A Closer Look At Modern Evasive Phishing Emails

Elyssa Boulila*‡, Marc Dacier†, Siva Prem Vengadessa Peroumal*, Nicolas Veys*, Simone Aonzo‡
* Amadeus IT Group, France † KAUST, Saudi Arabia ‡ EURECOM, France

***Abstract*—The prevalence of sophisticated evasion techniques employed by phishing attacks in circumventing anti-phishing and email security measures is on the rise. The present study offers an exhaustive analysis of user-reported phishing messages pertaining to five companies over a ten-month period. These messages are of particular interest as they evaded all state of the art security layers in place and were identified by the recipients themselves.**

To study these elusive phishing attempts, we developed an analysis infrastructure, CrawlerBox, designed to overcome cloaking tactics that exploit browser fingerprinting and bot detection challenges. CrawlerBox is made available as an open-source tool to assist other researchers in pursuing further studies.

Over the course of ten months, we gathered 1,551 user-reported messages that were confirmed to be malicious, with a particular focus on those targeting the harvesting of corporate credentials. Our analysis infrastructure enabled us to scan these messages and crawl any associated web resources, including URLs, embedded HTML, and JavaScript content.

Our findings indicate that the majority of observed phishing attacks are low-volume but meticulously planned, exhibiting a high degree of premeditation and strategic preparation. In particular, a substantial number of sites were registered and had obtained TLS certificates several weeks before the attacks in order to avoid being flagged based on their young age, a common practice used by products to defeat phishing websites. Furthermore, it was observed that phishing pages are now safeguarded by advanced evasion mechanisms, such as bot detection services and open-source fingerprinting libraries. Notably, QR codes are increasingly used to embed phishing content. The victim needs to use a personal phone to flash the code and access the site; this activity will typically fall outside the perimeter of the corporate security defenses. These findings underscore the evolving sophistication of phishing threats and the urgent need for resilient systems to counter these advanced techniques, further emphasizing the critical role of robust infrastructures like CrawlerBox in enhancing the security and dependability of email systems.

***Index Terms*—phishing, crawler, cloaking**

I. Introduction

Phishing attacks have evolved into one of the most pervasive and dangerous threats in the digital landscape, exploiting human vulnerabilities and technical loopholes alike. They have undergone extensive research, and various detection systems have been suggested. Nevertheless, this phenomenon is still on the rise and has recently reached an all-time high [1]. Combating malicious emails resembles a perpetual cat-and-mouse chase. New techniques rise continuously, making it possible to bypass detection. In this paper, we assess the evasion mechanisms used today by threat actors to slip through the cracks of commercial email security filters. We propose to study these techniques' sophistication, prevalence, and efficiency.

Through industry cooperation, we had access to ten months of email exchanges pertaining to five organizations: one major multinational technology corporation and four additional companies which email security is overseen by the first organization. The multinational corporation provides software solutions for the global travel and tourism industry, with more than 20,000 collaborators distributed across four continents. The four other companies each focus on different business activities, including travel platforms, content aggregation, booking engines, revenue management, merchandising, transportation, and payment solutions. This range of services illustrates the diverse activities within the studied environments. As is customary in such cases, the architecture of the system in question entails the preliminary analysis of any email sent to an employee by a number of automated commercial security mechanisms. If the message is classified as *clean*, i.e., it has not been found to contain any kind of threat, it is delivered to the recipient's inbox. However, such security mechanisms are far from perfect, as we will show in this paper. Some messages that are considered clean may actually be false negatives (FNs). Thanks to a strong security awareness program, some FNs are detected by end users who have been instructed to report malicious emails.

This study focuses on user-reported emails that have been subsequently confirmed as malicious by expert analysts. These are the messages that evaded the detection of the entire set of security layers in place, yet were identified by human recipients. The data in question represents a veritable gold mine of information, as it comprises the messages that were able to circumvent the main security layers.

To deceive the evasion techniques that they might be relying on, we develop our own analysis infrastructure. It extracts web resources (including URLs and HTML/JavaScript) embedded in the reported messages. It loads them afterward in a carefully crafted sandbox, which we dub *CrawlerBox*, constructed with high stealthiness requirements. Given that some phishing pages are protected with bot detection or browser fingerprinting services, as we will show in our paper, we made sure that *CrawlerBox* defeats such state-of-the-art services by incorporating an evasive crawler, which we refer to as *NotABot*.

In this work, we present our analysis infrastructure, the results obtained from crawling evasive phishing pages, and an inspection of the evasion techniques used by threat actors. Our findings, as observed by analyzing user-reported emails from the studied companies, indicate that the malicious messages in question are predominantly associated with low-volume attacks, namely, infrequent campaigns against which automated systems have received limited data, thereby impeding their ability to detect them on a large scale. Moreover, we observe that the majority of the landing domains are created long before the reception of the phishing messages. This suggests that attackers register their phishing domains in advance before starting their campaign. Interestingly, we notice that some rely on relatively new services or open-source fingerprinting libraries to bypass detection. In particular, QR codes are leveraged to embed malicious URLs. Access to the website will not be done from the corporate machine protected by the corporate defenses but from the smartphone that flashes the QR code. As of now, most personal smartphones are typically less monitored and protected than machines in the corporate network. This is another way to deceive the security measures in place within companies.

We even discovered a bug exploited in the wild that is used to hide a URL inside a QR code without getting extracted by email filters at scanning time. The wide usage and the diversity of the found evasion techniques suggest that attackers are putting great effort today into protecting their phishing pages from detection.

The main contributions of this article are:

- The development of an automated analysis infrastructure, including a crawler with the capacity to circumvent sophisticated browser fingerprinting and bot detection services. It is made available as open source [2].
- Providing a comprehensive insight into the nature of phishing based on an analysis of user-reported messages from five companies operating across diverse sectors.
- Identifying the most prevalent evasion techniques, thereby indicating the areas where technological resources should be concentrated to counter modern phishing.

All in all, this study offers a wealth of contributions that can significantly enhance the security and dependability of email systems by addressing both the technological and behavioral dimensions of email-based threats. Namely, by bridging the gap between human-reported observations and automated system capabilities, this research not only bolsters the technological infrastructure of email systems but also emphasizes the indispensable role of end-user awareness and vigilance in modern cybersecurity.

To discuss these topics, this article is structured as follows. Section II presents the state of the art and positions our work. Section III surveys the known evasion techniques. We introduce in Section IV our analysis infrastructure and in Section V our results and the newly discovered evasion implementations that threat actors arm their infrastructure with to bypass detection. In Section VI, we outline the key takeaways of our study and provide a detailed discussion of the obtained results. Then, in Section VII, we describe the limitations of this work. Finally, our paper concludes in Section VIII, where after drawing the final remark we describe our responsible disclosures.

II. Related Work

In this section, we discuss prior work pertaining to the phishing ecosystem measurement and analysis. We also describe how our work is positioned in relation to the state of the art.

A. *Studying the phishing ecosystem*

In order to study the phishing ecosystem, previous research often relied on public sources (mainly the Certificate transparency logs, and open-source blacklist databases) [3]–[12]. Our research focuses on phishing targeting specifically a corporate environment, based on live user-reported phishing messages received by multiple companies in the travel and tourism industry. We make a closer examination of the phishing landscape, and the techniques employed to evade detection.

Meaningfully assessing long-term trends in the volume of phishing attacks has been shown to be challenging [13]–[15]. In 2009, Moore et al. [16] examined both phishing websites and the associated spam messages. The study revealed that spam was predominantly sent around the time the phishing websites emerged, leading several products to use the registration time of the website and the creation of its TLS certificate as an indicator of maliciousness. As we will show in our study, this approach is becoming less effective as attackers have taken it into account in their modus operandi.

In 2020, Oest et al. [17] studied the end-to-end life cycle of phishing attacks while passively measuring victim traffic to phishing pages. It was found that the typical phishing attack lasted 21 hours from the first to the last victim's visit and that anti-phishing entities identified each attack nine hours after the first victim's visit. This highlights the fact that phishing attacks are generally short lived. Yet, attackers attempt to maximize any victim impact within that brief attack period. Later in 2021, Bijmans et al. [5] presented an empirical, longitudinal measurement study of the end-to-end life cycle of Dutch phishing campaigns. Interestingly, 69% of the phishing domains returned a blank screen and no favicon. This measurement confirmed that a large number of domains were either inactive at scanning time or were already implementing techniques, known as "cloaking mechanisms", to hide their true nature from anti-phishing actors. We present cloaking in detail in Section III. In 2022, Subramani et al. [4] developed a crawler that scanned phishing URLs while simulating user interactions. Modern user verification systems (including CAPTCHAs) were seen in some of the studied phishing websites.

In 2024, [18] found that national companies and organizations are far more often impersonated using malicious newly registered domains under their country's own ccTLD (country code Top-Level Domain). Whereas, international companies are impersonated using compromised domains, decreasing overall mimicry but avoiding any registration costs.

So called "phishing kits" are unified collections of tools used to deploy phishing sites on web servers [19]. In 2008, Cova et al. [20] focused on the analysis of 'free' phishing kits. They were found to contain backdoors which exfiltrated the gathered information. This conclusion was also verified and generalized in more recent work in 2022 by Tejaswi et al. [21].

Oest et al. [19] analyzed phishing kits in 2018 and revealed the used *.htaccess* server-side filtering techniques. In 2021, Bijmans et al. [5] identified 70 distinct phishing kits by fingerprinting the kits based on their unique properties (including file names, paths, and uncommon strings). The authors identify 10 different phishing kits families.

In 2022, Merlo et al. [22] analyzed PHP, JavaScript, and HTML codes from 20,871 kits collected from forensic teams of private security firms. The static similarity analysis revealed that 90% of the kits share 90% or more of their source code with other kits.

In 2016, Han et al. [23] proposed a web honeypot which attracted real attackers into installing phishing kits in a compromised web application. The authors observed an average blacklist detection latency of 12 days.

More recently, Kondracki et al. [24] presented in 2021 the first analysis of MITM phishing toolkits used in the wild. These toolkits operate as reverse proxy servers, positioned between victims and target web servers. They mimic web servers when interacting with victims while functioning as clients when connecting to the target web servers. The authors' findings revealed that phishing sites relying on MITM phishing toolkits are significantly underrepresented in anti-phishing blocklists. Specifically, only 43.7% of domains and 18.9% of IP addresses associated with these toolkits appeared on blocklists. These results emphasize the importance of developing more advanced anti-phishing bots for scanning malicious sites.

In 2024, Lee et al. [8] conducted a comprehensive study of phishing kits at the script level. The utilized web crawler (Selenium Chrome WebDriver) failed to evade most of the sophisticated cloaking mechanisms (e.g. fingerprinting-based techniques). During our study, we made sure to develop an evasive crawler in order to correctly assess the analyzed phishing landscape – presented in Section IV.

B. Comparison with our work

Previous work [8] confirmed that attackers are increasingly relying on evasion techniques, and more particularly on one subcategory called "cloaking" to avoid detection. Some security crawlers attempt to overcome this by simulating user interaction for example. Yet, many fail to bypass advanced evasion strategies [6], [9], [11], [25]–[28].

We propose in this paper to make a deep analysis of malicious user-reported emails from five companies. These messages reached the users' inboxes despite all the security defense layers in place, which makes them particularly stealthy. We build our own analysis infrastructure (Section IV) and we examine the prevalence of cloaking techniques associated to these messages (Section V). The objective is to have a good understanding of today's phishing ecosystem, while taking into consideration the threats that usually fly under the radar of automated security scanners.

III. Taxonomy

The lifecycle of a phishing message from its creation to delivery can be broken down into four distinct stages.

1) The message is prepared by the threat actor with the intention of luring the victim into visiting a malicious URL embedded within the message. This URL usually points to a website that allows the user to inadvertently leak sensitive information. To send the message, attackers may use compromised email servers or accounts, third-party services, or malicious sender accounts. They may also employ tactics such as domain spoofing or email header manipulation to disguise the true origin of the message.
2) The email is sent to the recipient using the Simple Mail Transfer Protocol (SMTP). At this point, some evasion techniques can be used, such as SMTP smuggling [29].
3) Once the email reaches the recipient's mail server, it undergoes various security checks before being delivered to the inbox. At this point, the server checks the integrity and the origin of the message with SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), and DMARC (Domain-based Message Authentication, Reporting and Conformance). The email is parsed and its attachments are scanned for known malware signatures. The embedded links are checked against regulary updated URL denylists or analyzed in real time in an isolated environment. Additional checks such as content filtering (based on, for example, phishing keywords, obfuscated text, or brand impersonation) and reputation-based scanning can be applied.
4) If the message is categorized as benign, it is delivered to the recipient's inbox. If not, it is quarantined or deleted.

This lifecycle is designed to ensure that phishing emails are detected and blocked before reaching the recipient. Nonetheless, sophisticated phishing campaigns might evade these defenses.

In this section, we propose a taxonomy of the evasion techniques employed to bypass the email security parsers and scanners. In particular, we categorize these techniques into two distinct classes:

1) **Message-level evasion:** it refers to techniques used to hide an embedded URL in a message, such that it is not successfully retrieved by email security parsers. This category also includes alterations made to a message to deceive anti-phishing tools.
2) **Cloaking:** it refers to techniques used when a client visits a phishing website. The goal is to conceal the phishing page from web crawlers and security bots (stage (3)), while keeping it visible to human victims [30]. If the visitor is identified as a bot, a benign content is displayed. Thereby, security detection efforts are obstructed.

A. Message-level evasion

Message level evasion can be based for instance on obfuscation, character substitution, and character encoding (where parts of the message are encoded in Base64 for example). Attackers might also embed malicious text in images to evade text-based filters [31]. This technique was proved to be efficient especially with the rise of malicious QR codes [32].

To evade content-based detection, attackers might add benign content (noise) to dilute malicious signals or to mimic the behavior of legitimate emails (e.g., including a lengthy newsletter or formal corporate email structure) [33]. In other cases, the message might contain minimal content (e.g., a single malicious link or one sentence), limiting the information available for behavioral analysis systems to flag the email [34].

B. Cloaking

Cloaking has gained popularity among attackers [9]. Yet, security scanners have struggled to keep pace with these evolving threats [6], [9], [28]. It can occur on both the client side and the phishing server side, depending on where the logic for concealing the phishing content is applied. Client-side cloaking involves techniques executed within the user's browser or system. For example, some checks are run on the client to collect information about the environment or the user's behavior. Depending on the result, a different content is displayed to the visitor (benign or malicious). Server-side cloaking is executed on the phishing server itself. In this case, the server decides what content to serve based on attributes of the incoming request, such as IP address or User-Agent. The phishing content is only displayed to targeted users.

In this Section, we present the known client-side and server-side cloaking techniques.

1) Client side cloaking techniques: In 2021, Zhang et al. [9] presented an in-depth analysis of client-side cloaking techniques found in the wild and classified them into three high-level categories:

1) *User interaction:* the phishing content remains hidden until a button in a pop-up window is clicked or if a mouse activity is detected. In other cases, user interaction such as scrolling or filling out form fields are needed to show the malicious content.
2) *Fingerprinting:* if the cookies are disabled or if the browser cache is enabled, the client is detected as an automated environment and the phishing page is not displayed. This detection can also be made based on the origin of the traffic (by checking the *Referrer* field), or on the user agent string, accessed via the `navigator.userAgent` property. Some additional checks can be made based on attributes like the screen resolution for instance.
3) *Bot behavior:* the phishing content is randomly displayed or is delayed by a few seconds. While scanning a URL, some security crawlers do not wait enough time before the page is reloaded with malicious content. This leads to a false negative verdict. Additionally, Google reCaptcha was seen in previous phishing campaigns since at least 2020 [25]. It presents the user with the "I'm not a robot" checkbox, prompting them to solve image-based puzzles.

2) Server-side cloaking techniques: Based on decisions made by the phishing server, they include:

1) *Delaying the activation of a phishing URL:* before its activation, all visitors are redirected to a benign page. This technique can be used to prevent email security filters from reaching the malicious page while scanning the URL extracted from an incoming message. For example, attackers might choose to send a phish message when the victim is supposed to be unavailable (in the middle of the night for instance). At delivery time, the URL does not display any malicious content. A few hours later, the URL is activated.
2) *User Agent filtering:* the phishing content is only accessible for specific clients, for example the ones associated with a mobile browser (such as Safari on iOS). This technique might be part of the attacker's strategy, for instance, when the phishing URL is embedded in a QR code, which should normally be decoded by a mobile phone, not a desktop or a laptop.
3) *IP blacklists:* used to block access to clients whose IPs are associated to known security scanners.
4) *Displaying the phishing site only if the GET request is made to a valid tokenized URL:* The attacker generates URLs containing unique tokens. For example, a URL might look like the following: *https://evil-site.com/dhfYWfH*, where the token is "dhfYWfH". Any request lacking a valid token is redirected to a benign webpage. Additionally, attackers can disable individual tokens, preventing even those URLs from displaying the phishing content.

IV. Analysis Infrastructure

In order to study evasive phishing messages, we had access to all the emails reported by the employees of a major multinational technology company, supervising the security of four additional companies. The multinational corporation specializes in software solutions tailored for the global travel and tourism industry, with a workforce exceeding 20,000 employees across four continents. The remaining four companies operate in various sectors, encompassing travel platforms, content aggregation, booking systems, revenue management, merchandising, transportation, and payment processing. This diversity underscores the wide-ranging operational environments considered in our study.

Figure 1 shows our analysis pipeline. It is composed of 3 phases: (1) fetching newly reported malicious messages and pruning them, (2) parsing and crawling the found web contents, and (3) logging the results.

A. Pruning the data

In total, these companies handle over 60 million inbound emails monthly. These email messages are initially intercepted and analyzed by state of the art security products. Only the messages classified as *clean* are delivered to the intended recipients. At this stage, 17% of all messages are filtered out because they were detected as malicious or spam. Among the remaining 83%, about 14,000 are monthly reported as suspicious by end-users (corresponding to 0.03% of the total delivered messages). The companies in question maintain a robust awareness policy, actively encouraging their staff and consultants to report any suspicious emails they receive and frequently engaging them in phishing simulation campaigns. Then, once a message is reported, a security expert examines it, and, in the end, one of three possible tags is given: malicious, spam, or legitimate.

The primary distinction between emails labeled as "spam" and those categorized as "malicious" lies in their intent and potential risk. Spam refers to unsolicited messages primarily used for undesired advertising and promotions. In contrast, malicious messages are associated with fraudulent activities, including phishing attempts. These messages seek to deceive recipients into divulging sensitive information, e.g., login credentials, credit card details, or confidential data, by masquerading as communications from trusted entities. On average, among the reported emails, about 3.7% are found to be malicious, while the

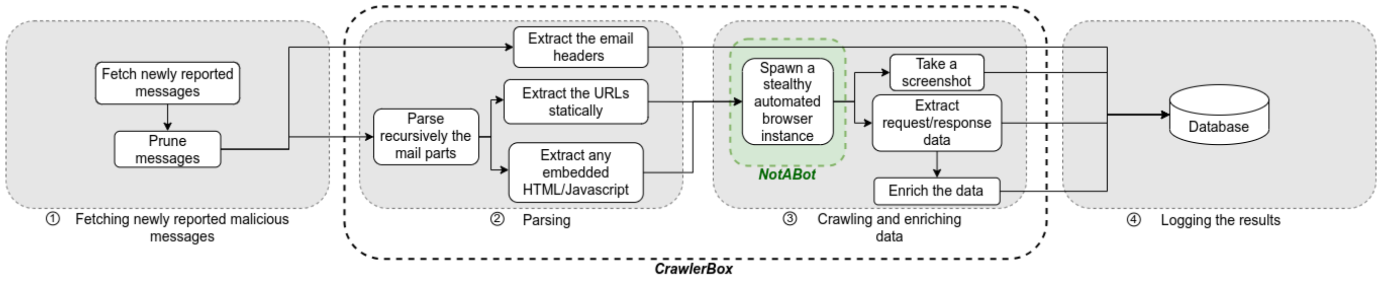
Fetch newly reported messages
Prune messages
① Fetching newly reported malicious messages
Parse recursively the mail parts
Extract the email headers
Extract the URLs statically
Extract any embedded HTML/Javascript
② Parsing
Spawn a stealthy automated browser instance
NotABot
Take a screenshot
Extract request/response data
Enrich the data
③ Crawling and enriching data
Database
④ Logging the results
CrawlerBox

Figure 1: *CrawlerBox* Analysis Pipeline

rest are flagged as either legitimate (35.0%) or spam (61.3%). In absolute numbers, 500 are reported and confirmed as malicious every month, i.e., 25 per working day on average. While spam emails present a fascinating area of research, they fall outside the scope of this work, which concentrates on emails with malicious intent.

In this work, we only consider emails that are tagged as “malicious”. As such, our objective is not to perform phishing detection but rather to carefully study stealthy messages and gain insights into the tactics used to fly under the radar of security products. Therefore, our ground truth is based on user-reported suspicious emails later confirmed as malicious by security experts. Among these “malicious” messages, and as we will further explain in Section V, 29.9% lead to active phishing, namely, such emails lead to an active web page with a login form impersonating a trusted entity.

To analyze this very interesting and rich dataset, we developed an analysis infrastructure which we call *CrawlerBox* (4600 lines of Python code), and we release its source code as a contribution of this work [2]. Moreover, given that the detection of automated tools (bots) follows a continuous adversarial cycle, *CrawlerBox* has been designed with a modular architecture, allowing for interchangeable use of the crawling component. We named our crawler *NotABot*. It was engineered to circumvent the evasion techniques presented in Section III. Finally, given that phishing URLs may be short-lived, *CrawlerBox* analyzes the reported emails as soon as they are tagged by experts.

B. CrawlerBox– Parsing phase

First, each message’s mail header is extracted. Then, we carefully developed a parser that scans recursively all the parts and subparts of an email message. Depending on its content type, as extracted from the associated ”Content-Type” header field [35], each part is parsed differently. Globally in our dataset, the most prevalent content types are: HTML, images, Octet Stream files (i.e., binary files), EML (short for electronic mail, in our case EML attachments), text (including Rich Text Format attachments), PDF, and ZIP files.

Our URL extraction methodology works as follows:

- URLs are statically extracted from text-based formats.
- Inline and attached images are scanned for the presence of URLs (using a combination of Optical Character Recognition libraries) and QR codes. URLs are extracted from the found QR codes if any.
- For PDF files, we use two approaches: (1) extracting embedded and text-based URLs, and (2) taking a screenshot of each page, which is then analyzed like the images described above.
- Octet Stream files are analyzed according to their file signature determined by magic numbers.
- Any discovered HTML or JavaScript code is dynamically loaded with our crawler, as we explain below. Dynamic analysis in our case is fundamental given the use of obfuscation to hide malicious URLs.
- ZIP files are unpacked, and each file within is subjected to the appropriate analysis described in this list.
- Finally, EML files are processed recursively.

C. CrawlerBox– Crawling phase

The crawling phase is thoroughly logged, capturing the visited domains, their associated TLS certificates, corresponding IP addresses, as well as the requests and responses exchanged with the browser (including the content, the loaded HTML and JavaScript). The collected data is enriched with WHOIS information, Shodan [36] service banners and Cisco Umbrella [37] details. Moreover, once the page is fully loaded, a screenshot is taken.

The objective of this phase is to visit the malicious sites while evading cloaking techniques. In this context, we introduce *NotABot*, the crawler component within *CrawlerBox*. We show in Section IV-D that it is able to bypass advanced bot detection solutions. *NotABot* is based on Puppeteer [38]. An instance of the original Chrome browser is launched for each crawled website or retrieved HTML/JavaScript code. We chose it since it is the most popular browser [39]. Running the real Chrome browser in non-headless mode helps to avoid a few browser inconsistencies that could be used to detect the crawler. To defeat some timing tests that can be used to detect when the browser is running in a virtual machine [40], we use a Dell Precision 3571 physical machine (Intel(R) Core(TM) i7-12800H 2.40 GHz, with 32 GB of RAM) running Windows 11. Moreover, detection techniques can identify bots by checking whether the associated IP address is associated with cloud providers, proxies, or VPNs [41]. To circumvent this issue, our machine is connected to the Internet using a 4G modem with a SIM card under a commercial mobile data plan.

Next, we hide the attributes that are injected by Puppeteer. Recent studies [42] [43] show that websites and bot detection companies extensively depend on attributes introduced by instrumentation frameworks, such as `navigator.webdriver`, to identify crawlers. This property indicates if the user agent is under automation control [44]. To bypass this detection, we disable the `AutomationControlled` flag, which sets the property value to *False* before loading the page content.

During our early experiments, to automatically log the requests and responses exchanged by the browser, we enabled request interception in Puppeteer and set up listeners for the traffic. By construction, once request interception is enabled, every request will stall unless it is continued, responded to, aborted, or completed using the browser cache [45]. Unfortunately, this setting resulted in a peculiar browser caching behavior, which made our crawler identifiable using the HTTP headers Cache-Control and Pragma. Interestingly, we noticed that this could be evaded by disabling the page request interception in Puppeteer while also keeping the handlers for logging the traffic.

Finally, to avoid behavioral analysis detection, *NotABot* generates fake mouse movements. Since Puppeteer relies on the Chrome DevTools protocol, these actions are recognized as legitimate by the browser, thus setting the browser's *isTrusted* event property to True [41]. Fingerprinting scripts often use this property to check if the browser ran programmatic actions (actions not initiated by a human).

D. *Fingerprint test of NotABot*

To test the robustness of our crawler *NotABot*, we check its associated verdict when challenged against popular browser fingerprinting services and advanced bot detection tools. Furthermore, we assess its performance in comparison to state-of-the-art open-source security crawlers.

1) *Passing basic bot detection tests*: We use BotD [46], an open-source library designed for detecting basic bots.
2) *Passing advanced JavaScript-based anti-bot challenges*: We employ Cloudflare's Turnstile [47], an alternative to conventional CAPTCHA systems. It executes a sequence of JavaScript challenges that collect data about the browser environment, incorporating methods like proof-of-work, proof-of-space, web API probing, and other techniques to detect browser quirks and human behavior.
3) *Evading the detection of an advanced anti-bot commercial solution*: We visit a website protected by an advanced commercial Web Application Firewall (WAF) with each tested crawler. Due to legal restrictions, we cannot explicitly name this WAF, and therefore, we will refer to it as AnonWAF. This solution employs sophisticated techniques, including TLS fingerprinting, behavioral analysis, JavaScript fingerprinting, and HTTP header inspection, to identify bot activity. We check the verdict associated with each visit by accessing the WAF's logs.

For our comparison, we choose seven open-source crawlers, selected for their reputation as state-of-the-art tools in this field. They are widely recognized for their advanced capabilities and frequent use in security research and threat analysis. First, we choose Kangooroo [48], a Java utility designed by the Canadian Center for Cyber Security for crawling malicious URLs. Today, it is integrated with Assemblyline's [49] URLDownloader service. Second, we select Lacus [50]. It is a web capturing system built on Playwright and is integrated within the AIL framework [51]). Third, we test Puppeteer with the `puppeteer-extra-plugin-stealth` package [52]. This plugin helps mitigate bot detection mechanisms by disguising headless browsing indicators. Fourth, we select Selenium associated with the `selenium-stealth` package [53]. According to its official description, this implementation helps to "prevent almost all selenium detections." Fifth, we choose `undetected_chromedriver` [54]. This tool is described as "an optimized Selenium Chromedriver patch which does not trigger anti-bot services." Finally, we used Nodriver [55] and Selenium-Driverless [56] for the sixth and seventh. These last two frameworks do not rely on Chromedriver or Selenium. Instead, they leverage low-level Chrome DevTools Protocol (CDP) commands to implement bot automation functions. These two projects are relatively recent, and our research predates their development. Furthermore, they have not been presented in any previously published peer-reviewed studies. However, for the sake of completeness and in recognition of their promising results, we have included them in our assessment.

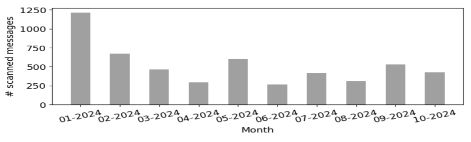

Figure 2: Number of the scanned messages per month

Table I presents a summary of our assessment. Only three out of eight crawlers, including *NotABot*, were able to bypass all the bot detection tools. In fact, *NotABot* passes both Turnstile and AnonWAF without requiring any interaction. As explained in more detail in the last section of the paper, we responsibly disclosed this finding to both companies. Cloudflare rewarded us with a bug bounty associated with a medium-severity vulnerability, while the anonymous company developing AnonWAF acknowledged the issue and is working on remediation. These recognitions highlight our implementation's novelty and effectiveness.

However, Nodriver and Selenium-Driverless also achieve the same stealthiness levels. The reason why we developed *NotABot* is because when the project started in January 2024, they were not available. For future work, we consider expanding *CrawlerBox* by integrating them. Diversifying crawler components within an analysis infrastructure such as *CrawlerBox* can only be beneficial in studying sophisticated phishing campaigns.

V. Analyzing User-Reported Emails

For 10 months, from January 2024 to October 2024, when the experts classified 5,181 reported emails as malicious,

Table I: Assessment of open-source crawlers in defeating SOTA bot detection tools

Crawler / Tool	Kangooroo	Lacus	Puppeteer + stealth plugin	Selenium + stealth plugin	undetected_ chromedriver	Nodriver	Selenium-Driverless	***NotABot***
BotD	×	√	√	×	√*	√	√	√
Turnstile	×	×	×	×	×	√	√	√
AnonWAF	×	×	×	×	√	√	√	√

[1] (√) The examined crawler is able to evade detection and pass the test.
[2] (×) The examined crawler fails the test.
* Only when used in non-headless mode.

CrawlerBox immediately analyzed them. Figure 2 presents the distribution of the scanned messages over time. Our infrastructure processed 518.1 messages per month on average, with a standard deviation equal to 278.4.

Furthermore, we had access to the final 10 months (March to December) of 2023 despite the fact that our study had not yet commenced at that time. During the aforementioned time frame, the experts identified an average of 885.2 user-reported messages per month as phishing [1], with a standard deviation of 454.7. In fact, the apparent peak observed in January 2024 is instead part of a downward trend. In the final three months of 2023, experts identified 1,959, 1,533, and 1,249 phishing messages, respectively.

A closer look at our dataset of 2024 revealed that: In 2,572/5,181 cases (49.6%), the messages do not contain any embedded web resources (such as URLs); these are generally associated with fraud when attackers try to establish first contact with the recipient. An example of such deceptive emails is a plain-text message impersonating the billing department of a partner company, falsely asserting a past-due balance and pressuring the recipient to reply urgently to facilitate payment. This tactic often employs the threat of service disconnection to enhance the perceived legitimacy of the fraudulent claim. Whereas 823 messages (15.9%) lead to error pages (NXDomain error, page unreachable, etc.), despite our efforts to analyze them in real time. These instances might correspond to phishing websites that had been deactivated. However, some may be attributed to phishing pages that *CrawlerBox* was unable to access due to server-side filtering mechanisms, such as geolocation restrictions or User-Agent filtering (e.g., requiring a mobile-specific User-Agent). Furthermore, 235 messages (4.5%) lead to pages requiring specific user interaction (e.g., a Dropbox document, a Google Drive page, or a website requiring solving a traditional CAPTCHA system involving image-based puzzles). Only five (0.1%) messages resulted in the download of external files, specifically ZIP archives. Each archive contains an HTA (HTML Application) file that retrieves a JavaScript file from a domain identified as malicious by VirusTotal (more than 16/94 detections). HTAs are Windows applications that utilize HTML and scripting languages. They are executed using `mshta.exe` and run with the same privileges as the user, granting them extensive access to system resources, including the ability to create, modify, or delete files and registry entries. This elevated level of trust makes HTAs a potential security risk; therefore *CrawlerBox* does not run them. To err on the side of caution, an analysis of the Endpoint Detection and Response (EDR) logs associated with our machine confirmed that no compromise had occurred. Finally, 1,551 out of 5,181 (29.9%) correspond to active phishing; namely, such emails redirect to a web page with a fake login form.

[1]A paired samples t-test was employed to compare the means of the 2023 and 2024 datasets (the two means originate from the same group and were measured at two distinct points in time), yielding a p-value of 0.008. Given a significance level of $\alpha = 0.05$, the null hypothesis is rejected, indicating a statistically significant difference between the two years.

A. Spear phishing

Given *CrawlerBox* takes a screenshot every time the browser loads a webpage, we compared such screenshots with the five known legitimate login pages of the companies under study.

However, screenshots associated with these messages often contain the victim's email address and some injected noise. To identify such malicious portals, we use fuzzy hashes: pHash (perceptual hash) and dHash (differential hash). The perceptual hash was also used in previous work [9] while studying client-side cloaking techniques. Both are robust against small alterations in the images, such as scaling, cropping, or noise. The (dis)similarity is measured by the hamming distance between the screenshot's hash and the hash of the real legitimate pages. We manually define a threshold under which we confirm that two images are considered similar. The combination of both hashes proved to result in better performance in identifying fake lookalike login pages.

Using this method, 1,137/1,551 (73.3%) messages were automatically classified with high confidence as spear phishing attacks because they led to the display of fake, well-known, lookalike login pages that were specific to a particular organization. This observation warrants careful interpretation. The phishing messages analyzed were those bypassing automated defense systems but subsequently reported by company employees. Notably, approximately three-quarters of these emails were targeted attacks specifically directed at the companies. This finding can be understood in two ways, not mutually exclusive. First, attackers often study defense mechanisms and craft phishing messages capable of evading detection. Second, employees may more readily recognize phishing attempts that mimic their own company's branding or target their specific organizational context. However, irrespective of the reason, this measure suggests a complementary relationship between technological solutions and human intervention, emphasizing the need for a multifaceted approach to dependable systems.

We collect a total of 1,438 distinct landing URLs and 522 distinct landing domains. Table II shows the distribution of the associated top-level domains (TLDs). We observe that domains with the `.com` TLD are the most common, confirming the result obtained by Kondracki et al. [24]. Surprisingly, the second most common TLD is `.ru`. The widespread presence of `.ru` domains suggests that registering Russia-based domains offers some advantages to phishers (cheaper or easier to register domain names for example). This theory is confirmed by Liu et al. [57]: due to Russia's `.ru` registry policy and price changes, the number of `.ru` spamming domains have increased in the past years. Note that the companies which we are collaborating with had no Russian headquarters during the analysis period, excluding that threat actors would use the `.ru` TLD for adding any sense of legitimacy in tricking victims. The associated `.ru` registrars are: REGRU-RU, R01-RU, RU-CENTER-RU, REGTIME-RU, and OPENPROV-RU.

After identifying the phishing URLs leading to fake lookalike login pages, we inspect the requests made by our automated browser while visiting these pages. We notice that in 339/1,137 cases (29.8%), the client downloads in real-time embedded resources, including the logo and the background image from the third-party domains belonging to the organization being impersonated. This is a crucial observation because by identifying referrals in requests made for the aforementioned web resources within their own systems, organizations can track, at early stages, pages impersonating their login sites.

Table II: Number of phishing domains discovered per TLD

Rank	TLD	Domains	Rank	TLD	Domains
1	.com	262 (50.2%)	6	.xyz	9 (1.7%)
2	.ru	48 (9.2%)	7	.org	8 (1.5%)
3	.dev	45 (8.6%)	8	.click	7 (1.3%)
4	.buzz	27 (5.2%)	9	.br	7 (1.3%)
5	.tech	9 (1.7%)	10	Other	100 (19.2%)

Next, we evaluate the phishing deployment timeline from creation to delivery by analyzing the landing domains' WHOIS information, TLS certificate details, and the associated messages delivery timestamps. For each domain, we compute the average time difference between the domain registration, the issuance of the TLS certificate, and the average delivery timestamp of the associated phishing messages.

We refer to "timedeltaA" as the time difference between the registration of the domain and the average delivery time of the messages. Similarly, we refer to "timedeltaB" as the time difference between the creation of the domain's TLS certificate and the average delivery time of the messages.

The kurtosis values associated with the distributions of timedeltaA and timedeltaB are equal to 8.4 and 6.8 respectively, indicating fat tails distributions (right-skewed). For better visualization of this data, we focus in Figure 3 on the domains where timedeltaA and timedeltaB are under 90 days. Notably, 102 domains have a timedeltaA over 90 days. But only 5 domains have a timedeltaB over 90 days. Four out of the five domains are legitimate but were compromised to host phishing. We choose the 90-day duration because several security and anti-spam firms use this value as a threshold [58]: domains younger than 90 days are given the "new domain" status, resulting in low reputation scores.

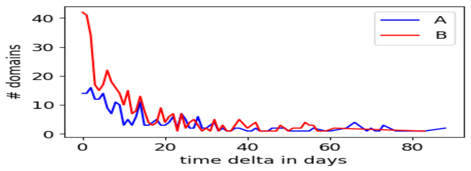

Figure 3: Domain count per average time difference under 90 days (A) between the domain registration and the average message delivery timestamp, and (B) between the TLS certificate creation time and the average message delivery timestamp

The median time for delivering a phishing message is 575 hours (approximately 24 days) after creating the landing domain, and 185 hours (approximately 8 days) after issuing the associated TLS certificate. Email security filters use a domain's reputation as a key factor in determining its maliciousness score, with the age of the domain frequently impacting its reputation. This would potentially explain why, nowadays, some threat actors carefully register in advance the phishing domains before sending the malicious messages. This approach contrasts with the tactics seen a few years ago, when spam was typically sent shortly after the associated phishing websites were created [16]. By registering phishing domains well before sending the associated emails, attackers demonstrate their ability to adapt to modern defensive measures that rely on reputation-based detection.

For both distributions, we check the outliers corresponding to 71 distinct domains out of 522 (having a timedeltaA over 273 days or a timedeltaB over 45 days). After a close inspection, we make the following observations: 42 of these domains are created from scratch. Whereas at least 20 domains are legitimate domains corresponding to small businesses which were exploited to host phishing. The remaining cases (9 domains) correspond to legitimate services abused to host malicious webpages (vercel.app, cloudflare-ipfs.com, workers.dev, r2.dev, oraclecloud.com, and cloudfront.net).

Our analysis also reveals that the average number of reported messages linked to the identified landing domains is 2.62. Some landing domains are linked to multiple reported phishing messages (maximum number equals 58). Yet, the median number is only 1.0. This suggests that many of these phishing attempts may be part of more targeted attacks rather than large-scale phishing campaigns [2]. This could indicate that a significant portion of the associated landing domains could be designed for specific, low-volume operations, making them harder to detect in targeted scenarios.

[2]Please note that these numbers are based only on self-reported messages from the studied companies. We cannot generalize to all email traffic due to legal reasons since this measurement would require accessing the content of all employees' emails.

To verify this statement, we use Cisco Umbrella. Using this tool, we have access to Cisco's massive passive DNS database, which is considered one of the largest in the world [37]. Umbrella resolvers analyze over 180 billion DNS requests daily. We examine the DNS query volumes for the malicious landing domains during the last 30 days before the reception of their associated message. We filter the domains by dropping the ones corresponding to compromised websites or some legitimate providers (e.g., cloudflare-ipfs.com) because we believe that their DNS query volume would not be representative of the phishing traffic. We find that the domains associated with only one phishing message have significantly lower DNS query volumes (median maximum query volume per day equals to 18.5 and median total number of DNS queries in the past 30 days equals to 43.0) compared to those associated with multiple messages (median maximum query volume per day equals to 50.5 and median total number of DNS queries in the past 30 days equals to 100.5). These median numbers are relatively low. Nonetheless, we notice that the domain associated with the most reported messages has also by far the highest DNS query volume in the past 30 days (665,126,135), confirming that it is not associated with a targeted campaign. The domains with the second (37,623,107) and third (15,362) highest query volume are associated to only 5 and 1 reported messages, respectively.

Next, we study the syntax of the 522 credential harvesting landing domains. Among these domains, only 15.7% (82/522) include combosquatting, target embedding, homoglyphs, keyword stuffing, or typosquatting. No domain included punycode. In prior work [24], [27], security scanners were shown to be limited by computational constraints, crawling only domains that appeared in the Certificate Transparency logs and employed at least one of the deceptive techniques mentioned above. Since most of the observed malicious landing domains do not use any of these tricks, they might have avoided being scanned by a significant number of security crawlers. This is another indication of the attention paid by malicious actors to understand how the mitigation tools work in order to defeat them.

Key Findings

- More than 29% of phishing pages dynamically load logos and background images from the impersonated organization's domains, providing a critical opportunity for organizations to detect and track these malicious activities early by monitoring referral requests for such resources.
- Usually, domains are registered long before their associated TLS certificate is created and before delivering phishing messages. Some of these domains are compromised, but the majority are created with malicious intent. Domains with the `.ru` TLD are unusually prevalent among phishing attacks
- Median DNS query volumes for phishing domains are remarkably low, signaling a shift toward low-profile, high-impact campaigns. These domains often evade detection due to their lack of common deceptive markers like typosquatting.
- Non-targeted attacks increasingly leverage HTML attachments that execute JavaScript locally to redirect users, bypassing detection tools.

B. Non-targeted attacks

In a subsequent analysis phase, we manually reviewed the remaining 414 (out of 1,551) active phishing messages, excluding the ones that we previously identified as spear phishing (i.e., 1,137 emails). During this process, we identified 130 (out of 414) unique web pages that do not replicate the legitimate login portals of the studied companies. These pages impersonate services commonly used by employees, such as Microsoft Excel (20 messages), OneDrive (12), Office 365 (11), generic fake Microsoft login pages (44), DocuSign (1), and others (42).

We observe that 29 phishing messages include an HTML file as an attachment separate from the main body of the email. In order to view the file's content, the victim must download the attachment locally and open it with their web browser. In such cases, the browser's address bar would display the file's URI (Uniform Resource Identifier). Such HTML file may contain JavaScript designed to update the browser's URL and reload the page to redirect the victim to an external website. In our analysis, we identify 19 HTML files (associated with 19 messages) that are loaded locally without changing the window's URL. External resources (such as the page's background) are retrieved within the browser's frames from legitimate services commonly used for managing and storing multimedia content (for example, `gyazo.com` and `freeimages.com`).

Finally, we examine the URLs of the collected landing pages, totaling 111 unique URLs. They are associated with 111 distinct domains, with only 11 adopting some deceptive techniques mentioned in the previous subsection.

C. Prevalence of evasion techniques

We examine the evasion techniques used by threat actors to avoid detection. Notably, we study the prevalence of the mechanisms discussed in Section III, and we identify new implementations of known categories. First, we check the message-level evasion tactics. Second, we inspect the employed cloaking techniques.

1) Message-based evasion: All the reported messages pass the three email authentication methods: SPF (Sender Policy Framework), DKIM (DomainKeys Identified Mail), and DMARC (Domain-based Message Authentication Reporting and Conformance). This means that they are either sent from legitimate, well established email addresses or from compromised or malicious accounts.

First, we notice in at least 270 cases a lengthy series of line breaks and a long random text after a fraudulent request urging the user to click on a malicious link or to open a malicious attachment. This added noise can dilute the signals of the malicious content. If the noise is sufficiently disruptive, AI-models or signatures might rely on erroneous features or patterns, thus resulting in false negatives – as in this case, in which human intervention was found to be necessary for proper classification.

Then, we have identified a novel approach that exploits a bug in QR code scanners employed by email filters. In 35 messages, we observed the presence of faulty QR codes (whose decoding generates a syntactically incorrect URL). The embedded string contains irrelevant characters preceding the malicious URL. For instance, it could be constructed as follows: ”`xxx https://evil-site.com/`” or ”`[https://evil-site.com/`”, where ”`xxx`” denotes any arbitrary sequence of ASCII characters. We tried both the latest iOS and Android versions and, despite the syntactical irregularities, scanning such a QR code with the default mobile phone camera app prompts the user to visit the malicious webpage. This occurs because the QR code extraction mechanism is able to successfully retrieve the embedded URL while disregarding any faulty characters. We tested the resilience of three leading commercial email security tools—ranked among the top ten email security products in the market [59]—against this attack. As of April 2024, two out of the three tools failed to detect the malicious link, classifying the corresponding messages as benign [3]. The mismatch between how QR codes are handled by the email security parsers and by mobile libraries leaves users exposed to undetected phishing attacks. It is worth emphasizing that the use of QR codes in phishing emails is particularly advantageous for attackers, as it encourages recipients to scan the code using their smartphones. This approach increases the likelihood of the malicious link being accessed through the victim's mobile internet connection, bypassing corporate security and monitoring tools.

2) Cloaking techniques: We rely on the same categorization discussed in Section III to present our findings. During our analysis, we examine the web resources loaded by phishing websites (HTML pages, JavaScript, CSS, images, etc.). By analyzing these resources, we identify the client-side evasion techniques deployed by threat actors, as well as the user data exfiltrated to C2 infrastructure for server-side filtering.

a) Client side. User interaction: By analyzing the collected JS code obtained from crawling embedded URLs, we have observed that, in at least 15 messages, cloaking exists based on the association of the following objects: the user agent string (accessed via `navigator.userAgent`), the timezone (accessed via `Intl.DateTimeFormat().resolvedOptions().timeZone`) and the browser language (retrieved from `navigator.language` or `navigator.userLanguage`).

In 47 other cases, the visitor lands on a webpage where they must enter a One-Time Password (OTP), sent to them in a separate message, in order to access the malicious login site. It is difficult for email security filters to flag phishing URLs involving OTP prompts since this tactic introduces multiple layers of complexity. Because the actual malicious login page is not immediately presented, scanning the URL may not detect any suspicious activity.

Furthermore, we found 11 messages that rely on custom challenge–response authentication. For example, some of them present a simple math equation that users must solve and enter the result in order to gain access to the landing sites. This technique is designed to create an additional layer of interaction that needs custom code to automate the analysis process.

b) Client side. Bot behavior: We notice the frequent usage of proprietary challenge–response authentication. Our observations revealed that phishers employ a variety of techniques to safeguard their activities from automated bots, which are also commonly utilized in legitimate business operations. The two principal free proprietary services used by these actors are Cloudflare Turnstile [47] and Google reCAPTCHA v3 [60]. On one hand, Turnstile was announced by Cloudflare in 2022 as an alternative to CAPTCHA. It is based on a series of JavaScript challenges that examine network-level data, such as IP address reputation, browser characteristics, and device fingerprints, in order to assess the likelihood of a user being a bot. Usually, the visitor also has to click on a box to validate the Turnstile. It is loaded before the landing page of the phishing site. From the perspective of the attacker, this represents a cost-free and sophisticated solution for preventing security crawlers from reaching and detecting the phishing site. Based on our collected data, Cloudflare Turnstile is employed in 943 out of the 1,267 identified phishing messages aimed at harvesting victims' credentials, accounting for 74.4% of the total. This highlights the significant prevalence of Turnstile in phishing campaigns reported by end-users. In comparison, Google reCAPTCHA [60] is associated with 314 of the reported phishing messages, representing 24.8% of the total. Usually, these verification checks are run after the visitor is presented with a Cloudflare Turnstile. This approach ensures at least two layers of protection, both leveraging browser fingerprinting techniques. Google reCaptcha is run in the background following Turnstile, thereby preventing the need for victims to interact with two CAPTCHA-like solutions consecutively. This strategy minimizes the risk of raising suspicion among potential targets.

Moreover, in at least 295 cases, we have seen scripts attempting to hijack the console's logging functions. By redefining them, the console methods are prevented from being used normally, potentially to prevent debugging or to obfuscate the console output. In at least 10 other cases, we have seen instances where a recurring timer is triggered every second. Each time it runs, it records the current time, invokes the debugger statement (which will pause execution if a debugger is open), and then records the time again. This could be used to detect debugging activity.

Interestingly, some phishing pages (associated with 39 messages) prevent the visitors from using the context menu by disabling right-clicking or prevent users from accessing

[3]For legal reasons, we cannot disclose the names of these security products; however, we have engaged with them following the usual responsible disclosure process.

browser developer tools (Inspect Element, JavaScript Console, or View Source) by disabling key combinations.

c) Client side. Fingerprinting: In comparison to the fingerprinting checks presented by Zhang et al. in 2021 [9], we measured a greater number of checks. The authors presented a limited set of browser characteristics, including cookies, the cache, the navigator User-Agent property, and the Referrer field. Concurrently, the techniques employed by attackers have become more sophisticated. Indeed, we have observed the use of advanced fingerprinting techniques based on open-source browser fingerprinting libraries: BotD [46] and FingerprintJS [61]. These libraries are responsible for computing a fingerprint associated to a page visitor and detecting bots. FingerprintJS was discussed previously in the literature [62]–[65] in the context of legitimate browser fingerprinting. It was considered as one of the most prevalent fingerprinting libraries in 2023 [65]. The libraries BotD and FingerprintJS are associated to only 5 malicious reported messages, all of which were received from July 9th to July 18th. It is worth noting that our analysis spans a ten-month period, from the beginning of January to the end of October 2024. The low number and the closeness of the messages' reception times might indicate a punctual use of a specific phishing kit for a targeted campaign.

d) Additional client-side evasion techniques: We observed that in 167 pages (corresponding to 103 distinct messages since we can extract multiple URLs from one message), a subtle visual effect is created on the rendered webpage. A JavaScript code (encoded in base64) is appended to each HTML document's `<head>` section. It runs before the rest of the document is fully parsed and loaded. It applies a color rotation of 4 degrees to the entire document using the CSS filter `hue-rotate(4deg)`. This technique could be used to evade the detection of tools relying on visual similarity checks to identify phishing pages. This trick is not efficient against *CrawlerBox* for two reasons. The first one is because we employ fuzzy hashes (pHash and dHash), which primarily work on grayscale information. The color rotation should not affect the generated hashes as long as the grayscale conversion results in a similar image. The second reason is that we chose a particular threshold under which we consider two images similar. This threshold is tailored to our needs and for detecting only pages impersonating the legitimate login pages of the companies under study.

e) Server side: We notice that phishing websites send AJAX requests including user data, before loading the malicious landing page. The requests are sent to a server controlled by the attacker (either the same hosting the phishing site, or a separate C2 server). The data includes the client's IP address, country, and user agent. We believe that some server-side filtering is deployed based on the collected information. For example, the phishing page might only be accessible to visitors from a targeted country or region, or for clients using mobile devices.

More particularly, we observe the usage of legitimate third-party services to get the client's IP and enrich it with additional data before sending them to a C2 server. Namely, the IP is retrieved by visiting httpbin.org [66] and is enriched with the results obtained from ipapi.co [67]. The collected information includes the city, the country, the ASN, and the network provider associated to the client's IP. Httpbin and ipapi were associated with 145 and 83 malicious messages, respectively.

Finally, looking at the loaded Javascript code, we find an obfuscated script shared between 38 distinct domains and associated to 151 reported messages. It implements a sleep function and hijacks the console's methods. Additionally, it extracts the victim's email address from the tokenized URL, checks if it is valid (using a regular expression), and performs a synchronous AJAX request to the attacker's server. The goal is to check if the victim's email address exists in the attacker's database. This highlights that attackers keep track of their potential victims and only load the phishing website if the corresponding URL targets a specific message recipient. Another similar obfuscated code is shared between 57 other distinct malicious domains, corresponding to 143 other reported messages.

Key Findings

- Some advanced techniques are employed today by threat actors to evade detection. Some of them rely on relatively new services or open-source libraries.
- QR codes are leveraged to embed phishing content. We even discovered a bug in email security filters which is actively exploited to hide malicious links.
- Phishing websites deploy advanced client-side techniques such as browser fingerprinting, OTP prompts, and challenge-response mechanisms, making detection by security tools increasingly difficult.
- A staggering three quarters of credential-harvesting phishing campaigns leverage Cloudflare Turnstile to block bots while appearing legitimate.
- Many phishing campaigns incorporate obfuscated JavaScript to track victims and personalize attacks.

VI. Final Discussion

Our research indicates a shift toward low-profile, high-impact campaigns that are carefully prepared for a significant period of time before the attacks are launched. This observation contrasts with previous findings by Moore et al. [16], which indicated that phishing emails were typically distributed around the time that phishing websites were launched. This shift highlights an evolution in attackers' strategies, likely in response to modern defenses. Among the evasion techniques observed, we note the emergence of relatively new trends, such as the carefully altered use of QR codes to evade email filters and to bypass corporate security by leveraging employees' personal devices. In addition, attackers are extensively using stealth techniques – such as browser fingerprinting and advanced bot detection methods – to protect their phishing infrastructure. Previous studies [9], [26], [30], [68], [69] present cloaking as an evasion technique. However, our research uncovers new implementations of known cloaking categories (such as the abuse of modern browser fingerprinting libraries and victim tracking). Determining the specific technique solely responsible for evading detection remains challenging, as we lack access to the backend of proprietary automated anti-phishing tools. However, our findings suggest that the combined use of the identified evasion techniques makes the analyzed phishing messages particularly stealthy and difficult to detect.

These observations highlight the urgent need for more adaptive defenses. For instance, we suggest building web security scanners that exhibit a fingerprint consistent with that of browsers operated by humans. It is also important to render the messages' contents in order to detect QR codes and possible hidden texts. However, we acknowledge that conventional scanning and sandboxing techniques may remain insufficient in the face of advanced adversarial methods, particularly those leveraging meticulous filtering and dynamically tailored payloads. Scalability and cost constraints may also pose significant obstacles to implementing defenses. Nonetheless, email filters can be enhanced through the integration of behavior-based and context-aware analysis to improve the detection of subtle, evasive threats – for instance, by identifying abnormal message contents or links that alter behavior based on the visitor's IP address or browser fingerprint.

Finally, we emphasize that the findings presented in this study are generalizable and transferable to other institutions. Given the high degree of source code reuse among phishing kits, as highlighted by Merlo et al. [22], and the proliferation of Phishing-as-a-Service platforms, it is likely that the strategies and evasion techniques identified in our study are also employed to target other organizations. In this context, we provide *CrawlerBox* as an open-source analysis infrastructure to facilitate the reproducibility of this measurement study by other organizations. It represents a comprehensive analysis pipeline for parsing phishing messages, crawling web resources, and logging the corresponding artifacts. For future work, we would like to enrich *CrawlerBox* with additional crawler components.

VII. Limitations.

Dataset. We acknowledge that our dataset might be subject to biases. First, the dataset consists solely of user-reported messages, meaning it excludes malicious emails that users fail to report. In reality, employees might be more likely to report certain types of threats while ignoring others, potentially leading to an unbalanced dataset. Additionally, observer bias may arise during the expert validation process, as security analysts may unconsciously apply subjective criteria when curating the dataset. Despite these limitations and as stated above, our objective is not to develop a phishing detection system. Instead, our primary aim is to uncover and characterize some evasion techniques employed by attackers in real-world scenarios. By leveraging this dataset, we identify and study such techniques.

Crawlerbox. It was tested within a consistent environment, including identical hardware and network conditions. However, the findings should be interpreted with caution, as factors beyond the crawler's implementation may affect bot detection outcomes. For example, a visitor's historical behavior and the reputation of the associated IP address are critical factors that can significantly influence detection verdicts. In order to reproduce these results, it is essential to have a neutral testing environment. Finally, our crawler *NotABot* is not able to solve traditional CAPTCHAs (such as Google reCaptcha v2). Nevertheless, prior research [68] has addressed this challenge by proposing an AI-powered system designed to emulate human behavior for CAPTCHA solving. Integrating such a system with modern security crawlers could enhance their capabilities, enabling them to access and analyze uncloaked phishing content more effectively.

VIII. Conclusion

This paper presents a comprehensive analysis of evasion techniques used by threat actors to bypass anti-phishing detection mechanisms. Drawing on user-reported phishing messages provided by a multinational technology organization responsible for email security across four additional companies, the study uncovers active strategies employed by attackers in real-world scenarios. Our findings emphasize the increasing sophistication and adaptability of phishing tactics, reflecting the persistent evolution of threats in response to advancing detection technologies. By detailing these evasion techniques, this research contributes to a deeper understanding of the contemporary phishing landscape, highlighting the urgent need for continuous innovation in detection and mitigation strategies.

Responsible Disclosure and Ethical Considerations. In accordance with responsible disclosure practices, all flaws identified during the course of our research were promptly reported to the affected parties prior to the submission of this paper. Our findings were acknowledged by the affected vendors. Specifically, the two email security products tested are now able to extract links from faulty QR codes built as presented in our study. Additionally, our crawler's implementation (*NotABot*) was awarded a bug bounty by Cloudflare for bypassing its Turnstile. Finally, due to legal agreements between the company that provided us with the data and the company that developed "AnonWAF", we cannot disclose its real name, but they have acknowledged the problem and are currently integrating a fix into the product. Finally, it is worth noting that the methods adopted in this research are consistent with our institution's ethical guidelines and the data collection and storage are compliant with the law in which our institution and the companies reside.

Dataset sharing. Unfortunately, we cannot share our dataset due to the inclusion of sensitive information, such as employee email addresses and message contents. Personally identifiable information may be present in obfuscated or encrypted formats and can be located in various fields, such as in the phishing URLs' fragments, in the loaded web resources, and in other related data. Ensuring the thorough anonymization of this information without compromising its integrity would require substantial effort and cannot be guaranteed to meet the necessary standards for privacy and security.

Acknowledgements

This work benefited from two government grants managed by the French National Research Agency with references: “ANR-22-PECY-0007” and “ANR-23-IAS4-0001”.

References

[1] R. Zieni, L. Massari, and M. C. Calzarossa, “Phishing or not phishing? a survey on the detection of phishing websites,” *IEEE Access*, vol. 11, pp. 18 499–18 519, 2023.

[2] “Crawlerbox,” https://github.com/AmadeusITGroup/CrawlerBox, 2025.

[3] S. Chhabra, A. Aggarwal, F. Benevenuto, and P. Kumaraguru, “Phi.sh/$ocial: the phishing landscape through short urls,” in *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference*, 2011, pp. 92–101.

[4] K. Subramani, W. Melicher, O. Starov, P. Vadrevu, and R. Perdisci, “Phishinpatterns: measuring elicited user interactions at scale on phishing websites,” in *Proceedings of the 22nd ACM Internet Measurement Conference*, 2022, pp. 589–604.

[5] H. Bijmans, T. Booij, A. Schwedersky, A. Nedgabat, and R. van Wegberg, “Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3757–3774.

[6] A. Oest, Y. Safaei, P. Zhang, B. Wardman, K. Tyers, Y. Shoshitaishvili, and A. Doupé, “{PhishTime}: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists,” in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 379–396.

[7] S. Sheng, B. Wardman, G. Warner, L. Cranor, J. Hong, and C. Zhang, “An empirical analysis of phishing blacklists,” 2009.

[8] W. Lee, J. Hur, and D. Kim, “Beneath the phishing scripts: A script-level analysis of phishing kits and their impact on real-world phishing websites,” in *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security*, 2024, pp. 856–872.

[9] P. Zhang, A. Oest, H. Cho, Z. Sun, R. Johnson, B. Wardman, S. Sarker, A. Kapravelos, T. Bao, R. Wang *et al.*, “Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing,” in *2021 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2021, pp. 1109–1124.

[10] S. Bell and P. Komisarczuk, “An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank,” in *Proceedings of the Australasian Computer Science Week Multiconference*, 2020, pp. 1–11.

[11] A. Oest, Y. Safaei, A. Doupé, G.-J. Ahn, B. Wardman, and K. Tyers, “Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists,” in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1344–1361.

[12] Y. Choi, W. Lee, and J. Hur, “Phishinwebview: Analysis of anti-phishing entities in mobile apps with webview targeted phishing,” in *Proceedings of the ACM on Web Conference 2024*, 2024, pp. 1923–1932.

[13] A. Bergholz, G. Paass, F. Reichartz, S. Strobel, M.-F. Moens, and B. Witten, “Detecting known and new salting tricks in unwanted emails.” in *CEAS*, vol. 9, 2008.

[14] T. Moore and R. Clayton, “How hard can it be to measure phishing?” *Mapping and Measuring Cybercrime*, 2010.

[15] A. Bergholz, J. H. Chang, G. Paass, F. Reichartz, and S. Strobel, “Improved phishing detection using model-based features.” in *CEAS*, 2008.

[16] T. Moore, R. Clayton, and H. Stern, “Temporal correlations between spam and phishing websites.” in *LEET*, 2009.

[17] A. Oest, P. Zhang, B. Wardman, E. Nunes, J. Burgis, A. Zand, K. Thomas, A. Doupé, and G.-J. Ahn, “Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale,” in *29th {USENIX} Security Symposium ({USENIX} Security 20)*, 2020.

[18] G. C. Moura, M. Korczyński, M. Müller, and S. Castro, “Characterizing and mitigating phishing attacks at cctld scale,” in *OARC 43*, 2024.

[19] A. Oest, Y. Safei, A. Doupé, G.-J. Ahn, B. Wardman, and G. Warner, “Inside a phisher’s mind: Understanding the anti-phishing ecosystem through phishing kit analysis,” in *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2018, pp. 1–12.

[20] M. Cova, C. Kruegel, and G. Vigna, “There is no free phish: An analysis of” free” and live phishing kits.” *WOOT*, vol. 8, pp. 1–8, 2008.

[21] B. Tejaswi, N. Samarasinghe, S. Pourali, M. Mannan, and A. Youssef, “Leaky kits: the increased risk of data exposure from phishing kits,” in *2022 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 2022, pp. 1–13.

[22] E. Merlo, M. Margier, G.-V. Jourdan, and I.-V. Onut, “Phishing kits source code similarity distribution: A case study,” in *2022 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*. IEEE, 2022, pp. 983–994.

[23] X. Han, N. Kheir, and D. Balzarotti, “Phisheye: Live monitoring of sandboxed phishing kits,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 1402–1413.

[24] B. Kondracki, B. A. Azad, O. Starov, and N. Nikiforakis, “Catching transparent phish: Analyzing and detecting mitm phishing toolkits,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 36–50.

[25] S. Maroofi, M. Korczyński, and A. Duda, “Are you human? resilience of phishing detection to evasion techniques based on human verification,” in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 78–86.

[26] L. Invernizzi, K. Thomas, A. Kapravelos, O. Comanescu, J.-M. Picod, and E. Bursztein, “Cloak of visibility: Detecting when machines browse a different web,” in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 743–758.

[27] K. Tian, S. T. Jan, H. Hu, D. Yao, and G. Wang, “Needle in a haystack: Tracking down elite phishing domains in the wild,” in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 429–442.

[28] B. Acharya and P. Vadrevu, “{PhishPrint}: Evading phishing detection crawlers by prior profiling,” in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3775–3792.

[29] “Smtp smuggling,” https://smtpsmuggling.com/, 2024, [Accessed 13-09-2024].

[30] W. Li, S. Manickam, S. U. A. Laghari, and Y.-W. Chong, “Uncovering the cloak: A systematic review of techniques used to conceal phishing websites,” *IEEE Access*, 2023.

[31] N. Potti, J. B. Wendt, Q. Zhao, S. Tata, and M. Najork, “Hidden in plain sight: Classifying emails using embedded image contents,” in *Proceedings of the 2018 World Wide Web Conference*, 2018, pp. 1865–1874.

[32] Checkpoint, “What is quishing (qr phishing)?” https://www.checkpoint.com/cyber-hub/threat-prevention/what-is-phishing/what-is-quishing-qr-phishing/, 2024, [Accessed 03-03-2024].

[33] C. Wang, D. Zhang, S. Huang, X. Li, and L. Ding, “Crafting adversarial email content against machine learning based spam email detection,” in *Proceedings of the 2021 International Symposium on Advanced Security on Software and Systems*, 2021, pp. 23–28.

[34] L. Ke, B. Li, and Y. Vorobeychik, “Behavioral experiments in email filter evasion,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 30, no. 1, 2016.

[35] “Content-type header field,” https://datatracker.ietf.org/doc/html/rfc2045#section-5.

[36] “Shodan: Search engine for the internet of everything,” https://www.shodan.io/, 2024.

[37] “Cisco umbrella,” https://umbrella.cisco.com/blog/new-passive-dns-enhancements-for-cisco-umbrella-investigate, 2024.

[38] “Puppeteer,” https://pptr.dev/, 2024, [Accessed 19-08-2024].

[39] C. Han, Q. Niyaz, and A. Javaid, “Browser security comparison using cookie analysis,” in *2024 IEEE International Conference on Electro Information Technology (eIT)*. IEEE, 2024, pp. 256–261.

[40] G. Ho, D. Boneh, L. Ballard, and N. Provos, “Tick tock: building browser red pills from timing side channels,” in *8th USENIX Workshop on Offensive Technologies (WOOT 14)*, 2014.

[41] A. Vastel, W. Rudametkin, R. Rouvoy, and X. Blanc, “Fp-crawlers: studying the resilience of browser fingerprinting to block crawlers,” in *MADWeb’20-NDSS Workshop on Measurements, Attacks, and Defenses for the Web*, 2020.

[42] H. Jonker, B. Krumnow, and G. Vlot, “Fingerprint surface-based detection of web bot detectors,” in *Computer Security–ESORICS 2019: 24th European Symposium on Research in Computer Security, Luxembourg, September 23–27, 2019, Proceedings, Part II 24*. Springer, 2019, pp. 586–605.

[43] J. Jueckstock and A. Kapravelos, “Visiblev8: In-browser monitoring of javascript in the wild,” in *Proceedings of the Internet Measurement Conference*, 2019, pp. 393–405.

[44] “Navigator: webdriver property,” https://developer.mozilla.org/en-US/docs/Web/API/Navigator/webdriver, 2024.

[45] “Puppeteer: Page.setrequestinterception() method,” https://pptr.dev/api/puppeteer.page.setrequestinterception, 2024.

[46] “Botd,” https://github.com/fingerprintjs/BotD, 2024.

[47] "Cloudflare turnstile," https://developers.cloudflare.com/turnstile/, 2024.

[48] "Kangooroo," https://github.com/CybercentreCanada/kangooroo.

[49] "Assembyline," https://github.com/CybercentreCanada/assemblyline, 2024.

[50] "Lacus," https://github.com/ail-project/lacus, 2024.

[51] "Ail framework," https://github.com/ail-project/ail-framework, 2024.

[52] "puppeteer-extra-plugin-stealth," https://github.com/berstend/puppeteer-extra/tree/master/packages/puppeteer-extra-plugin-stealth, 2024.

[53] "Selenium-stealth," https://github.com/diprajpatra/selenium-stealth, 2020.

[54] "undetected-chromedriver," https://github.com/ultrafunkamsterdam/undetected-chromedriver, 2024.

[55] "Nodriver," https://github.com/ultrafunkamsterdam/nodriver, 2024.

[56] "Selenium-driverless," https://github.com/kaliiiiiiiiii/Selenium-Driverless, 2024.

[57] H. L. Liu, K. Levchenko, M. Félegyházi, C. Kreibich, G. Maier, and G. M. Voelker, "On the effects of registrar-level intervention," in *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET 11)*, 2011.

[58] A. R. Omar, S. Taie, and M. E. Shaheen, "From phishing behavior analysis and feature selection to enhance prediction rate in phishing detection," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 5, 2023.

[59] "Email security reviews and ratings," https://www.gartner.com/reviews/market/email-security, 2024, [Accessed 13-09-2024].

[60] "Google recaptcha," https://www.google.com/recaptcha/, 2024.

[61] "fingerprintjs," https://github.com/fingerprintjs/fingerprintjs, 2024.

[62] A. Sjösten, D. Hedin, and A. Sabelfeld, "Essentialfp: Exposing the essence of browser fingerprinting," in *2021 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, 2021, pp. 32–48.

[63] L. Hui, H. Xudong, G. Fan, W. KaiLun, and Y. Enze, "Web service access control based on browser fingerprint detection," *Journal of Web Engineering*, vol. 20, no. 5, pp. 1587–1622, 2021.

[64] P. Baumann, S. Katzenbeisser, M. Stopczynski, and E. Tews, "Disguised chromium browser: Robust browser, flash and canvas fingerprinting protection," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, 2016, pp. 37–46.

[65] T. Stephenson, "A comparative study on analyses of browser fingerprinting," Ph.D. dissertation, Wesleyan University, 2023.

[66] "Httpbin.org," https://httpbin.org/ip, 2024.

[67] "ipapi," https://ipapi.co, 2024.

[68] X. Teoh, Y. Lin, R. Liu, Z. Huang, and J. S. Dong, "{PhishDecloaker}: Detecting {CAPTCHA-cloaked} phishing websites via hybrid vision-based interactive models," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 505–522.

[69] P. Zhang, Z. Sun, S. Kyung, H. W. Behrens, Z. L. Basque, H. Cho, A. Oest, R. Wang, T. Bao, Y. Shoshitaishvili *et al.*, "I'm spartacus, no, i'm spartacus: Proactively protecting users from phishing by intentionally triggering cloaking behavior," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 3165–3179.