Deep Dive into Client-Side Anti-Phishing: A Longitudinal Study Bridging Academia and Industry

Rana Pourmohamad
Arizona State University
rpourmoh@asu.edu

Steven Wirsz
Arizona State University
swirsz@asu.edu

Adam Oest
Arizona State University
aoest@asu.edu

Tiffany Bao
Arizona State University
tbao@asu.edu

Yan Shoshitaishvili
Arizona State University
yan@asu.edu

Ruoyu Wang
Arizona State University
fishw@asu.edu

Adam Doupé
Arizona State University
doupe@asu.edu

Rida A. Bazzi
Arizona State University
bazzi@asu.edu

ABSTRACT

Client-side anti-phishing methods are crucial for safeguarding individuals against phishing attacks, offering a proactive approach beyond traditional blocklisting strategies. This study expands the scope to include a comprehensive evaluation of client-side anti-phishing techniques within the Chrome browser, alongside an in-depth analysis of academic research in the field of phishing over the past five years. Our findings highlight the inherent limitations of current client-side anti-phishing measures, which demonstrated a detection rate of only 14% for phishing websites and blocked merely 10% of login-based phishing sites within the first hour, resulting in a substantial false negative rate. Additionally, our analysis reveals that attackers can readily circumvent these defenses by altering the content of phishing websites. The study also critically assesses recent academic contributions to understand their alignment and potential integration with client-side anti-phishing frameworks. Based on these insights, we propose targeted recommendations to enhance the efficacy and responsiveness of the client-side anti-phishing ecosystem, addressing the challenges of low detection coverage, slow response times, and high rates of false negatives.

CCS CONCEPTS

• **Security and privacy → Web application security**; **Phishing**.

KEYWORDS

Client-side Anti-Phishing, Google SafeBrowsing, Blocklist

ACM Reference Format:
Rana Pourmohamad, Steven Wirsz, Adam Oest, Tiffany Bao, Yan Shoshitaishvili, Ruoyu Wang, Adam Doupé, and Rida A. Bazzi. 2024. Deep Dive into Client-Side Anti-Phishing: A Longitudinal Study Bridging Academia and Industry. In *Proceedings of ACM Asia Conference on Computer and Communications Security (ASIA CCS '24).* ACM, New York, NY, USA, 17 pages. https://doi.org/10.1145/3634737.3657027

ASIA CCS '24, July 1–5, 2024, Singapore, Singapore

ACM ISBN 979-8-4007-0482-6/24/07
https://doi.org/10.1145/3634737.3657027

1 INTRODUCTION

Phishing remains a significant threat, with a record number of attacks reported in the third quarter of 2022 [6]. This rise in phishing incidents highlights the ongoing challenge posed by sophisticated phishing websites, kits, and evasion techniques like cloaking [25, 32, 54, 63]. These attacks not only result in direct financial losses for victims but also cause reputational harm to impersonated entities and threaten business infrastructures [75].

To combat phishing, the industry uses server-side blocklists, which are highly accurate in identifying phishing sites with few false positives. However, their effectiveness is limited by update latency, allowing 75% of victims to access phishing pages before they're blocked [24, 55]. Additionally, blocklists are less effective against targeted attacks like spear phishing [11]. The delay in updating blocklists creates a vulnerability window, underscoring the need for more timely solutions.

Client-side anti-phishing complements server-side blocklists by detecting phishing directly in the browser, identifying suspicious elements and patterns using machine learning [45]. This approach effectively counters crawler-evasive tactics, delays in blocklist updates, and targeted attacks like spear-phishing. Acting as a last line of defense, client-side detection not only protects users but also informs server-side blocklists about detected threats.

While blocklists are widely studied in the literature on anti-phishing defenses [8, 24, 51, 52, 55, 56, 64, 69, 70, 76], less research has focused on client-side anti-phishing. This work aims to (1) understand the development and effectiveness of client-side anti-phishing as a defense, and (2) examine its role in the broader anti-phishing ecosystem. Additionally, it seeks to (3) comprehend the constraints shaping client-side anti-phishing compared to server-side defenses, and (4) use this insight to evaluate the applicability of academically proposed anti-phishing techniques for client-side detection.

In our work, we (1) conducted a three-year longitudinal study on Chrome's client-side anti-phishing, analyzing its evolution, effectiveness, and impact on server-side blocklisting. We also (2) investigated how client-detected phishing data in Chrome is shared with servers. Furthermore, we (3) identified design constraints in Chrome's client-side anti-phishing and (4) used these insights to evaluate the suitability of recent academic phishing defense strategies for client-side detection, focusing on research from the past five years in major security conferences.

We focused this study on Chrome because it has the largest user base globally, with approximately three billion active users. Additionally, results from Chrome's client-side anti-phishing feeds into the Google Safe Browsing blocklist that protects most mainstream browsers, not only Chrome. Browsers that use Google Safe Browsing blocklist include, in addition to Chrome, Firefox, Safari, and Opera [67].

To understand Chrome's client-side anti-phishing system, we conducted a detailed manual analysis of its source code across versions 79(2019) to 108(2023) over three years. This included code review and reverse engineering to comprehend its design and functionality. Our analysis revealed the evolution of Chrome's anti-phishing features, including the introduction of *Enhanced Protection* mode, visual features, and an updated detection model that operates independently of Chrome's core updates.

Classifier Evaluation. We evaluated Chrome's client-side anti-phishing classifier using real-world phishing samples from PhishTank [15] over three years. As shown in Section 4) this assessment revealed that the classifier had a high rate of failure and false negatives in detecting known phishing attacks.

Evaluating Client-side Classification Impact on Server-side Blocklisting. We conducted a long-term experiment using the PhishFarm framework [51] to assess the impact of client-side anti-phishing on blocklist performance in Chrome. This involved testing 14 groups of 35 phishing websites over seven days. Our May 2020 experiment showed that client-side anti-phishing had little effect on blocklisting, with websites being blocklisted mainly when reported to Google Safe Browsing. A follow-up experiment in January 2021 confirmed these findings.

Disclosure. In February 2021, we reported our findings about the effectiveness of the anti-phishing features in Google Chrome to the Chrome team. They agreed with our findings and said that they were working on improving client-side anti-phishing detection.

We conducted experiments in May and July 2022, and April 2023 to assess server-side blocklisting's impact. These experiments showed that Chrome's anti-phishing system had adopted a visual classification model, enhancing detection but still struggling with high failure rates and false negatives on PhishTank websites. By April 2023, there was notable improvement: non-cloaked phishing sites had a 10% blocklisting rate within the first hour and 51% after seven days. Although, cloaked sites using evasion techniques remained largely unblocked.

We discovered that phishing websites without a login form were consistently not blocked by client-side detection. This was confirmed by a specific experiment focusing solely on such non-login phishing sites. This suggests a potential overreliance of client-side detection on login forms, either in visual or page features, indicating a need for broader detection criteria.

Design Constraints and the Academic Work. Our study identified the main constraints that dictate some of the design decisions of client-side anti-phishing. The identified constraints are: (1) locality of detection, (2) solution simplicity, (3) efficiency (4) generality. These constraints make it possible for an attacker to *completely bypass the client-side detection system*! For example, we verified that attackers could bypass detection by adding an excessive amount of DOM elements to a phishing page. Such bypasses are real threats and have been used by real-world attackers—in one experiment, 34% of PhishTime websites (i.e., known and already detected phishing websites) bypassed detection from the classifier.

In our review of 25 academic papers on phishing from major security conferences, we found that most of the proposed defenses don't adequately address client-side anti-phishing needs. They either fail to meet the specific design constraints of client-side systems or are not suitable for automatic phishing detection.

Our results paint a dire picture of the state of client-side anti-phishing detection as currently deployed in practice and indicate the need for more academic research on the topic. Our study is a first step in this direction by providing an understanding and an analysis of solution requirements for client side anti-phishing.

Rather than be disheartened by these results, the community should take these findings as a challenge to design improved and more robust client-side detection frameworks and mechanisms, especially solutions that extend beyond the current state of affairs in which it seems that the client-side detection is over fitting on a single features: phishing websites with a login form.

In summary, in this paper we make the following contributions:

- We evaluated the evolution and effectiveness of Chrome's client-side anti-phishing classification model and the impact of client-side anti-phishing in Chrome on server-side blocklists over a period of three years. This is the first such effort of this magnitude.
- We identified some of the constraints that seem to be guiding the design of client side anti-phishing as deployed in Chrome.
- We studied the academic literature in major security conference over the last 5 years to assess the suitability of current academic work for client-side anti-phishing. Our indicate that current efforts are in general not suitable for client-side anti-phishing and more work is needed.

The rest of this paper is organized as follows. Section 2 gives an overview of phishing and the various strategies employed to combat it. Section 3 studies the design and evolution of client-side anti-phishing in Chrome. Section 4 presents an evaluation of the performance and accuracy of client-side anti-phishing in Chrome. Section 5 discusses the impact of client-side anti-phishing on the anti-phishing ecosystem. Sections 3, 4 and 5 identify constraints on client-side anti-phishing. Section 6 presents an analysis of academic works for their suitability for client-side anti-phishing. Section 7 discusses the work limitations and our disclosure of our findings. Section 8 covers related works and Section 9 concludes the paper.

2 BACKGROUND

Phishing involves attackers posing as trusted entities to extract sensitive information. Defense strategies include blocklists (precise but weak against new threats) and machine learning (capable of identifying new attacks but prone to false results). Browsers like Chrome and Edge combine these methods to balance accuracy and adaptability [4, 5, 10, 18, 21, 27, 28, 48, 49, 51, 66].

Phishing Attacks. Phishing attacks proceed in three major stages: In the first stage, attackers create a fake website that masquerades as a trusted website. Phishers with little technical knowledge can deploy these fake websites without technical knowledge by using phishing kits (a set of software tools) [47].

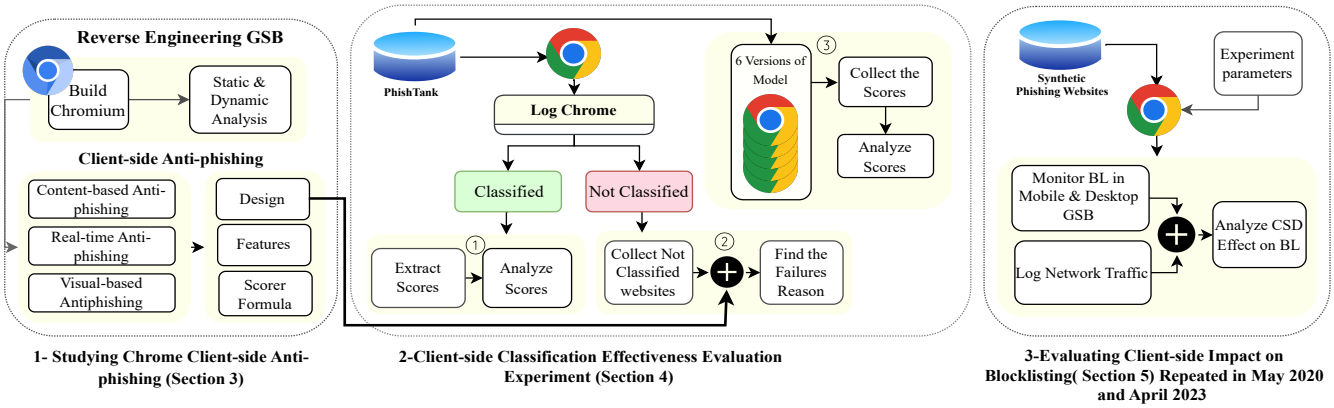

Figure 1: Evaluation Framework for Chrome's Client-Side Anti-Phishing Measures: Part 1 entails dynamic and static analysis of Chromium source code; Part 2 involves the examination of Chrome debug logs, including ① analyzing scores to determine the percentage of real phishing websites scored above the threshold, ② identifying the reasons for client-side failures, and ③ extracting scores of client-side anti-phishing on 100 real phishing websites across six versions of Chrome from September 2019 to April 2023; Part 3 investigates the influence of client-side anti-phishing on blocklisting performance, with investigations initially conducted in May 2020 and revisited in April 2023.

In the second stage, attackers send messages to users to convince them to click on the phishing website's link and then use some aspect of social engineering to deceive users to take action [16, 69]: to provide the phishing websites with sensitive information. In the third stage, phishers extract the victim's information from the phishing website and monetize that information [23].

Blocklists. Blocklists are the first layer of defense in browsers against phishing attacks and they are enabled by default in all major web browsers (Chrome, Firefox, Opera, Edge, and Safari), both on desktop and mobile platforms. When users visit a malicious website, blocklists are used to identify the malicious site and the browser displays prominent warnings in place of the malicious content. Blocklists were the first large-scale anti-phishing defense deployed in browsers. Blocklists are updated server-side and propagated to browsers. Researchers demonstrated that blocklists have a significant effect on stopping phishing campaign [64]. However, blocklists have noticeable shortcomings [55].

Although Google incorporated a real-time anti-phishing system for detecting phishing URLs and improved the speed of Google Safe Browsing (GSB) from the prior delay of "up to 30 minutes," this feature is not enabled by default, and it suffers from delay of caching and updating the URL blocklist database, which provides attackers a window of opportunity to launch their phishing campaign [57]. This change also applies to the mobile version of Chrome [57]. Due to the short lifetime of phishing websites, a few hours of latency can affect the blocklist performance in a significant way, and relying on blocklisting can increase the risk to users [55, 64].

ML-based Anti-phishing. Machine Learning (ML)-based anti-phishing solutions extract features of the web page or domain and feed these features into a trained model to output a *phishy-ness* score. However, studies have shown that ML-based classifiers are susceptible to evasion attacks [3, 5, 7, 13, 30, 38, 40, 62]. Evasion attacks are effective against practical ML-based classifiers, and, at the same time, these attacks can be successfully launched without changing the web page functionality and appearance [40]. Most major browsers, including Chrome, Firefox, and Edge, implement ML-based anti-phishing in the browser (client-side) [49, 74].

3 CHROME CLIENT-SIDE ANTI-PHISHING

We first aim to study the structure, design, and components of Chrome's client-side anti-phishing. To our knowledge, no prior research has attempted to reverse engineer and document the entire ecosystem. While previous studies have focused on analyzing only the classifier component [41], we include all client-side anti-phishing features, as well as the pre- and post-classification stages and other anti-phishing components of the Chrome browser.

Figure 2 shows the Chrome anti-phishing ecosystem. In ①, the client-side classifier periodically downloads a visual-based model and a content-based model, along with corresponding thresholds. When the user visits a website, ② features for the two models are extracted. After feeding the features to the model, the ③ scores are compared against the downloaded threshold, and if any score is over the threshold the browser will display a warning to the user. At the same time, the browser will report the URL, along with the features (and other information) to Google Safe Browsing server-side component. From there, server-side classification occurs (we do not have visibility on how this is done), then, if it is determined to be a phishing URL, then ④ the URL is added to the Google Safe Browsing blocklist. Finally, ⑤ this blocklist is propagated to other end-users browsers, thus protected them from the phishing URL.

Along with this high-level understanding, we performed a static and dynamic analysis of Chromium's implementation to extract Chrome's built-in classifier. Over our observation period of three years we identified significant changes to Chrome's client-side anti-phishing, which have yet to be studied or evaluated in prior work [41]. These changes include (1) new anti-phishing components, (2) new features, (3) changes to overarching algorithms used by the Chrome client-side anti-phishing ecosystem, (4) changes to

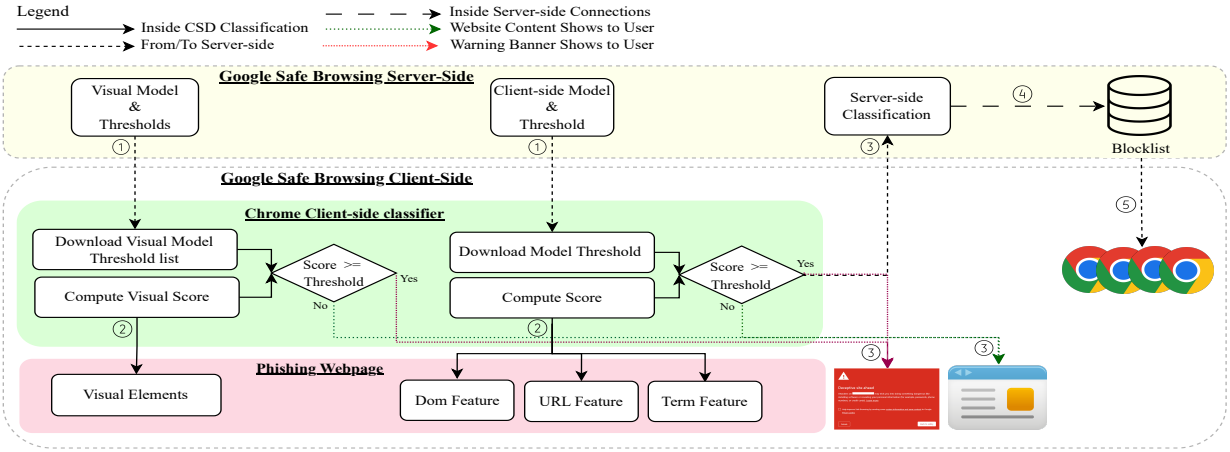

Figure 2: The Chrome anti-phishing ecosystem is composed of client-side and server-side components. First ①, the Chrome browser periodically downloads two client-side anti-phishing machine learning models trained on the server side (a visual-based model and a content-based model). ② When a user visits a webpage, the browser then extracts features, including the DOM, URL, and term features for the content-based model, as well as visual elements from the webpage. The browser calculates the final scores for these two models and compares them with the provided thresholds. ③ If the scores surpass the any threshold, Chrome displays a warning page, and notifies the server-side component. ④ If server-side classification confirms that the URL is a phishing website, the blocklist is updated, and ⑤ this updated blocklist is then propagated to other users' browsers.

the client-side model, and (5) the classification threshold. In addition, we found that Chrome provides three different layers of security: (1) no protection, (2) standard protection (the default setting), and (3) enhanced protection, as shown in Table 1. Figure 3 provides a detailed depiction of the current state of Chrome's client-side anti-phishing, where new changes are highlighted in yellow, and shows the protection level, security policy, and mechanism behind each of the layers. Figure 4 also presents a comprehensive timeline of Chrome's evolving anti-phishing technologies, capturing key updates in 2008, 2012, 2019, 2020, and 2023.

3.1 Client-side Anti-phishing Classification

In our study, we embarked on a detailed examination of Chrome's client-side anti-phishing classification, acknowledging the significant challenge posed by the Chromium project's extensive size, encompassing over 10 million lines of code. Our methodical approach included:

1. Compiling the Extensive Chromium Codebase: We compiled the expansive Chromium source code on our systems, a critical step in understanding the comprehensive structure and mechanisms of such a large-scale browser project.

2. Static Analysis of the Chromium Codebase: We methodically analyzed the source code statically, which allowed us to identify potential weaknesses and improvement areas within this vast codebase without executing the code.

3. Dynamic Analysis and Observation: Through dynamic analysis, we observed the Chromium browser's real-time behavior in response to various phishing and legitimate websites.

4. Classifier Reverse-Engineering in a Complex Code Environment: We undertook reverse-engineering of Chrome's anti-phishing classifier, a significant task given the extensive nature of the Chromium code.

5. Log File and Network Traffic Analysis: Our study involved analyzing Chrome's log files and network traffic to gain insights into the operational dynamics of the anti-phishing classifier within this large software environment.

6.In-depth Examination of Chromium's Source Code: We conducted a thorough examination of the Chromium source code, particularly focusing on the Google Safe Browsing components.

7.Verification and Comparative Analysis: We verified the client-side anti-phishing functionality in both Chrome and Chromium and conducted a comparative analysis of the browser's response to a range of websites.

8. Identification of a Dual-Classification System: Our investigation revealed that Chrome's anti-phishing system utilizes two classifiers: a content-based classifier and a visual-based classifier, within this extensive code environment.

9.Debugging the Extensive Chromium Code: Debugging the Chromium source code for different website types was a key part of our research, presenting unique challenges due to the project's scale.

The enormity of the Chromium project added a layer of complexity to our analysis but also provided a unique opportunity to explore the efficacy of Chrome's anti-phishing strategies in a real-world, large-scale software setting. This study not only illuminates the intricate technical aspects of Chrome's anti-phishing strategies but also emphasizes the extensive efforts required to analyze such a large and complex codebase for enhancing browser security.

3.1.1 Content-based Anti-phishing Classifier. The content-based anti-phishing classifier is a crucial component of GSB's client-side anti-phishing classification system. This classifier employs a machine learning model, likely trained server-side and then deployed on the client, to analyze local browser-based features of websites, encompassing URL, DOM elements, and term-based attributes.

Once the features are extracted, the model calculates an anti-phishing score, representing the probability that a given website is a phishing website. The content-based classifier uses a threshold from the server and compares the score with the threshold. If the score is higher than the threshold, client-side detection shows a warning page to the user and sends information about the URL and the features to the Google Safe Browsing server-side. Table 7 in Appendix A provides a complete list of the features that the classifier extracts and uses to calculate the phishing score. Our study of these features show that they are all local to the client side which means that they are locally available and the client doesn't need any remote resources to calculate them. This lead to the identification of the first constraint that seems to guide the design of client-side anti-phishing detection:

C1 - Locality: features used in phishing detection should be local to the client.

3.1.2 Visual-based Anti-phishing Classifier. GSB's client-side anti-phishing classification system added a new and important component: the visual-based anti-phishing classifier. This classifier focuses on visual elements on the website rather than content-based features, providing additional protection against phishing attempts.

This classifier operates similarly to the content-based classifier: using a downloaded visual-based detection, which includes a list of visual targets and thresholds for each target. The classifier then extracts visual elements from the page and calculates a score for each one, comparing them to their respective thresholds. If at least one score is higher than its threshold, the page is detected as phishing. Visual classification or visual matchingin the client-side phishing classifier, involves comparing visual elements of a webpage against known phishing indicators or patterns. Here's a basic overview of how it works:

1. Image Processing: The software captures images or visual elements from a webpage.

2. Feature Extraction: Key features from these images (like color histograms, shapes, text, or even more abstract features) are extracted. This can include blurring the image to focus on general shapes and colors rather than details.

3. Pattern Matching: The extracted features are then compared to known patterns associated with phishing websites. These patterns are derived from previously identified phishing attacks and can include specific layouts, color schemes, or other visual cues.

4. Hashing and Comparison: Sometimes, images are hashed to create a unique identifier, which can be compared against a database of hashes from known phishing sites.

5. Decision Making: If the visual elements match closely enough with known phishing indicators, the software may flag the page as potentially malicious, triggering a warning or further analysis.

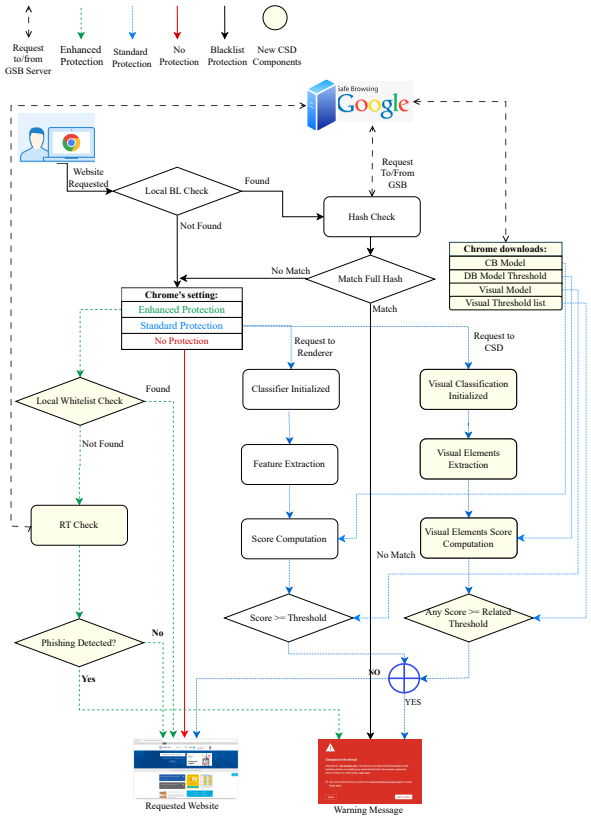

Figure 3: Chrome Client-side anti-phishing design. This diagram presents Chrome's client-side anti-phishing in different layers of security (described in Table 1). The Blue boxes are the new components of GSB that are not studied in the previous works. The purple arrow shows the connection with the GSB server, the red arrow displays the "No protection layer's" mechanism, the Blue arrow pictures the "Standard protection," which is the default setting for the Chrome browser, and the green arrow presents the mechanism of the third security level, "Enhanced protection."

3.1.3 Client-side Anti-phishing Classification Algorithm. The anti-phishing analysis process includes three phases: pre-classification, main classification, and after-classification.

Phase 1: Pre-classification.

Content-based anti-phishing classification starts with checks in the pre-classification phase, which classifies only [X]HTML documents and HTTP/S websites. It does not classify any tab in incognito mode, any URL allowlisted by enterprise policies, or any URL that is in the local client-side detection allowlist database, and it does not classify an IP address in the private IP ranges [61].

Phase 2: Main classification.

Once the pre-classification conditions are met, Google Safe Browsing's content-based and visual-based anti-phishing classifiers begin the process of computing scores based on the features and models to see if the webpage is a likely phishing website. This process consists of three steps:

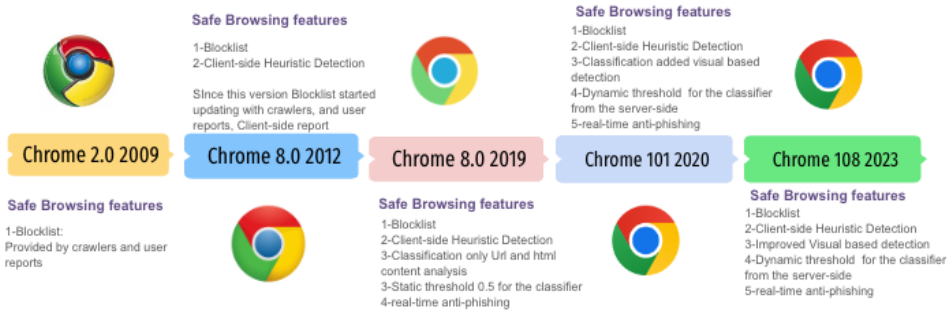

Figure 4: Chronological Evolution of Chrome's Anti-Phishing Techniques (2008-2023): This figure illustrates the progressive enhancement of Google Safe Browsing's client-side anti-phishing mechanisms, highlighting key developments in five major iterations - 2008, 2012, 2019, 2020, and 2023. Each version marks a significant advancement in combating phishing threats, showcasing Google's commitment to evolving cybersecurity measures.

First, if not already present, the system downloads the trained machine learning models for both the content-based and visual-based classifiers from the Google Safe Browsing server.

Next, for content-based classification, the system extracts URL, DOM, and term features from the visited website, while for visual-based classification, it extracts visual features. They have 500 ms threshold for feature extraction, and beyond this time the classification don't start. Finally, the system runs the scorer and calculates the final score for the content-based classifier and each of the extracted visual elements. Our study of the models used for client-side anti-phishing detection in Chrome lead us to identify a second constraint that seems to be essential for client-side solutions:

> **C2 - Simplicity**: Machine learning models used in phishing detection should be simple.

We realize that simplicity is not precisely defined, but it seems clear that it is an important consideration in client-side solutions.

Phase 3: Post-classification.

Once the main classification process is complete, the website will be identified as a potential phishing site if either of the following conditions is met: (1) the computed content-based phishy-ness score exceeds the threshold or (2) the visual-based scores for at least one visual element exceed the related threshold.

If any classifier determines that a website is potentially phishing, Chrome extracts features. These features include 21 different attributes that are listed in Table 6 in Appendix A. Chrome appends these features to the phishing URL and sends it to the server for further evaluation of the suspicious website.

While we do not have visibility into the server-side component, we suspect that the server-side will analyze the report, potentially visit the phishing URL, and make a final decision about whether the page is a phishing site or not.

If the server-side determines that the page is a phishing site, it blocks access and adds the site to the Google Safe Browsing blocklists. This information is then shared with all browsers that use the this blocklist, such as Google Chrome, Mozilla Firefox, and Apple Safari.

Security Level	Blocklist	Content based	Visual based	Real-time
No Protection	✓	✗	✗	✗
Standard Protection (Default)	✓	✓	✓	✗
Advanced Protection	✓	✓	✓	✓

Table 1: Chrome security levels. Chrome provides three levels of security, and each level uses related anti-phishing components.

3.2 Real-time Anti-phishing

In September 2019, Chrome version 79 was released with a new defense technique, called real-time anti-phishing, to address the blocklist latency problem and prevent attackers from exploiting the 30-minute blocklist refresh time. As of this writing, real-time anti-phishing is enabled only at the Advanced Protection level (shown in Table 1). We find that real-time anti-phishing works as follows: Chrome first checks visited websites against the allowlist database on the user's system containing thousands of popular websites known to be safe. If the visited website's URL does not match with the local whitelist database, then the browser sends the URL to the Google Safe Browsing server to check if the user is visiting a malicious website [20, 22].

4 CLASSIFIER EVALUATION

Our study evaluated Chrome's client-side anti-phishing classifier using real phishing samples from PhishTank [15]. Our preliminary findings revealed a high failure rate and many false positives in its detection capabilities. This led to a multi-year assessment focusing solely on the classifier's ability to detect phishing websites, crucial for user protection. The scope excluded an analysis of the false positive rate.

The experiment involved visiting verified phishing websites from PhishTank and testing them against the classifier. Figure 6 shows that results: only 66% of the phishing websites even underwent classification (they failed in the Pre-classification stage (Section 4.2), and we will describe the root cause in Section 4.2). In May 2020, using 300 phishing websites, we found that only 66% were even classified, and out of those, just 14% were correctly identified as

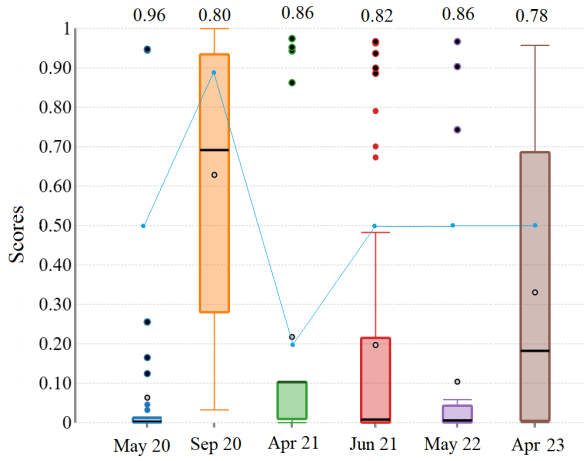

Figure 5: The result of classification of the content-based model across six versions of Chrome. For each experiment, we selected 100 new phishing websites from PhishTank and evaluated them against the model. The range of scores (the classifier's result) is different in these versions as well as the threshold. The number of the scores above the line that connected the thresholds are detected as phishing websites. The False Negative ratio is indicated above each box in the figure. For instance, in May 2020, the Client-side model had a False Negative ratio of 96%.

phishing, indicating a high false negative rate. The detailed results and analysis of the pre-classification failures are elaborated in a subsequent section.

Browser Validation Experiment. Because the classifier was based on Chromium (discussed in Section 3), we conduct a follow-up experiment to check for any differences between client-side anti-phishing models in Chrome and Chromium. In this experiment, we visited 100 additional (new) phishing websites (sourced from PhishTank) using both Chrome and Chromium. The classification outcomes were identical in both browsers, which confirmed that classifiers in Chrome and Chromium function in similar manners and validated our understanding of the client-side anti-phishing system in Chrome (and Chromium).

4.1 Longitudinal Model Drift

We continued to analyze the client-side classification in Chromium to understand how the model changed over time. In September 2020, we noticed a shift in classification results. By reverse engineering the updated source code, we identified changes to the content-based anti-phishing model and the classification algorithm, including a significant adjustment to the phishy-ness threshold. The previous model used a hard-coded threshold of 0.5, while the new model employed a dynamic threshold, which was tuned by GSB and sent to the client.

Therefore, during our observation period, each time the content-based classifier model changed, we re-evaluated its performance on 100 new phishing websites from PhishTank. This approach evaluates the classifier model's effectiveness when encountering new phishing websites.

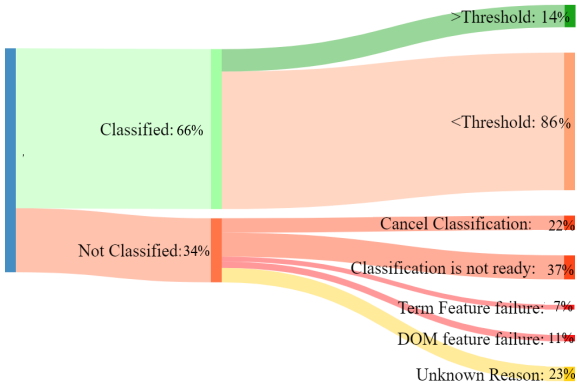

Figure 6: Classification results categories from the May 2020 classification experiment. This figure presents the percentage of the 300 phishing websites visited, on which Chrome performs classification, the resulting classification percentages, and the portion of websites that bypass content-based anti-phishing. Additionally, it illustrates the reasons for classification failure.

Figure 5 shows the results of six different experiments on the content-based. Before September 2020, the classifier used by Google Chrome used a hard-coded threshold of 0.5 (the blue line in Figure 5). However, in subsequent versions, the threshold was retrieved from the server and downloaded along with the classification model. We observed that the downloaded thresholds varied across different versions, with values of 0.8999976, 0.2, and a subsequent return to 0.5 observed in the last four models.

The results indicate that the number of false negative scores (scores below the threshold) remains noticeably high, even after tuning the threshold. For instance, the second new threshold was 0.2, and the scores accordingly changed with the changed model. As Figure 5 shows, accuracy in the new version of the model with 0.5 thresholds is improved from 0.02% in the earlier model with the static threshold to 0.14% in the last model.

4.2 Exploring Failures in Client-Side Detection Beyond Classifiers

In our initial experiment of 300 phishing websites, 34% bypassed the classifier completely, which is shown in Figure 6. To investigate these failures, we analyzed and categorized the Chrome log files, and matched with the Chromium source code. In some cases, we cannot determine the reason due to incomplete logs or ambiguous results, and we mark these as "Unknown Reasons."

We discover significant failures that can happen *before the classification process*, and we call these failures *avoidances*. Therefore, even if Chrome had a perfect client-side classification model, it would still fail to detect over 34% phishing websites because of these avoidances. We found four reasons for classification failures: model loading failure (37%), web page recapturing issues or not being reloaded (22%), DOM features extraction failure (11%), and scorer loading failure (7%). The failures caused by DOM features timing out indicate that attackers can bypass client-side classification, even if the classifier's model was perfectly accurate. The extraction

of DOM and Term features is constrained by a 500-millisecond timeout. If this duration is exceeded, classification is aborted, potentially allowing exploitation by attackers. The existence of a timeout of a short duration even at the expense of increased vulnerability to phishing attacks lead us to identify a third constraint that seems to guide the design of the client-side anti-phishing solution in Chrome:

C3 - Efficiency: The speed of detection is paramount for a client-side phishing detection

5 EVALUATION OF CLIENT-SIDE EFFECT ON BLOCKLISTING

Now that we have evaluated the client-side classification in Section 4, we turn our attention to analyzing the *impact* of client-side classification on blocklisting. In essence, this allows us to shed light on the outcomes of the server-side classification (Steps ③ and ④ in Figure 2) process, which ultimately measures how client-side classification protects *other* users.

5.1 Methodology

To evaluate blocklist performance, we used two main metrics: (*Discovery*) (the provider's ability to identify new suspected URLs) and (*Detection*) (the provider's ability to blocklist discovered URLs), further broken down into *Coverage* and *Speed* [52] as sub-metrics. Our methodology, primarily focused on Google Safe Browsing, is also applicable to other providers like Microsoft SmartScreen and Opera's protection services.

We utilized the PhishFarm framework[1] from the authors [51], modifying it for our experiments. We generated various inert phishing websites using a phishing kit, categorizing them based on the experimental conditions. Each site was hosted on a new, unique .com domain, with naming structures varying per group. We then tracked the time taken for these sites to be blocklisted on Chrome (both mobile and desktop), post-reporting to Google Safe Browsing or upon client-side detection, over a seven-day period.

5.2 Blocklist Experiments

In our comprehensive study, we conducted three key experiments to assess the efficacy of client-side anti-phishing methods in blocklisting, involving a total of 735 unique phishing websites.

5.2.1 May 2020 Experiment: For our initial experiment, we created 14 different experimental groups (with 35 URLs in each experimental group) that varied the following settings: random URLs vs suspicious URLs (impersonating brand names), content-based detection enabled or disabled, real-time phishing enabled or disabled, if the URL was reported directly to Google Safe Browsing, and if cloaking evasion technique was used. In addition, all websites were from one phishing kit that impersonated one brand as shown in Table 2. As shown in Table 2, this initial experiment revealed a major weakness in the client-side anti-phishing ecosystem. The phishing websites were only blocklisted if they were reported directly to Google Safe Browsing (GSB), and the content-based classification and real-time anti-phishing did not have any impact on the blocklisting process.

5.2.2 2021 Replication Studies: Following up on these findings, we conducted two smaller studies in 2021, confirming the limited impact of content-based and real-time anti-phishing on blocklisting. These studies led to an acknowledgment from Google and indicated ongoing improvements in their detection mechanisms.

We communicated our findings to Google in February 2021, who acknowledged them and indicated efforts to enhance client-side anti-phishing. However, a subsequent study in July 2021 showed that the issues still remained.

5.2.3 April 2023 Follow-Up: We conducted this experiment in April 2023, and Table 3 shows the seven different experimental conditions. In this experiment we also added the new phishing trend that we observed: websites that don't show a login form immediately. These include two types: traditional login websites and non-login sites, where users are prompted to scan a QR code or perform an action to access the login. We also used three different phishing kits to deploy the phishing websites that spoofed three different types of websites (two normal and one non-login phishing). The results pointed to ongoing challenges in effectively detecting and blocklisting phishing sites without direct reporting. For this experiment, we focused only on non-reporting conditions, and varied: random URLs vs suspicious URLs, Standard Protection vs Enhanced Protection, cloaking, and type of phishing page. Standard protection now includes both content-based and visual-based classifiers, while enhanced protection still includes standard protection plus real-time protection. We conducted this experiment in April 2023, and Table 3 shows the seven different experimental conditions.

Overall, our experiments revealed critical insights into the limitations and evolution of client-side anti-phishing strategies in the face of sophisticated phishing tactics.

Blocklist Coverage Experiments. Client-side detection with Google Safe Browsing blocked about half of phishing attacks in non-cloaking scenarios. However, as indicated in latency graphs (Figure 9 and Figure 10), these blocks were delayed, with none within the first 30 minutes and only 10% within the first hour. Despite this latency, 75% of victims still accessed the phishing sites. This marks a significant improvement from our initial experiments where client-side detection had zero impact on blocklist coverage.

C4 - Generality: Client-side anti-phishing solutions should be general and not restricted to specific attacks.

However, client-side measures were ineffective against cloaked phishing sites, underscoring the ease with which attackers can bypass these defenses. Notably, even the latest client-side detection failed to identify non-login phishing sites, highlighting the need for further advancements in anti-phishing technologies.

A key observation from our May 2020 experiment (Table 2) was the lack of blocklisting for cloaked sites on mobile Chrome, unlike desktop Chrome. This inconsistency, also noted by Oest et al. [51] 2018 in their research, persisted in our study. Our latest

[1] https://phishfarm-project.com

Experiment Settings						Total Number of Websites per group	Number of blocklisted websites: Desktop Chrome	Number of blocklisted websites: Mobile Chrome
Group ID	URL	CB	RT	Report	Evasion Technique			
1	Rand	✓	✗	✗	Cloaking	35	0	0
2	Rand	✓	✗	✗	No Cloaking	35	0	0
3	Susp	✓	✗	✗	No Cloaking	35	0	0
4	Rand	✓	✓	✗	No Cloaking	35	0	0
5	Susp	✓	✓	✗	Cloaking	35	1	0
6	Rand	✓	✓	✓	No Cloaking	35	33	35
7	Susp	✗	✓	✓	Cloaking	35	34	0
8	Susp	✓	✓	✓	No Cloaking	35	34	31
9	Susp	✓	✓	✓	Cloaking	35	34	33
10	Rand	✓	✗	✓	Cloaking	35	35	0
11	Susp	✓	✗	✓	Cloaking	35	35	2
12	Rand	✓	✗	✓	No Cloaking	35	35	33
13	Rand	✗	✓	✓	Cloaking	35	35	0
14	Rand	✗	✓	✓	No Cloaking	35	35	31

Table 2: May 2020 Evaluation of Blocklist Effectiveness for Unreported Phishing Websites The table displays the blocklist coverage performance for 14 distinct groups of phishing websites with login forms. The first five columns detail the varying experiment settings, encompassing URL type, Standard Protection, Enhanced Protection, Reporting Status, and the Evasion Techniques employed. Each experiment involves 35 websites, as indicated in column six. The final two columns demonstrate the blocklist coverage results for desktop and mobile Chrome browser platforms.

Experiment settings						Websites per Group	Desktop Chrome Blocklisted Sites	Mobile Chrome Blocklisted Sites
Group ID	URL	Standard Protection	Enhanced Protection	Evasion Techniques	Phishing Type			
1	Rand	✓	✗	Cloaking	Login	35	1	1
2	Rand	✓	✗	No cloaking	Login	35	18	18
3	Susp	✓	✗	No cloaking	Login	35	16	16
4	Rand	✗	✓	No Cloaking	Login	35	23	23
5	Susp	✗	✓	Cloaking	Login	35	3	3
6	Rand	✓	✗	No Cloaking	Non-Login	35	0	0
7	Rand	✗	✓	No Cloaking	Non-Login	35	0	0

Table 3: April 2023 Blocklist Efficacy Assessment for not reported Phishing Websites The table presents the blocklist coverage for seven unique phishing website categories not submitted to blocklist providers and detected exclusively via client-side methods. Organized into seven columns, the table outlines experiment parameters, including URL type, Standard and Enhanced Protection measures, Evasion Techniques used, and the classification of phishing websites (with or without a login page). The final two columns demonstrate the blocklist coverage effectiveness on both desktop and mobile Chrome browser platforms.

experiment in April 2023, however, shows Google has achieved consistent coverage between mobile and desktop browsers, though with varying response speeds.

Blocklist Speed Experiments. The goal of client-side anti-phishing is to minimize blocklist update delays. In May 2020, median blocklisting times were 88 minutes for desktop Chrome and 3 hours for mobile Chrome, with unreported sites not contributing to blocklisting. Our final experiment showed an improved median blocklisting time of 110 minutes for client-detected sites, still posing a risk for phishing attacks. This duration allows ample opportunity for attackers [55]. Figures 7 and 8 display these blocklisting speeds.

5.3 Effectiveness of Real-time Anti-phishing

Google claims that the new real-time anti-phishing component (September 2019) improves the detection of un-seen phishing websites by 30%. Our results demonstrate that even this additional component do not efficiently protect the user against new phishing websites. Blocklist experiment results for groups 5 and 6 (as shown in Table 2) for 70 phishing websites in these two groups, real-time anti-phishing does not affect the blocklisting process. For these 70 phishing websites, Chrome with content-based and real-time anti-phishing methods enabled blocklisted one phishing website.

6 ACADEMIC WORKS: A FIVE-YEAR REVIEW

We conducted a review of 25 papers from major security conferences, from 2018 to 2023, including S&P, Usenix, CCS, NDSS, AsiaCCS, ACM journals, and the WIC conference, representing a significant body of work in the field. The chosen conferences are known for rigorous peer-review processes and for hosting pioneering works in security, making them ideal sources for our analysis. Our review evaluated each of these papers against the four constraints that we identified for client-side anti-phishing solutions and which we already highlighted in green boxes in the paper. Our objective was to evaluate the applicability and relevance of these academic findings to the practical needs of current industrial client-side anti-phishing solutions, thereby bridging the gap between theoretical research and practical application.

We divided the papers into two categories: those proposing phishing detection models and those that did not. For those proposing models, we further analyzed their suitability for client-side detection based on the four constraints that we identified:

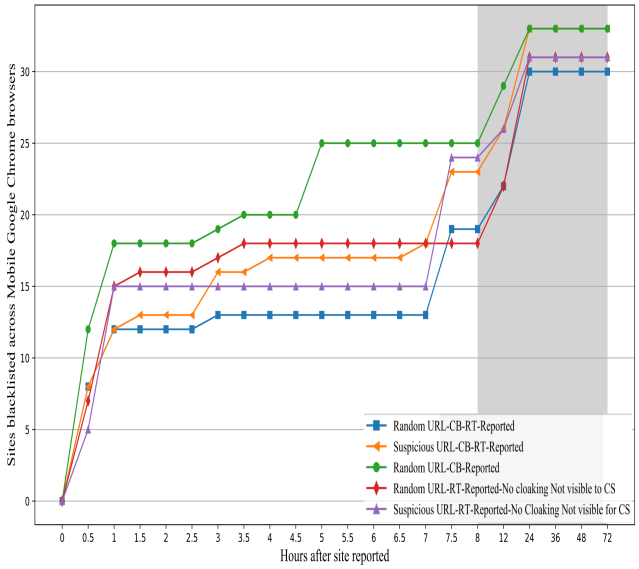

Figure 7: Mobile blocklisting Performance. This figure shows the delay between initial reporting using the mobile Chrome browser to blocking for five different groups of websites. Each group was set up differently. For example, the group with websites that had random URLs used server-side cloaking and were reported to blocklist entities when visited from mobile Chrome with content-based anti-phishing enabled was blocked faster than the other groups. This information is from the initial experiment conducted in May 2020.

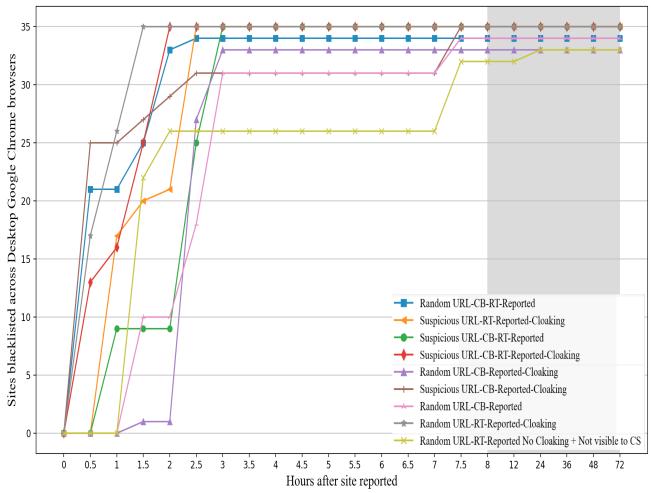

Figure 8: Desktop blocklisting Performance. This figure shows the delay between initial reporting using the desktop Chrome browser to blocking for nine different groups of websites that we did in May 2020. Each group had a different setup that you can see in Table 2. For instance, group 8, which had websites with suspect URLs, used cloaking on the server-side and reported to blocklist entities when visited from desktop Chrome with anti-phishing enabled, were blocked faster than other groups.

1. Locality: This criterion assesses methods that necessitate resources beyond local capabilities, encompassing features unavailable locally, such as domain provider information. The method outlined in paper [34], for instance, stands out in terms of efficiency and accuracy. Nonetheless, it requires external resources, like domain provider information, to conduct the classification process.

2. Simplicity: This constraint evaluates the complexity of machine learning methods in the proposed anti-phishing solutions. For instance, the Phishpedia [42] model demonstrates high accuracy and minimal runtime overhead. However, despite these strengths, deep learning models like Phishpedia often encounter challenges in real-world scenarios, particularly regarding their adaptability and complexity in uncontrolled environments. This highlights the need for solutions that balance complexity and practical applicability.

3. Efficiency: This aspect evaluates the detection speed and the model's size. The benchmark for Chrome's anti-phishing feature is a 500 ms response time. However, certain studies, such as [77], report a higher time overhead for detection: approximately 800 ms for benign websites and 3000 ms for malicious sites.

4. Generality: This aspect assesses if the anti-phishing methods are universally applicable or tailored to specific phishing types, such as financial phishing. For instance, the authors referenced in [29] offer key insights on browser-level defenses, focusing primarily on IDN-based phishing. However, this specialized approach may not fully integrate with Google Chrome's wider anti-phishing strategies, which are designed to address a broader range of phishing threats. Beside the specific attack, some papers [71] focus on specific platform like android.

Some of the reviewed papers do not directly propose anti-phishing solutions, but still contributes to the field of anti-phishing research. For instance, the study presented in [53] focuses on aspects related to phishing but doesn't outline a specific anti-phishing method. Despite this, the findings from 'PhishTime' could offer valuable insights to enhance anti-phishing measures in browsers like Google Chrome, particularly in assessing the efficacy of blocklists. However, the integration and relevance of these insights would largely depend on how they align with Google Chrome's existing anti-phishing strategies and implementation frameworks.

A summary of the evaluations appear in Tables 4 and 5. The current state of client-side anti-phishing detection, as observed in practice, is concerning and highlights the need for more focused academic research in this area. Our analysis is a step towards understanding the solution requirements for effective client-side anti-phishing. These challenges should motivate the community to design more robust and innovative client-side detection frameworks, moving beyond the existing paradigms that appear to be limited in addressing evolving phishing threats.

7 DISCUSSION

Our study's primary purpose was to evaluate content-based anti-phishing, and our initial experiment revealed surprising results in this regard. Our study in May 2020 found that the client-side anti-phishing system aimed to reduce the time lag in blocklists

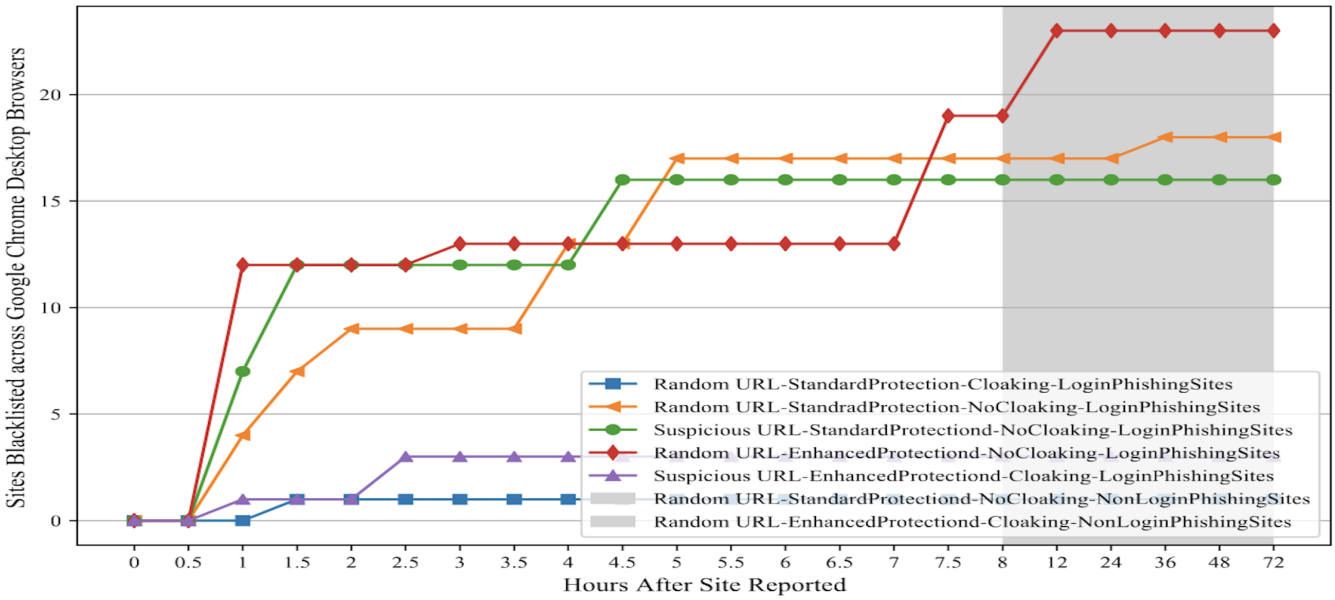

Figure 9: Desktop blocklisting Performance in the latest experiment in April 2023. This figure shows the delay between initial reporting using the desktop Chrome browser to blocking for our final experiments that we did in April 2023. We divided the websites into five different groups; each group had a different setup that you can see in Table 2.

Not Efficient	Not Simple	Not Local	Not General	Not Relevant
[50] [77]	[39] [42] [76]	[33] [34] [36] [35] [56]	[12] [29] [37] [72] [26] [71]	[9] [14] [17] [43] [53] [58] [59] [60] [65]

Table 4: Suitability of recent academic works for client-side anti-phishing

Not Efficient	Not Simple	Not Local	Not General
High Overhead (speed) [77]	Hybrid Deep Learning method [42]	Not-local detection [34] [36] [35]	Specific for Crypto Phishing [12]
Resource Efficiency [50]	Complexity of Cloaking Detection features [76]	Needs Human Interaction [33]	Specific for Financial Phishing [72]
	Use Multiple Models [39]	Search Engine Query needs [56]	Specific for inter-organization phishing [37]
			Specific for Android Platform [71]
			Specific for IDN attacks [29]
			Specific for Email system [26] [72]

c

Table 5: Categorization of Academic Research Based on Anti-Phishing Criteria

protecting users from new phishing attacks. However, our subsequent experiments conducted in April 2023 showed that, despite the improvements made to the system by Google, client-side detection only led to a 10% increase in the number of websites blocklisted within the first hour. This implies that there is still scope for improvement in the overall anti-phishing ecosystem.

However, our findings show that for phishing websites that are reported without cloaking techniques, content-based anti-phishing can accelerate blocklisting.

Privacy is also another unsolved challenge in Chrome's client-side anti-phishing. Google Safe Browsing client-side anti-phishing claims that it protects users' sensitive information [20]; however, our experiment shows that content-based anti-phishing and password protection sends a request to the server that includes the visited URL and the list of URLs with the same stored passwords. These URLs might be transmitting sensitive information in the context of a false positive classification result.

Our findings could be used to improve content-based classification accuracy. Continuous monitoring of recent phishing websites' structure—including URLs and the content—training the related machine-learning model, and refining the classification algorithm with this data, can further improve content-based detection.

Our empirical blocklist experiments on Chrome and GSB could be adapted to analyze other modern browsers and threats, including spam filters and malware.

7.1 Limitations

Source Data.Our study highlighted weaknesses in Chrome's client-side anti-phishing ecosystem, but it's important to consider the limitations of our analysis. Initially, we focused on a single organization, potentially biasing our findings. We addressed this in our final experiment by including more brands and phishing styles, ensuring diverse data and reducing bias. **Source Classifier**. We used Chromium's open-source code for evaluating Chrome's anti-phishing defenses, as Chrome isn't open-source, but this didn't affect our results due to the presence of the Google Safe Browsing system in both. **Reporting**. Our reporting was based on a single

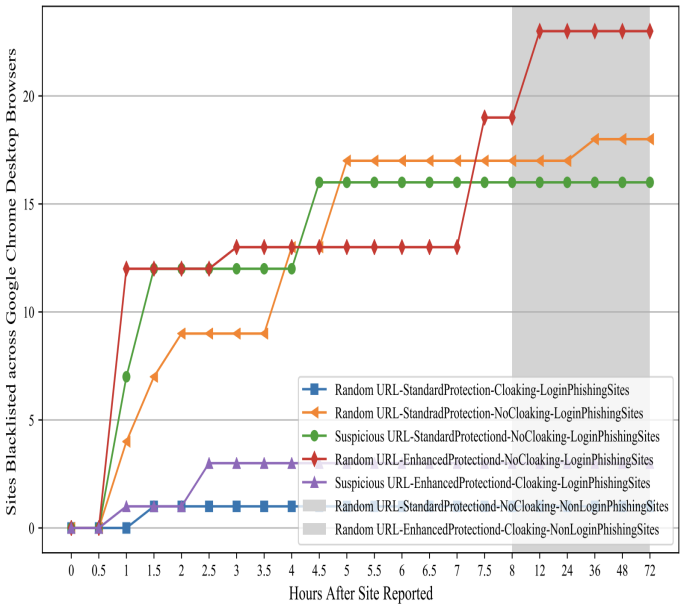

Figure 10: Mobile Blocklisting Performance in the latest experiment in April 2023. This figure shows the delay between initial reporting using the mobile Chrome browser to blocking for our final experiments on the mobile Chrome browser in April 2023. We divided the websites into five different groups; each group had a different setup (see Table 2.)

submission per phishing site, which might differ from the multiple reports that could occur in real-world scenarios. Although our study revealed Chrome's client-side anti-phishing ecosystem's weaknesses in the real world, our analysis should be considered along with its limitations.

7.2 Disclosure

In February 2021, we reported to Google the shortcomings in content-based anti-phishing and mobile blocklisting gaps. Google recognized these problems and was working on enhancing their client-side anti-phishing. Later evaluations showed some improvements in the classifier, but our results in Figure 5, still highlighted persistent weaknesses in content-based anti-phishing.

8 RELATED WORK

Our study represents the first thorough evaluation of client-side content-based anti-phishing, including empirical performance analysis and bridging the gap between industry and academia. Unlike prior research focusing on blocklisting [1, 64, 73], browser security warnings [2, 16, 19, 68], and anti-phishing toolbars [31, 44, 46, 70, 75], our work distinctively assesses the efficacy of in-browser classifiers alongside blocklisting strategies, addressing a previously unexplored area in the field.

Oest et al. [51] assessed the timeliness of anti-phishing blocklists and the efficacy of evasion techniques. They used the PhishFarm framework to test cloaking techniques against anti-phishing entities by deploying fake phishing websites. Similarly, we adapted that framework to evaluate content-based and real-time anti-phishing effectiveness in major web browsers.

Liang et al. [41] demonstrated the vulnerabilities of Chrome's content-based classifier by showing how feature modifications could drastically alter a site's phishing score, though they didn't assess its broader ecosystem impacts. In contrast, our study uncovers new attack methods that evade classification entirely.

Lei et al. [40] assessed Chrome's phishing detection in 2019, focusing on evasion attacks and proposing methods to increase classifier robustness. Our research extends this by empirically examining client-side anti-phishing's effects on blocklist coverage and response times.

Oest et al. [55] studied the lifecycle of large-scale phishing attacks, identifying detection gaps and the timing of attacks. They introduced the 'Golden Hour' concept, finding that rapid blocklisting is crucial to mitigate the impact of these attacks and emphasized the importance of effective content-based anti-phishing in the early stages to protect potential victims.

9 CONCLUSION

In our study, we analyzed the effectiveness of client-side classification in detecting never-before-seen phishing websites. We assessed the impact of client-side detection on facilitating blocklisting from May 2020 to April 2023. Our analysis encompassed the strengths and weaknesses of the Google Safe Browsing client-side anti-phishing ecosystem, including blocklists, content-based, visual-based, and real-time detection, which are components of different layers of protection in Chrome, including standard protection and enhanced protection. Through a longitudinal evaluation, we aimed to comprehend how client-side detection could be enhanced to better safeguard end-users from new and evolving phishing attacks.8

After reporting our findings to Google and their subsequent commitment to making improvements, the final experiment conducted in April 2023 showed an increase of zero in blocklisting efficacy in the first 30 minutes and only 10% in the first hour, with a 51% increase over seven days. While this improvement was noteworthy, the system still encountered difficulties in effectively blocking new and evolving phishing attacks, particularly those that did not include login pages. Our study specifically tested non-login phishing websites, highlighting the challenges in detecting these types of attacks. Additionally, there are still design vulnerabilities that attackers could potentially exploit to bypass client-side detection measures.

Our study of the evolution of Chrome's client-side anti-phishing and our identification of constraints that seem to mandate some design decisions was useful in understanding proposed solutions in the academic literature as possible improvements for practically deployed client-side anti-phishing systems such as Chrome. Our academic literature review indicates that there is need for more work and attention from academia to the client-side anti-phishing problem. While the constraints that we identified are not the last word on the unavoidable constraints that practical systems must be contend with, we believe that they are an important first step towards identifying such constraints.

10 ACKNOWLEDGEMENTS

This material is based upon work supported by the Advanced Research Projects Agency for Health (ARPA-H) under Contract No. SP4701-23-C-0074 and the Defense Advanced Research Projects Agency (DARPA) and Naval Information Warfare Center Pacific (NIWC Pacific) under Contracts No. N66001-22-C-4026 and No. N66001-20-C-4020. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of ARPA-H, DARPA, or NIWIC Pacific.

REFERENCES

[1] R Abrams, Orlando Barrera, and J Pathak. 2013. Browser Security Comparative Analysis. *NSS Labs* (2013).

[2] Devdatta Akhawe and Adrienne Porter Felt. 2013. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 257–272.

[3] Ahmed AlEroud and George Karabatis. 2020. Bypassing Detection of URL-based Phishing Attacks Using Generative Adversarial Deep Neural Networks. In *Proceedings of the Sixth International Workshop on Security and Privacy Analytics*. 53–60.

[4] Ahmed Aleroud and Lina Zhou. 2017. Phishing environments, techniques, and countermeasures: A survey. *Computers & Security* 68 (2017), 160–196.

[5] Charles Anderson. 2019. Adversarial Sampling Attacks Against Phishing Detection. In *Data and Applications Security and Privacy XXXIII: 33rd Annual IFIP WG 11.3 Conference, DBSec 2019, Charleston, SC, USA, July 15–17, 2019, Proceedings*, Vol. 11559. Springer, 83.

[6] Anti-Phishing Working Group (APWG). 2022. Phishing Activity Trends Report: Q3 2022. https://docs.apwg.org/reports/apwg_trends_report_q3_2022.pdf. (2022). Accessed on April 20, 2023.

[7] Giovanni Apruzzese, Mauro Conti, and Ying Yuan. 2022. SpacePhish: The Evasion-space of Adversarial Attacks against Phishing Website Detectors using Machine Learning. In *Proceedings of the 38th Annual Computer Security Applications Conference*. 171–185.

[8] Simon Bell and Peter Komisarczuk. 2020. An Analysis of Phishing Blacklists: Google Safe Browsing, OpenPhish, and PhishTank. In *Proceedings of the Australasian Computer Science Week Multiconference*. 1–11.

[9] Bitaab. 2023. BEYOND PHISH: Toward Detecting Fraudulent e-Commerce Websites at Scale. In *S and P*.

[10] Marzieh Bitaab, Haehyun Cho, Adam Oest, Penghui Zhang, Zhibo Sun, Rana Pourmohamad, Doowon Kim, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. 2020. Scam Pandemic: How Attackers Exploit Public Fear through Phishing. In *2020 APWG Symposium on Electronic Crime Research (eCrime)*.

[11] AJ Burns, M Eric Johnson, and Deanna D Caputo. 2019. Spear phishing in a barrel: Insights from a targeted phishing campaign. *Journal of Organizational Computing and Electronic Commerce* 29, 1 (2019), 24–39.

[12] Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. 2020. Phishing scams detection in ethereum transaction network. *ACM Transactions on Internet Technology (TOIT)* 21, 1 (2020), 1–16.

[13] Sen Chen, Minhui Xue, Lingling Fan, Shuang Hao, Lihua Xu, Haojin Zhu, and Bo Li. 2018. Automated poisoning attacks and defenses in malware detection systems: An adversarial machine learning approach. *computers & security* 73 (2018), 326–344.

[14] Tom Chothia, Stefan-Ioan Paiu, and Michael Oultram. 2018. Phishing attacks: Learning by doing. In *2018 USENIX Workshop on Advances in Security Education (ASE 18)*.

[15] Cisco Talos Intelligence Group. 2023. PhishTank: Phishing URL data-set operated by Cisco Talos Intelligence Group. https://phishtank.com. (2023). Accessed on April 20, 2023.

[16] Rachna Dhamija, J Doug Tygar, and Marti Hearst. 2006. Why phishing works. In *Proceedings of the SIGCHI conference on Human Factors in computing systems*. 581–590.

[17] Matt Dixon, James Nicholson, Dawn Branley-Bell, Pam Briggs, and Lynne Coventry. 2022. Holding your hand on the danger Button: observing user phish detection strategies across Mobile and desktop. *Proceedings of the ACM on human-computer interaction* 6, MHCI (2022), 1–22.

[18] Ronald C Dodge Jr, Curtis Carver, and Aaron J Ferguson. 2007. Phishing for user security awareness. *computers & security* 26, 1 (2007), 73–80.

[19] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. 2008. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1065–1074.

[20] Google. 2020. Google Chrome Privacy Whitepaper. https://www.google.com/chrome/privacy/whitepaper.html. (2020).

[21] Google. accessed May 4, 2023. Google Safe Browsing Transparency Report. https://safebrowsing.google.com/. (accessed May 4, 2023).

[22] Google. accessed May 4, 2023. Safe Browsing APIs (v4). https://developers.google.com/safe-browsing/v4/. (accessed May 4, 2023).

[23] Brij B Gupta, Nalin AG Arachchilage, and Kostas E Psannis. 2018. Defending against phishing attacks: taxonomy of methods, current issues and future directions. *Telecommunication Systems* 67, 2 (2018), 247–267.

[24] Xiao Han, Nizar Kheir, and Davide Balzarotti. 2016. Phisheye: Live Monitoring of Sandboxed Phishing Kits. (2016), 1402–1413 pages.

[25] Shuang Hao, Matthew Thomas, Vern Paxson, Nick Feamster, Christian Kreibich, Chris Grier, and Scott Hollenbeck. 2013. Understanding the domain registration behavior of spammers. In *Proceedings of the 2013 conference on Internet measurement conference*. 63–76.

[26] Grant Ho, Asaf Cidon, Lior Gavish, Marco Schweighauser, Vern Paxson, Stefan Savage, Geoffrey M Voelker, and David Wagner. 2019. Detecting and characterizing lateral phishing at scale. In *28th USENIX security symposium (USENIX security 19)*. 1273–1290.

[27] Thorsten Holz, Markus Engelberth, and Felix Freiling. 2009. Learning more about the underground economy: A case-study of keyloggers and dropzones. In *European Symposium on Research in Computer Security*. Springer, 1–18.

[28] Jason Hong. 2012. The state of phishing attacks. *Commun. ACM* 55, 1 (2012), 74–81.

[29] Hang Hu, Steve TK Jan, Yang Wang, and Gang Wang. 2021. Assessing Browser-level Defense against {IDN-based} Phishing. In *30th USENIX Security Symposium (USENIX Security 21)*. 3739–3756.

[30] Weiwei Hu and Ying Tan. 2017. Black-box attacks against RNN based malware detection algorithms. *arXiv preprint arXiv:1705.08131* (2017).

[31] Huajun Huang, Shaohong Zhong, and Junshan Tan. 2009. Browser-side countermeasures for deceptive phishing attack. In *2009 Fifth International Conference on Information Assurance and Security*, Vol. 1. IEEE, 352–355.

[32] Federal Bureau Of Investigation. 2018. Business E-mail Compromise The 12 Billion Dollar Scam. (2018).

[33] Amir Kashapov, Tingmin Wu, Sharif Abuadbba, and Carsten Rudolph. 2022. Email summarization to assist users in phishing identification. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. 1234–1236.

[34] Taeri Kim, Noseong Park, Jiwon Hong, and Sang-Wook Kim. 2022. Phishing URL Detection: A Network-based Approach Robust to Evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1769–1782.

[35] Taeri Kim, Noseong Park, Jiwon Hong, and Sang-Wook Kim. 2022. Phishing URL Detection: A Network-based Approach Robust to Evasion. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 1769–1782.

[36] Brian Kondracki, Babak Amin Azad, Oleksii Starov, and Nick Nikiforakis. 2021. Catching transparent phish: Analyzing and detecting MITM phishing toolkits. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 36–50.

[37] Daniele Lain, Kari Kostiainen, and Srdjan Čapkun. 2022. Phishing in organizations: Findings from a large-scale and long-term study. In *2022 IEEE Symposium on Security and Privacy (SP)*. IEEE, 842–859.

[38] Pavel Laskov et al. 2014. Practical evasion of a learning-based classifier: A case study. In *2014 IEEE symposium on security and privacy*. IEEE, 197–211.

[39] Jehyun Lee, Pingxiao Ye, Ruofan Liu, Dinil Mon Divakaran, and Mun Choon Chan. 2020. Building robust phishing detection system: an empirical analysis. *NDSS MADWeb* (2020).

[40] Yusi Lei, Sen Chen, Lingling Fan, Fu Song, and Yang Liu. 2020. Advanced Evasion Attacks and Mitigations on Practical ML-Based Phishing Website Classifiers. *arXiv preprint arXiv:2004.06954* (2020).

[41] Bin Liang, Miaoqiang Su, Wei You, Wenchang Shi, and Gang Yang. 2016. Cracking classifiers for evasion: a case study on the google's phishing pages filter. In *Proceedings of the 25th International Conference on World Wide Web*. 345–356.

[42] Yun Lin, Ruofan Liu, Dinil Mon Divakaran, Jun Yang Ng, Qing Zhou Chan, Yiwen Lu, Yuxuan Si, Fan Zhang, and Jin Song Dong. 2021. Phishpedia: A hybrid deep learning based approach to visually identify phishing webpages. In *30th USENIX Security Symposium (USENIX Security 21)*. 3793–3810.

[43] Zane Ma, Joshua Reynolds, Joseph Dickinson, Kaishen Wang, Taylor Judd, Joseph D Barnes, Joshua Mason, and Michael Bailey. 2019. The impact of secure transport protocols on phishing efficacy. In *12th USENIX Workshop on Cyber Security Experimentation and Test (CSET 19)*.

[44] Samuel Marchal, Giovanni Armano, Tommi Gröndahl, Kalle Saari, Nidhi Singh, and N Asokan. 2017. Off-the-hook: An efficient and usable client-side phishing prevention application. *IEEE Trans. Comput.* 66, 10 (2017), 1717–1733.

[45] AbdelKarim Mardini. 2019. Better password protections in Chrome. https://blog.google/products/chrome/better-password-protections. (Dec 2019).

[46] Noman Mazher, Imran Ashraf, and Ayesha Altaf. 2013. Which web browser work best for detecting phishing. In *2013 5th International Conference on Information and Communication Technologies*. IEEE, 1–5.
[47] Heather McCalley, Brad Wardman, and Gary Warner. 2011. Analysis of backdoored phishing kits. In *IFIP International Conference on Digital Forensics*. Springer, 155–168.
[48] D Kevin McGrath and Minaxi Gupta. 2008. Behind Phishing: An Examination of Phisher Modi Operandi. *LEET* 8 (2008), 4.
[49] Microsoft. accessed May 4, 2023. Microsoft Defender SmartScreen Overview. https://docs.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen/microsoft-defender-smartscreen-overview. (accessed May 4, 2023).
[50] Akihito Nakamura and Fuma Dobashi. 2019. Proactive phishing sites detection. In *IEEE/WIC/ACM International Conference on Web Intelligence*. 443–448.
[51] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. 2019. 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, 1344–1361.
[52] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. 2020. PhishTime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 379–396.
[53] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. 2020. {PhishTime}: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *29th USENIX Security Symposium (USENIX Security 20)*. 379–396.
[54] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. 2018. Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*. IEEE, 1–12.
[55] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. 2020. 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)*.
[56] Thomas Kobber Panum, Kaspar Hageman, René Rydhof Hansen, and Jens Myrup Pedersen. 2020. Towards Adversarial Phishing Detection. In *13th {USENIX} Workshop on Cyber Security Experimentation and Test ({CSET} 20)*.
[57] Vasilii Sukhanov Patrick Nepper, Kiran C. Nair and Varun Khaneja. 2019. Better password protections in Chrome - How it works. https://security.googleblog.com/2019/12/better-password-protections-in-chrome.html. (Dec 2019).
[58] Peng Peng, Chao Xu, Luke Quinn, Hang Hu, Bimal Viswanath, and Gang Wang. 2019. What happens after you leak your password: Understanding credential sharing on phishing sites. In *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*. 181–192.
[59] Nikolas Pilavakis, Adam Jenkins, Nadin Kökciyan, and Kami Vaniea. 2023. "I didn't click": What users say when reporting phishing. In *Symposium on Usable Security and Privacy (USEC) 2023*. The Internet Society, 1–13.
[60] Benjamin Reinheimer, Lukas Aldag, Peter Mayer, Mattia Mossano, Reyhan Duezguen, Bettina Lofthouse, Tatiana Von Landesberger, and Melanie Volkamer. 2020. An investigation of phishing awareness and education over time: When and how to best remind users. In *Sixteenth Symposium on Usable Privacy and Security (SOUPS 2020)*. 259–284.
[61] Yakov Rekhter, Robert Moskowitz, Daniel Karrenberg, Geert Jan de Groot, and Eliot Lear. 1996. *Address Allocation for Private Internets*. RFC 1918. RFC Editor. 1–9 pages. https://datatracker.ietf.org/doc/html/rfc1918
[62] Bushra Sabir, M Ali Babar, and Raj Gaire. 2020. An Evasion Attack against ML-based Phishing URL Detectors. *arXiv preprint arXiv:2005.08454* (2020).
[63] Mahmood Sharif, Sruti Bhagavatula, Lujo Bauer, and Michael K Reiter. 2016. Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition. In *Proceedings of the 2016 acm sigsac conference on computer and communications security*. 1528–1540.
[64] Steve Sheng, Brad Wardman, Gary Warner, Lorrie Cranor, Jason Hong, and Chengshan Zhang. 2009. An empirical analysis of phishing blacklists. (2009).
[65] Hossein Siadati, Sean Palka, Avi Siegel, and Damon McCoy. 2017. Measuring the effectiveness of embedded phishing exercises. In *10th USENIX workshop on cyber security experimentation and test (CSET 17)*.
[66] Aditya K Sood and Richard J Enbody. 2013. Crimeware-as-a-service—a survey of commoditized crimeware in the underground market. *International Journal of Critical Infrastructure Protection* 6, 1 (2013), 28–38.
[67] StatCounter Global Stats. 2020. Desktop vs Mobile vs Tablet Market Share Worldwide. https://gs.statcounter.com/platform-market-share/desktop-mobile-tablet/worldwide. (2020).
[68] Joshua Sunshine, Serge Egelman, Hazim Almuhimedi, Neha Atri, and Lorrie Faith Cranor. 2009. Crying Wolf: An Empirical Study of SSL Warning Effectiveness.. In *USENIX security symposium*. Montreal, Canada, 399–416.
[69] Kurt Thomas, Damon McCoy, Chris Grier, Alek Kolcz, and Vern Paxson. 2013. Trafficking fraudulent accounts: The role of the underground market in Twitter spam and abuse. In *22nd {USENIX} Security Symposium ({USENIX} Security 13)*. 195–210.
[70] Nikolaos Tsalis, Nikos Virvilis, Alexios Mylonas, T Apostolopoulos, and Dimitris Gritzalis. 2014. Browser blacklists: the Utopia of phishing protection. In *International Conference on E-Business and Telecommunications*. Springer, 278–293.
[71] Güliz Seray Tuncay, Jingyu Qian, and Carl A Gunter. 2020. See no evil: phishing for permissions with false transparency. In *29th USENIX Security Symposium (USENIX Security 20)*. 415–432.
[72] Amber Van Der Heijden and Luca Allodi. 2019. Cognitive triaging of phishing attacks. In *28th USENIX Security Symposium (USENIX Security 19)*. 1309–1326.
[73] Nikos Virvilis, Nikolaos Tsalis, Alexios Mylonas, and Dimitris Gritzalis. 2014. Mobile devices: A phisher's paradise. In *2014 11th International Conference on Security and Cryptography (SECRYPT)*. IEEE, 1–9.
[74] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-scale automatic classification of phishing pages. (2010).
[75] Chuan Yue and Haining Wang. 2008. Anti-phishing in offense and defense. In *2008 Annual Computer Security Applications Conference (ACSAC)*. IEEE, 345–354.
[76] Penghui Zhang, Adam Oest, Haehyun Cho, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kpravelos, Tiffany Bao, Ruoyu Wang, Yan Shoshitaishvili, Adam Doupé, and Gail-Joon Ahn. 2021. CrawlPhish: Large-scale Analysis of Client-side Cloaking Techniques in Phishing. In *Proceedings of the 42nd IEEE Symposium on Security and Privacy (Oakland)*.
[77] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, et al. 2022. 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*. 3165–3179.

A APPENDIX

Feature	Brower features	Description of the features
History features	kUrlHistoryVisitCount	Number of visits to that URL stored in the browser history
	kUrlHistoryTypedCount	Number of times the URL was typed in the Omnibox
	kUrlHistoryLinkCount	Number of times the URL was reached by clicking a link
	kUrlHistoryVisitCountMoreThan24hAgo	Number of times URL was visited more than 24h ago
	kHttpHostVisitCount	Number of user-visible visits to all URLs on the same host/port as the URL for HTTP and HTTPS
	kHttpsHostVisitCount	
	kFirstHttpHostVisitMoreThan24hAgo	Boolean feature which is true if the host was visited for the first time more than 24h ago (only considers user-visible visits)
	kFirstHttpsHostVisitMoreThan24hAgo	
Browse features	kHostPrefix	prefixes appended to features that tell for which page type the feature pertains
	kReferrer	Referrer
	kHasSSLReferrer	True if the referrer was stripped because it is an SSL referrer
	kPageTransitionType	Stores the page transition
	kIsFirstNavigation	True if this navigation is the first for this tab
	kRedirectUrlMismatch	Feature that is set if the URL from the navigation entry doesn't match the URL at the end of the redirect chain
	kRedirect	The redirect chain that leads to the named page
	kSecureRedirectValue	If a redirect is SSL, we will use this value instead of the actual redirect so we don't leak any SSL websites
	kHttpStatusCode	The HTTP status code for the main document
	kSafeBrowsingMaliciousUrl kSafeBrowsingOriginalUrl kSafeBrowsingIsSubresource kSafeBrowsingThreatType	Fields from the UnsafeResource if there is any

Table 6: Browser features are the features that content-based anti-phishing extracts if the classifier has scored more than the threshold and detects the website as a phishing website. Content-based anti-phishing extracts these features and adds to the request that it sends to the server to run the server-side classification on the suspected website.

DOM feature type	Model DOM features	Page DOM features
DOM HTML form features	PageHasForms	Page has any form element
	PageActionOtherDomainFreq	The fraction of form elements whose "action" attribute point URL on a different domain
	PageActionURL	The token feature containing each URL that an "action" attribute points to
	PageHasTextInputs	The page has any input type="text" element
	PageHasPswdInputs	The page has any input type="password" element
	PageHasRadioInputs	The page has any input type="radio" element
	PageHasCheckInputs	The page has any input type="checkbox" element
DOM HTML link features	PageExternalLinksFreq	The fraction of links on the page that point~ to other domains
	PageLinkDomain	The token feature containing each external domain that is linked to
	PageSecureLinksFreq	The fraction of links on the page that use HTTPS
DOM HTML script features	PageNumScriptTagsGTOne	The number of script elements on the page is greater than 1?
	PageNumScriptTagsGTSix	The number of script elements on the page is greater than 6?
Other DOM HTML features	PageImgOtherDomainFreq	The fraction of images whose source attribute points to an external domain

Table 7: DOM features. Content-based classifier uses the content of the visited website to evaluate its phishy-ness. It extracts the features from the website, creates a feature map, and computes the phishy-ness score based on the model's rules' features and their weights.

BL Performance Metrics		Description
Discovery		blocklist's ability to identify new suspected URLs
Detection	Coverage	The proportion of the discovered URLs that are correctly blocklisted at any point
	Speed	The time delay between discovery and blocklisting

Table 8: Blocklisting evaluation metrics [52].