I'm SPARTACUS, No, I'm SPARTACUS: Proactively Protecting Users from Phishing by Intentionally Triggering Cloaking Behavior

Conference Paper · November 2022

DOI: 10.1145/3548606.3559334

CITATION
1

READS
258

12 authors, including:

Zhibo Sun
Arizona State University
11 PUBLICATIONS **139** CITATIONS

SEE PROFILE

Hans Walter Behrens
Arizona State University
9 PUBLICATIONS **38** CITATIONS

SEE PROFILE

Adam Doupé
Arizona State University
81 PUBLICATIONS **2,283** CITATIONS

SEE PROFILE

I'm Spartacus, No, I'm Spartacus: Proactively Protecting Users from Phishing by Intentionally Triggering Cloaking Behavior

Penghui Zhang
Penghui.Zhang@asu.edu
Arizona State University

Zhibo Sun
eric.sun@drexel.edu
Drexel University

Sukwha Kyung
skyung1@asu.edu
Arizona State University

Hans Walter Behrens
hwb@asu.edu
Arizona State University

Zion Leonahenahe Basque
zbasque@asu.edu
Arizona State University

Haehyun Cho
haehyun@ssu.ac.kr
Soongsil University

Adam Oest
aoest@paypal.com
PayPal, Inc.

Ruoyu Wang
fishw@asu.edu
Arizona State University

Tiffany Bao
tbao@asu.edu
Arizona State University

Yan Shoshitaishvili
yans@asu.edu
Arizona State University

Gail-Joon Ahn
Gail-Joon.Ahn@asu.edu
Arizona State University

Adam Doupé
doupe@asu.edu
Arizona State University

ABSTRACT

Phishing is a ubiquitous and increasingly sophisticated online threat. To evade mitigations, phishers try to "cloak" malicious content from defenders to delay their appearance on blacklists, while still presenting the phishing payload to victims. This cat-and-mouse game is variable and fast-moving, with many distinct cloaking methods—we construct a dataset identifying 2,933 real-world phishing kits that implement cloaking mechanisms. These kits use information from the host, browser, and HTTP request to classify traffic as either anti-phishing entity or potential victim and change their behavior accordingly.

In this work we present Spartacus, a technique that subverts the phishing status quo by disguising user traffic as anti-phishing entities. These intentional false positives trigger cloaking behavior in phishing kits, thus hiding the malicious payload and protecting the user without disrupting benign sites.

To evaluate the effectiveness of this approach, we deployed Spartacus as a browser extension from November 2020 to July 2021. During that time, Spartacus browsers visited 160,728 reported phishing URLs in the wild. Of these, Spartacus protected against 132,247 sites (82.3%). The phishing kits which showed malicious content to Spartacus typically did so due to ineffective cloaking—the majority (98.4%) of the remainder were detected by conventional anti-phishing systems such as Google Safe Browsing or VirusTotal, and would be blacklisted regardless. We further evaluate Spartacus against benign web sites sampled from the Alexa Top One Million List for impacts on latency, accessibility, layout, and CPU overhead, finding minimal performance penalties and no loss in functionality.

CCS '22, November 7–11, 2022, Los Angeles, CA, USA

ACM ISBN 978-1-4503-9450-5/22/11…$15.00
https://doi.org/10.1145/3548606.3559334

CCS CONCEPTS

• **Security and privacy → Phishing**.

KEYWORDS

phishing, web security, cloaking, social engineering

ACM Reference Format:
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), November 7–11, 2022, Los Angeles, CA, USA.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3548606.3559334

1 INTRODUCTION

Despite concerted effort by the security community, phishing attacks not only remain prevalent [12, 41], but have, in fact, recently replaced online malware as the biggest web-based threat [11, 35]. The reasons are a familiar story for any anti-malware or immunology researcher. Similar to computer viruses and malware, our strongest weapon in the fight against phishing is timely detection (through Internet-scale web crawling and analysis) and quarantine (through blacklisting such as Google Safe Browsing) of phishing sites [25, 52]. However, advanced phishing web sites use *evasion* techniques (also known as *cloaking* techniques), to avoid being detected by anti-phishing systems [38]. By successfully evading being blacklisted, even for a brief time, these phishing sites significantly extend the window in which they can damage users [25].

All cloaking techniques aim to show phishing content to visitors that they decide are a "real human," while showing a benign web page to those who are identified as an "anti-phishing crawler." Cloaking techniques are categorized into two groups: client-side and server-side. Client-side cloaking fingerprints a potential victim's browser, typically through the execution of JavaScript code. Though there is an ongoing cat-and-mouse game in that space, recent research shows promise in detecting client-side cloaked phishing sites through analysis of the client-side cloaking code [52].

Server-side cloaking uses *fingerprinting-based cloaking techniques* (a term that we use synonymously with server-side cloaking) that leverage IP addresses, User-Agent HTTP headers, or Referrer HTTP headers to identify visitors [14]. Because this cloaking occurs so early in the process, client-side anti-phishing techniques are unlikely to obtain the content they need to detect phishing sites [23, 24]. Though the anti-phishing ecosystem can *eventually* blacklist phishing web sites that use server-side cloaking techniques (e.g., through user reports), as demonstrated by prior work and our own experiments (in section 7), these techniques significantly increase the time that a phishing web site can victimize users [22, 23, 52].

In studying the ecosystem, we realized that anti-phishing research tends to focus on improving the analysis power of anti-phishing systems, and that this approach fares poorly against the small amount of information provided by a server-side cloaked site. However, server-side cloaking techniques can use the relatively large amount of information available to them to easily distinguish between HTTP requests made by legitimate users and HTTP requests made by anti-phishing systems.

In this paper, we consider the anti-phishing problem from a different angle and strike at the core reason behind the effectiveness of server-side cloaking techniques: rather than attempting to detect server-side cloaking through improved analysis tools, we instead *make the legitimate users themselves look like anti-phishing crawlers, so that server-side cloaked phishing pages will decline to phish them.* In this way, users can evade phishing content in real time, without prior knowledge of the phishing web site, by *leveraging*, instead of fighting, the cloaking functionality in the phishing site itself.

To realize this idea, we propose Spartacus, a framework that disguises user browsers as anti-phishing crawlers when requesting web page content. When visiting web sites, Spartacus mutates the information that phishers may fingerprint in the HTTP request such as User-Agent, Referrer, or IP address[1] to make the request appear to be from an anti-phishing crawler. When the server-side cloaking script examines the HTTP request, it will decide that the visit is from an anti-phishing system, and will return a benign-looking web page to the user, sparing them from the phishing attack.

By conducting the following evaluations, we demonstrate that Spartacus can effectively protect users from server-side cloaking. First, to estimate the potential benefits of Spartacus, we measured the prevalence of server-side fingerprinting-based cloaking techniques in phishing kits (programs used by phishers to easily create phishing web sites) using an automated analysis. In total, we analyzed 2,933 phishing kits and discovered that 96.52% (2,831) of them contain server-side fingerprinting-based cloaking techniques.

Then, we performed an evaluation to see if Spartacus can trigger evasion in real-world phishing sites. In our large-scale evaluation of the framework, over a period of nine months from late-2020 to mid-2021, Spartacus visited 160,728 real phishing web sites (provided by the Anti-Phishing Work Group) *and evaded 82.28% of them without relying on blacklists or other anti-phishing techniques.*

Because the Spartacus framework is designed to protect end-users directly, its impact on the functionality of benign web sites is a concern. We evaluated the performance and functionality impact of Spartacus on benign web sites both automatically and manually. We found that with Spartacus installed, the tested benign web sites displayed properly (i.e., benign web sites do not perform server-side cloaking or otherwise change the HTML that they send based on the mutated HTTP headers). When visiting benign web sites hosted on providers that employ security mechanisms such as Akamai and Cloudflare, Spartacus could successfully visit the majority of them (99.84% out of 10,000). Complex components in these web sites, such as buttons/links, online chat, register/login, shopping carts, checkout, etc. functioned correctly without any error. The authors also installed Spartacus in their daily-use browsers to visit web sites for one month, and Spartacus did not cause any issues when used in real-world web browsing for a month.

We also evaluated current anti-phishing systems against modern phishing web sites and compared that to Spartacus. We submitted the 45,526 phishing web sites that Spartacus visited as part of our large-scale evaluation to anti-phishing systems and monitored the result. After waiting five days for blacklists to update, 24,154 (53.1%) phishing sites that were evaded by Spartacus (i.e., they showed benign content to Spartacus) *were not detected by anti-phishing systems*, 16,698 (36.7%) were evaded and detected, 4,598 (10.1%) were not evaded but were detected, and 76 (0.2%) were neither evaded nor detected. In other words, Spartacus alone can protect users against 89.8% of this subset of phishing sites we analyzed *in real time*, and the combination of Spartacus and current techniques can protect users against 99.8% of them, whereas existing techniques alone protect against only 46.8% (and these have a median blacklisting lag time of 2.58 hours, compared to Spartacus' real-time effect).

These results suggest that the idea of Spartacus traps phishers in a dilemma: to attack Spartacus users, phishers should disable at least some server-side cloaking criteria and therefore allow more HTTP requests to successfully retrieve the phishing content. However, this strategy allows anti-phishing crawlers to view the phishing content and use content-based detection techniques.

In summary, our contributions are as follows:

- An analysis of modern phishing kits to understand the prevalence of fingerprinting cloaking techniques.
- An automated client-side framework called Spartacus that can evade phishing web sites with fingerprinting-based cloaking with little impact to users' browsing activity.
- An evaluation of Spartacus as a browser extension to measure its effectiveness and efficiency compared to current anti-phishing systems.

To protect Internet denizens from phishing attacks, we release our Spartacus extension for public use[2].

[1] IP address is optional due to privacy concerns, as discussed in Section 5.

[2] https://mega.nz/folder/zboygDBL#t6l1QOSZlYNgwaM_0AL89g.

Cloaking Type	Attributes	Category
Server-side	HTTP Request	IP Cloaking User-agent Cloaking Referrer Cloaking
Client-side	Client-side Characteristics w/ JavaScript	User Interaction Cloaking Fingerprinting Cloaking Bot Behavior Cloaking

Table 1: Summary of cloaking types used in phishing.

2 BACKGROUND

The research community proposed many anti-phishing techniques such as URL-based phishing detection [4, 5, 13, 16] and web content analysis techniques [3, 6, 49, 51, 53]. Those approaches introduced URL blacklists, malicious infrastructure analysis techniques, and e-mail spam filters. In addition, commodity URL blacklists such as Google Safe Browsing [45] and Microsoft SmartScreen [21] support the anti-phishing ecosystem in the backend, which use machine learning classifiers to warn users that the web site they are visiting is suspicious. A downside of such defenses is that only when anti-phishing systems retrieve phishing content can they precisely classify phishing web sites [21, 45, 50]

To exploit the downside of the anti-phishing systems, modern phishing campaigns use cloaking techniques that attempt to distinguish potential victim visits from anti-phishing entity visits in an attempt to hide from the latter party [48]. Several research works have shown that cloaking techniques are effective at delaying or disabling detection [18, 22, 24]. There are two categories of cloaking techniques: server-side and client-side (Table 1 categorizes each type). Client-side cloaking techniques execute JavaScript in the user's browser to distinguish visitors and display different web page content [52]. However, server-side cloaking techniques analyze the HTTP request to identify visits from anti-phishing entities [14, 23]. Among them, *fingerprinting-based cloaking techniques* are widely used in advanced phishing web sites. Figure 1 depicts how phishing web sites use fingerprinting-based cloaking techniques. Cloaking code in the phishing web server fingerprints the profile in the HTTP request and responds with different web page content (with the goal of showing phishing content only to potential victims).

However, as phishing kits continue to evolve, the number of identified fingerprints increases. As shown in Figure 2, any match of the IP address, hostname, or User-Agent will result in a `404 Page Not Found` error response. Therefore, visits from anti-phishing systems trigger fingerprinting-based cloaking techniques in the phishing server. As a result, anti-phishing systems cannot retrieve phishing content, which leads to mis-detection or delayed detection.

In this work, we consider the anti-phishing problem from a different perspective: Phishers try their best to evade visits from anti-phishing entities, so let users camouflage themselves as anti-phishing crawlers when visiting phishing web sites. In this way, users will not see the phishing content. Consequently, we can protect them from the very first visit to unknown phishing web sites.

3 PREVALENCE OF FINGERPRINTING-BASED CLOAKING

Oest et al. [23] analyzed *.htaccess* files in phishing kits and demonstrated that fingerprinting-based cloaking is popular. To examine the prevalence of fingerprinting-based cloaking techniques used

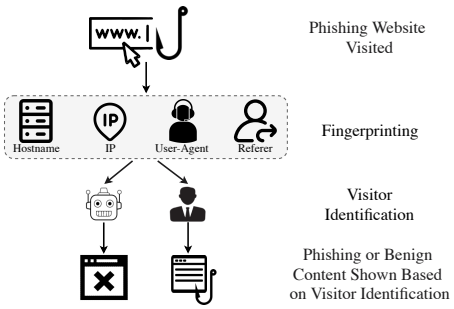


Figure 1: Typical operation of server-side fingerprinting-based cloaking in phishing web sites.

```php
$hostname = gethostbyaddr($_SERVER['REMOTE_ADDR']);
$blocked_words = array("above","google","softlayer","amazonaws","cyveillance",
"phishtank","dreamhost", "netpilot","calyxinstitute","tor-exit", "paypal", ...);
foreach($blocked_words as $word) {
    if (substr_count($hostname, $word) > 0) {
    header("HTTP/1.0 404 Not Found");
        die("<h1>404 Not Found</h1>
        The page that you have requested could not be found.");

    }
}
$bannedIP = array("^66.102.*.*", "^38.100.*.*", "^107.170.*.*", "^149.20.*.*",
"^64.233.160.*", "^72.14.192.*", "^66.102.*.*", "^64.18.*.*", "^194.52.68.*",
"^212.143.*.*", "^212.150.*.*", "^212.235.*.*", "^217.132.*.*", "^50.97.*.*",
"^217.132.*.*", "^209.85.*.*", "^66.205.64.*", ...);
if(in_array($_SERVER['REMOTE_ADDR'],$bannedIP)) {
     header('HTTP/1.0 404 Not Found');
     exit();
} else {
     foreach($bannedIP as $ip) {
          if(preg_match('/' . $ip . '/',$_SERVER['REMOTE_ADDR'])){
               header('HTTP/1.0 404 Not Found');
               die("<h1>404 Not Found</h1>
               The page that you have requested could not be found.");
          }
     }
}

if(strpos($_SERVER['HTTP_USER_AGENT'], 'google')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'msnbot')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'Yahoo! Slurp')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'YahooSeeker')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'Googlebot')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'crawler')
    or strpos($_SERVER['HTTP_USER_AGENT'], 'facebookexternalhit') !== false
    or ...)
{ header('HTTP/1.0 404 Not Found'); exit; }
```

Figure 2: Simplified PHP code snippet of fingerprinting-based cloaking in a phishing kit, checking IP, Hostname, and User-Agent.

in advanced phishing kits, we manually inspect a random 10.93% (56) of phishing kits from a dataset by *phishunt.io* [28], and extract common patterns of fingerprinting-based cloaking techniques.

The patterns include (1) blocked words, for example, any potential crawler identification such as "google," "facebook," or "phishtank" (e.g., `blocked_words` in Figure 2), (2) IP checks that block visit from IP addresses, such as `bannedIP` in Figure 2, and (3) error responses (i.e., returning an error status code and an error web page). These attributes reflect the implementation of fingerprinting-based cloaking techniques in real-world phishing kits.

We use the identified patterns to automatically find fingerprinting-based cloaking techniques in phishing kits. Among the inspected kits in *phishunt.io* [28] from May 2020 to July 2021, 410 of 512 contain fingerprinting-based cloaking techniques through IP, Referrer, or User-Agent (297, 5, and 318, respectively). We also analyze 2,421 phishing kits provided to us by Cisco and find that all of these phishing kits implement fingerprinting-based cloaking techniques. Among them, 1,983 of the phishing kits contain a User-Agent check, and 1,660 contain an IP address check. In total, 96.52% (2,831) out of 2,933 phishing kits contain fingerprinting-based cloaking techniques.

Popular fingerprinting words. To understand the popularity of blocked User-Agent words (e.g., "bot" and "curl") that phishing kits

Sensitive Word	Amount	Sensitive Word	Amount
bot	1,273	crawler	371
curl	581	facebook	223
google	447	phishtank	200
amazonaws	446	atn	125
compatible	432	spinner	113

Table 2: Top 10 sensitive words appeared in the phishing kits we examined.

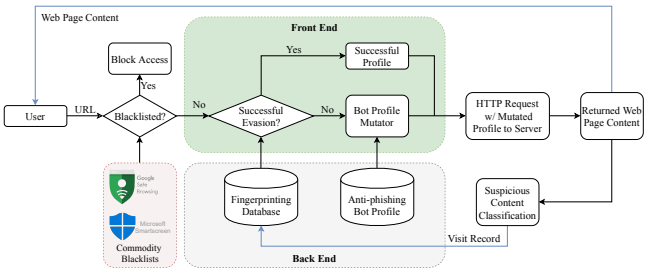


Figure 3: Spartacus architecture and its workflow.

use to evade anti-phishing crawler, we counted the appearance of 407 unique words in the User-Agent checks of phishing kits. Table 2 displays the top 10 blocked words. Note that one phishing kit can contain multiple rules using a word. The result shows that if a request contains "bot" or "curl" in the HTTP User-Agent header, phishers will return an error rather than phishing content. A potential victim's HTTP request, however, does not include those blocked words [44]. Based on this analysis, we can use the frequently blocked words and turn them into *trigger words* that will cause an HTTP request to evade the phishing content by triggering the phishing web site's fingerprinting-based cloaking.

4 DESIGN

By turning fingerprinting-based cloaking against phishing web sites, we can create a proactive anti-phishing defense for users who attempt to visit such web sites. To this end, we design, implement, and evaluate Spartacus, a framework that automatically and selectively mutates HTTP requests made by a user's web browser to resemble those of an anti-phishing entity. If a user protected by Spartacus was to visit a phishing web site with fingerprinting-based cloaking, the mutated request would trigger the cloaking, which will then display benign content rather than a phishing page.

First, we define a *profile* as the set of fingerprintable attributes and their values. For instance, one profile might contain a User-Agent string with `bot`, an empty Referrer, or an AWS IP address. Profiles are used by Spartacus to generate an appropriate HTTP request for the target web site. Whenever phishers add new fingerprints to fingerprinting-based cloaking, these attributes can also be added to the Spartacus profiles.

4.1 Overview

Figure 3 illustrates the Spartacus architecture. The framework consists of two parts, the *front end* and the *back end*. The front end is responsible for the primary functionality, such as deciding if and how to mutate profiles, and the back end stores information.

When a user visits a URL, it is first checked against blacklists maintained by current anti-phishing systems, and if the URL is found, Spartacus outright *blocks* the access (by displaying a prominent phishing warning). Otherwise, the front end queries the *Fingerprinting Database* to see if it has processed the URL before. If the URL is found, and was previously successfully mitigated, Spartacus will use the same profile to request the web site. To mitigate privacy concerns, we design the Fingerprinting Database to operate locally, which does not share visit history among Spartacus users. However, the Fingerprinting Database could be extended to be shared through a centralized server (while maintaining users' privacy), which we discuss in section 8.

If the URL is neither blacklisted nor in the database, Spartacus will mutate the profile based on the *Anti-phishing Bot Profile Database*, skipping any mutations that previously failed to trigger cloaking[3]. Then, the HTTP request, with the mutated profile, is sent to the server. After receiving the page content, we determine whether it has suspicious content by using a classification engine that executes concurrently with Spartacus and runs in the background to avoid delaying page rendering. The purpose of the suspicious content classification is to verify if the mutated profile was effective at triggering cloaking. The Fingerprinting Database is, therefore, updated with the visited URL, the mutated profile Spartacus used, and the classification result.

4.2 Visit Pre-filters

When a user visits a URL, it must first pass through two pre-filters.
Blacklist Filter. With the contributions of the anti-phishing ecosystem, Spartacus can filter known phishing URLs that have already been blacklisted by commodity blacklists, specifically Google Safe Browsing and Microsoft SmartScreen. Any match will block access to the URL without further action.
Prior Visits. If the URL is not blacklisted, Spartacus will examine if the URL was previously visited by querying the Fingerprinting Database. If the URL was already visited and was successfully evaded, then Spartacus will use the successful profile mutation.

4.3 Bot Profile Mutator

The Bot Profile Mutator is responsible for profile mutation to trigger fingerprinting-based cloaking in advanced phishing web sites. Items in the profile can be modified or changed to camouflage the user as anti-phishing crawlers, for instance the User-Agent HTTP header. Generally, anti-phishing crawlers contain "bot" and "crawler," or the name of the company such as "Google" and "Facebook" in the User-Agent HTTP header. The mutator uses the *trigger words* that we automatically extracted from phishing kits (discussed in Table 2). Spartacus is extendable to accept more trigger words to mutate users' profile to evolve with the state of anti-phishing crawlers/bots and server-side cloaking.

Another aspect is the Referrer HTTP header. Typically, the potential victim visits from the phisher's phishing lures. Therefore, phishers can block all visits that are not from the phishing lures.

Optionally, the profile mutator can leverage proxy servers to camouflage the user's IP address. For example, a proxy server on *AWS EC2* is useful because phishers have inferred that some anti-phishing crawlers use *AWS EC2* (according to our analysis

[3]We populate the Anti-phishing Bot Profile Database with known crawler-looking profiles (e.g., extracted from phishing kits as discussed in section 3).

in Table 2). In this case, the bot mutator can proxy the request through an evaded IP. Even though the proxy server can help evade fingerprinting-based cloaked phishing websites, it can also raise privacy concerns (discussed in section 5). Therefore, users must consent to the privacy implications before enabling it.

In the mutation process, Spartacus appends one trigger word from the Anti-phishing Bot Profile Database to the user's own User-Agent string, following the order of the popularity in Table 2. Spartacus also avoids using trigger words that were not successful for the same URL. Additionally, Spartacus sets the Referrer to None (to remove the header). As for the optional IP/Hostname mutation, Spartacus reroutes the request to a proxy server, whose IP is in one of the most popular blocked IP ranges.

4.4 Suspicious Content Classification

After submitting the HTTP request to the server, Spartacus will receive an HTTP response from the server. After the suspicious content classification, there are four different possibilities:

(1) The server is benign and responds with benign content.
(2) The server is benign and responds with suspicious content.
(3) The server is malicious and responds with benign content (e.g., error page or redirection to a benign web site, as shown in Figure 4c).
(4) The server is malicious and responds with suspicious content.

We consider a successful evasion whenever there is no features such as a login form, sensitive (phishy) words (e.g., "username" and "password"), and a submission button [50], because we do not otherwise know if the content is benign. For example, if a user first visits *paypal.com*, which is benign (but Spartacus does not know this), Spartacus mutates the profile and the web page will still contain "username" and "password" fields because PayPal returns them. In this case, Spartacus will treat it as unsuccessful because there is still "phishy" content. Similarly, if a user visits a PayPal phishing page such as *paypal-verify.com* that does not contain cloaking, Spartacus mutates the profile but still receives a phishing web page. In this case, Spartacus will also treat it as unsuccessful for the same reason.

For case (1), there is no security risk to the user, so we consider it a *successful* evasion. However, there is a possibility that the benign web site serves different content to Spartacus (due to User-Agent sniffing or other server-side techniques) and that Spartacus breaks the functionality of the web site for the user. Our experimental results (presented in Section 6.6) demonstrate that very few modern web sites change functionality/responses based on our changes to the HTTP request. In addition, we present in Section 6.6 a proposal, with experimental results, of adding another pre-filter to Spartacus to apply Spartacus to only suspicious web sites.

Case (2) happens when the user visits benign web sites that contain phishing-like features such as a login form, sensitive (phishy) words (e.g., "username" and "password"), and a submission button. As these are often indistinguishable from phishing pages (in fact, they are copied by phishers), we still consider it as an *unsuccessful* evasion, because we consider *all* unknown web sites with a login form as suspicious. In the suspicious content classification, we still determine it as "suspicious" and the mutation will be marked as "unsuccessful." Similar to case (1), this situation does not affect users' browsing activities on those web sites, according to the experimental result in Section 6.6. When the user visits the same web site next time, Spartacus will mutate the profile with other available trigger words in the Anti-phishing Bot Profile Database. Based on the evaluation result in Section 6.6, changing trigger words does not impact the web site's accessibility, layout, or functionality.

In case (3), the server is malicious with evasion techniques and determines that the visit from Spartacus is an anti-phishing bot visit. So it either returns an error web page, or redirects the visit to a benign web site. Consequently, Spartacus *successfully* prevents users from seeing phishing content, by triggering the fingerprinting-based cloaking techniques in the phishing web sites.

In case (4), the phishing web site either (a) does not perform any fingerprinting-based cloaking or (b) the profile failed to trigger the fingerprinting-based cloaking. In the former case (a), Spartacus cannot trigger any cloaking behavior in the phishing site, so we consider it as an *unsuccessful* evasion. However, our results in Section 7 demonstrate that these phishing sites are quickly detected by the anti-phishing ecosystem (median detection of 28 minutes). In the latter case (b), Spartacus will store the failed profile to help inform future visits to this URL to trigger the fingerprinting-based cloaking. Note that, from the perspective of the browser, this case is the same as case (2), which is why we cannot simply block the URL (we do not know if the server is malicious or benign).

5 PRIVACY

The Spartacus framework, as discussed in Section 4, is a client-side framework that runs solely on the user's machine. However, we strive to protect as much of the user's privacy as possible. In addition, these privacy protections are important to enable a potential centralized Spartacus in future work (discussed in Section 8).

5.1 Privacy Information

Spartacus requires four types of sensitive information from users: (1) *visited URL*: When Spartacus mutates the profile, it operates on a hashed URL, and only stores the hashed URL, along with the successfulness of evasion. (2) *The HTTP profile used*: Spartacus stores the user's HTTP profile to modify it. The privacy information in the profile includes the User-Agent string, which contains browser version, browser type, and operating system information, and the Referrer. (3) *Returned HTTP response*: Spartacus analyzes, but does not store, the returned HTTP response to inspect whether the HTTP response is suspicious. If valid content is returned, an external classification process will determine its suspiciousness by searching for content such as sensitive words and login forms [50]. The classification result is then stored to mark if the corresponding profile mutation successfully evades suspicious content. (4) *Potential proxy server monitoring*: if a user wants improved evasion performance beyond the User-Agent/Referrer mutation, he/she can choose to turn on the proxy server option in Spartacus. In this case, all user's HTTP requests will be sent through a proxy server, but the user can be susceptible to monitoring.

5.2 Privacy Consent and Protection

We ensure that Spartacus reasonably considers users' privacy, so we implemented both consent and protection methodologies to notify users of the data that is collected and prevent their information from being stolen and abused.

Consent. To ensure that users are aware of the privacy information Spartacus access and stores, a privacy policy consent notice is presented on first use. First, the privacy information used by Spartacus is summarized. Then, using Privacy Policies [29], a privacy policy is created for Spartacus. The information Spartacus collects and how it uses the information are described on the privacy policy page. Users can choose to opt out and not install Spartacus if they do not consent. Especially, when users want to turn on the proxy server option to improve evasion performance, they must understand and consent to the option's trade-off. We created a dedicated privacy policy page for the proxy server option.

Protection Methodology. The privacy information Spartacus collects does not contain any PII, which minimizes potential harm. Spartacus stores the hashed URL, bot profile (IP address, User-Agent triggering word, and Referrer), and evasion success.

6 EVALUATION

We implemented Spartacus as a Chrome browser extension, and we evaluate Spartacus through three perspectives: effectiveness, latency, and functionality impact. These three aspects demonstrate the feasibility of the Spartacus framework in practice because it can successfully evade advanced phishing web sites, introduce negligible latency to user browsing, and not introduce breakage.

Dataset. In our evaluation, we used two different datasets, a malicious one, to test the effectiveness of Spartacus, and a benign one, to understand its potential impact on benign web sites.

(1) *APWG Dataset*: For the effectiveness evaluation, Spartacus visited 160,728 live phishing web sites from November 2020 to July 2021 using the Anti-Phishing Working Group (APWG) URL feed [32], which is a curated dataset of reported phishing URLs, supported by a large number of collaborating members. Additionally, we leveraged another 8,474 live phishing web sites in the APWG Dataset to evaluate the effectiveness of IP mutation.

(2) *Benign Dataset*: To evaluate the impact on benign web sites, we collected a dataset of 60,848 benign domains, randomly selected from 629,843 domains in the Alexa Top One Million Domain List [1].

6.1 Effectiveness

We evaluate the effectiveness of Spartacus by visiting the same phishing web sites with two different browser configurations: one with default settings (default browser) and the other with Spartacus installed (Spartacus browser). The phishing URLs for both visits are from the APWG Dataset. To reduce the impact of the selection of trigger words on the results, for each URL the Spartacus browser uses a profile with a random trigger word from the 407 trigger words, no Referrer header, and no IP proxy. We use a simple profile because, per our analysis of phishing kit behavior in Section 3, any cloaking rule match will trigger the cloaking. Note that in Section 6.2 we evaluate the effectiveness of each trigger word and in Section 6.4 we evaluate the effectiveness of proxying the IP address. For each visit, we record the final landing web page content and URL.

In the experiment on 160,728 phishing URLs from APWG from November 2020 to July 2021, 132,247 (82.28%) did not contain malicious content in Spartacus. We consider an HTTP response from the Spartacus browser benign if its web page (1) is different from the one shown on the default browser and (2) does not contain suspicious content, such as "phishy" words or bad forms, according to the content features in our reimplementation of CANTINA+ [50].

Figure 4 demonstrates the difference of response web page content between the default and the Spartacus browser visit for a cloaked phishing web site. The content in Figure 4a shows the phishing content when the default browser visits it. When Spartacus mutates the HTTP profile to include a random trigger word and removes the Referrer header, the phishing web site shows the error web page in Figure 4b. Other phishing web sites redirect visitors to a benign URL instead of returning an error page. In this way, Spartacus receives the web page content shown in Figure 4c, also indicating a successful evasion.

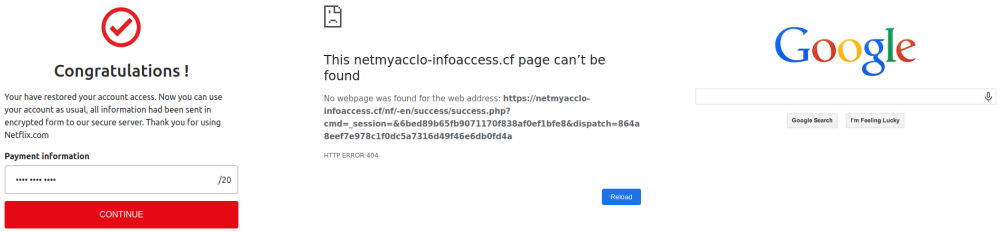
This result shows that Spartacus can camouflage users as anti-phishing entities and prevent users from phishing content on phishing sites with fingerprinting-based cloaking techniques. Because the users see benign content (either an error page or a benign URL), the user is never exposed to the phishing attack, thus proactively preventing the user from falling victim to the phishing attack—even if they are the first user to visit the phishing URL.

6.2 Effectiveness of Trigger Words

In the prior Spartacus experiment, we used random trigger words for each URL visit. While this limited the impact of trigger word selection on the results, in our design of Spartacus the profile mutator selects trigger words in order of their popularity. Therefore, we evaluate the effectiveness of each trigger word on actually triggering fingerprinting-based cloaking techniques. The trigger words are retrieved through the analysis of phishing kits, discussed in Section 3.

We tested all the trigger words by visiting each phishing web site with a different profile consisting of the trigger word, no Referrer header, and no IP proxy. In the evaluation, one profile means one trigger word because each profile contains only one trigger word. Similar to the effectiveness evaluation, we also visit the web site using a default browser as a comparison. We conduct the experiment on 916 phishing web sites. 725 of them show web page differently at least under one trigger word between Spartacus browser and default browser.

In the 725 cloaked phishing web sites, each trigger word has different evasion capabilities. Table 3 is the result of the top 10 trigger words in successfully evading phishing content. The word *bot* has the most sites evasion. 99.31% of the cloaked phishing web sites can be evaded by appending *bot* in the User-Agent. Compared with the popularity rank in Table 2, it is also the most popular blocked word in the phishing kits we examined. The effectiveness of this word in practice confirms its popularity in the phishing kits. Similarly, the top trigger words such as "amazonaws," "phishtank," and "google" also have high usage in phishing kits. An interesting note



(a) Default browser visit. **(b) Spartacus browser visit with error.** **(c) Spartacus browser visit with content.**

Figure 4: Web page content from default and Spartacus browser visits for a cloaked phishing web site.

is that "bot" and "amazonaws" combined can trigger cloaking in all phishing web sites that used fingerprint-based cloaking techniques.

This result shows that trigger words can effectively evade phishing web sites with fingerprinting-based cloaking techniques. Furthermore, a small number of trigger words can evade a myriad of cloaked phishing web sites.

6.3 Effectiveness of Only User-Agent or Referrer

Spartacus achieved over 80% evasion rate by mutating both User-Agent and Referrer headers, but we hope to understand the effectiveness of Spartacus mutating either User-Agent or Referrer header. To this end, we performed evaluation where we had three different browsers visiting the same dataset of phishing web sites: (1) a browser with Spartacus only mutating the User-Agent string (UA Spartacus browser); (2) a browser with Spartacus only mutating the Referrer (REF Spartacus browser); and (3) a normal browser (normal browser).

In this evaluation, we visited 4,905 phishing web sites using the above three browsers. The analysis was conducted similar to subsection 6.1, where we compared the web page contents of each web site from the UA Spartacus browser and the REF Spartacus browser with those from the normal browser, respectively, and thus we can detect the amount of web pages of either the UA Spartacus browser or the REF Spartacus browser that did not contain suspicious content. Among the visited phishing web sites, the UA Spartacus browser evaded 4,028 of them, while the REF Spartacus browser evaded 16 of them. Spartacus could evade a small amount of phishing websites by only mutating the Referrer, because a limited number of phishing kits contained Referrer inspection (our analysis showed 5 out of 410 phishing kits from *phishunt.io* in section 3). We consider Referrer as an option in Spartacus because phishers may check the Referrer in future phishing kits. Combined, Referrer and User-Agent could evade 82.44% of the phishing web sites. These results show that Spartacus only mutating the User-Agent header could evade more phishing web sites than only mutating the Referrer, however mutating *both* of them improves protection than only one mutation.

6.4 Effectiveness of Proxy Server Option

Even though Spartacus can successfully evade phishing content in over 80% of the phishing web sites by mutating User-Agent and Referrer headers, we want to analyze the effectiveness of mutating the IP address. Therefore, we conducted another experiment with a Spartacus profile that had default User-Agent header, has Referrer header, but proxied the connection through a server with an Amazon AWS IP address. Since the proxy server option may introduce privacy implications, we turn off this option by default. Users can opt to use this feature in Spartacus only if they read, understand, and consent to the privacy implications.

Trigger Word	Count	Trigger Word	Count
bot	720	atn	717
amazonaws	718	curl	717
phishtank	718	facebook	716
dwcp	717	crawler	716
google	717	katipo	713

Table 3: Top 10 trigger words that evaded phishing content.

In this experiment we used 8,474 phishing web sites. Similar to the evaluation procedure in Section 6.1, we visited those web sites in both the default and the Spartacus browsers (only changing the IP address according to the profile). Then, we compared the web page contents of each phishing web site on both visits and detected if the web page of the Spartacus browser does not contain suspicious content. Among the visited phishing web sites, Spartacus evaded 88.98% (7,540) of them through the proxy server. Therefore, Spartacus can evade phishing web sites that implement IP, User-Agent, or Referrer cloaking. Phishers may design emerging cloaking techniques in the future, however Spartacus was designed as an extensible framework so that fingerprint features can be added.

6.5 Efficiency and Latency

We next explore Spartacus' impact on the user experience when they visit benign web sites. By design Spartacus may introduce latency to the HTTP request, due to the database query, HTTP profile mutation, and returned content inspection.

We conducted an experiment to measure the latency of Spartacus from the following three perspectives: database query, profile mutation, and content inspection. We used *exthouse* [39], which analyzes the impact of a browser extension on web performance, as our test bench, which contains five major measurements:

- Time to Interactive (TTI): the time it takes for the page to become fully interactive with the extension.
- First Input Delay (FID Δ): the time from when a user first interacts with the web site to the time when the browser is actually able to begin processing event handlers in response to that interaction.

Name	Score		FID Δ (ms)		Scripting Δ (ms)		TTI (ms)		Added Long Task (ms)		Extra CPU Time (ms)	
	B	M	B	M	B	M	B	M	B	M	B	M
Grammarly for Chrome	50	60	20	190	300	1,000	1,300	3,500	380	1,740	300	961
Adblock Plus	59	100	20	20	0	100	900	3,300	0	0	0	0
Skype	82	74	140	130	100	300	900	3,300	130	250	141	277
Avira Browser Safety	94	90	60	50	100	200	1,100	3,600	0	110	63	112
Avast SafePrice	99	68	120	90	100	200	1,100	4,000	310	400	67	82
AdBlock	100	100	50	20	0	100	1,000	3,300	0	0	0	0
Google Translate	100	100	20	20	0	100	900	3,200	0	0	0	0
Pinterest Save Button	100	100	30	30	0	100	800	3,200	0	0	0	0
Tampermonkey	100	100	20	20	0	100	1,000	3,400	0	0	0	0
uBlock Origin	100	100	20	20	0	100	1,000	3,600	0	0	0	0
Spartacus	**100**	***100***	**20**	***20***	**0**	***100***	**800**	***3,200***	***0***	***0***	***0***	***0***

Table 4: Exthouse metrics of top 10 Chrome extensions [39] along with Spartacus when visiting Benign and Malicious websites.

- Scripting Time (Scripting Δ): the amount of time JavaScript execution in the extension.
- Long Task (Added Long Task): this value represents a sum of Long Tasks added by the extension, where Long Tasks are defined as a task that blocks the main thread for 50 ms or more[4].
- Extra CPU Consumption (Extra CPU Time): the extra CPU consumption of the extension for each URL the browser visits.

The lower the factors are, the better the web site performs with the tested extension. Lastly, *exthouse* creates a score for the extension. A higher score reflects a better performance of the extension.

Table 4 illustrates the *exthouse* scores of the top 10 Chrome extensions [39] and Spartacus when visiting benign and malicious web sites. We tested these extensions with 100 web sites, including half benign and half malicious, and used the average in the metrics. Spartacus has a score of 100, based on a 20 ms FID, 0 scripting delta, and 800 ms of TTI when visiting benign web sites. The metrics of Spartacus visiting malicious web sites also outscores those of other popular extensions. Even though it takes a longer time to interact with the malicious web site, it is still acceptable because Spartacus needs time to mutate the profile, which is still less time than other extensions. For instance `Avira Browser Safety` (ABS) is an extension that warns users if the web site is unsafe. It adds long tasks and extra CPU time when visiting malicious web sites.

This evaluation result shows that Spartacus adds minimal overhead to web browsing. The inspection result shows that Spartacus outscores popular Chrome extensions and has negligible impact on the performance of the web sites, compared with other extensions.

6.6 Impact on Benign Web Sites

Merzdovnik et al. discovered that add-ons can cause some web sites to malfunction (e.g., they found that the *PrivacyBadger* extension led to a large number of timeouts and therefore to a potentially large number of malfunctioning web sites) [20]. Therefore, it is important for Spartacus to minimize the negative impacts on benign URL visits. Impacts may include the ability to access web sites, the correct display of web site layout, and the correct web site functionality. To evaluate functionality of benign web sites, we conducted two experiments: a Coarse-Grained (a large-scale evaluation with automated analysis) and Fine-Grained (a small-scale evaluation with in-depth manual analysis). In each experiment, for each URL, the Spartacus browser used a profile with a random trigger word, no Referrer header, and no IP proxy.

[4]https://developer.mozilla.org/en-US/docs/Web/API/Long_Tasks_API

Experiment	Tested	Passed	Blocked	Different Layout
W/out Algorithm 1	60,848	60,574	150	124
W/ Algorithm 1	60,848	60,688	29	39

Table 5: Coarse-grained experiment result of Spartacus with and without applying the Logic of mutating HTTP profile discussed in Algorithm 1.

6.6.1 Coarse-Grained Experiment. In the Coarse-Grained experiment, we intended to evaluate if the Spartacus framework has negative impacts on access to the web site or the web site layout through automated analysis of results on a large crawl of benign domains. We randomly sampled 60,848 (9.66%) from the 629,843 URLs in Alexa Top One Million Domain List [1] and visited them in both default and Spartacus browsers. We visited them using both browsers with existing sessions similar to the user's browser. We compared the resulting web page screenshot and HTML similarity on the visited URLs. The result is shown in Table 5: 0.25% (150) have different layouts, and 0.20% (124) block access to the Spartacus browser.

We first manually examined the results to identify why the Spartacus browser shows different layouts from the default browser. We found that although the screenshot and HTML were dissimilar between the default and Spartacus browser visits, such differences did not impact the use of the web site. Figure 8 and Figure 6 show typical differences in browser rendering between the default and Spartacus browser visits. For example, the web page is rendered differently in terms of screenshot similarity between the default and Spartacus browser visits shown in Figure 8. The difference here is due to the shape of the buttons, different background color, and content spacing. In Figure 6, a window popped up to ask for permission to use cookies in the default browser, but did not in Spartacus's visit. The cookie request pop-up that was missing in the Spartacus browser is not due to the extension: in 10 visits in different default browsers, the pop-up appeared only three times.

Evaluation on a larger dataset of benign web sites. To further evaluate Spartacus' potential impact on benign web sites, we conducted an experiment where we visited 629,843 benign web sites in the Alexa Top One Million Domain List. We found that only 3,023 (0.48%) benign web sites either blocked access or showed a different web page layout to the Spartacus browser. This result confirms the prior results that most benign web sites do not block Spartacus' visit and do not deliver a different web page content from a normal visit one to a Spartacus browser.

Benign web sites with security mechanisms. Some benign web sites are built on web hosting services such as Cloudflare and Akamai, and the services contain security mechanisms such as anti-DDoS and anti-crawling. Therefore, to make sure that users can successfully visit those web sites with the protection of Spartacus, we visited 5,000 Cloudflare-based and 5,000 Akamai-based benign web sites using Spartacus. In total, we could successfully visit 99.86% of 10,000 web sites. 14 benign sites were inaccessible. It is mainly because the web site owners employ traffic filtering mechanisms over the CDNs. With such low false-evasion rate, users can visit most benign web sites hosted on the CDNs successfully,

Algorithm 1
Logic of mutating HTTP profile

```
p = default_profile
u = url_to_visit
if
    (reputation(u) ≤ Neutral) and
    (duration(u) ≥ 1,501) or
    (u.domain NOT in top reviewed sub-domains) then
    p = mutate_http_profile(p)
end if
send_request(p)
```

and can report the falsely evaded benign web sites to the Spartacus provider, who can asynchronously inspect them and force Spartacus to use the default profile to visit the web sites.

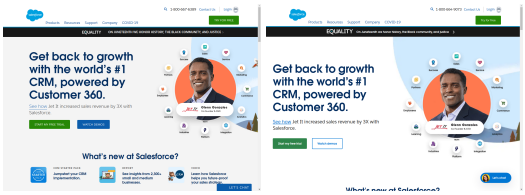
Potential Improvement. Although Spartacus has low false positives when visiting benign web sites, there is still the possibility to reduce false positives. We inspected the 150 benign web sites that were falsely evaded by Spartacus and 150 cloaked phishing web sites that were successfully evaded by Spartacus. To further differentiate the two categories, we extracted domain reputation [37], domain age [7], and top viewed sub-domains [43] of URLs from benign and malicious web sites. Cisco Talos[5] defines reputation of a domain using five categories: Trusted, Favorable, Neutral, Questionable, and Untrusted [37]. Reputation-wise, 136 out of 150 phishing URLs have a reputation lower or equal to Neutral level, the lowest of which is Untrusted. In contrast, only 24 of 150 benign URLs have a lower reputation than Favorable, the lowest of which is Questionable (and only one is Questionable). In terms of domain lifespan, the mean value of domain duration since registration for benign URLs is 4,692 days and the median is 4,521 days. However, the average lifespan of the malicious domains is 1,618 days, with a median of 900 days. Moreover, all 150 benign URLs fall into the top viewed sub-domains of the corresponding domain names, while none of the phishing ones matches.

Therefore, we summarize that a legitimate domain has a higher reputation and longer life than a malicious one, and they are within top viewed sub-domains. We can further reduce the possibility of Spartacus falsely evading benign web sites by introducing another pre-filter to Spartacus before mutating the HTTP profile.

We choose the phishing domain age that resides on 75% in the list (1,501) and the Neutral level as thresholds because these thresholds clearly divided trustworthy domains from un-trustworthy domains. With this pre-filter, if a URL has a lifespan lower than 1,501 days and its reputation level is Neutral or worse, or its sub-domain is not top viewed, then Spartacus will mutate the HTTP profile. The logic is shown in Algorithm 1.

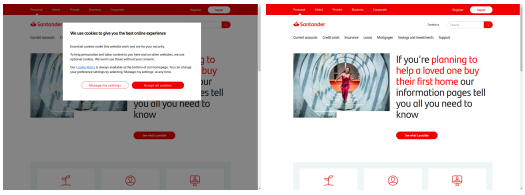
We evaluated this augmented version of Spartacus and found that 29 legitimate domains show different web page content on the default and augmented version of Spartacus browsers, as listed in Table 5. Hence, only 0.04% of 60,848 domains result in a false positive detection of a phishing web site. This is because the 29 domains do not meet the trustworthiness requirements and therefore Spartacus mutated the profile when visiting them. As one possible mitigation, we can provide a channel for users to report falsely evaded web sites. After receiving a report, we can conduct a manual inspection and force Spartacus to trust the false-positives. In comparison, phishing URLs can still be evaded through Spartacus because they pass through the augmented pre-filter.

[5]Cisco Talos is a threat intelligence service and used by other studies [26, 27, 31].



(a) Default browser visit. (b) Spartacus visit

Figure 5: Difference due to the shape of buttons.



(a) Default browser visit. (b) Spartacus visit

Figure 6: Difference due to popup.

Evaluation Perspective		Amount	
		Default	Spartacus
Accessibility		58	58
Correct Layout		58	58
Proper Functionality	Click Buttons	58	58
	Online Chat	3	3
	Shopping Cart Add	5	5
	Registration/Login	22/22	22/22

Table 6: Fine-grained experiment result of Spartacus visit, compared with the result of default browser visit.

6.6.2 Fine-Grained Experiment. Inspired by the methodology used by Snyder et al. [34] and Trickel et al. [40], in the Fine-Grained experiment, we aim to manually evaluate the operation of web sites visited through Spartacus.

This methodology concentrates on the operation of a web site from the perspective of the user. Even though Spartacus may introduce an error to a web site, if the users do not perceive any difference when browsing, then we consider that Spartacus does not negatively impact the web site. This methodology focuses on user-facing impacts to benign web site functionality, and the experiment was performed by the authors manually.

The experiment includes the evaluation of a web site's rendering and interactions between visitors and the web site. There are four steps in the experiment methodology: (1) Open legitimate domains in a browser with Spartacus installed and also in a browser with default settings. (2) Inspect the accessibility of the web site, similar to the Coarse-Grained experiment. (3) After successful web page content retrieval, compare the layouts between different visits. (4) Interact with links, buttons, and other activities such as register/login, online chat, or shopping, as long as the web site allows us to do so, to ensure that the web site performs correctly. (5) Finally,

test the authentication functionality to ensure that Spartacus will not impact it.

We randomly selected 60 domains from the Alexa Top One Million List, choosing 20 every 200,000 (for an even distribution), and the result is displayed in Table 6. As a comparison, the default browser had the same result as that of Spartacus. Among the 60 legitimate domains, 58 were accessible. Two domains were inaccessible even in the default browser, so we suspect that they are offline. For the 58 accessible domains, we followed the steps described previously to inspect them. All of them have the same layout as the visit from the default browser. Then, we interacted with the 58 web sites by clicking buttons, chatting online, and adding items into the cart if it is a shopping web site. All 58 web sites performed normally. Lastly, we registered an account on 15 web sites, and all were successful in Spartacus. Even though we can successfully register an account, we still need to make sure that we can log in properly with those accounts to test the authentication process under Spartacus. The results shows that all the accounts we registered during the Fine-Grained experiment could be logged in successfully.

With the experimental results from both the Coarse- and Fine-Grained experiments, we can summarize that Spartacus has little impact on the accessibility and visibility on benign web sites. Therefore, Spartacus can protect users from visiting advanced phishing web sites while keeping their normal browsing activities.

Trigger words on benign web sites. When browsing benign web sites, Spartacus can think that the visit resulted in evasion failure, and then mutate future visits with new trigger words. Therefore, we conducted an experiment to show that these trigger words do not impact the layout and functionality of benign web sites. We tested the top 10 popular trigger words in Table 2 on 30 randomly selected benign domains. We then compared the screenshot and HTML similarity between the default browser visit and 10 Spartacus browser visits using the different trigger words. The result shows that each of the 30 benign domains have the same layout using different trigger words. Additionally, we manually tested the functionality as in the Fine-Grained experiments and found that all web sites performed correctly with different trigger words.

Benign web sites with risk-based authentication mechanism. Risk-based authentication (RBA) mechanism prevents web sites from requiring users to use Two-Factor Authentication by inspecting the features in an HTTP request such as IP addresses and/or User-Agent string [46]. As major web sites such as Amazon, Google, LinkedIn, and Facebook have employed such a mechanism [46], we evaluate Spartacus's ability to trigger Two-Factor Authentication in RBA enhanced web sites. To this end, we visited eight web sites that are known to employ RBA using Spartacus[6]. These web sites cover all three types of RBA implementations mentioned in prior work [46]. The result shows that we could successfully visit all eight web sites without encountering a Two-Factor Authentication prompt, which demonstrates that Spartacus does not cause inconvenience for users visiting RBA protected benign web sites.

6.6.3 Long-term Use. To determine long-term impact on the user experience, we evaluate how Spartacus performs in a long term usage scenario. The authors installed Spartacus in their primary browsers for daily use for a period of one month. During the experiment, we looked for abnormalities, such as unexpected access blocking, frequent risk-based authentications (e.g., reCAPTCHA and two-factor authentication), and slow page rendering. After the one-month experiment, the authors did not encounter with any abnormal actions during normal browsing.

[6] Amazon, Facebook, GOG.com, Google, iCloud, LinkedIn, Steam, and Twitch [46].

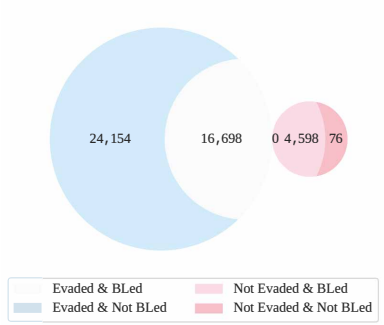


Figure 7: Venn Diagram describing the ability of Spartacus and current anti-phishing systems against phishing web sites. The unit is the amount of phishing web sites.

6.6.4 Reason of Low Breakage. In both Fine-Grained and Coarse-grained experiments, the Spartacus browser could successfully request and render benign web sites and allow users to interact with them as usual. It is mainly because we only append one of the trigger words in the User-Agent string, instead of replacing the string with a crawler one. As we discovered in section 3, phishing servers will deny the request because they employ an aggressive filtering mechanism—blocking access as long as there is any suspicious patterns in the User-Agent string. However, benign web sites perform anti-crawling in a different way. For example, they monitor new or existing user accounts with high levels of activity and no purchases, or they detect abnormally high volumes of product views as a sign of non-human activity [8]. Additionally, benign web apps such as *WordPress* [47] check the User-Agent string mainly because they need the visitor's browser version to deliver the best web page layout and according functionalities. Therefore, the difference between benign and malicious server's anti-crawling mechanisms allows Spartacus to evade phishing sites and access benign ones.

7 ECOSYSTEM SUPPORT

Note that in our experiments (detailed in Section 6.1) Spartacus cannot evade 17.72% of the URLs. We hypothesize that these phishing web sites do not rely on fingerprinting-based cloaking techniques. Nowadays, the anti-phishing ecosystem such as Google Safe Browsing and VirusTotal can quickly detect and/or blacklist such phishing attacks. Therefore, we can rely on support from the ecosystem to handle the phishing web sites that Spartacus cannot evade.

To verify our hypothesis, we evaluated the blacklist/detection speed of current anti-phishing systems on the examined phishing URLs. We submitted the phishing URLs to Google Safe Browsing and VirusTotal at the same time that Spartacus visited them, and

queried the results every 15 minutes to calculate the speed. Due to the deployment time of this experiment, we submitted 45,526 phishing URLs: 40,852 that Spartacus could evade and 4,674 that Spartacus could not evade. Among the 4,674 submitted phishing URLs that Spartacus could not evade, Google Safe Browsing and VirusTotal combined blacklisted 4,598 of them. The remaining 76, after manual inspection, were found that they were not phishing web sites and were falsely reported to APWG. This evaluation result verifies that the ecosystem currently can protect users from phishing web sites that Spartacus cannot evade, which acts as a complement to Spartacus. In contrast, we submitted 40,852 phishing web sites that Spartacus successfully evaded. This result shows that 24,154 of them were not detected or blacklisted by the anti-phishing systems.

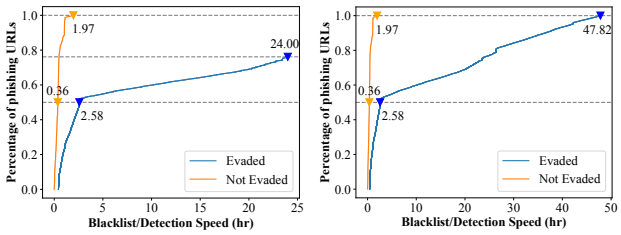
Figure 7 overviews the ability of Spartacus along with support from the anti-phishing ecosystem. Within our dataset, advanced phishing web sites significantly outnumber basic ones, which is shown in Figure 7 as the blue circle and red circle, respectively. However, the anti-phishing ecosystem detects only 40.87% of the submitted evaded phishing URLs. As a comparison, Spartacus alone can evade 89.73% of submitted phishing URLs. The ecosystem can mostly handle the non-evaded phishing URLs as a complement to Spartacus, and hence both advanced and basic phishing web sites can be evaded or detected.

We also measure the detection speed of current anti-phishing systems, which is visualized in Figure 8a (within 24 hours) and Figure 8b (whole frame). All submitted phishing web sites that Spartacus cannot evade are detected/blacklisted in two hours, and 50% of these can be detected within 22 minutes. As a comparison, current anti-phishing systems do not perform well against phishing web sites that can be evaded by Spartacus. The median detection time is 154 minutes. However, within 24 hours, they only detect/blacklist 76.16% of the cloaked phishing web sites. Moreover, it can take as long as 47.82 hours to finally detect a cloaked phishing web site. As a comparison, Spartacus provides a nearly real-time protection against cloaked phishing web sites. This reflects the ability of the current anti-phishing ecosystem against phishing web sites: for basic phishing web sites, current anti-phishing systems can react and blacklist them quickly, however, for advanced phishing, it takes a long time to detect, which is exploited by phishers to victimize users. Spartacus, however, only needs an average of 3.2 seconds (based on Table 4) to evade advanced phishing web sites. Therefore, Spartacus can not only greatly shorten the *golden hour* [25] left by the current anti-phishing ecosystem, but also evade cloaked phishing web sites that the ecosystem cannot detect.

8 MITIGATING SERVER-SIDE CLOAKING

According to our observation and analysis in Section 3, phishers use fingerprinting-based cloaking techniques to accomplish their phishing attacks. We expect that the sophistication of phishing web sites will continue to improve, and that advanced phishing kits will create more fingerprints to inspect, match, and block requests that appear to be from anti-phishing entities.

Although researchers and organizations have proposed mitigations for phishing web sites [19, 50], they all require malicious web page content to feed the classifier and make decisions. Server-side



(a) CDF of Blacklist/Detection time within *24 hours* of current anti-phishing systems against detected phishing URLs evaded and not evaded by Spartacus.

(b) CDF of Blacklist/Detection time of current anti-phishing systems against systems against detected phishing URLs evaded and not evaded by Spartacus.

Figure 8: CDF of Blacklist/Detection time.

Model	Precision (%)	Recall (%)	FPR (%)	FNR (%)	F1	ACC (%)
CANTINA+	95.45	54.60	2.60	45.40	0.69	76.00

Table 7: Evaluation metrics of CANTINA+ [50] against cloaked phishing websites.

cloaking techniques deny requests from the anti-phishing systems, and their methodologies may not be useful. To demonstrate this, we selected 500 cloaked phishing web sites that can be evaded by Spartacus and 500 benign web sites from the Alexa Top One Million List to test our implementation of CANTINA+ [50], which is a phishing classifier with URL-, web-, and HTML-based features. Specifically, we re-implemented CANTINA+ to perform the classification of returned web page content. We then collected the features that CANTINA+ needs and used the machine learning algorithm proposed in the original paper for training and testing. The result, depicted in Table 7, shows a high false-negative rate when classifying cloaked phishing web sites. Therefore, the ecosystem should ensure that existing and new detection and mitigation systems are capable of adapting to fingerprinting-based cloaking techniques.

While Spartacus attempts to modify the user's HTTP requests to appear as anti-phishing systems, to best combat server-side cloaking anti-phishing systems should carefully modify the HTTP requests of their crawlers to mimic users. In this case, the anti-phishing systems can bypass the cloaking techniques and retrieve the actual malicious content. For example, they should avoid sending requests based on the IP addresses of well-known anti-phishing entities.

Additionally, Spartacus can be extended to share resources among users. Instead of only locally recording visited URLs and successfulness of profile mutations, Spartacus can merge the visit history from users in a centralized server. Then, the server can distribute and update periodically to the clients. In this way, users can benefit from an up-to-date Fingerprinting Database because Spartacus knows to block access if it ever successfully evaded the web site from other users. Furthermore, the Fingerprinting Database is designed to minimize privacy issues in this centralized setting, because all URLs are hashed and only the triggering words and proxy server IP are stored in the database.

9 COUNTERMEASURES TO SPARTACUS

Even though Spartacus can successfully prevent users from seeing phishing content in fingerprinting-cloaked phishing web sites, attackers will explore countermeasures to mitigate it.

9.1 Using Other Cloaking Techniques

There are different types of phishing web sites in the wild, roughly categorized into basic and advanced. Within advanced phishing web sites, server-side and client-side cloaking techniques are two techniques that help evade detection from the anti-phishing ecosystem. Because Spartacus focuses on the evasion of phishing attacks with server-side cloaking techniques, attackers can use other types of cloaking techniques to harm individuals and organizations.

Phishers can use basic phishing web sites, phishing web sites with client-side cloaking, or those with User Interaction Cloaking (e.g., CAPTCHAs). As we presented in Section 7 and with the analysis results from Oest et al. [24], basic phishing web sites can be quickly detected and blacklisted by anti-phishing systems in a median of 28 minutes. As for client-side cloaking techniques, phishers can implement them into their web sites to bypass Spartacus. However, Zhang et al. [52] proposed a methodology to detect such evasion by force-executing JavaScript. Hence, client-side cloaked phishing web sites can also be detected using prior techniques.

Finally, phishers can create a CAPTCHA web page as a barrier before showing phishing content. Such a technique can bypass evasion from Spartacus. However, using CAPTCHA may lower phishers' profit because it is time consuming and challenging for potential victims [30]. Secondly, it does not distinguish real users from anti-phishing crawlers because every visitor needs to solve the puzzle [30]. Recently, researchers have proposed methodologies of bypassing CAPTCHAs [33, 36, 52], which further allows anti-phishing crawlers to bypass them.

In a future with Spartacus deployed and support from the anti-phishing ecosystem, it is challenging to bypass all anti-phishing methodologies while allowing only potential victim traffic to visit. Such dilemmas force attackers to either spend resources inventing new evasion techniques or accept reduced profit.

9.2 Emerging Phishing Based on Spartacus

We assume that phishers can gain full knowledge of Spartacus and develop countermeasures accordingly.

For example, phishers could develop stateful server-side cloaking techniques: allowing the first person with a matching template to evade the phishing content, then change the cloaking so that future visits would see phishing content. This could affect the second user who visits the same phishing web site, because Spartacus uses the "successful" profile for the user. Hence, the user views the phishing content due to the stateful nature of the server-side cloaking. However, by design, the suspicious content classification module is run externally and determines whether the returned web page has suspicious content, no matter if Spartacus mutates the profile or uses a successful one. Therefore, when the classification module determines that the profile is unsuccessful to evade phishing, Spartacus will mark it as failed and will select a new one in the future.

As another example, phishers could enumerate the bot profiles in Spartacus and develop high-fidelity cloaking techniques which identify anti-phishing ecosystem HTTP profiles more precisely. For example, phishers could only cloak visits that contain the exact User-Agent string of anti-phishing crawlers[7]. This technique could bypass Spartacus's evasion. However, these precise fingerprints increase the ease of anti-phishing systems to successfully bypass the cloaking by trivially changing their User-Agent string. Finally, phishers could develop complex and advanced fingerprinting techniques to use fingerprints that Spartacus does not consider, such as the order of HTTP headers, TCP/IP fingerprints, support for esoteric HTTP features (e.g., supporting the `100 Continue` response code), timing side-channels, and so on. While some of these fingerprints could be added to Spartacus, it might not be technically feasible to add all of them. Therefore, we could work with anti-phishing entities (or they could deploy Spartacus themselves) to integrate the exact Spartacus framework into their anti-phishing systems, so that these low-level fingerprints would be identical to Spartacus users.

10 LIMITATIONS

Even though Spartacus can protect users from a diverse array of sophisticated phishing web sites using server-side cloaking techniques in the wild, our framework should be considered alongside certain limitations.

10.1 Spartacus Design

Phishing Classification. The Spartacus framework is not a phishing classification system. Instead, it camouflages users as security crawlers when they visit web sites with cloaking techniques and can evade malicious content if they use fingerprinting-based cloaking. This approach proactively protects users in nearly real time. As evaluated in section 6, Spartacus can evade 82.28% of phishing web sites in real time using only User-Agent and Referrer mutation, with a negligible impact on benign web sites. Previous work has proposed methodologies classifying phishing web sites with high accuracy [19, 45]. Therefore, with Spartacus and existing classification methodologies, the anti-phishing ecosystem can cover a broader range of phishing attacks.

HTTP request mutation. As discussed in section 2, fingerprinting-based cloaking techniques can inspect IP, Hostname, User-Agent, Referrer, and other fingerprints to classify whether the visitor is an anti-phishing crawler or a potential victim. In Spartacus's design, we consider mutating User-Agent and Referrer in the HTTP request, along with changing the IP address using proxy servers. There is a limitation where Spartacus cannot evade phishing web sites that only identify crawlers/bots by new fingerprints that Spartacus does not mutate. One solution for the potential limitation is that we intentionally designed Spartacus as an extendable framework. In this case, Spartacus can remain up-to-date to evade new cloaked phishing web sites.

[7] E.g., phishers only block visits whose User-Agent perfectly matches *Mozilla/5.0 (Linux; Android 5.0; SM-G920A) AppleWebKit (KHTML, like Gecko) Chrome Mobile Safari (compatible; AdsBot-Google-Mobile; +http://www.google.com/mobile/adsbot.html).*

10.2 Spartacus Deployment and Evaluation

Phishing kit analysis. In the analysis to understand the prevalence of fingerprinting-based cloaking, we hope to include as many phishing kits as possible. Due to resource limitations, we only analyzed phishing kits from *phishunt.io* [28] and those from the public dataset from Cisco. Within both resources, we analyzed 2,933 phishing kits and summarized that the fingerprinting-based cloaking techniques exist in 96.52% of the phishing kits. We believe that this analysis demonstrates the prevalence of fingerprinting-based cloaking techniques.

Data collection. We selected the APWG Dataset to evaluate the effectiveness of Spartacus. Due to infrastructure and resource limitations, we were only able to test Spartacus over total of nine months from November 2020 to July 2021. Even though additional data crawling would be desirable to evaluate Spartacus, the APWG Dataset provides a breadth of phishing data collection because it contains diverse types of phishing web sites targeting different brands. The phishing web sites are submitted periodically by collaborating members including anti-phishing systems and financial organizations impersonated by phishing web sites. Overall, we tested Spartacus on over 130,000 live phishing web sites and verified that Spartacus could evade malicious content by triggering fingerprinting-based cloaking. Therefore, we believe this limitation is mitigated to the extent allowed by current resources.

11 RELATED WORK

Researchers have studied phishing for several decades. They have proposed several methodologies to detect phishing attacks based on features from the URL, content, etc., and then warn users before visiting the deceptive web sites. Some work analyzes the URL of a suspicious web site based on the lexical features or URL ranking to determine the maliciousness of the site [5, 10, 15, 17]. Others collect web page content and detect phishing web sites with textual and visual similarity features [9, 51, 53]. Combining these techniques with other available features, including both URL and web page content, researchers developed blacklist-based anti-phishing systems such as Google Safe Browsing [45] to protect users from visiting suspicious web sites. All the proposed methodologies in the past, however, have a limitation that they are detection systems, and require certain features to classify the maliciousness, which can take time to acquire (due to cloaking). Furthermore, Bijmans et al. [2] found that the median uptime for phishing domains is 24 hours, showing the fast move of phishers. The delay of detection from anti-phishing systems and quick action of phishers extends the gap to mitigating phishing attacks [25].

With the large-scale implementation of cloaking techniques in phishing attacks [22–25], researchers realize that sophisticated phishing attacks are responsible for a substantial portion of damage and that the whole ecosystem should prioritize mitigating phishing with evasion techniques. Cloaking techniques make anti-phishing more challenging because it becomes more and more difficult to retrieve the phishing content, which most anti-phishing systems depend on. With a very limited amount of web site features, current anti-phishing systems cannot precisely determine the maliciousness.

Therefore, analysis and detection of server-side and client-side cloaking techniques have been proposed to fight against such sophistication. For client-side cloaking techniques, Zhang et al. [52] proposed CrawlPhish to force-execute JavaScript snippets in the HTML response to reveal malicious content. As for server-side cloaking in phishing, previous work [14, 23, 42] categorizes types of server-side cloaking through analysis of compromised phishing kits.

We consider the nature and prevalence of cloaked phishing web sites [22, 23] and provide a novel methodology to proactively prevent users from seeing phishing content. Rather than bypass cloaking techniques in phishing web sites, Spartacus deliberately triggers them and hence prevents users. Our framework is also extensive with the ability to add fingerprints that phishers use in the future.

12 CONCLUSION

Through our analysis of compromised phishing kits, we understand that fingerprinting-based cloaking techniques are largely implemented in the sophisticated phishing attacks and help to evade visits from anti-phishing entities. Such evasion is difficult to mitigate because phishers can always include new features of the up-to-date anti-phishing crawlers and identify them.

We consider this problem from a different perspective. The current state of the ecosystem is that the anti-phishing entities identify the fingerprints that phishing kits use to trigger cloaking and design new crawlers without those fingerprints, which are then learned by phishers. The phishers then add new fingerprints—in a never-ending cycle.

The Spartacus system proposes a new angle for the anti-phishing ecosystem to fight against cloaked phishing web sites. Rather than attempting to circumvent cloaking techniques, Spartacus beats the advanced phishing web sites at their own game by deliberately triggering the cloaking behavior. For benign web sites, we demonstrated that Spartacus has negligible impact on users' access, web layout, and web functionality.

Due to the rise of sophisticated phishing web sites in the wild, we believe that automated evasion systems such as Spartacus are essential to keep trapping phishers in a lose-lose dilemma where they cannot differentiate real users from anti-phishing crawler visits. Methodologies such as ours can be incorporated into the ecosystem to more expeditiously and reliably evade sophisticated phishing, thus proactively protecting users from phishing attacks.

ACKNOWLEDGMENTS

Many thanks to the anonymous referees for their thoughtful reviews. We would also like to thank our shepherd, Pierre Laperdrix.

This material is based upon work supported in part by Army Research Office (ARO) Grant No. W911NF17-1-0370, Defense Advanced Research Projects Agency (DARPA) Grant No. N66001-20-C-4020 and HR00112190093, and the Korea Internet & Security Agency (KISA) grant funded by the Personal Information Protection Commission (PIPC) (No. 1781000003). Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of United States Government or any agency thereof.

REFERENCES

[1] Amazon. 2021. Alexa Top Sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[2] 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)*. 3757–3774.

[3] Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding Malicious Domains Using Passive DNS Analysis.. In *Ndss*. 1–17.

[4] Sun Bin, Wen Qiaoyan, and Liang Xiaoying. 2010. A DNS based anti-phishing approach. In *2010 Second International Conference on Networks Security, Wireless Communications and Trusted Computing*, Vol. 2. IEEE, 262–265.

[5] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. 2010. Lexical feature based phishing URL detection using online learning. In *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*. 54–60.

[6] Davide Canali, Davide Balzarotti, and Aurélien Francillon. 2013. The role of web hosting providers in detecting compromised websites. In *Proceedings of the 22nd international conference on World Wide Web*. ACM, 177–188.

[7] Danny Cork. 2021. A Python package for retrieving WHOIS information of domains. https://github.com/DannyCork/python-whois.

[8] DataDome. 2019. Web scraping protection: How to protect your website against crawler and scraper bots. https://datadome.co/bot-management-protection/scraper-crawler-bots-how-to-protect-your-website-against-intensive-scraping/#2.

[9] Matthew Dunlop, Stephen Groat, and David Shelly. 2010. Goldphish: Using images for content-based phishing analysis. In *2010 Fifth international conference on internet monitoring and protection*. IEEE, 123–128.

[10] Mohammed Nazim Feroz and Susan Mengel. 2015. Phishing URL detection using URL ranking. In *2015 ieee international congress on big data*. IEEE, 635–638.

[11] Google. 2019. Google Transparency Report. (2019). https://transparencyreport.google.com/safe-browsing/overview?hl=en.

[12] 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*. 1273–1290.

[13] Huajun Huang, Liang Qian, and Yaojun Wang. 2012. A SVM-based technique to detect phishing URLs. *Information Technology Journal* 11, 7 (2012), 921–925.

[14] 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)*. IEEE, 743–758.

[15] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. 2012. Enhancing phishing e-mail classifiers: A lexical url analysis approach. *International Journal for Information Security Research (IJISR)* 2, 1/2 (2012), 40.

[16] Mahmoud Khonji, Andrew Jones, and Youssef Iraqi. 2011. A novel Phishing classification based on URL features. In *2011 IEEE GCC conference and exhibition (GCC)*. IEEE, 221–224.

[17] Anh Le, Athina Markopoulou, and Michalis Faloutsos. 2011. Phishdef: Url names say it all. In *2011 Proceedings IEEE INFOCOM*. IEEE, 191–195.

[18] 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.

[19] 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)*.

[20] Georg Merzdovnik, Markus Huber, Damjan Buhov, Nick Nikiforakis, Sebastian Neuner, Martin Schmiedecker, and Edgar Weippl. 2017. Block me if you can: A large-scale study of tracker-blocking tools. In *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 319–333.

[21] 2019. Windows Defender SmartScreen. (2019). https://github.com/MicrosoftDocs/windows-itpro-docs/blob/public/windows/security/threat-protection/windows-defender-smartscreen/windows-defender-smartscreen-overview.md.

[22] 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 *40th*. Oakland, CA, 764–781.

[23] Adam Oest, Yeganeh Safaei, 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.

[24] 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.

[25] 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)*.

[26] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. 2018. Made: Security analytics for enterprise threat detection. In *Proceedings of the 34th Annual Computer Security Applications Conference*. 124–136.

[27] 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.

[28] Phishunt. 2021. Exposing phishing kits seen from phishunt.io. https://github.com/danlopgom/phishing_kits.

[29] Privacy Policies. 2021. #1 Privacy Policy Generator - Privacy Policies . https://www.privacypolicies.com/.

[30] radware bot manager. 2021. How CAPTCHA Is Used To Block Bots, And Why We Do Not Recommend Using It. https://www.radwarebotmanager.com/when-to-use-and-when-not-to-use-captcha/.

[31] Arya Renjan, Karuna Pande Joshi, Sandeep Nair Narayanan, and Anupam Joshi. 2018. Dabr: Dynamic attribute-based reputation scoring for malicious ip address detection. In *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE, 64–69.

[32] Foy Shiver. 2016. APWG and the eCrime Exchange: A Member Network Providing Collaborative Threat Data Sharing. https://www.first.org/resources/papers/valencia2017/shiver-foy_slides.pdf.

[33] Suphannee Sivakorn, Jason Polakis, and Angelos D Keromytis. 2016. I'm not a human: Breaking the Google reCAPTCHA. *Black Hat* (2016), 1–12.

[34] Peter Snyder, Cynthia Taylor, and Chris Kanich. 2017. Most websites don't need to vibrate: A cost-benefit approach to improving browser security. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 179–194.

[35] Verizon Enterprise Solutions. 2019. Data Breach Investigations Report (DBIR). (2019).

[36] Fabian Stark, Caner Hazırbas, Rudolph Triebel, and Daniel Cremers. 2015. Captcha recognition with active deep learning. In *Workshop new challenges in neural computation*, Vol. 2015. Citeseer, 94.

[37] Cisco Talos. 2021. IP & Domain Reputation Center. https://www.cisco.com/c/en/us/products/security/talos.html.

[38] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. 2017. Data breaches, phishing, or malware?: Understanding the risks of stolen credentials. In *Proceedings of the 2017 ACM SIGSAC conference on computer and communications security*. ACM, 1421–1434.

[39] Treo. 2020. Exthouse: Analyze the impact of a browser extension on web performance. . https://github.com/treosh/exthouse.

[40] Erik Trickel, Oleksii Starov, Alexandros Kapravelos, Nick Nikiforakis, and Adam Doupé. 2019. Everyone is different: Client-side diversification for defending against extension fingerprinting. In *28th USENIX Security Symposium (USENIX Security 19)*. 1679–1696.

[41] Amber Van Der Heijden and Luca Allodi. 2019. Cognitive triaging of phishing attacks. In *28th USENIX Security Symposium*. 1309–1326.

[42] David Y Wang, Stefan Savage, and Geoffrey M Voelker. 2011. Cloak and dagger: dynamics of web search cloaking. In *Proceedings of the 18th ACM conference on Computer and communications security*. 477–490.

[43] WEBrate. 2022. Webrate.org - Rate the web. https://webrate.org/.

[44] Tech Blog (wh). 2012. Most Common User Agents. https://techblog.willshouse.com/2012/01/03/most-common-user-agents/.

[45] Colin Whittaker, Brian Ryner, and Marria Nazif. 2010. Large-scale automatic classification of phishing pages. (2010).

[46] Stephan Wiefling, Nils Gruschka, and Luigi Lo Iacono. 2019. Even Turing Should Sometimes Not Be Able To Tell: Mimicking Humanoid Usage Behavior for Exploratory Studies of Online Services. In *24th Nordic Conference on Secure IT Systems (NordSec 2019)* (Aalborg, Denmark) *(Lecture Notes in Computer Science, Vol. 11875)*. Springer Nature, 188–203. https://doi.org/10.1007/978-3-030-35055-0_12

[47] wordpress.org. 2022. WordPress Source Code. https://github.com/WordPress/WordPress.

[48] Baoning Wu and Brian D Davison. 2005. Cloaking and Redirection: A Preliminary Study.. In *AIRWeb*. 7–16.

[49] Min Wu, Robert C Miller, and Greg Little. 2006. Web wallet: preventing phishing attacks by revealing user intentions. In *Proceedings of the second symposium on Usable privacy and security*. ACM, 102–113.

[50] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. 2011. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)* 14, 2 (2011), 21.

[51] Haijun Zhang, Gang Liu, Tommy WS Chow, and Wenyin Liu. 2011. Textual and visual content-based anti-phishing: a Bayesian approach. *IEEE transactions on neural networks* 22, 10 (2011), 1532–1546.

[52] Penghui Zhang, Adam Oest, Haehyun Cho, Zhibo Sun, 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)*. San Francisco, CA.

[53] Yue Zhang, Jason I Hong, and Lorrie F Cranor. 2007. Cantina: a content-based approach to detecting phishing web sites. In *Proceedings of the 16th international conference on World Wide Web*. 639–648.