What You See Is Not What You Get: Measuring SVG Abuse on Suspicious Landing Pages

Bence Beke-Szabó*
bence.beke-szabo@kuleuven.be
DistriNet, KU Leuven
Leuven, Belgium

Daan Vansteenhuyse*
daan.vansteenhuyse@kuleuven.be
DistriNet, KU Leuven
Leuven, Belgium

Lieven Desmet
lieven.desmet@kuleuven.be
DistriNet, KU Leuven
Leuven, Belgium

Abstract

Phishing and other scams remain a persistent threat, and their operators increasingly rely on cloaking to hide malicious content from the scanners defenders use to detect them. Scalable Vector Graphics (SVG) has been identified in recent industry reporting as a rising vehicle for such evasion, specifically as email attachments. An SVG is a document that can embed scripts, inline encoded resources, and reference external content, yet security tooling often treats it as a static image. Despite this attention, the actual prevalence of these techniques on suspicious web pages has not been measured. This paper presents a first measurement testing whether the SVG evasion techniques documented in email-delivered attacks also appear in the inline SVGs served on suspicious landing pages, using a corpus of citizen-reported suspicious pages compared against a benign Tranco baseline.

We find that the active-content techniques most associated with malicious SVG, such as embedded scripts, are almost entirely absent from served landing pages and no more common in the suspicious corpus than in the benign one. Suspicious pages rely more on embedded and linked imagery, mostly loaded cross-origin, whereas popular pages load from the same origin. We confirm these observations by manually looking into SVGs, finding open redirects, fake Captchas and an entire scam embedded in a single SVG. Our findings suggest that effort spent scanning landing-page SVGs for active content is largely misdirected, and that inspection is better aimed at how SVGs embed and inline resources, which could be used as a feature for phishing detectors.

CCS Concepts

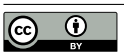
• **Security and privacy → Web application security**.

Keywords

phishing, scalable vector graphics, cloaking

ACM Reference Format:
Bence Beke-Szabó, Daan Vansteenhuyse, and Lieven Desmet. 2026. What You See Is Not What You Get: Measuring SVG Abuse on Suspicious Landing Pages. In *11th Workshop on Traffic Measurements for Cybersecurity (WTMC '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3846375.3849095

*Both authors contributed equally to this research.



WTMC '26, The Hague, Netherlands

ACM ISBN 979-8-4007-3017-7/2026/11
https://doi.org/10.1145/3846375.3849095

1 Introduction

Phishing and other malicious campaigns remain among the most persistent forms of cybercrime, luring victims to attacker-controlled pages that impersonate trusted brands and services. Defenders counter these campaigns at scale with automated crawlers and scanners while attackers try to avoid detection by employing *cloaking*: evasion techniques whose purpose is to hide malicious content from automated analysis while still presenting it to the human victim [1, 31, 35, 36]. Knowing which techniques attackers actually use, and how widely, is essential for directing detection effort where it matters.

1.1 SVG as a vehicle for malicious content

Scalable Vector Graphics (SVG) is an XML-based image format widely used on the modern Web for logos, icons and illustrations. Unlike raster formats, an SVG is a document: it can reference external resources, inline base64-encoded content, embed scripts, and even wrap entire fragments of HTML via `<foreignObject>`. This expressiveness makes SVG a powerful tool for web developers but also attracts malicious intent. A wave of 2025 industry reports, documenting SVG abuse in email-delivered attacks, noted that security tooling frequently treats SVG as a static image and does not inspect its contents [6, 7, 32, 34]. In these attacks, SVG is delivered as an attachment in an email which, on opening, shows the phishing payload or redirects to a malicious page. To the best of our knowledge, however, there has been no measurement of the prevalence of these techniques within the SVGs served on suspicious *landing pages* themselves, nor any comparison against a benign baseline that would establish whether such usage is in fact distinctive.

1.2 Our contribution

We present a measurement study of SVG usage on suspicious pages, using a corpus of sites drawn from citizen-reported suspicious messages. Since these reports include not only phishing but also spam and other suspicious content, our corpus offers a broader and more representative view of what ordinary users encounter than feed-based phishing datasets alone [33]. We characterize the SVGs served on these pages and compare them against a benign baseline, crawled using the same browser configuration, from the Tranco top-10k list, capturing the rendered DOM after JavaScript execution in both cases.

Our measurements yield two main findings. First, active-content evasion in SVG (embedded scripts, CDATA-wrapped payloads, `<foreignObject>`) is almost entirely absent from served landing pages, in both the suspicious and the benign corpus, indicating that the SVG threat highlighted in industry reporting is a property of email delivery rather than of the landing page. Second, we observe

a clear difference in how resources are loaded, with the suspicious dataset being 30 times more likely to contain an embedded image (31.8% vs. 0.9%), which in turn is often loaded from a different origin whereas popular websites mostly load images from the same origin.

2 Context

In order to detect phishing websites, a variety of scanners exist, both static [16, 20, 24, 29] and dynamic [11, 19, 30], which analyze the content for signs of malicious activity. Others, particularly reference-based phishing detectors, rely on visual identifiers such as brand logos or investigate screenshots [14, 16, 18–20]. Such visual models, however, are costly and time-consuming to run on a large scale [15]. More practical deployments, therefore, resort to analyzing the delivered DOM [2, 28, 29]. By parsing the HTML, such DOM analyzers extract valuable features.

These DOM analyzers, though, are especially prone to client-side cloaking techniques. These are strategies cybercriminals employ to hide themselves from anti-phishing scanners. On the server-side, these techniques can be used to redirect known scanners to a benign website based on e.g. IP address or User-Agent [10, 13]. Client-side techniques are used to hide or obfuscate the malicious payload from benign entities. Examples of these techniques are misusing Captchas [21, 31], timeouts to stall execution [4] or obfuscated JavaScript to thwart detectors [25].

3 SVG Abuse Techniques

One understudied cloaking technique makes use of the `<svg>` tag. SVGs are a way of delivering high-quality images defined as geometric entities which are expressed as nested XML tags and rendered directly in the browser. Scanning tools often classify these as simple images and skip parsing them for further references [6]. However, an SVG is not a simple image but a document embedded in the page: it can carry executable code, reference external resources, and even contain other markup. This mismatch between how SVG is treated and what it can actually do is what makes it an attractive vehicle for hiding content behind a tag perceived as a harmless image. We briefly describe how SVGs can be abused to hide malicious content.

Embedding active content. Beyond graphical primitives such as `<circle>` and `<line>`, an SVG may contain a `<script>` element, allowing arbitrary JavaScript to execute in the context of the page. If a scanner treats the SVG as an image, the content is never parsed. Scripts can also be included by wrapping it in a `<![CDATA[` section, which prevents the XML parser from interpreting the enclosed characters as markup.

Embedding entire pages. The `<foreignObject>` element allows arbitrary XHTML to be embedded inside an SVG and rendered as part of the page. In principle this permits an attacker to place an entire phishing page, credential form and all, inside what appears to be a single image file. A tool that inspects only the top-level HTML, or that stops at the SVG boundary, would never encounter the smuggled form.

Inlining content with base64. SVG references, such as image sources, may be expressed as `data:` URIs carrying base64-encoded content rather than as ordinary URLs. This lets an attacker inline a resource directly into the markup so that no external fetch occurs: there is no additional URL for a crawler to resolve, follow, or match against a blocklist. The same mechanism can inline an impersonated brand's logo, so that the logo renders pixel-perfectly for the victim while never appearing as a retrievable image resource that a logo- or resource-based detector could flag.

Fetching and linking external resources. Conversely, SVG can also reach outward. The `<image>` and `<feImage>` elements load external images, and the `<a>` element embeds hyperlinks directly within the graphic. An SVG can further nest additional `<svg>` elements, composing content from multiple sources. Each of these provides a channel through which an SVG perceived as a static image can pull in or point to further content that a naive scanner never examines.

4 Methodology

In this section, we elaborate on the datasets used, and the analysis performed to identify likely instances of cloaking.

4.1 Datasets

To measure the abuse of SVGs, we leverage two datasets: a crowdsourced dataset [33] and the Tranco toplist [12]. The crowdsourced dataset consists of websites flagged as suspicious by users. Most reports are made over email, often including spam and different types of scams. This dataset, therefore, gives a broader view on potential SVG misuses than merely phishing. Note that due to user reports not all content of this dataset is malicious; previous work identified the varying accuracy of crowdsourced data [5, 23, 33]. We crawled one week's worth of reports consisting of 129, 809 unique websites.

In order to make a comparison with benign web traffic, we resort to the Tranco list of which we took the top 10k where we assume, similar to previous work [16, 17, 19, 20], that popular websites are likely benign.

Both datasets were crawled using Chromium `new-headless` with a manually set User-Agent whilst also employing anti-bot evasion techniques such as manually setting WebGL parameters and implementing the permissions API. All redirects were followed, JavaScript was executed and after 30 seconds without network activity the final HTML was stored. From these all inline SVGs were extracted. 84.8% of the suspicious websites were reachable, compared to 75.8% of the benign set.

4.2 Features

To assess the prevalence of abused SVGs, we extract a set of structural and content features from every unique SVG in both corpora. Note that several features do not necessarily indicate abuse, they merely provide insights into how SVGs are used. All features are computed per SVG via case-insensitive pattern matching.

Active and obfuscated content. These features capture an SVG's capacity to carry executable or hidden payloads. *Has* `<script/>` flags an embedded script element, rendering SVG able to execute custom JavaScript. *Content wrapped in CDATA* detects `<![CDATA[` sections, which can be used to shield script or markup from XML parsing. *href contains base64* identifies `data:` URIs carrying base64-encoded content, a means of inlining resources without an external fetch making it harder for static scanners to analyze the delivered

content. *Has* `<foreignObject/>` detects the embedding of arbitrary XHTML inside an SVG.

Included elements. Additionally, we record some interesting elements that are included in the SVGs. Our selection is based on elements that can either load and link resources (`<a/>`, `<image/>`, `<use/>`, `<feImage/>`, `<style/>` and `<animate/>`) or create re-usable elements (`<defs/>`, `<svg/>`, `<symbol/>`).

5 Results

We first highlight the difference in SVG prevalence across both datasets, showing how SVGs are more widely used on popular websites. Second, we dive deeper into the extracted SVGs and analyze the features defined in the previous section.

5.1 Prevalence of SVGs

Table 1: Comparison between two datasets on SVG prevalence.

	User-reports	Tranco toplist
Crawled sites	129,809	10,000
Sites with an SVG	3,655	5,015
Total SVGs	131,791	377,712
Unique SVGs	9,245	71,805

Across both datasets, the number of websites with an SVG varies wildly (Table 1). Only 2.8% of websites from the suspicious dataset contain an SVG, but averaging 36 SVGs per website while 50.2% of the most popular sites have an SVG averaging 75 SVGs per page. This suggests that SVGs are mostly a feature of legitimate, professionally built sites; suspicious pages mostly do not use them at all.

Additionally, the majority of SVGs appear multiple times across the dataset. In the suspicious dataset, on average an SVG is repeated 14.3 times across all websites, whereas for popular websites this is lower with 5.3 repetitions. This is likely due to two things. Malicious campaigns often reuse code and deploy over multiple domains [4, 27] causing same SVGs to end up multiple times in our dataset. Secondly, the popular pages often reuse the same libraries or website structure (e.g. using WordPress) which under the hood use SVGs.

5.2 Comparison with Benign Baseline

Table 2 reports the prevalence of each feature described in Section 4.2 across the two corpora. The upper block covers active and obfuscated content; the lower block covers the included elements. To account for the large majority of duplicated SVGs, only unique ones were kept for this analysis. All values are per-SVG prevalence over the unique SVGs in each corpus.

Table 2: Prevalence of SVG features: user-reported suspicious pages vs. Tranco top 10k baseline.

Metric	User-reports (n=9,245)	Tranco toplist (n=71,805)
Has `<script/>`	0.01%	0.02%
Content wrapped in CDATA	0.00%	0.00%
href contains base64	0.89%	0.49%
Has `<foreignObject/>`	0.08%	0.07%
Has `<a/>`	0.23%	0.03%
Has `<image/>`	31.76%	0.89%
Has `<use/>`	62.55%	21.95%
Has `<feImage/>`	0.01%	0.00%
Has `<style/>`	1.36%	1.28%
Has `<animate/>`	0.10%	0.24%
Has `<defs/>`	10.27%	10.74%
Has nested `<svg/>`	0.08%	0.45%
Has `<symbol/>`	0.98%	1.63%

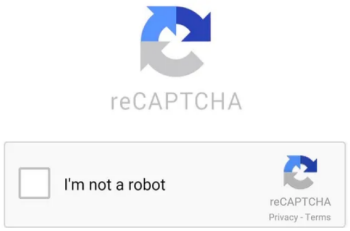


Figure 1: Example render of a malicious SVG mimicking a legitimate Captcha.

Listing 1: Simplified example of found SVG abuse in the wild where a Google Docs page hosts a reCAPTCHA image. If the image is clicked, the user is redirected to a malicious website exploiting an open redirect vulnerability in Google.

```
<svg>
   <image xlink:href="https://docs.google.com/[...]"/>
   <a rel="noreferrer" target="_blank"
         xlink:href="https://www.google.com/url?q=malicious.com">
   </a>
</svg>
```

The active-content features are rare in both corpora. Embedded `<script>` elements occur in 0.01% of reported SVGs and 0.02% of baseline SVGs, CDATA sections in neither, and `<foreignObject>` in under a tenth of a percent of each. A manual analysis of the SVG that used the `<script>` tag reveals that the one in the suspicious dataset embeds an entire investment scam, including the information-harvesting form, within a single SVG. In the Tranco dataset, scripts are often used for animations (4 cases) or are just empty (13 cases), likely leftovers from SVG-generating tools.

The only active-content feature that differs substantially is base64-encoded `href` content, present in 0.89% of reported SVGs against 0.49% of the baseline, roughly twice as common, though uncommon in absolute terms in both.

Anchors (`<a>`), though rare, are more common in the reported corpus (0.23% vs. 0.03%). 14 out of 21 suspicious SVGs with the

anchor element do so as a means to redirect users to a different scam page after clicking the element. The clickable element in these cases consisted of an image depicting a fake Google Captcha challenge. The redirect itself uses an open redirect vulnerability [22] where the effective scam is embedded into the query parameters of a benign looking link. In this case they use a `google.com` redirect after clicking on a fake Google Captcha. This might fool users into believing the benignity of the website after solving the fake challenge. An example of this SVG can be found in Listing 1 and Figure 1.

The included-element block is more varied. Embedded images (`<image>`) appear in 31.76% of reported SVGs but only 0.89% of baseline SVGs, and `<use>` references in 62.55% against 21.95%. The embedded images also differ in origin between the two datasets (Table 3). In the suspicious dataset, 86.05% of images embedded in the SVG are loaded from a different origin compared to 16.79% from the Tranco baseline.

These results differ from the industry reports that found active content in SVG being used as attack vector via email. The active-content techniques most often associated with malicious SVG are absent from served landing pages. We can summarize these results into two main findings.

Table 3: Image origins with n = number of `<image/>` elements, showing SVGs in the user-reported dataset more often include images from a different origin compared to popular websites.

	User-reports (n=3,248)	Tranco toplist (n=2,138)
Same-origin	13.95%	83.21%
Cross-origin	86.05%	16.79%

Finding 1: active-content evasion is virtually absent from served pages. The features most directly tied to SVG-based evasion in industry reporting, such as embedded scripts, CDATA-shielded payloads, and `<foreignObject>`, are essentially non-existent in both the suspicious and the benign corpus. This suggests that the SVG threat highlighted for email-delivered attacks does not carry over to the SVGs served on landing pages. However, our crawler only captured the final HTML after JavaScript has executed and possible redirects happened. Therefore, it is possible active content has already executed without leaving a lingering trace for tools to identify.

Finding 2: suspicious pages embed more images, while predominantly loading them from different origins. The clearest distinction between the datasets is structural. SVG from the reported dataset are more than thirty times as likely to contain an embedded `<image>`. When looking into where these images are loaded from, the high majority is loaded cross-origin, differing from the popular websites that mostly load images in SVGs same-origin. This indicates that malicious (and spam) campaigns likely use multiple domains to host the same content, whereas popular domains likely use CDNs hosted on the same origin. Thus, where email-borne SVG abuse is about executing hidden content, landing-page SVG abuse is about embedding and referencing content across origins.

6 Related Work

Our work fits into the broader category of cloaking studies [1, 10, 25, 26, 35] where we focus on how SVGs are leveraged as evasion technique. The security risks of SVGs were first established by Heiderich et al., who showed that an SVG is effectively a scriptable, one-file web application that can embed HTML and JavaScript [8] and be abused for cross-site scripting [9]. More recently, industry reports have documented SVGs being used to hide malicious payloads [6, 7, 32, 34], but almost exclusively as email attachments. These works demonstrate the capability for abuse but leave the measurement on landing pages unexplored.

7 Limitations and Future Work

Our study has several limitations. First, the two corpora differ in SVG density per page (36 vs. 75 SVGs per SVG-bearing site); as our feature table is computed per SVG, some element-prevalence differences may partly reflect page composition rather than malicious intent. Second, our dataset consisting of user reports contains noise and might miss trends more present in pure phishing sites [33]. Future work should therefore expand our analysis to a broader, perhaps curated dataset complemented with a larger subset of benign websites. Finally, the metrics used to analyze the different SVGs can be expanded or complemented with an additional LLM step to catch more advanced techniques and provide explainability on how more obscure SVGs function.

8 Conclusion

We presented the first measurement of SVG usage on suspicious landing pages, testing whether the evasion techniques documented in email-delivered SVG attacks also appear on the Web, comparing SVGs drawn from citizen-reported suspicious messages against a benign Tranco baseline. Around half of popular domains use an SVG, while only 2.8% of suspicious webpages do. The active-content techniques most associated with malicious SVG in industry reporting (embedded scripts, CDATA-shielded payloads, and `<foreignObject>`) are almost entirely absent from served pages, and no more common in the suspicious corpus than in the Tranco one.

What distinguishes the suspicious corpus is structural: a far heavier reliance on embedded and linked imagery, and, in a small but real fraction of cases, the use of base64 encoding to inline content that a resource-based scanner would otherwise fetch and inspect. Our small manual analysis showed that when SVG is abused, it is used to embed and to inline rather than to execute, whereas linked images often are fetched from a different origin. Scanners should not treat an SVG as a static image, but as a document that can embed active content, inline encoded resources, and reference external ones where inspection should extend into the SVG's contents accordingly.

9 Ethics Considerations

All websites were crawled only once to minimize the traffic and we adhered to the Menlo report [3]. User reports were stripped of PII on a best-effort basis with all data being stored in local environments. Due to the sensitive nature of this data, it will not be made public.

10 Generative AI Usage

Claude Opus 4.8 was used to assist with writing code and as a writing assistant tasked with rephrasing and checking for grammar errors.

Acknowledgments

This research is partially funded by the Internal Funds KU Leuven, and by the Cybersecurity Research Program Flanders.

References

[1] Bhupendra Acharya and Phani Vadrevu. 2021. PhishPrint: Evading Phishing Detection Crawlers by Prior Profiling. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3775–3792. https://www.usenix.org/conference/usenixsecurity21/presentation/acharya

[2] Lucas Torrealba Aravena, Pedro Casas, Tailai Song, Javier Bustos-Jiménez, and Ivana Bachmann. 2026. PHISHGRAPH - When HTML Graph Simplicity Outsmarts Deep and Reference-Based Phishing Detectors. In *NOMS 2026-2026 IEEE Network Operations and Management Symposium*. 1–10.

[3] Michael Bailey, David Dittrich, Erin Kenneally, and Doug Maughan. 2012. The Menlo Report. *IEEE Security And Privacy* 10, 2 (2012), 71–75. doi:10.1109/MSP.2012.52

[4] Hugo Bijmans, Tim Booij, Anneke Schwedersky, Aria Nedgabat, and Rolf van Wegberg. 2021. Catching Phishers By Their Bait: Investigating the Dutch Phishing Landscape through Phishing Kit Detection. In *30th USENIX Security Symposium (USENIX Security 21)*. USENIX Association, 3757–3774. https://www.usenix.org/conference/usenixsecurity21/presentation/bijmans

[5] Xander Bouwman, Victor Le Pochat, Pawel Foremski, Tom Van Goethem, Carlos H. Ganan, Giovane C. M. Moura, Samaneh Tajalizadehkhoob, Wouter Joosen, and Michel van Eeten. 2022. Helping hands: Measuring the impact of a large threat intelligence sharing community. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, 1149–1165. https://www.usenix.org/conference/usenixsecurity22/presentation/bouwman

[6] Cloudflare. 2025. SVGs: the hacker's canvas. https://www.cloudflare.com/cloudforce-one/research/svgs-the-hackers-canvas/

[7] Ben Gibney. 2025. An Old Vector for New Attacks: How Obfuscated SVG Files Redirect Victims. https://www.forcepoint.com/blog/x-labs/obfuscated-svg-files-redirect-victims

[8] Mario Heiderich, Tilman Frosch, Meiko Jensen, and Thorsten Holz. 2011. Crouching tiger - hidden payload: security risks of scalable vectors graphics. In *Proceedings of the 18th ACM conference on Computer and communications security (CCS '11)*. ACM, 239–250. doi:10.1145/2046707.2046735

[9] Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, and Jörg Schwenk. 2012. Scriptless attacks: stealing the pie without touching the sill. In *Proceedings of the 2012 ACM conference on Computer and communications security (CCS '12)*. ACM, 760–771. doi:10.1145/2382196.2382276

[10] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. 2016. Cloak of Visibility: Detecting When Machines Browse a Different Web. In *2016 IEEE Symposium on Security and Privacy (SP)*. 743–758. doi:10.1109/SP.2016.50

[11] Takashi Koide, Daiki Chiba, and Mitsuaki Akiyama. 2020. To Get Lost is to Learn the Way: Automatically Collecting Multi-step Social Engineering Attacks on the Web. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security*. ACM, 394–408. doi:10.1145/3320269.3384714

[12] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society. doi:10.14722/ndss.2019.23386

[13] Wenhao Li, Selvakumar Manickam, Shams Ul Arfeen Laghari, and Yung-Wey Chong. 2023. Uncovering the Cloak: A Systematic Review of Techniques Used to Conceal Phishing Websites. *IEEE Access* 11 (2023), 71925–71939. doi:10.1109/ACCESS.2023.3293063

[14] Yuexin Li, Chengyu Huang, Shumin Deng, Mei Lin Lock, Tri Cao, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2024. KnowPhish: Large Language Models Meet Multimodal Knowledge Graphs for Enhancing Reference-Based Phishing Detection. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 793–810. https://www.usenix.org/conference/usenixsecurity24/presentation/li-yuexin

[15] Yuexin Li, Hiok Kuek Tan, Qiaoran Meng, Mei Lin Lock, Tri Cao, Shumin Deng, Nay Oo, Hoon Wei Lim, and Bryan Hooi. 2025. PhishIntel: Toward Practical Deployment of Reference-Based Phishing Detection. In *Companion Proceedings of the ACM on Web Conference 2025 (WWW '25)*. ACM, 2863–2866. doi:10.1145/3701716.3715192

[16] 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)*. USENIX Association, 3793–3810. https://www.usenix.org/conference/usenixsecurity21/presentation/lin

[17] Rebecka Lindkvist, Linn Petersson, Carl Magnus Bruhner, David Hasselquist, Martin Arlitt, and Niklas Carlsson. 2025. Characterizing the Trust Dilemma: Comparing Web Security of Malicious and Benign Domains. In *2025 9th Network Traffic Measurement and Analysis Conference (TMA)*. 1–4. doi:10.23919/TMA66427.2025.11096998

[18] Ruofan Liu, Yun Lin, Xiwen Teoh, Gongshen Liu, Zhiyong Huang, and Jin Song Dong. 2024. Less Defined Knowledge and More True Alarms: Reference-based Phishing Detection without a Pre-defined Reference List. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 523–540. https://www.usenix.org/conference/usenixsecurity24/presentation/liu-ruofan

[19] Ruofan Liu, Yun Lin, Xianglin Yang, Siang Hwee Ng, Dinil Mon Divakaran, and Jin Song Dong. 2022. Inferring Phishing Intention via Webpage Appearance and Dynamics: A Deep Vision Based Approach. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, 1633–1650. https://www.usenix.org/conference/usenixsecurity22/presentation/liu-ruofan

[20] Ruofan Liu, Yun Lin, Yifan Zhang, Penn Han Lee, and Jin Song Dong. 2023. Knowledge Expansion and Counterfactual Interaction for Reference-Based Phishing Detection. In *32nd USENIX Security Symposium (USENIX Security 23)*. USENIX Association, 4139–4156. https://www.usenix.org/conference/usenixsecurity23/presentation/liu-ruofan

[21] Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. 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)*. ACM, 78–86. doi:10.1145/3419394.3423632

[22] José Martinho, Diogo Mendes, and Pedro Pinto. 2024. ORAT - An Open Redirect Analysis Tool. In *2024 12th International Symposium on Digital Forensics and Security (ISDFS)*. 1–5. doi:10.1109/ISDFS60797.2024.10527319

[23] Tyler Moore and Richard Clayton. 2008. Evaluating the Wisdom of Crowds in Assessing Phishing Websites. In *Financial Cryptography and Data Security*, Gene Tsudik (Ed.). Springer, 16–30. doi:10.1007/978-3-540-85230-8_2

[24] Ala Mughaid, Shadi AlZu'bi, Adnan Hnaif, Salah Taamneh, Asma Alnajjar, and Esraa Abu Elsoud. 2022. An intelligent cyber security phishing detection system using deep learning techniques. *Cluster Computing* 25, 6 (2022), 3819–3828.

[25] Aleksandr Nahapetyan, Kanv Khare, Kevin Schwarz, Bradley Reaves, and Alexandros Kapravelos. 2026. Characterizing Phishing Pages by JavaScript Capabilities. doi:10.48550/arXiv.2509.13186

[26] Hiroki Nakano, Takashi Koide, and Daiki Chiba. 2025. PhishParrot: LLM-Driven Adaptive Crawling to Unveil Cloaked Phishing Sites. doi:10.48550/arXiv.2508.02035

[27] Adam Oest, Yeganeh Safei, Adam Doupe, 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. doi:10.1109/ECRIME.2018.8376206

[28] Linshu Ouyang and Yongzheng Zhang. 2021. Phishing Web Page Detection with HTML-Level Graph Neural Network. In *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. 952–958. doi:10.1109/TrustCom53373.2021.00133

[29] Tailai Song, Pedro Casas, and Michela Meo. 2026. Phishing the Phishers with SpecularNet: Hierarchical Graph Autoencoding for Reference-Free Web Phishing Detection. doi:10.48550/arXiv.2603.01874

[30] Karthika Subramani, William Melicher, Oleksii Starov, Phani Vadrevu, and Roberto Perdisci. 2022. PhishInPatterns: measuring elicited user interactions at scale on phishing websites. In *Proceedings of the 22nd ACM Internet Measurement Conference*. ACM, 589–604. doi:10.1145/3517745.3561467

[31] Xiwen Teoh, Yun Lin, Ruofan Liu, Zhiyong Huang, and Jin Song Dong. 2024. PhishDecloaker: Detecting CAPTCHA-cloaked Phishing Websites via Hybrid Vision-based Interactive Models. In *33rd USENIX Security Symposium (USENIX Security 24)*. USENIX Association, 505–522. https://www.usenix.org/conference/usenixsecurity24/presentation/teoh

[32] KnowBe4 Threat. 2025. 245% Increase in SVG Files Used to Obfuscate Phishing Payloads. https://blog.knowbe4.com/245-increase-in-svg-files-used-to-obfuscate-phishing-payloads

[33] Daan Vansteenhuyse, Hadji Musaev, and Lieven Desmet. 2026. From Reports to Insights: Challenges and Opportunities in Citizen-Driven Malicious Website Datasets. In *Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb 2026)*. doi:10.14722/madweb.2026.23033

[34] Austin Zeizel and Austin Bryant. 2025. Weaponized SVGs: Inside a global phishing campaign targeting financial institutions. https://www.ibm.com/think/x-force/weaponized-svgs-inside-a-global-phishing-campaign-targeting-financial-institutions

[35] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, RC Johnson, Brad Wardman, Shaown Sarker, Alexandros Kapravelos, 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 *2021 IEEE Symposium on Security and Privacy (SP)*. 1109–1124. doi:10.1109/SP40001.2021.00021

[36] Penghui Zhang, Zhibo Sun, Sukwha Kyung, Hans Walter Behrens, Zion Leonahenahe Basque, Haehyun Cho, Adam Oest, Ruoyu Wang, Tiffany Bao, Yan Shoshitaishvili, Gail-Joon Ahn, and Adam Doupé. 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 (CCS '22)*. ACM, 3165–3179. doi:10.1145/3548606.3559334