada, Australia, South Korea). For each `ip_location`, we specified three `network_providers` (Datacenter, Residential, Mobile), resulting in $17 \times 10 \times 3 = 510$ total combinations. We used these two systems as baselines for our comparative evaluation with PhishParrot.

TABLE II: Prompt for Obtaining Optimal User Profile

Tactics	Prompt
Persona (System)	You are a cybersecurity analyst specializing in phishing detection and anti-cloaking techniques. Your role is to analyze website behavior patterns and provide recommendations for optimal crawling environments to detect malicious sites that use cloaking techniques.
Instruction (System)	When analyzing cloaking behavior, you should: 1. Consider geographical targeting patterns and past attack trends. 2. Evaluate HTTP header configurations, including User-Agent strings. 3. Assess network provider characteristics, particularly those commonly used by attackers. 4. Compare provided cloaking and non-cloaking examples to identify behavioral differences. Additionally, prioritize settings that reflect environments attackers typically target based on historical evidence of their tactics, techniques, and procedures (TTPs). Incorporate insights into network providers, IP ranges, and User-Agent configurations most likely to bypass cloaking mechanisms.
Output Format (System)	Your responses should be provided in JSON format with specific keys: - http_header: HTTP headers for optimal browsing environment. - ip_location: IP location of optimal browsing environment. - network_provider: Optimal browsing environment network provider. - target_victim: Nature of users targeted by attackers. - reason: Logical explanation for the recommendations. DO NOT include markdown, code blocks, or any text outside the JSON.
Analysis Request (User)	Analyze the following website information for potential cloaking behavior and recommend an optimal crawling environment. URL: {`url`} Reference Examples: 1. Successful Crawling Example (examples of successful accesses): {`successful_examples`} 2. Failed Crawling Example (examples of sites using cloaking techniques): {`failed_examples`} Based on this information and the reference examples, provide recommendations for: 1. The necessary browsing environment to avoid being cloaked 2. Optimal header information (e.g. User-Agent) for HTTP requests 3. Source IP address characteristics (location and network provider) Please format your response as JSON with the specified keys: target_victim, http_header, ip_location, network_provider, and reason.

Phishing Detectors. To evaluate our system against various detection techniques, we reimplemented three state-of-the-art phishing detectors: ChatPhishDetector [6], VisualPhishnet [3], and StackModel [16]. We chose these systems because they employ distinct approaches: ChatPhishDetector leverages LLM-based contextual analysis of HTML and screenshots; VisualPhishnet performs visual similarity analysis on screenshots using a triplet convolutional neural network; and StackModel combines URL and HTML features with stacked machine learning models. These diverse perspectives provide a comprehensive evaluation of our system's effectiveness.

Input Suspicious URLs. We collected suspicious URLs from three feeds: CrowdCanary [17], which collects phishing URL reports from X using hashtags like "#phishing"; Urlscan.io

TABLE III: Performance Comparison of PhishParrot by LLM

	Avg. of 3 Phishing Detectors						
LLM	Acc	TPR	TNR	Pre	F1	Execution Time	Cost
o3-mini	87.7%	89.6%	85.0%	87.9%	0.886	26.54 seconds	$0.016
GPT-4o	85.9%	87.9%	83.3%	85.9%	0.868	19.13 seconds	$0.027
GPT-4o mini	88.0%	90.5%	85.3%	87.4%	0.889	16.69 seconds	$0.002

URLs tagged as "phishing" or "@phish_report"; and URLs from PhishTank, a community-based phishing verification platform. We excluded URLs with invalid domain names (NXDOMAIN), domain parking, or legitimate site domains (matching the top 100,000 sites in the Tranco list [18]).

Crawling Results DB Settings. PhishParrot requires a sufficient number of crawling results labeled as successful crawling or failed crawling. Therefore, we randomly selected from the aforementioned 540 crawling environments and applied them to suspicious URLs from February 1, 2025, to February 21, 2025, collecting a total of 33,432 crawling results. We labeled these results as successful crawling (reaching phishing content) or failed crawling (content was cloaked) based on consensus among three security engineers with all engineers agreeing on each classification. This process yielded 624 successful entries and 1,223 failed entries, which were stored in the Crawling Results DB. Furthermore, if all Phishing Detectors judged a site to be phishing when accessed by PhishParrot, but they judged it to be non-phishing when accessed by both the Standard Analysis System and the Typical User System, we assumed a high likelihood of cloaking and also stored that crawling result in the Crawling Results DB. This setting allows PhishParrot to capture the latest attack trends and adapt to rapidly changing phishing tactics.

LLMs. PhishParrot uses o3-mini, GPT-4o, and GPT-4o mini from OpenAI. These are the latest models with excellent inference performance and speed. By comparing various models, we can evaluate which is optimal for adoption in PhishParrot.

Metrics. We evaluate system performance using seven metrics: Accuracy, True Positive Rate, True Negative Rate, Precision, F1 score, Execution Time, and Cost, with VirusTotal as ground truth. VirusTotal provides analysis results from approximately 90 antivirus engines but requires continuous evaluation due to detection delays for new phishing sites. We conducted continuous monitoring for one week and adopted criteria from previous research to suppress false detections [19], [20]. URLs determined as "malicious" by five or more engines are classified as true phishing sites; others as non-phishing sites.

Using this ground truth, we evaluate: Accuracy (Acc) as the proportion of URLs correctly classified by the system; True Positive Rate (TPR) as correct phishing classifications among true phishing sites; True Negative Rate (TNR) as correct non-phishing classifications among true non-phishing sites; Precision (Pre) as true phishing sites among system-identified phishing sites; F1 score as the harmonic mean of Precision and TPR; Execution Time as per-URL analysis time from input to output; and Cost as LLM API cost per URL (as of March 2025).

TABLE IV: Performance Comparison Between PhishParrot (GPT-4o mini) and Baseline Systems

Systems	Phishing Detector	Acc	TPR	TNR	Pre	F1	Execution Time
PhishParrot	ChatPhishDetector	**92.7%**	**95.0%**	**89.6%**	**91.8%**	**0.933**	26.71 seconds
	VisualPhishnet	84.7%	86.5%	82.9%	84.5%	0.855	12.00 seconds
	StackModel	86.7%	89.9%	83.5%	86.1%	0.877	11.37 seconds
Typical User System	ChatPhishDetector	64.4%	68.0%	58.1%	63.1%	0.652	22.15 seconds
	VisualPhishnet	52.4%	50.1%	54.8%	52.5%	0.513	7.44 seconds
	StackModel	54.4%	52.8%	46.0%	53.8%	0.529	6.81 seconds
Standard Analysis System	ChatPhishDetector	58.9%	61.9%	52.4%	61.3%	0.612	20.31 seconds
	VisualPhishnet	50.8%	46.4%	55.2%	50.8%	0.485	5.60 seconds
	StackModel	54.6%	53.9%	45.4%	53.8%	0.534	4.97 seconds

TABLE V: Top 5 Suggested Optimal User Profile

Optimal User Profile (OS, Browser, Network, Location)	#
Windows, Google Chrome, Residential, Japan	6,128 (18.87%)
Android, Google Chrome, Residential, Japan	5,834 (17.97%)
Windows, Google Chrome, Datacenter, United States	5,492 (16.92%)
macOS, Safari, Residential, Japan	2,847 (8.77%)
Android, Google Chrome, Mobile, United States	2,432 (7.49%)

TABLE VI: Top 5 Suggested User-agent and Network Type

User-agent Type	#
Windows, Google Chrome	10,243 (31.53%)
Android, Google Chrome	9,325 (28.70%)
macOS, Google Chrome	5,543 (17.06%)
macOS, Safari	5,136 (15.81%)
iOS, Safari	853 (2.63%)

Network Type	#
Residential, Japan	17,487 (53.83%)
Datacenter, United States	9,978 (30.71%)
Mobile, United States	2,432 (7.49%)
Residential, United States	625 (1.92%)
Mobile, Japan	468 (1.44%)

B. *Evaluation Results*

We conducted a real-time crawling of 32,487 suspicious URLs collected over 21 days from March 1, 2025, to March 21, 2025, using PhishParrot and two baseline systems. We then applied three Phishing Detectors and performed continuous monitoring with VirusTotal. According to VirusTotal's inspection results, 15,309 URLs were identified as phishing sites, while 17,178 URLs were classified as non-phishing sites.

Summary. Table III presents the performance comparison of PhishParrot across different LLMs, while Table IV compares the accuracy between PhishParrot and baseline systems. The evaluation results reveal that GPT-4o mini demonstrates the optimal balance between detection effectiveness and processing efficiency among the compared LLMs. Furthermore, the PhishParrot system using GPT-4o mini achieved Accuracy rates of 92.7%, 84.7%, and 86.7% for ChatPhishDetector, VisualPhishnet, and StackModel, respectively. This represents an improvement of up to 33.8% compared to the baseline system, achieved with only an average increase of about 7.6 seconds in execution time, demonstrating PhishParrot's practicality. These results remain valuable even if some phishing sites are inactive or employ sophisticated cloaking (e.g., advanced bot checks) requiring manual user interaction.

LLMs. In evaluating PhishParrot, GPT-4o mini demonstrated the best overall performance with the highest accuracy, shortest execution time, and lowest cost. While o3-mini recorded the highest precision, it required longer processing time. GPT-4o showed the lowest accuracy and highest cost among the tested models. The difference in detection accuracy among LLM models is within 2.1%, which is not statistically significant. The analysis cost per URL ($0.002–$0.027) is negligible for practical use, and execution time increases can be addressed through parallel processing. Considering these results, GPT-4o mini is determined to be the optimal choice for PhishParrot.

Baseline Systems. PhishParrot using GPT-4o mini significantly outperformed both baseline crawling approaches across all Phishing Detectors. Compared to the Typical User System, PhishParrot achieved accuracy improvements of 28.3%, 32.3%, and 32.3% for ChatPhishDetector, VisualPhishnet, and StackModel, respectively. When compared to the Standard Analysis System, the difference expanded further, showing improvements of 33.8%, 33.9%, and 32.1%. This remarkable improvement demonstrates that the adaptive selection of crawling environments based on LLM analysis and similar case extraction effectively evades cloaking techniques that hinder detection. The increase in execution time was moderate (about 5-6 seconds) compared to the dramatic improvement in detection capability. Notably, PhishParrot showed consistent performance improvement across all detector types, demonstrating the versatility and robustness of the proposed system.

Examples of Optimal User Profile. Analysis of PhishParrot's suggested user profiles revealed distinct patterns in attackers' targeting strategies. Table V shows the top 5 complete combinations, with Japanese users on residential networks dominating (top 2 combinations: 36.84% combined). Windows Google Chrome from Japanese residential networks ranked first (18.87%), followed by Android Google Chrome from Japanese residential networks (17.97%). United States-based profiles also appeared prominently, with Windows Google Chrome from United States datacenters third (16.92%). Notably, macOS Safari from Japanese residential networks ranked fourth (8.77%), demonstrating attackers' awareness of Apple users as valuable targets.

Individual component analysis (Table VI) showed Google Chrome's dominance across all platforms (77.29% total), reflecting the attackers' focus on widely-adopted environments. Geographic analysis revealed a strong concentration in Japanese residential networks (53.83%) and United States datacenters (30.71%). The combined Japanese targeting (55.27%) suggests attackers' particular focus on this region, potentially related to economic characteristics and cybersecurity awareness differences [21]. These insights demonstrate PhishParrot's ability to identify sophisticated, region-specific targeting strategies. This level of granular insight into attacker TTPs (Tactics, Techniques, and Procedures) underscores the limitations of static, rule-based analytical environments.

V. Limitation

PhishParrot has two main limitations. First, the crawling environment selection process introduces additional execution overhead. However, this overhead is offset by the improved phishing site detection rate, and the system's benefits can be maximized by selectively applying it to suspicious URLs. Second, the system requires a certain number of successful and failed examples to reference crawling results from similar attacks. Nevertheless, once several hundred initial examples are prepared (as in our experiment), the system begins to automatically accumulate cases of trending phishing attacks and cloaking patterns. This adaptive mechanism is achieved through continuous operation alongside Phishing Detectors. Therefore, after building the initial dataset, the system is designed to self-improve while adapting to the latest attack trends, maintaining PhishParrot's long-term effectiveness.

VI. Conclusion

This research introduced PhishParrot, a crawling environment optimization system that uses LLMs to counter cloaked phishing sites. By accumulating diverse crawling information and using LLM analysis to mimic attackers' targets, PhishParrot significantly improved phishing detection rates compared to standard analysis systems and typical user systems. The system's ability to construct flexible environments tailored to attackers' targets demonstrates the effectiveness of combining data accumulation with LLM contextual understanding to bypass cloaking techniques. This adaptive approach represents an advancement in phishing detection and provides a foundation for effective cybersecurity defenses against evolving threats.

References

[1] Penghui Zhang et al. Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *Proc. IEEE SP*, 2021.

[2] Penghui Zhang et al. I'm spartacus, no, i'm SPARTACUS: proactively protecting users from phishing by intentionally triggering cloaking behavior. In *Proc. ACM CCS*, 2022.

[3] Sahar Abdelnabi et al. Visualphishnet: Zero-day phishing website detection by visual similarity. In *Proc. ACM CCS*, 2020.

[4] Yun Lin et al. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *Proc. USENIX Security*, 2021.

[5] Ruofan Liu et al. Inferring phishing intention via webpage appearance and dynamics: A deep vision based approach. In *Proc. USENIX Security*, 2022.

[6] Takashi Koide et al. Chatphishdetector: Detecting phishing sites using large language models. *IEEE Access*, 2024.

[7] Bhupendra Acharya and Phani Vadrevu. Phishprint: Evading phishing detection crawlers by prior profiling. In *Proc. USENIX Security*, 2021.

[8] Ruofan Liu et al. Knowledge expansion and counterfactual interaction for reference-based phishing detection. In *Proc. USENIX Security*, 2023.

[9] Daiki Chiba et al. Domaindynamics: Advancing lifecycle-based risk assessment of domain names. *Comput. Secur.*, 2025.

[10] Luca Invernizzi et al. Cloak of visibility: Detecting when machines browse a different web. In *Proc. IEEE SP*, 2016.

[11] Xiwen Teoh et al. Phishdecloaker: Detecting captcha-cloaked phishing websites via hybrid vision-based interactive models. In *Proc. USENIX Security*, 2024.

[12] Takashi Koide et al. Phishreplicant: A language model-based approach to detect generated squatting domain names. In *Proc. ACSAC*, 2023.

[13] Woonghee Lee et al. Beneath the phishing scripts: A script-level analysis of phishing kits and their impact on real-world phishing websites. In *Proc. Asia CCS*, 2024.

[14] Jaime G. Carbonell and Jade Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proc. SIGIR*, 1998.

[15] Patrick S. H. Lewis et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proc. NeurIPS*, 2020.

[16] Yukun Li et al. A stacking model using URL and HTML features for phishing webpage detection. *Future Gener. Comput. Syst.*, 2019.

[17] Hiroki Nakano et al. Canary in twitter mine: Collecting phishing reports from experts and non-experts. In *Proc. ARES*, 2023.

[18] Victor Le Pochat et al. Tranco: A research-oriented top sites ranking hardened against manipulation. In *Proc. NDSS*, 2019.

[19] Peng Peng et al. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proc. IMC*, 2019.

[20] Shuofei Zhu et al. Measuring and modeling the label dynamics of online anti-malware engines. In *Proc. USENIX Security*, 2020.

[21] Global Anti-Scam Alliance. 2024 state of scams in japan report, 2024. https://www.gasa.org/post/2024-state-of-scams-in-japan-report.