

Uncovering Flaws in Anti-Phishing Blacklists for Phishing Websites Using Novel Cloaking Techniques

Wenhao Li^{1,2}^a, Yongqing He², Zhimin Wang², Saleh Mansor Alqahtani¹ and Priyadarsi Nanda¹^b

¹Faculty of Engineering and IT, University of Technology Sydney, NSW 2007, Australia

²Chengdu MeetSec Technology Co., Ltd., Chengdu, China


Keywords: Cloaking Techniques, Evasion Techniques, Phishing Blacklists, Anti-Phishing Entities, Phishing Websites.


Abstract: The proliferation of phishing attacks pose substantial threats to global prosperity amidst the Fourth Industrial Revolution. Given the burgeoning number of Internet users and devices, cyber criminals are harnessing phishing toolkits and Phishing-as-a-Service (PhaaS) platforms to spawn numerous fraudulent websites. In retaliation, assorted detection mechanisms, with anti-phishing blacklists acting as a primary line of defense against phishing sites, have been proposed. Yet, adversaries have contrived cloaking techniques to dodge this detection method. This study endeavors to unearth the shortcomings of prevailing blacklists and thereby bolster the efficacy of detection strategies for Anti-Phishing Entities (APEs). This paper presents an exhaustive analysis of innovative and practicable attacks on current anti-phishing blacklists, unmasking potential weaknesses in these protection mechanisms hitherto unexplored in prior research. Additionally, we divulge potential loopholes exploitable by attackers and appraise their effectiveness against popular browser blacklists.

1 INTRODUCTION

Phishing, a prevalent form of cybercrime leveraging social engineering, manipulates victims' trust to purloin funds or sensitive data (Pujara and Chaudhari, 2018). Recent years have witnessed a threefold surge in the incidence of these attacks (APWG, 2022). As projected by Cybercrime Magazine, the global cost of cybercrime is set to reach \$10.5 trillion by 2025 (Freeze, 2018), manifesting cybercrime as a significant threat to prosperity in the Fourth Industrial Revolution (World Economic Forum, 2020). Perpetrators execute phishing attacks via diverse channels such as social media, email, and text messaging, often deploying fraudulent websites mimicking legitimate ones (Gupta et al., 2016). A striking escalation is evident in the quantity of distinct phishing websites identified—316,747 in December 2021, up from a mere 60,926 in December 2017 (APWG, 2017; APWG, 2022). This trend owes to the expanding Internet user base and the proliferation of connected devices, offering an appealing landscape for criminal activity. Additionally, the advent of phishing

toolkits and PhaaS platforms enable the creation of numerous phishing websites at a reduced cost, requiring minimal coding expertise (Han et al., 2016; Alabdan, 2020). Numerous approaches have been developed to detect phishing websites, including blacklist/whitelist-based, visual similarity-based, heuristic-based, machine learning-based, and deep learning-based detection (Al-Ahmadi et al., 2022). These methods function independently or form part of the mechanisms integrated into the anti-phishing ecosystem, where browsers wield significant influence via blacklists—access control mechanisms that issue warnings for identified phishing websites (Bell and Komisarczuk, 2020). Various browsers rely on distinct feed sources for their blacklists. For instance, Safari, Google Chrome, and Firefox utilize the Google Safe Browsing (GSB) blacklist, Edge deploys Microsoft SmartScreen's (MS SmartScreen) blacklist, while other browsers may source blacklists from a combination of APEs. Despite serving as critical first-line defenses against phishing websites (Oest et al., 2019), these blacklists operate as Blackbox, employing security crawlers to automate detection. However, the intrinsic divergence of this detection method from normal user requests has spurred the development of evasion techniques,

^a <https://orcid.org/0009-0007-4342-6676>

^b <https://orcid.org/0000-0002-5748-155X>

allowing phishing websites to evade detection and continue luring unsuspecting users. Consequently, an analysis of the potential shortcomings in current blacklists can enhance the effectiveness of APEs' detection strategies and better shield users from phishing attacks.

In this paper, we explore innovative and practical attacks against current anti-phishing blacklists to shed light on and rectify potential weaknesses in existing protective mechanisms. We propose and scrutinize advanced cloaking techniques hitherto overlooked, uncovering significant potential deficiencies in APEs that may be exploited by attackers. The main contributions of this paper include:

- Comprehensive investigation of current cloaking techniques used by phishing websites, suggesting potential improvements for APEs from an attacker's perspective.
- Proposition and exploration of novel cloaking techniques aimed at detecting APEs. We conduct preliminary experiments to understand APE responses to our techniques, thereby revealing several uncovered flaws in popular APEs such as GSB, MS SmartScreen, APWG, and ESET.
- Presentation of a scheme and design of a framework, supplemented by an experiment evaluating our cloaking techniques against current anti-phishing blacklists, highlighting potential threats to Internet users.

2 RELATED WORKS

Understanding the prevalent cloaking techniques against anti-phishing blacklists and potential strategies, attackers might exploit to avoid APE detection is crucial. Earlier studies have categorized the cloaking techniques employed by phishing sites as: server-side cloaking, client-side cloaking, and fingerprinting. These schemes are implementation specific and subdivide these categories further into User Interaction, Fingerprinting, and Bot behavior (Zhang et al., 2021). In this paper, we focus on server-side and client-side cloaking techniques, bot behavior, and fingerprinting as client-side techniques encompass user interaction techniques.

Server-side cloaking leverages attributes of HTTP Requests such as Hostname, Referrer, User Agent, Cookie, and client IP addresses. These attributes allow phishers to distinguish between requests from known anti-phishing entities or security crawlers and genuine user requests. Phishers utilize these attributes to gain additional information about users, enabling

them to filter or redirect requests server-side to evade detection (Oest et al., 2018; Oest et al., 2019; Oest et al., 2020a).

Client-side cloaking entails JavaScript code to verify user interaction where verification methods include sending pop-up alerts, tracking mouse behavior, employing reCAPTCHA, or requiring session initiation or apply obfuscation methods, thereby thwarting static code analysis by APEs. Such methods aid in filtering out APE detections and luring genuine users (Maroofi et al., 2020; Oest et al., 2020b; Zhang et al., 2021).

Bot behavior identification metrics encompass timing and randomization (Zhang et al., 2021). Phishers may delay the display of malicious content, leveraging the fact that bots or human inspectors may not stay on the page long enough for the content to load. Similarly, discrepancies in JavaScript execution times can indicate bot activity (Acharya and Vadrevu, 2021). Randomization techniques, like employing a random number generator to determine content display, can further evade detection by APEs (Zhang et al., 2021).

Basic fingerprinting techniques utilize HTTP requests and JavaScript to extract parameters like User Agent, Referrer, and Cookie, enabling phishers to identify APEs. Advanced fingerprinting techniques deploy web APIs such as Canvas and WebGL to collect unique client fingerprints. These sophisticated methods facilitate more precise identification, enhancing phishers' evasion capabilities (Acharya and Vadrevu, 2021).

Our research identified two principal strategies employed to elude detection by APEs: identification of authentic human users and discernment of APEs. In order to comprehend state-of-the-art cloaking methodologies employed by contemporary phishing sites, we examined multiple PhaaS providers' sites to investigate their functionalities and potential evasion capabilities. Our observations revealed the application of a new function, **Virtual Machine Detection**, which potentially aids in evasion. Phishing sites use numerous features to detect virtual machines, including CPU core count, memory size, WebGL renderer and User Agent information, Webdriver, the quantity of installed browser plugins, and language information sourced from WebGL or JavaScript. We questioned whether this information could effectively identify APEs. Moreover, we delved into potential browser and protocol level features that could enable precise fingerprinting or assist in APE identification. Recent privacy research highlighted two significant tracking techniques: **Cache-based** tracking and the use of **Accept-CH** headers in HTTP

requests (Ali et al., 2023). We questioned if these tracking techniques could offer additional avenues for phishing sites to identify APEs. Subsequently, we conducted experiments to assess their potential in evading anti-phishing detection and evaluated the efficacy of these innovative techniques against anti-phishing blacklists.

3 PRELIMINARIES

In our investigation, we introduce three novel techniques and execute three sets of preliminary experiments to comprehend the responses of APEs to our methods. To evaluate this, we select security crawlers like GSB and MS SmartScreen, which supply blacklist feeds to prevalent browsers including Chrome, Safari, Firefox, and Edge, owing to their extensive user base. We also incorporate two critical security crawlers, APWG and ESET, given their importance in the anti-phishing ecosystem. PhishTank is excluded as it is no longer accepting registrations. The aim of these preliminary experiments is to evaluate the feasibility of using the proposed techniques to elude APE detection, and expose the potential threats these security crawlers present to APEs.

3.1 Virtual Machine Detection

3.1.1 Background

Virtual machine detection, frequently utilized by ransomware, trojans, and other malicious software to differentiate user environments, is a growing concern in the cybersecurity domain. Such software discerns whether it's operating within a virtual environment, thus allowing it to modify its behavior, evade detection, and target real users. Phishing websites now exploit this strategy, gathering environmental fingerprints to identify virtual machines. They leverage Web APIs like WebGL and Canvas, as well as JavaScript Objects, to collect visitor-specific data such as screen rendering details and color depth. Dark web PhaaS providers have reportedly integrated virtual machine detection tools into their services, enabling them to filter genuine users. As indicated by (Lin et al., 2022), numerous active phishing websites are employing these techniques to harvest browser fingerprint data—an increasingly prevalent practice. This harvested data enables attackers to circumvent Two Factor Authentication (2FA) and misuse stolen credentials. Our initial experiment aimed to investigate if techniques related to virtual machine detection could assist phishers in distinguishing between nor-

mal user requests and security crawlers, thereby potentially evading detection strategies of APEs.

3.1.2 Experimental Set Up

In our survey of existing browser-side virtual machine detection techniques and prevalent user device information, we selected features such as memory size, CPU core count, screen color depth/width/height, User-Agent headers, and WebGL-extracted data for virtual machine detection and subsequent feature dataset creation. Figure 1 presents an overview of the design aspects in our experiment. Initially, we integrated JavaScript code that gathered feature information into our experimental website pages. These sites were deployed on cloud servers and classified into four groups. The URLs of each group were then individually submitted to GSB, MS SmartScreen, APWG, and ESET, which are four prominent APEs, to prompt anti-phishing crawler requests. The JavaScript code executed in APEs' security crawler browsers or during manual inspection, amassing pertinent data and returning it to the experimental website servers to log features matching our dataset. The backend of these sites processed the data for storage in a database. Upon data collection completion, we analyzed these records and the additional database-stored data to evaluate the feasibility of using browser-side virtual machine detection techniques to evade detection from APEs.

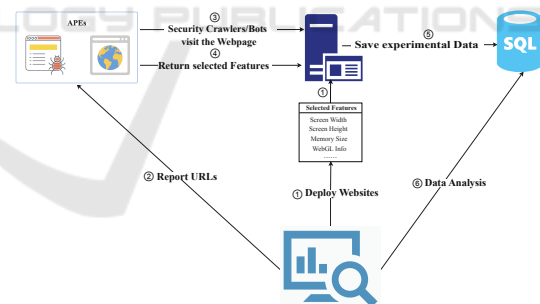


Figure 1: The Experiment Design of Virtual Machine Detection.

3.1.3 Observed Results

GSB. Our experiment reveals distinct differences between the characteristics of GSB's crawler and those of typical users, particularly in memory size, number of browser plugins installed, and screen dimensions. These findings suggest that virtual machine detection techniques could potentially bypass GSB's existing phishing detection mechanism. Analysis of the crawler's feature data sent to the server, upon removing duplicates, showed few unique values in the Renderer feature acquired through WebGL. Notably,

we observed consistent Renderer information from crawlers across geographically dispersed phishing website servers. This concurs with previous findings on utilizing WebGL to generate hidden images and compute image hashes (Acharya and Vadrevu, 2021), albeit our direct method of acquiring renderer information via WebGL differs from their schemes.

MS SmartScreen. MS SmartScreen exhibited similar traits to GSB, with clear differences in memory size, screen dimensions, and WebGL-derived Renderer features compared to typical user profiles.

APWG. Crawlers from APWG displayed significant variations in screen dimensions, number of browser plugins, and WebGL-obtained Renderer features, compared to an ordinary user. After removing duplicates, we noted a paucity of unique renderer information from APWG as well.

ESET. We observed substantial disparities in the number of CPU cores, memory size, User-Agent headers, number of browser plugins, and Renderer features obtained via WebGL between ESET and ordinary users. ESET also demonstrated a less number of unique renderer values.

3.2 Cache-based Mechanism

3.2.1 Background

Web cache mechanisms aim to alleviate server request loads, diminish bandwidth utilization, and enhance system performance. Prominent web caching technologies encompass database cache, CDN cache, DNS cache, proxy server cache, and browser cache. The latter involves a browser storing resources obtained from HTTP requests locally, enabling direct access to webpages via cache during subsequent visits and precluding repeated server requests. Typically, browser caching strategies, implemented via HTTP header settings, consist of local cache and negotiated cache. The former relies on resource expiration time parameters (i.e., "expires" and "cache-control"), while the latter hinges on resource modification status (i.e., "last-modified," "If-modified-since," "Etag," and "If-None-Match"). Both strategies result in client-side, rather than server-side, resource loading. Essentially, when a browser accesses a resource, if the local cache matches, cached content is used; otherwise, a server request verifies a negotiated cache match. Consequently, we suggested a technique to determine identity based on security crawler access behavior, and conducted preliminary experiments to assess whether APEs demonstrate cache behavior consistent with regular browser users when visiting phishing websites.

3.2.2 Experimental Set UP

For this experiment, we constructed a webpage (route: /index) that incorporated an image (route: /img.jpg), with the image's HTTP response header featuring a Cache-Control header (max-age=86400), mandating a 24-hour browser cache. To enforce this caching behavior, we designed a redirection that compelled multiple visits to our webpage. Clients were redirected to the index page, with varying parameters conveyed in the URL via the HTML Meta tag, and redirection was capped at three instances. Consequently, under default cache strategies employed by Chrome, Firefox, Edge, and Safari, clients would initiate the following request sequence: /index->/img.jpg (Cached)->/index?A->/index?B->/index?C, where, A, B, and C represent query strings in the URL.

Our experimental websites were deployed on a cloud server and the URLs were categorized into four groups, each submitted separately to the four major APEs: GSB, MS SmartScreen, APWG, and ESET. The backend of these websites recorded each request for /index and /img.jpg. During post data collections, we analyzed requests with URL query strings named NULL, A, B, and C, along with /img.jpg requests, treating them as one group. By assessing the number of /img.jpg requests within this group, we could gauge the likelihood of using a browser caching mechanism to evade detection from APEs.

Importantly, we observed during our experiment that some APEs tended to shift IPs upon redirection, hindering our ability to trace redirections and tally accesses initiated by a single security crawler. We thus updated our method with a unique design (Figure 2) to track specific security crawler requests using a sequence *RequestGroupID*: [RandomID-1, RandomID-2, ..., RandomID-N]. Initially, the first RequestGroupID and associated RandomID are generated and the server responds with the RandomID, which is subsequently used as the URL's query string for the next redirection. Each client visit to the index page generates a new RandomID related to the RequestGroupID until the preset redirection limit is reached. The initial RandomID is used as the URL parameter for all subsequent image requests, enabling us to count the number of image file requests containing the same RandomID under the same RequestGroupID in the URL.

3.2.3 Observed Results

GSB. Our experiment revealed that GSB's anti-phishing crawlers exhibit distinct behaviours when accessing static website resources compared to regular user browsers, initiating new requests for re-

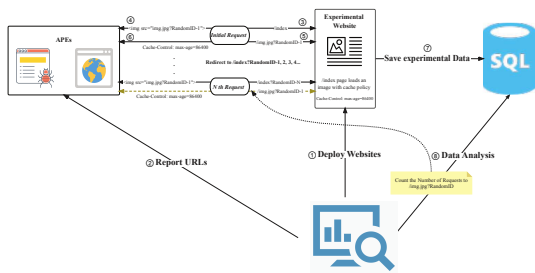


Figure 2: The Experiment Design of Cache-based Mechanism.

sources that should have been cached. Hence, cache mechanisms might effectively identify GSB crawlers and could potentially bypass the current GSB phishing detection process.

MS SmartScreen. The experiment data suggest that MS SmartScreen crawlers’ behaviour towards cached resources is identical to that of regular users, implying cache mechanisms cannot identify these crawlers.

APWG. In this experiment, APWG demonstrated adherence to cache policies on some occasions—requesting `/img.jpg` only once upon the initial visit and not on subsequent redirected requests. However, instances of non-adherence—requesting `/img.jpg` multiple times within the same request set—were also observed. APWG’s anti-phishing crawlers exhibited varied behaviours, leading to the discovery of some issues:

- In some of APWG’s request data, we found that APWG’s anti-phishing crawlers could not effectively execute all redirect requests, sometimes failing to execute the third redirect, either only requesting the `/index` page without parameters, or only executing the `/index?A` page after the first redirect, or at most requesting the `/index?B` page after the second redirect, which is also significantly different from the behavior of normal users.
- APWG’s crawlers invariably disregarded page refresh delay times set in HTML, executing all redirects to the `/index` page and `/img.jpg` requests far quicker than the specified refresh time.

ESET. Similar to APWG, ESET showed partial compliance with browser cache policies during the experiment. Notably, the behaviours of their anti-phishing crawlers also exposed the same issues as APWG:

- In all of ESET’s request data, we found that ESET’s anti-phishing crawlers could not effectively execute all redirect requests, these anti-phishing crawlers always failed to execute the third redirect, either only requesting the `/index` page without parameters, or only executing the `/index?A` page after the first redirect, or at most requesting the `/index?B` page after the second redi-

rect, which is also significantly different from the behavior of normal users.

- Like APWG’s crawlers, ESET’s ignored HTML-set page refresh delays, executing all possible redirect `/index` page and `/img.jpg` requests much faster than the allotted refresh time.

Therefore, we postulate that cache mechanisms can partially identify anti-phishing crawlers from APWG and ESET. This experiment also unveiled additional abnormal behaviours by APWG and ESET when visiting phishing websites, potentially exploitable alongside cache mechanisms.

3.3 Client Hints in HTTP Header

3.3.1 Background

The use of Client Hints in the HTTP header, akin to browser cache, is a technique employed to optimize performance and enhance user experience. This method allows clients to actively convey certain characteristics to the server, such as device type, operating system, and network information. Consequently, the server can deliver personalized content based on these characteristics, fostering improved browsing experiences. Upon a user’s webpage visit, the server can issue an `Accept-CH` request to the browser, soliciting Client Hints. The client reciprocates by returning the requested data in the HTTP header. Notwithstanding the fact that `Accept-CH` lacks universal browser compatibility, it is supported by major desktop browsers, including Chrome, Edge, and Opera. Interestingly, recent cybersecurity and privacy research suggests potential user tracking via this vector (Ali et al., 2023). Thus, we conducted a preliminary experiment to ascertain whether the Client Hints returned by security crawlers differ from those of regular users, thereby exploring the feasibility of identifying APEs using Client Hints.

3.3.2 Experimental Set Up

Figure 3 depicts the design of this experiment, wherein we crafted a single-page website featuring an image. When `/index` was accessed from a regular user’s browser, it would automatically solicit `/img.jpg`. The response returned encompassed the `Accept-CH` directive, incorporating all hints that the server sought. Desktop browsers compatible with `Accept-CH` appended some or all of the listed client hint headers in subsequent requests. We launched these websites on a cloud server, categorizing the URLs into four groups for submission to the four key APEs: GSB, MS SmartScreen, APWG, and ESET.

The backend of these websites logged all acquired data, including HTTP headers. Following data collection, we scrutinized the HTTP headers in requests related to the Client Hints we outlined in Accept-CH and contrasted these with data from real user browsers supporting Client Hints. This facilitated an assessment of the potential for utilizing Client Hints in request headers to evade detection by security crawlers.

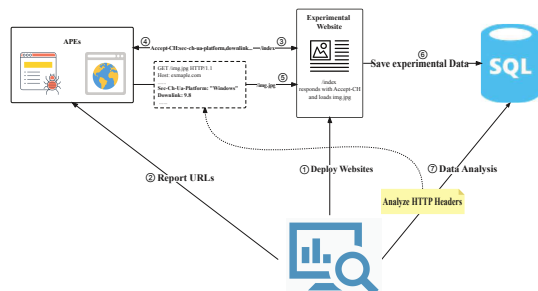


Figure 3: The Experiment Design of Client Hints.

3.3.3 Observed Results

GSB. Our experiment reveals that GSB’s anti-phishing crawlers fully support Accept-CH in relation to browser type and version. However, these crawlers inconsistently adhere to the Accept-CH directive, often neglecting to incorporate all specified Client Hints in subsequent requests. A noticeable omission is all User Agent Client Hints (UA-CH) beginning with Sec-CH-UA. This discrepancy results in a near 50% reduction in Client Hints in the HTTP request headers compared to normal users under identical Accept-CH specifications. This significant difference suggests that Client Hints could serve as an effective identifier for GSB, potentially offering a method to bypass current phishing detection mechanisms.

MS SmartScreen. The performance of MS SmartScreen’s anti-phishing crawlers closely mirrors that of normal users in terms of Client Hints’ content and items in HTTP request headers. Consequently, it is not possible to distinguish MS SmartScreen based solely on Client Hints.

APWG. A majority of APWG’s anti-phishing crawlers support Client Hints, with a mere 2% of requests originating from unsupported browsers. The compliance of these crawlers with Accept-CH closely aligns with that of GSB, primarily ignoring UA-CH headers beginning with Sec-CH-UA. Interestingly, a subset of APWG’s crawlers return NULL values for these headers, differing from typical user behavior. This information suggests that Client Hints can serve as an effective identifier for APWG, potentially offering a method to bypass its current phishing detection mechanisms.

ESET. Our study of ESET reveals a diverse array of clients with varying degrees of Client Hints support. Approximately 10% of these clients exhibit behavior consistent with typical users in their adherence to Accept-CH. The remaining majority display behavior similar to GSB or APWG, either ignoring Sec-CH-UA starting UA-CH headers or returning NULL values. This suggests that Client Hints can be effectively used to identify ESET crawlers, providing a potential method to bypass their existing phishing detection mechanisms.

3.4 Summary

Upon conducting preliminary investigations into Virtual Machine Detection, Cache-based Mechanisms, and the use of Client Hints from HTTP requests, we posit that these methods offer innovative avenues for APE identification and evasion, to varying degrees. This analysis also uncovers the potential vulnerabilities these four APEs might encounter. Here, we enumerate the primary risks:

- GSB crawlers clearly indicate virtualization technology usage. Through the use of Web APIs and JavaScript code, potential attackers may identify specific browser and device features, thereby distinguishing GSB crawlers and evading GSB detection. The range of Renderers obtained via WebGL for GSB crawlers is significantly limited, permitting attackers to acquire this information, construct a feature dataset, and subsequently identify GSB crawlers. Lastly, GSB crawlers do not adhere to browser caching policies and do not fully adhere to the Accept-CH directive of Client Hints when visiting websites. These distinctive behaviors could be exploited by attackers to filter GSB requests and dodge detection.
- MS SmartScreen crawlers display user-like behavior when visiting phishing websites, posing challenges for identification and evasion through the cache-based mechanism or Client Hints. Despite this, MS SmartScreen does reveal limitations in handling virtualization feature detection and Renderer diversity. These shortcomings provide potential avenues for detection evasion.
- APWG crawlers emulate real-user behaviors when accessing phishing websites, including partial adherence to browser cache behaviors. This similarity confers a degree of resistance to cache-based mechanisms. However, APWG’s defense against virtual machine detection and the use of Client Hints is inadequate. Additionally, its handling of web page redirection and delayed

page refresh diverges from typical user behavior. These disparities allow attackers to identify APWG crawlers and evade detection by employing multiple redirects or delayed webpage loading.

- ESET has a more obvious use of virtualization technology, with similar resilience as APWG in virtual machine detection techniques and use of Client Hints in HTTP requests. It also has inconsistent behaviors in resource consumption activities (redirection and delayed page refresh) compared to the behavior of real user. Attackers can exploit these flaws to evade detection from ESET.

Table 1 presents a comparative analysis of the effectiveness of three potential cloaking techniques against APEs. To further evaluate these methods against anti-phishing blacklists and to assess their potential risks to Internet users, we have designed an additional experiment to observe the real blacklisting behavior of mainstream browsers against these techniques.

Table 1: The comparison of effectiveness of potential cloaks against APEs

Potential Cloaks	APEs			
	GSB	MS SmartScreen	APWG	ESET
Virtual Machine Detection	✓	✓	✓	✓
Cache-based Mechanism	✓	✗	↔	↔
Client Hints	✓	✗	✓	✓

✓ -> More effective ↔ -> Less effective ✗ -> Not effective

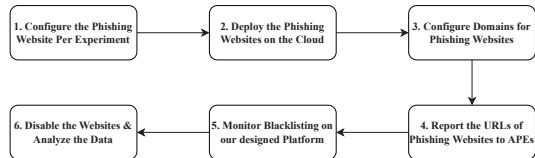


Figure 4: The overview process of the experiment.

4 PROPOSED SCHEME

Figure 4 offers an overview of our experimental procedure. Initially, we developed corresponding simulated phishing websites incorporating the aforementioned techniques. 20 such websites including three experimental groups and one control were deployed on the cloud, featuring independent domain names and unique random paths to circumvent any effects of unrelated internet scanning or accidental crawling. Given that no primary browsers publicly acknowledge the use of APWG and ESET for their blacklist

data, we focused on GSB and MS SmartScreen whose blacklists extend to dominant browsers such as Safari, Chrome, Firefox, and Edge. This allowed our results to reflect the broad impact of cloaking techniques on internet users.

Subsequently, we scrutinized the blacklisting behavior on Windows desktop browsers and macOS platforms over a 72-hour period. In line with past research citing an average phishing website lifespan of around 21 hours (Oest et al., 2020a), we deemed this timeframe sufficient. Post monitoring, we discontinued these simulated websites and analysed the obtained data.

To automate our process, we engineered a system that chiefly embraces a BS/CS hybrid architecture, featuring five modules: Control Node, Monitoring Node, Task Manager, Cloud Server Manager, and Visualized Management Platform (illustrated in Figure 5). This system enables automated deployment of simulated phishing websites, scheduled monitoring of blacklisting behavior, and data visualization. Operating in a distributed fashion, it offers scalability, flexibility, and automated data and resource adjustment.

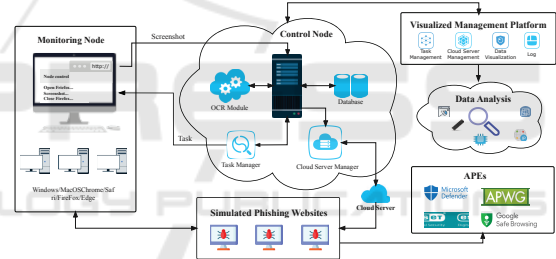


Figure 5: The modules and workflow of proposed system.

5 EXPERIMENT SETUP

We conducted an experiment encompassing three experimental and one control group to evaluate the efficacy of three potential cloaking techniques against anti-phishing blacklists: virtual machine detection, the cache-based mechanism, and the application of Client Hints in HTTP headers. Each experimental group utilized one technique to simulate five identical phishing websites.

Experiment A – Baseline: The websites in Experiment A imitated Facebook’s login page without any cloaking techniques. This group served as a blacklisting effectiveness baseline in comparison to the other groups that utilized cloaking techniques.

Experiment B – Virtual Machine Detection: This experiment’s websites used virtual machine detection techniques. Leveraging the limited diversity of

the WebGL Renderer noted in the preliminary experiment, we processed previously collected WebGL Renderer data to compile a feature dataset. These features, along with other detection features from the dataset, were integrated into the phishing website using the GoFrame framework. The backend received various data from crawlers through frontend JavaScript code and compared these features with the dataset. Requests initially landing on a benign page were classified based on their match with the dataset, and accordingly, directed to either a benign or phishing webpage.

Experiment C – Cache-based Mechanism: This group’s websites employed a Cache-based Mechanism. Based on crawler behavior observed in the preliminary experiment, we implemented two redirects on the websites with an image on the page. Similar to the preliminary experiment method, we identified requests initiated by the same crawler and logged the image request frequency on the backend. After an initial landing on a benign webpage, the image request count determined the subsequent redirection, either to a benign or phishing webpage.

Experiment D – Utilization of Client Hints: Based on preliminary experiment data, we developed a set of rules for Client Hints in HTTP headers and implemented them in the backend of all Experiment D websites. Clients initially accessed a benign landing page, receiving a server response with Accept-CH, specifying required client hints for the HTTP headers. If no match was found with our rules set, the client was redirected to a phishing webpage. Conversely, a match identified the client as a security crawler, redirecting it to a benign webpage to avoid detection.

6 RESULT AND ANALYSIS

For 72 hours, our monitoring platform observed six desktop browsers, each accessing five URLs per experiment, culminating in a total of 30 monitored URLs per experiment. Figure 6 reveals that all phishing websites from Experiment A were blacklisted within 12 hours by Chrome and Firefox on Windows, and by Chrome, Firefox, and Safari on macOS. In contrast, three websites from Experiment A remained unblacklisted by Edge on Windows at the end of our observation period. Remarkably, the simulated phishing websites in Experiments B and D, employing virtual machine detection techniques and HTTP Client Hints headers respectively, evaded blacklisting by all browsers throughout the experiment. As depicted in Figure 7, only three websites from Experiment C, utilizing a cache-based mechanism, were blacklisted by

Edge on Windows after 7 to 7.5 hours; all others remained unblacklisted.

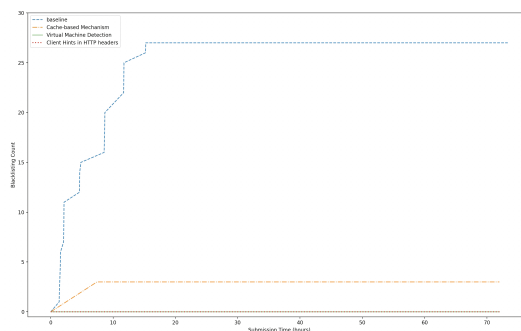


Figure 6: Number of blacklisted websites in all browsers of test over the submission time.

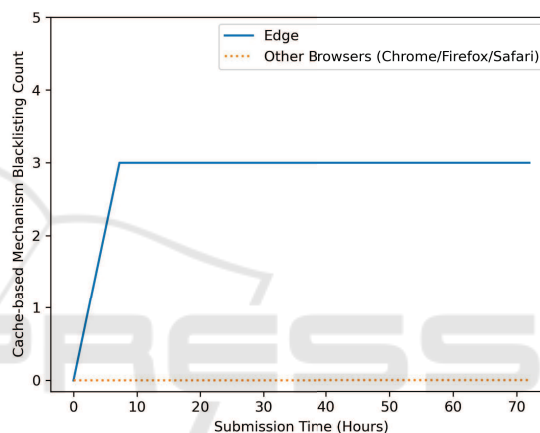


Figure 7: Number of blacklisted websites in all browsers of Cache-based Mechanism over the submission time.

Our findings suggest that both virtual machine detection techniques and the use of Client Hints in HTTP headers can effectively evade detection from GSB and MS SmartScreen crawlers, thereby avoiding blacklisting and potentially exposing Internet users to risk. Similarly, the cache-based mechanism demonstrated substantial evasion capabilities against GSB crawlers and to a lesser extent against MS SmartScreen, posing potential risks to users.

7 CONCLUSION

Anti-phishing blacklists constitute a critical mechanism for safeguarding Internet users from phishing websites and are extensively integrated into a multitude of browsers. This paper explores contemporary cloaking techniques deployed by phishing websites, shedding light on vulnerabilities and potential hazards inherent in current anti-phishing blacklists. We propose three novel cloaking strategies that have adeptly

circumvented detection from mainstream APEs and assess their efficacy using a bespoke framework. This paper also underlines potential strategies for bypassing APE detection. Future studies could further delve into these avenues, investigating real-world application of these cloaking techniques or conducting large-scale evaluations of these methodologies.

REFERENCES

- Acharya, B. and Vadrevu, P. (2021). Phishprint: Evading phishing detection crawlers by prior profiling. In *USENIX Security Symposium*, pages 3775–3792.
- Al-Ahmadi, S., Alotaibi, A., and Alsaleh, O. (2022). Pdgan: Phishing detection with generative adversarial networks. *IEEE Access*, 10:42459–42468. <https://doi.org/10.1109/ACCESS.2022.3168235>.
- Alabdan, R. (2020). Phishing attacks survey: Types, vectors, and technical approaches. *Future Internet*, 12(10). <https://doi.org/10.3390/fi12100168>.
- Ali, M. M., Chitale, B., Ghasemisharif, M., Kanich, C., Nikiforakis, N., and Polakis, J. (2023). Navigating murky waters: Automated browser feature testing for uncovering tracking vectors. In *Proceedings 2023 Network and Distributed System Security Symposium. Network and Distributed System Security Symposium, San Diego, CA, USA*. <https://doi.org/10.14722/ndss.2023.24072>.
- APWG (2017). Phishing activity trends report, 4th quarter 2017. https://docs.apwg.org/reports/apwg_trends_report_q4_2017.pdf.
- APWG (2022). Phishing activity trends report, 4th quarter 2021. https://docs.apwg.org/reports/apwg_trends_report_q4_2021.pdf.
- Bell, S. and Komisarczuk, P. (2020). An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank. In *Proceedings of the Australasian Computer Science Week Multiconference, ACSW '20, New York, NY, USA*. Association for Computing Machinery. <https://doi.org/10.1145/3373017.3373020>.
- Freeze, D. (2018). Cybercrime To Cost The World \$10.5 Trillion Annually By 2025. <https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021/>.
- Gupta, S., Singhal, A., and Kapoor, A. (2016). A literature survey on social engineering attacks: Phishing attack. In *2016 International Conference on Computing, Communication and Automation (ICCCA)*, pages 537–540. <https://doi.org/10.1109/CCAA.2016.7813778>.
- Han, X., Kheir, N., and Balzarotti, D. (2016). PhishEye: Live Monitoring of Sandboxed Phishing Kits. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS '16*, pages 1402–1413, New York, NY, USA. Association for Computing Machinery. <https://dl.acm.org/doi/10.1145/2976749.2978330>.
- Lin, X., Ilija, P., Solanki, S., and Polakis, J. (2022). Phish in sheep's clothing: Exploring the authentication pitfalls of browser fingerprinting. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 1651–1668.
- Maroofi, S., Korczyński, M., and Duda, A. (2020). Are you human? resilience of phishing detection to evasion techniques based on human verification. In *Proceedings of the ACM Internet Measurement Conference, IMC '20*, page 78–86, New York, NY, USA. Association for Computing Machinery. <https://doi.org/10.1145/3419394.3423632>.
- Oest, A., Safaei, Y., Doupé, A., Ahn, G.-J., Wardman, B., and Tyers, K. (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)*, pages 1344–1361. <https://doi.org/10.1109/SP.2019.00049>.
- Oest, A., Safaei, Y., Zhang, P., Wardman, B., Tyers, K., Shoshitaishvili, Y., Doupé, A., and Ahn, G. (2020a). Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *Proceedings of the 29th USENIX Security Symposium, Proceedings of the 29th USENIX Security Symposium*, pages 379–396. USENIX Association.
- Oest, A., Safaei, Y., Doupé, A., Ahn, G.-J., Wardman, B., and Warner, G. (2018). Inside a phisher's mind: Understanding the anti-phishing ecosystem through phishing kit analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12. <https://doi.org/10.1109/ECRIME.2018.8376206>.
- Oest, A., Zhang, P., Wardman, B., Nunes, E., Burgis, J., Zand, A., Thomas, K., Doupé, A., and Ahn, G. (2020b). Sunrise to sunset: Analyzing the end-to-end life cycle and effectiveness of phishing attacks at scale. In *Proceedings of the 29th USENIX Security Symposium, Proceedings of the 29th USENIX Security Symposium*, pages 361–377. USENIX Association.
- Pujara, P. and Chaudhari, M. (2018). Phishing website detection using machine learning: A review. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, 3(7):395–399.
- World Economic Forum (2020). Partnership against Cybercrime. https://www3.weforum.org/docs/WEF_Partnership_against_Cybercrime_report_2020.pdf.
- Zhang, P., Oest, A., Cho, H., Sun, Z., Johnson, R., Wardman, B., Sarker, S., Kapravelos, A., Bao, T., Wang, R., Shoshitaishvili, Y., Doupé, A., and Ahn, G.-J. (2021). Crawlphish: Large-scale analysis of client-side cloaking techniques in phishing. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 1109–1124. <https://doi.org/10.1109/SP40001.2021.00021>.