Are You Human? Resilience of Phishing Detection to Evasion Techniques Based on Human Verification

Sourena Maroofi
sourena.maroofi@
univ-grenoble-alpes.fr
Univ. Grenoble Alpes, CNRS,
Grenoble INP, LIG
France

Maciej Korczyński
maciej.korczynski@grenoble-inp.fr
Univ. Grenoble Alpes, CNRS,
Grenoble INP, LIG
France

Andrzej Duda
andrzej.duda@grenoble-inp.fr
Univ. Grenoble Alpes, CNRS,
Grenoble INP, LIG
France

ABSTRACT

Phishing is one of the most common cyberattacks these days. Attackers constantly look for new techniques to make their campaigns more lucrative by extending the lifespan of phishing pages. To achieve this goal, they leverage different anti-analysis (i.e., evasion) techniques to conceal the malicious content from anti-phishing bots and only reveal the payload to potential victims. In this paper, we study the resilience of anti-phishing entities to three advanced anti-analysis techniques based on human verification: Google re-CAPTCHA, alert box, and session-based evasion. We have designed a framework for performing our testing experiments, deployed 105 phishing websites, and provided each of them with one of the three evasion techniques. In the experiments, we report phishing URLs to major server-side anti-phishing entities (e.g., Google Safe Browsing, NetCraft, APWG) and monitor their occurrence in the blacklists. Our results show that Google Safe Browsing was the only engine that detected all the reported URLs protected by alert boxes. However, none of the anti-phishing engines could detect phishing URLs armed with Google re-CAPTCHA, making it so far the most effective protection solution of phishing content available to malicious actors. Our experiments show that all the major server-side anti-phishing bots only detected 8 out of 105 phishing websites protected by human verification systems. As a mitigation plan, we intend to disclose our findings to the impacted anti-phishing entities before phishers exploit human verification techniques on a massive scale.

CCS CONCEPTS

• **Security and privacy → Phishing**;

ACM Reference Format:
Sourena Maroofi, Maciej Korczyński, and Andrzej Duda. 2020. Are You Human? Resilience of Phishing Detection to Evasion Techniques Based on Human Verification. In *ACM Internet Measurement Conference (IMC '20), October 27–29, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3419394.3423632

IMC '20, October 27–29, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8138-3/20/10…$15.00
https://doi.org/10.1145/3419394.3423632

1 INTRODUCTION

Phishing is a form of social engineering with the goal of collecting credentials of end-users usually achieved by either email spoofing [1] or directly luring victims to enter their sensitive information into a fake website that matches the look and feel of the legitimate one [2]. There have been much research [3–7] and industry efforts to combat phishing, just to mention the Anti-Phishing Working Group (APWG) [8], PhishTank [9], or recently formed the COVID-19 Cyber Threat Coalition [10]. Nevertheless, according to the latest report from IBM X-force, phishing is the number one initial infection vector among attackers [11].

As with any other cybercriminal activity, phishers and security organizations are constantly in battle. While phishers try to develop new or misuse the existing techniques such as URL shorteners [12] to make their attacks more effective, anti-phishing organizations try to adapt their methods to detect phishing attacks swiftly. One of the most effective techniques used by miscreants is the *anti-analysis* also known as *evasion* [13]. The term refers to a wide range of techniques used by attackers to prevent automatic threat analysis [14]. While these techniques are very popular among malware developers [15], phishers also begin to use them [16, 17]. Malicious actors leverage evasion techniques to tell anti-phishing bots and humans apart. If the end-user is human, they reveal the malicious payload while for anti-phishing bots, they deliver a benign page to evade detection.

Previous work analyzed the existing and well known evasion techniques such as web-cloaking [18], URL redirection [19], using URL shorteners [12], and code obfuscation [20]. These techniques can affect the detection time, yet all major anti-phishing systems can cope with them [21]. For example, to detect web-cloaking, we can send two requests to the server, one with a user-agent related to a known anti-phishing bot (e.g., *googlebot*) and the other one with a typical browser user-agent (e.g., Mozilla Firefox). If the destination of the two requests are different (e.g., different domains), then we can infer the existence of web-cloaking [22].

However, we have recently observed more advanced anti-analysis techniques that can severely impede the performance of phishing detection systems or heavily extend the lifespan of phishing attacks. Human verification is one such technique that not only hampers the verification of phishing content but may also induce users to visit a phishing site due to the use of CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) on benign websites [23]. CAPTCHA is considered an effective anti-bot protection solution (e.g., against page scraping) but it is also misused by cybercriminals to protect their malicious pages from security

organizations. A real-world example of such a usage is a recently detected phishing campaign with more than 128,000 emails sent to victims with a link to a fake Microsoft login page protected by Google reCAPTCHA [24].

In this paper, we study three advanced anti-analysis techniques observed in real-world phishing attacks: Google reCAPTCHA [25], alert box, and session-based evasion techniques. To evaluate the effectiveness of the three techniques on anti-phishing bots, we have designed an experiment in which we emulate the operation of phishing websites—we register 105 domains, deploy harmless phishing websites, and protect each of them with one of the three evasion techniques. Then, we evaluate the detection performance of seven major anti-phishing engines: the Google Safe Browsing (GSB), Anti-Phishing Working Group (APWG), NetCraft, OpenPhish, PhishTank, Microsoft Defender SmartScreen, and Yandex Safe Browsing (YSB), as well as six most popular client-side anti-phishing extensions for Mozilla Firefox including the Avast Online Security and Avira Browser Safety. We show that almost all anti-phishing engines do not detect the phishing websites and mark the URLs as benign.

Our main contributions are as follows:

(1) we qualitatively analyze a new category of anti-analysis techniques observed on real-world phishing websites,
(2) we design a semi-automated and scalable framework for experimentally testing the evasion techniques (it is available on request for researchers to support reproducibility),
(3) we empirically study the effects of three human verification evasion techniques on major server-side and client-side anti-phishing engines,
(4) we show that all the major server-side anti-phishing bots only detected 8 out of 105 phishing websites protected by human verification systems.

2 PHISHING DETECTION AND ANTI-ANALYSIS TECHNIQUES

To better understand the three evasion techniques, we explain them in detail and discuss the possible approaches for phishing detection. Note that we have observed examples of phishing pages armed with all three techniques in real-world attacks (see Figures 1, 2, and 3 in Appendix A). The main source of our data is the *unverified* phishing section of the PhishTank [9] where the submitted URLs are not directly published as phishing but instead are pending for 'voters' to manually verify them as phishing URLs or false positives [26].

2.1 Server-Side vs. Client-Side Detection

Phishing page detection involves either client-side or server-side systems. In the server-side approach (e.g., GSB), the candidate URL is sent to the server (either by direct human report or by automatic URL collection by crawling). With the URL, the server starts to collect data (e.g., page content, domain content, lexical features, etc.) and decides whether it is malicious or benign. The client-side detection systems (e.g., browser add-ons and extensions) have direct access to the URLs and page content that end-users visit. Therefore, it is possible to collect features from the visited pages and decide on the phishing character of the pages either directly on the client-side or later on by sending data to server-side systems.

Each detection type has its advantages and drawbacks. The server-side approach can be used globally by every browser that can interact with the system using an application programming interface (API). The internal detection algorithm appears as a black-box to attackers. Users' privacy is preserved by sending the hashed version of the URLs to the server. The drawback is that there is no guarantee that, having the same URL, the server and the client fetch the same content. If the attacker can detect the anti-phishing bot (e.g., IP-based or user-agent detection [21]), it will serve benign content to the bot.

Client-side phishing detection extensions do not suffer from this problem. They have access to the very same content as users visit because they are installed on browsers. However, the lack of sufficient support for extensions in different browsers [27], the inconsistency between desktop and mobile phone browsers [21], and privacy concerns related to third-party extensions [28] are the factors that affect the popularity of client-side extensions. In this paper, we study seven server-side and six client-side phishing detection systems.

Below, we describe the three evasion techniques used by phishing websites to avoid detection.

2.2 Alert Box Evasion

Alert boxes in JavaScript are *modal* i.e., they pause script execution and do not allow the visitor to interact with the rest of the page until the window has been dismissed. Figure 1 in Appendix A shows an example of such a protection. The code to create a complete PayPal phishing page with alert box protection can be written in one PHP file and, for example, injected into one of the legitimate pages on a compromised machine. After a random number of seconds, the alert box appears on the screen by blurring the whole page and displaying the message: 'Please Sign In To Continue'. When the user clicks on the 'OK' button, an AJAX request is sent to the server to retrieve the malicious payload and replace the current page content with it. Code Listing 2 in Appendix C presents example PHP code of this technique.

To bypass the alert box, anti-phishing engines need to detect it first. Most browser emulation libraries, e.g., the Selenium[1] project, can distinguish the alert box window if it is present. They can also confirm or cancel the alert box.

2.3 Session-Based Evasion

Major companies like Google or Facebook have used the multi-page sign-in method for several years [29]. In this method, a user provides her username on the first page, submits the page, and enters her password on the second page. Note that the user can only be redirected to the second page if a session was generated on the first one. Phishers use this technique to reveal the second (malicious) page only to the users who have already visited the first page and clicked on the submit button. Figure 2 in Appendix A shows a real-world example of a session-based phishing attack. The first page (top figure) shows the 'Join Chat' button to persuade the victim to join a WhatsApp chat group and after pressing the button, the second page shows a fake Facebook login page.

[1] https://www.selenium.dev

One way to detect this type of evasion techniques is to automatically press the buttons and submit the forms on the suspicious page. In Section 4.1, we show that some anti-phishing engines submit forms on the suspicious pages if they can find the HTML *form* tag.

2.4 CAPTCHA-Based Evasion

Google reCAPTCHA is considered as an effective anti-bot protection solution against page scraping [25]. Figure 3 in Appendix A shows the phishing page protected by the Google reCAPTCHA v2 checkbox. By solving the CAPTCHA challenge, the form is submitted to the server and the phishing payload (e.g., PayPal login page) is served to the end-user without any URL redirection (Figure 3 in Appendix A bottom). To better understand the technique, Code Listing 1 in Appendix C shows the PHP code to create the phishing page. The first page is completely benign without an HTML *form* tag. When the user solves the CAPTCHA challenge, the HTML *form* tag is dynamically generated (using JavaScript) and then submitted to the server to reload the page using the same URL but with malicious content. Since the URL has not changed, the built-in browser anti-phishing system (e.g., GSB in Chrome) or the installed third-party extension (e.g., NetCraft toolbar) does not resend it to the server and serves instead the cached result usually valid for 5 to 60 minutes [30]. This behavior makes the detection process difficult (or impossible) and extends the lifespan of the phishing page.

There is no universal solution to bypass this type of evasion techniques. However, using client-side extensions, there is no need to automatically bypass CAPTCHA. If the end-user solves CAPTCHA and visits the second page (potentially malicious), the anti-phishing extension has also access to the content of the second page. So, it can detect phishing using the revealed page content.

3 EXPERIMENT METHODOLOGY

To investigate the effectiveness of human verification techniques to protect phishing websites, we have designed a testing experiment and evaluated the performance of seven server-side anti-phishing entities. The experiment has two phases: the initial test to check if anti-phishing engines can detect the payload itself and the main test in which we protect the same payload using one of the evasion techniques and report it to anti-phishing entities. Both phases follow the same process as explained below.

For security considerations, we do not publish the source code of the proposed framework. However, it is available to researchers upon request to support reproducibility.

Tested Server-Side Anti-Phishing Entities. We choose seven major anti-phishing entities: the Google Safe Browsing (GSB), NetCraft, Anti-Phishing Working Group (APWG), OpenPhish, PhishTank, Microsoft Defender SmartScreen, and Yandex Safe Browsing (YSB). GSB is used by Google Chrome, Mozilla Firefox, and Apple Safari that cover 87% of the end-user browsers both on desktop and mobile devices in 2020 [31]. Internet Explorer (IE) and Microsoft Edge use Microsoft Defender SmartScreen. The Opera browser uses two blacklist services: NetCraft and PhishTank [32]. To the best of our knowledge, OpenPhish and APWG are not directly used by any browser. However, they are two important blacklist feeds that might be used by anti-phishing engines, so we also consider them in our experiments. Finally, YSB is used by the Yandex browser, the second most popular browser in Russia as of May 2020[2].

Registering Domains. We first register 7 domains for the initial experiment and 105 for the main experiment, 112 domains in total. Oest et al. [21] performed a similar experiment to evaluate the effectiveness of web-cloaking in phishing attacks and registered fresh domains in bulk. In our study, we make sure that our domains are reputed enough to emulate the conditions of compromised domains rather than maliciously registered ones as observed in real-world cases. Therefore, we first register 50 drop-catch domains [33, 34] with the following new method:

(1) First, we scan the top 1M domains in the Alexa list [35] for 'SOA' and 'NS' DNS records and only keep the domains with the NXDOMAIN answer (770 domains).
(2) We use Godaddy and Porkbun APIs (two major registrars) to check the availability of the domains for registration from step 1 (251 domains).
(3) We collect WHOIS data for 251 domains from the previous step and only select those with 'NOT FOUND' answer to make sure they are not registered (244 domains).
(4) We submit these domains to VirusTotal[3] and Google Safe Browsing to make sure they have not been recently used in malicious activity (244 domains).
(5) We select the domains archived at least once by the Internet Archive[4] to make sure they have their web history (50 domains).
(6) We select the domains indexed at least once by the Google search engine based on the *site:domain* query (50 domains).

For the set of the remaining domains, we randomly generate keywords from the Unix dictionary and register 21 domains from new generic top-level domain (gTLDs) and the rest from legacy gTLDs (.com, .net, and .org). To reduce the impact of bulk registration patterns, we register our domains manually during two weeks in April 2020 with the OVH[5] registrar and deploy DNSSEC for all domains. All steps are taken to reduce the chances of being blacklisted due to the low reputation of the domain.

Website Content and Web Servers. Compromised domains are intrinsically legitimate but hacked to host and serve malicious content in addition to legitimate content. Therefore, we have to generate a full-fledged website for each domain. To achieve this, we propose and develop a fake website generator using the following algorithm:

(1) We extract meaningful keywords from the registered domain name.
(2) For each keyword, we find synonyms using the Datamuse[6] API.
(3) For each related keyword, we download the related page from the English version of Wikipedia along with their corresponding images.
(4) For each domain, we randomly generate 30 pages (with .php extensions) with different names and different directories.

[2] https://gs.statcounter.com/browser-market-share/all/russian-federation
[3] https://www.virustotal.com
[4] https://archive.org
[5] https://www.ovh.com
[6] http://www.datamuse.com/api

Then, we generate hyperlinks from one page to another to create a fully functional website.

Having the fake website generator, it takes 2 minutes to generate a fully functional website with 30 different pages in an automated manner. The output is a .zip package ready to upload and install on the server. We upload all the generated websites to our hosting infrastructures in one of the European countries with 22 different IP addresses and the Nginx web server. Finally, we issue TLS certificates for all the domains and keep all websites online without uploading phishing kits for one week.

Phishing Kits. The next step is to automatically generate phishing kits and upload them to our servers. We keep in mind the following considerations:

(1) The aim is to assess the effectiveness of phishing detection bots in bypassing human verification evasion techniques. Therefore, we have to make sure first that the 'naked' phishing payload, i.e., not armed with any evasion technique, can be easily detected.
(2) So, we have designed our own phishing pages instead of using existing phishing kits to prevent possible detection based on the *previously observed* attacks.
(3) The design of the phishing kits targets three major services: Gmail, PayPal, and Facebook login pages.
(4) We download all the external resources (e.g., pictures) from each target and generate a .zip package. This step is necessary since external resources like web favicons (favorite icons), logos as well as web beacons [36] play an important role for anti-phishing companies to track and detect phishing attacks for their websites [23].
(5) For Facebook and PayPal login pages, we just clone their HTML source code, remove JavaScript code, and external requests. We design a Gmail login page from scratch since the original page uses a heavily obfuscated JavaScript code to generate HTML tags at run-time. All the three generated phishing pages look exactly like their true versions.

Reporting and Monitoring Process. We submit phishing reports by either using an online form (GSB, SmartScreen, Net-Craft, and YSB) or sending an email (OpenPhish, PhishTank, and APWG). In the main experiment, we only generate one phishing URL for each domain. We never submit a domain to more than one anti-phishing bot. To monitor the results of the reports for each URL, we call GSB API to check if any of the URLs appeared in the GSB blacklist. Regarding OpenPhish, PhishTank, and APWG, we download the corresponding blacklist feeds every half an hour to check if they blacklisted our URLs and when. NetCraft notifies the reporter through emails by sending the results of the reports. SmartScreen has no publicly available API endpoint to check the results of the reports. Therefore, we develop a Python script to open the reported URLs using the Microsoft Edge browser, taking screenshots every 10 minutes for the first 72 hours and every 5 hours for the rest of the experiment, for several days.

Previous work showed some inconsistency among different browsers that use the same anti-phishing entities like Mozilla Firefox and Google Chrome (both use GSB) because of the different caching implementations of the GSB Update API (v4) in each browser [21]. In this experiment, our goal is not to check which browser faster blacklists phishing pages but instead, we test which anti-phishing engines can bypass the human verification evasion techniques.

4 EXPERIMENT RESULTS

Oest et al. [21], showed that the average blacklist time (i.e., the time between the URL submission and its appearance in one of the blacklists) was 126 minutes without using the web-cloaking technique and 238 minutes with web-cloaking. They also showed that anti-phishing engines could only detect 23% of the phishing URLs armed with web-cloaking. We expect a lower detection rate for the phishing sites instrumented with the three human verification techniques because they are more advanced and less observed in real-world phishing attacks.

4.1 Preliminary Test

In our initial test, we have submitted three URLs to seven anti-phishing entities targeting PayPal, Facebook, and Gmail without using any evasion technique. The test lasted for 24 hours. Based on the results from the previous study [21], 24 hours is enough to check if a reported URL is classified as malicious. For all seven anti-phishing bots, we received traffic to our webserver within the first 30 minutes after we reported the URLs. Table 1 shows the results of the preliminary tests. Since YSB was unable to detect even one of the phishing sites, we had to exclude it from the main experiment.

We also observed that except for GSB and NetCraft, none of the anti-phishing bots could identify the fake Gmail login pages as phishing attacks. Therefore, we also excluded Gmail from our target list in the final experiment. We suspect that the Gmail login page was more difficult to detect compared to PayPal and Facebook pages because of the different design approaches we used. As mentioned earlier, we have implemented the Gmail login page from scratch, whereas for PayPal and Facebook, we cloned the original HTML code. Moreover, log inspection shows that NetCraft, OpenPhish, and PhishTank submit the HTML *form* tags automatically by filling the 'username' field with different values (we do not log the password filed on our servers).

The results presented in Table 1 show that:

(1) There exist a relationship between different vendors. For example, the URLs we reported to OpenPhish also appeared in other blacklist feeds. The results also suggest that GSB uses other major blacklist feeds.
(2) For the URLs submitted to OpenPhish and PhishTank, we received abuse notification emails from PhishLabs [37] sent to the registered abuse notification email address related to our IP addresses.
(3) Within the first two hours after reporting the URLs to OpenPhish, we received a high amount of requests sent to our servers (81,967 requests). Their analysis reveals that anti-phishing bots looked for files related to: *i)* famous web-shells, *ii)* possible phishing kits (.zip files), and *iii)* possibly stolen credentials stored on the server (.log and .txt files).

4.2 Main Experiment

Table 2 shows the results of the main experiment to evaluate the effectiveness of six major anti-phishing entities in detecting phishing websites protected by evasion techniques. The experiment lasted

Table 1: Preliminary test results after reporting the Gmail (G), Facebook (F), and PayPal (P) phishing URLs.

Reported to	# of requests	Unique IPs	Reported pages	Also blacklisted by	Blacklisted targets
GSB	8,396	69	G, F, P	-	G, F, P
NetCraft	6,057	63	G, F, P	GSB	G, F, P
APWG	2,381	86	G, F, P	GSB	F, P
OpenPhish	81,967	852	G, F, P	PhishTank, GSB, APWG, SmartScreen	F, P
PhishTank	4,929	275	G, F, P	OpenPhish, GSB	F, P
SmartScreen	1,590	81	G, F, P	GSB	F, P
YSB	82	34	G, F, P	-	-

Table 2: Results of the main experiment after reporting phishing URLs. X/Y shows the number of detected URLs (X) out of all submitted ones (Y), where Alert box (A), Session-based (S), or Google reCAPTCHA (R) are used to hide phishing sites from server-side anti-phishing bots.

Anti-phishing bots	Facebook			PayPal		
	A	S	R	A	S	R
GSB	3/3	0/3	0/3	3/3	0/3	0/3
NetCraft	0/3	2/3	0/3	0/3	0/3	0/3
APWG	0/3	0/3	0/3	0/3	0/3	0/3
OpenPhish	0/3	0/3	0/3	0/3	0/3	0/3
PhishTank	0/3	0/3	0/3	0/3	0/3	0/3
SmartScreen	0/2	0/2	0/2	0/3	0/3	0/3

for two weeks in May 2020. Nevertheless, we received about 90% of the traffic during the first 2 hours after reporting the URLs.

Regarding the alert box protection, GSB was the only engine that detected all 6 reported URLs, on average 132 minutes after submission, which means that GSB can detect phishing websites protected by alert boxes by handling them in a browser simulation. The log analysis on our server reveals that GSB bots clicked on the 'confirm' button in the alert box and successfully retrieved phishing content, while other anti-phishing engines never reached the phishing content because they failed to click on the alert box.

When it comes to session-based evasion, NetCraft was the only engine that detected 2 out of 6 reported URLs (6 and 9 minutes after the submission). Interestingly, the log analysis indicates that NetCraft bypassed all six session-based pages and reached the phishing sites, but only 2 of them were detected as such. No other anti-phishing engine bypassed the session-based protection.

Regarding Google reCAPTCHA, as expected, none of the anti-phishing engines could detect even one out of 35 reported URLs, which means that it is so far the most effective protection solution of phishing websites available to malicious actors.

The results shown in Table 2 indicate that, in general, human verification techniques raise serious challenges to anti-phishing bots. In total, all the major server-side anti-phishing bots could only detect 8 out of 105 phishing URLs used in the main experiment. Fetching the phishing content depends on bypassing the evasion technique, which, as our experiment shows, is not a trivial task for anti-phishing bots. One possible solution to this problem is to let the end-users solve the challenge and reveal the final page content to client-side anti-phishing engines.

5 TESTING CLIENT-SIDE ANTI-PHISHING EXTENSIONS

Client-side extensions have access to the same content as users. If the user bypasses CAPTCHA (e.g., confirming alert box, solving Google reCAPTCHA, or pressing 'proceed' button in session-based pages) and visits the second page, the extensions have also access to the new possibly malicious content.

Therefore, in a separate experiment, we install the six most popular anti-phishing extensions on Mozilla Firefox (see Table 3). To make sure there is no conflict between the extensions, we use different Firefox profiles for each extension and disable GSB. For each extension, we submit 9 phishing URLs (3 URLs per evasion technique). We also use the Burp Suite[7] tool to capture the exchanged traffic related to each extension. In this way, we can read all TLS-encrypted requests between extensions and their servers. Then, we visit each URL three times with a 5-hour window between them.

Table 3 shows that none of the extensions could detect our phishing pages. Monitoring the traffic generated by the extensions reveals that they only collect the URLs visited by the user, send them to their servers, and check the URLs against their own blacklists. As shown in Table 3, four out of six extensions send 'naked' URLs (without hashing) along with all the query parameters to their servers. Since they do not create any feature vector on the client machine and only send the URL to the server for further analysis, they operate like their server-side counterparts, and thus, they are unable to detect CAPTCHA-protected phishing attacks.

5.1 Discussion

In Section 4, we have presented three quite effective evasion techniques based on human verification to avoid detection by anti-phishing engines. The important question is how popular these techniques are among attackers and what is the solution for effective handling of the attacks that use these techniques.

With respect to popularity, it is unlikely to find these attacks in public blacklist feeds because they mainly rely on anti-phishing engines that cannot bypass advanced types of evasion techniques. In a parallel and independent research study by Oest et al. [38], the authors analyzed 4,393 URLs and only found five of them that used the CAPTCHA challenge and two using the alert box technique to avoid detection. In recent work, Maroofi et al. [17] studied a reCAPTCHA-protected phishing URL that prevents crawlers from

[7] https://portswigger.net/burp

Table 3: Client-side anti-phishing extensions. Number of installations is the sum of installations for Chrome and Firefox. X/Y shows the number of detected URLs (X) out of all submitted ones (Y).

Extension	Company	# of installations	Sending URLs	Sending Params	X/Y
Avast Online Security	Avast	10,800,000+	✓(plain)	✓	0/9
Avira Browser safety	Avira	7,350,000+	✓(plain)	✓	0/9
TrafficLight	BitDefender	665,000+	✓(plain)	✓	0/9
Emsisoft Browser security	Emsisoft	80,000+	✓(hashed)	✗	0/9
NetCraft Anti-phishing	NetCraft	58,000+	✓(hashed)	✗	0/9
Online Security Pro	Comodo	14,000+	✓(plain)	✓	0/9

fetching the real malicious content. Although the URL was submitted to Phishtank, a community-based URL blacklist based on user reports, it was not confirmed by any other user and thus, it did not appear on the official blacklist. Therefore, due to the difficulties related to detecting such attacks, it is not straightforward to evaluate their popularity.

Another concern is to develop a suitable solution so that anti-phishing engines can detect these attacks and blacklist the URLs. Regarding the alert box technique using server-side detection, the solution is trivial since most automation frameworks (e.g., Selenium library) can interact with alert boxes and modal windows. For session-based attacks (server-side detection), one possible solution is to simulate form submissions. As shown in Section 4.1, NetCraft submitted the HTML form for each report but was only able to detect 2 malicious URLs. Finally, bypassing CAPTCHA by a server-side anti-phishing engine is not easy in general since there is no prior information about the characteristics of the CAPTCHA challenge used by attackers.

For client-side detection systems (in the form of extensions installed on the browsers), there is no need to implement any extra mechanism. If the user solves the challenge and visits a malicious page, it is also visible to extensions for the detection process.

6 CONCLUSION

In this paper, we evaluate the resilience of anti-phishing detection systems to evasion techniques based on human verification in a controlled experiment. We registered 105 previously-unseen as well as reputed drop-catching domains. For each domain, we automatically generated a full-fledged website and a phishing page protected by either Google reCAPTCHA, alert box, or session-based techniques, and reported them to major anti-phishing entities. An alarming picture emerges from our experiments: the overwhelming majority of phishing sites were not detected. Google Safe Browsing outperforms other entities by only bypassing the alert box anti-analysis technique. NetCraft was able to detect 2 URLs protected with the session-based technique. Other server-side and client-side detection systems were unable to detect even one of the phishing URLs as malicious.

The prevalence of human verification techniques in phishing kits can severely hinder the detection process and raises serious challenges for existing anti-phishing engines. As a mitigation plan, we intend to disclose our findings to the impacted anti-phishing entities before phishers exploit human verification techniques on a massive scale.

ACKNOWLEDGMENTS

This work has been carried out in the framework of the COMAR project funded by SIDN, the .NL Registry and AFNIC, the .FR Registry. This work has been partially supported by the ANR projects: the Grenoble Alpes Cybersecurity Institute CYBER@ALPS under contract ANR-15-IDEX-02, PERSYVAL-Lab under contract ANR-11-LABX-0025-01, and DiNS under contract ANR-19-CE25-0009-01.

REFERENCES

[1] Hang Hu and Gang Wang. End-to-end Measurements of Email Spoofing Attacks. In *27th USENIX Security Symposium*, pages 1095–1112, 2018.
[2] Tom N Jagatic, Nathaniel A Johnson, Markus Jakobsson, and Filippo Menczer. Social Phishing. *Communications of the ACM*, 50(10):94–100, 2007.
[3] Eric Medvet, Engin Kirda, and Christopher Kruegel. Visual-Similarity-Based Phishing Detection. In *4th International Conference on Security and Privacy in Communication Netoworks*, pages 1–6, 2008.
[4] Mahmood Moghimi and Ali Yazdian Varjani. New Rule-Based Phishing Detection Method. *Expert Systems with Applications*, 53:231–242, 2016.
[5] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. Machine Learning Based Phishing Detection from URLs. *Expert Systems with Applications*, 117:345–357, 2019.
[6] Jian Mao, Wenqian Tian, Pei Li, Tao Wei, and Zhenkai Liang. Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity. *IEEE Access*, 5:17020–17030, 2017.
[7] Samuel Marchal, Jérôme François, Radu State, and Thomas Engel. Phishstorm: Detecting Phishing with Streaming Analytics. *IEEE Transactions on Network and Service Management*, 11(4):458–471, 2014.
[8] Anti-Phishing Working Group (APWG): Cross-Industry Global Group Supporting Tackling the Phishing Menace. http://www.antiphishing.org, 2020.
[9] PhishTank: A Nonprofit Anti-Phishing Organization. http://www.phishtank.com, 2020.
[10] COVID-19 Cyber Threat Coalition. https://www.cyberthreatcoalition.org, 2020.
[11] X-Force Threat Intelligence Index, 2020. URL https://www.ibm.com/account/reg/us-en/signup?formid=urx-42703.
[12] Sidharth Chhabra, Anupama Aggarwal, Fabricio Benevenuto, and Ponnurangam Kumaraguru. Phi.sh/$oCiaL: The Phishing Landscape through Short URLs. In *8th Annual Collaboration, Electronic Messaging, Anti-Abuse and Spam Conference*, pages 92–101, 2011.
[13] Fortinet: Threat Landscape Report. https://www.fortinet.com/content/dam/fortinet/assets/threat-reports/threat-report-q2-2019.pdf, 2019.
[14] Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. A Survey on Automated Dynamic Malware-Analysis Techniques and Tools. *ACM Computing Surveys (CSUR)*, 44(2):1–42, 2008.
[15] Alexei Bulazel and Bülent Yener. A Survey on Automated Dynamic Malware Analysis Evasion and Counter-Evasion: PC, Mobile, and Web. In *1st Reversing and Offensive-oriented Trends Symposium*, pages 1–21. ACM, 2017.
[16] Adam Oest, Yeganeh Safei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Gary Warner. Inside a Phisher's Mind: Understanding the Anti-phishing Ecosystem Through Phishing Kit Analysis. In *2018 APWG Symposium on Electronic Crime Research (eCrime)*, pages 1–12. IEEE, 2018.
[17] Sourena Maroofi, Maciej Korczyński, Cristian Hesselman, Benoit Ampeau, and Andrzej Duda. COMAR: Classification of Compromised versus Maliciously Registered Domains. In *5th IEEE European Symposium on Security and Privacy, Euro S&P*, 2020.
[18] Luca Invernizzi, Kurt Thomas, Alexandros Kapravelos, Oxana Comanescu, Jean-Michel Picod, and Elie Bursztein. Cloak of Visibility: Detecting When Machines Browse a Different Web. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 743–758. IEEE, 2016.

[19] Ian Fette, Norman Sadeh, and Anthony Tomasic. Learning to Detect Phishing Emails. In *16th International Conference on World Wide Web*, pages 649–656, 2007.
[20] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There Is No Free Phish: An Analysis of "Free" and Live Phishing Kits. *USENIX Workshop on Offensive Technologies*, 8:1–8, 2008.
[21] Adam Oest, Yeganeh Safaei, Adam Doupé, Gail-Joon Ahn, Brad Wardman, and Kevin Tyers. PhishFarm: A Scalable Framework for Measuring the Effectiveness of Evasion Techniques against Browser Phishing Blacklists. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1344–1361. IEEE, 2019.
[22] Yi-Min Wang and Ming Ma. Detecting Stealth Web Pages That Use Click-Through Cloaking. In *Microsoft Research Technical Report, MSR-TR*, 2006.
[23] Adam Oest, Penghui Zhang, Brad Wardman, Eric Nunes, Jakub Burgis, Ali Zand, Kurt Thomas, Adam Doupé, and Gail-Joon Ahn. Sunrise to Sunset: Analyzing the End-to-end Life Cycle and Effectiveness of Phishing Attacks at Scale. In *29th USENIX Security Symposium*, 2020.
[24] Threat Spotlight: Malicious use of reCaptcha, 2020. URL https://blog.barracuda.com/2020/04/30/threat-spotlight-malicious-recaptcha/.
[25] Google reCAPTCHA v2, 2020. URL https://developers.google.com/recaptcha/docs/display.
[26] PhishTank FAQ, 2020. URL https://www.phishtank.com/faq.php.
[27] Comparison of web browsers, 2020. URL https://en.wikipedia.org/wiki/Comparison_of_web_browsers.
[28] Oleksii Starov and Nick Nikiforakis. Extended Tracking Powers: Measuring the Privacy Diffusion Enabled by Browser Extensions. In *26th International Conference on World Wide Web*, pages 1481–1490, 2017.
[29] Multiple Sign-In Pages, 2015. URL https://support.google.com/mail/forum/AAAAK7un8RUoAsE-6wmaSU/?hl=en&gpf=%23!topic%2Fgmail%2FoAsE-6wmaSU.
[30] Safe Browsing APIs (v4) - Caching, 2020. URL https://developers.google.com/safe-browsing/v4/caching.
[31] Browsers Market Share, 2020. URL https://netmarketshare.com/browser-market-share.aspx.
[32] Opera Browser FAQ, 2020. URL https://security.opera.com/mobile-browsers-faq/.
[33] Najmeh Miramirkhani, Timothy Barron, Michael Ferdman, and Nick Nikiforakis. Panning for gold.com: Understanding the Dynamics of Domain Dropcatching. In *In 27th International Conference on World Wide Web*, pages 257–266, 2018.
[34] Tobias Lauinger, Abdelberi Chaabane, Ahmet Salih Buyukkayhan, Kaan Onarlioglu, and William Robertson. Game of Registrars: An Empirical Analysis of Post-Expiration Domain Name Takeovers. In *26th USENIX Security Symposium*, pages 865–880, 2017.
[35] Alexa: Actionable Analytics for the Web. https://www.alexa.com.
[36] Janice C Sipior, Burke T Ward, and Ruben A Mendoza. Online Privacy Concerns Associated with Cookies, Flash Cookies, and Web Beacons. *Journal of Internet Commerce*, 10(1):1–16, 2011.
[37] Phishlabs. https://www.phishlabs.com.
[38] Adam Oest, Yeganeh Safaei, Penghui Zhang, Brad Wardman, Kevin Tyers, Yan Shoshitaishvili, and Adam Doupé. Phishtime: Continuous longitudinal measurement of the effectiveness of anti-phishing blacklists. In *29th USENIX Security Symposium*, pages 379–396, 2020.
[39] Phishing Activity Trends Report: 1st Quarter 2020 APWG. https://docs.apwg.org/reports/apwg_trends_report_q1_2020.pdf, 2020.

A EXAMPLES OF EVASION TECHNIQUES

Figure 1 shows an example of a phishing attack page protected by the alert box. Due to the modal nature of the alert boxes in JavaScript, the end-users (or bots) are only allowed to visit the phishing payload (Figure 1 bottom) if they press the 'OK' button (Figure 1 top).

Figure 2 presents an example of a phishing attack protected by PHP sessions. End-users can only visit the Facebook login page (Figure 2 bottom) if they first visit the cover page and click on the 'Join Chat' button (Figure 2 top).

Figure 3 shows the phishing attack protected by Google reCAPTCHA. Solving CAPTCHA on the first page (Figure 3 top) redirects the user to the second page without changing the final destination URL (Figure 3 bottom).

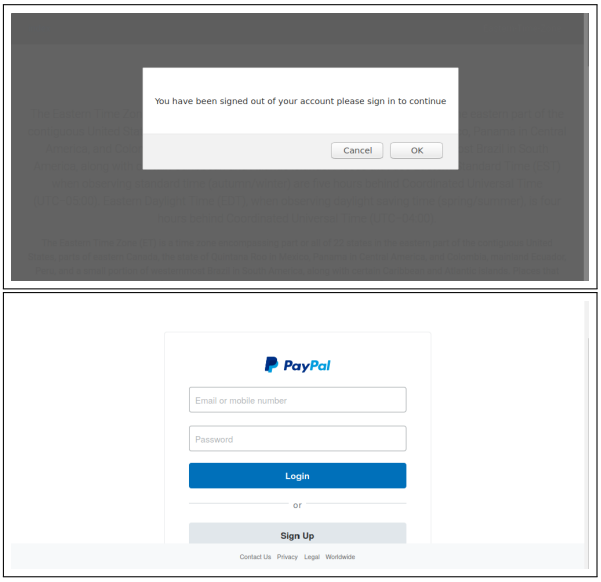

Figure 1: Alert box evasion technique. Alert box protected page (top) and phishing payload targeting PayPal (bottom).

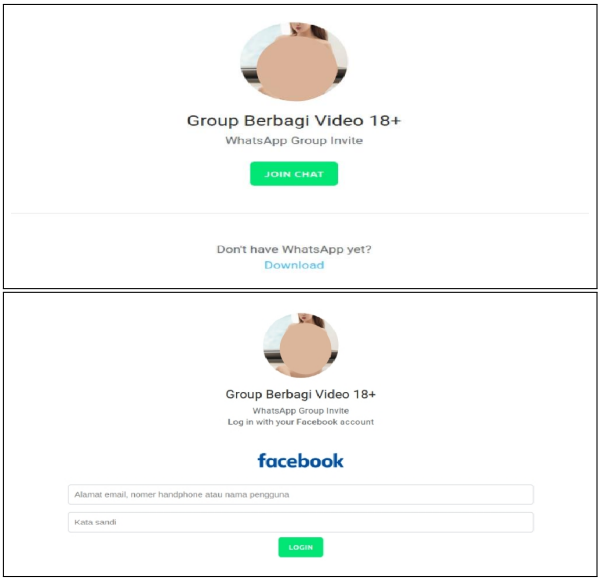

Figure 2: Session-based evasion technique. Cover page on the first visit (top) and the phishing payload targeting Facebook (bottom).

B ETHICAL CONSIDERATIONS

During our experiment, we were careful that no actual users would visit our websites. To accomplish that: *i)* we only reported our phishing URLs directly to anti-phishing bots either through emails or online forms, and never published our URLs online or on any public website and *ii)* for those possible users who accidentally visited our websites (with low probability), we removed all the sensitive

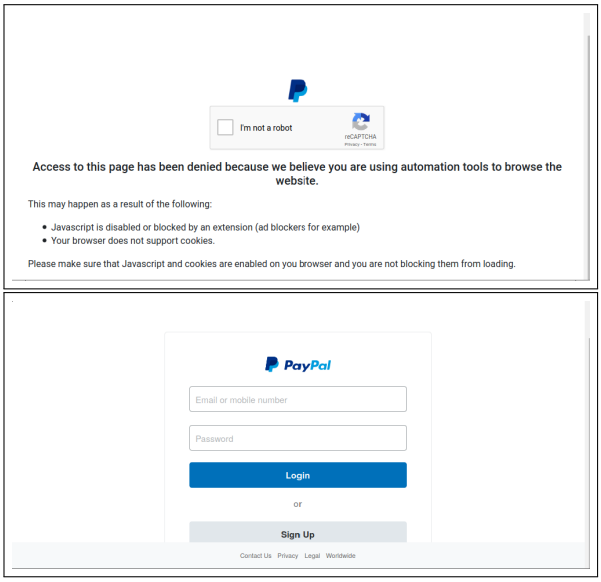

Figure 3: Google reCAPTCHA message evasion technique. Google reCAPTCHA-protected page (top) and the phishing payload targeting PayPal (bottom).

information (e.g., credentials) from the submission form. We also issued TLS certificates to make sure that there is no information leakage if an actual user accidentally enters her credentials.

Considering all the feasible precautions in place to ensure the safety of the users, the only remaining concern is the impact of our measurements on the overall performance of the anti-phishing engines. However, a comparison of the number of the reported phishing URLs with the daily number of captured phishing attacks by anti-phishing engines suggests that our reports cannot affect the overall performance of these engines. For example, for APWG, we have reported 18 URLs in one day while in March 2020, more than 2,000 URLs were blacklisted on a daily basis [39].

C PHP SAMPLE CODE

Code Listing 1 shows the sample PHP code to create a single-page phishing attack protected by the Google reCAPTCHA v2 checkbox.

Code Listing 2 shows the sample PHP code to create and monitor the alert box protected phishing attack. If the anti-phishing bot is able to successfully confirm the alert box, we log it on the server by submitting an HTML form tag with '*getData*' value as shown in the code (line 40-44).

```php
<?php
    $isvalid = false;
    if (isset($_POST['gresponse'])){
        $secret = 'Google CAPTCHA secret key';
        $captcha = $_POST['gresponse'];
        /* Check CAPTCHA result */
        $ans = chk_captcha($secret,$captcha);
        if ($ans->success)
            $isvalid = true;
        else
            $isvalid = false;
    }
    if ($isvalid){
        echo "Serve phishing payload HTML";
    }else{
        echo "Serve CAPTCHA page HTML";
    }
?>
<script>
    function capback(g_response){
        $form = $("<form>").attr({method:'post'});
        $input = $("<input>");
        $input.attr({name:"gresponse"});
        $input.attr({value:g_response});
        $form.append($input);
        $('body').append($form);
        $form.submit();
    }
</script>
```

Listing 1: Single-page PHP phishing code with Google reCAPTCHA protection.

```php
<?php
$log_file="name of log file";
$a = $_POST['get_data'];
if (isset($a) && $a == 'getData'){
    /*Anti-phishing engine or user managed
    to confirm the alert box*/
    $d = array('ip' => getip(),'page'=>'payload');
    log_data($d,$log_file);
    echo "SERVE PHISHING HTML";
}else{
    $d = array('ip' => get_ip(),'page'=>'benign');
    log_data($d,$log_file);
    echo "SERVE BENIGN CONTENT WITH ALERT BOX";
}
?>
<script>
    /*Creating JS check variable
    for the second page load*/
    <?php
        if (isset($a) && $a == 'getData'){
            echo 'first_visit = false;';
        }else{
            echo 'first_visit = true;';
        }
        $u = $_POST['login_email'];
        $p = $_POST['login_pass'];
        if (isset($u) && isset($p)){
             echo 'already_served = true;';
        }else{
             echo 'already_served = false;';
        }
    ?>
    window.onload = function(){
         /*execute after the window
            is loaded completely*/
        if (first_visit && already_served){
            setTimeout(get_real_data,2000);
        }
    }
    function get_real_data(){
        var msg='Please sing in to continue...'
        var result = confirm(msg);
        if (result){
            /*
            dynamically generate and
            submit a form with hidden
            value 'getData'
            */
        }else{
            /*
            submit an empty form
            */
        }
    }
</script>
```

Listing 2: PHP phishing code with alert box protection.