A Measurement Approach to Understanding the Data Flow of Phishing from Attacker and Defender Perspectives

Peng Peng

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Gang Wang, Chair
Bimal Viswanath
Danfeng Yao

November 22, 2019
Blacksburg, Virginia

Keywords: Phishing Attacks; Measurement; Credential Sharing; Online Scan Engines

A Measurement Approach to Understanding the Data Flow of Phishing from Attacker and Defender Perspectives

Peng Peng

(ABSTRACT)

Phishing has been a big concern due to its active roles in recent data breaches and state-sponsored attacks. While existing works have extensively analyzed phishing websites and detection methods, there is still a limited understanding of the data flow of the phishing process. In this thesis, we perform an empirical measurement to draw a clear picture of the data flow of phishing from both attacker and defender perspectives. First, from attackers' perspective, we want to know how attackers collect the sensitive information stolen from victims throughout the end-to-end phishing attack process. So we collected more than 179,000 real-world phishing URLs. Then we build a measurement tool to feed fake credentials to live phishing sites and monitor how the credential information is shared with the phishing server and potentially third-party collectors on the client side. Besides, we also obtain phishing kits to analyze how credentials are sent to attackers and third-parties on the server side. Then, from defenders' perspective, online scan engines such as VirusTotal are heavily used by phishing defenders to label phishing URLs, however, the data flow behind phishing detection by those scan engines is still unclear. So we build our own phishing websites, submit them to VirusTotal for scanning, to understand how VirusTotal works and the quality of its labels. Our study reveals the key mechanisms for information sharing during phishing attacks and the need for developing more rigorous methodologies to assess and make use of the labels obtained from VirusTotal.

A Measurement Approach to Understanding the Data Flow of Phishing from Attacker and Defender Perspectives

Peng Peng

(GENERAL AUDIENCE ABSTRACT)

Phishing attack is the fraudulent attempt to lure the target users to give away sensitive information such as usernames, passwords and credit card details. Cybercriminals usually build phishing websites (mimicking a trustworthy entity), and trick users to reveal important credentials. However, the data flow of phishing process is still unclear. From attackers' perspective, we want to know how attackers collect the sensitive information stolen by phishing websites. On the other hand, from defenders' perspective, we are trying to figure out how online scan engines (e.g., VirusTotal) detect phishing URLs and how reliable their detection results are. In this thesis, we perform an empirical measurement to help answer the two questions above. By monitoring and analyzing a large number of real-world phishing websites, we draw a clear picture of the credential sharing process during phishing attacks. Also, by building our own phishing websites and submitting to VirusTotal for scanning, we find that more rigorous methodologies to use VirusTotal labels are desperately needed.

Acknowledgments

I want to express my deepest appreciation to my advisor Dr. Gang Wang, for his unwavering guidance and valuable advice during the development of this work. His constructive feedback, insightful suggestions and patience made this work possible. I would like to extend my sincere gratitude to my committee members Dr. Bimal Viswanath and Dr. Danfeng Yao for their helpful comments on this work. Many thanks to Dr. Linhai Song, Dr. Bimal Viswanath, Limin Yang, Chao Xu, Luke Quinn and Hang Hu for their contributions in this work. I'm extremely grateful to my parents for their support throughout the years. Special thanks to all colleagues at Knowledge Works II.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Phishing attack is a persistent threat on the Internet. It exploits human factors to lure the target users to give away critical information. In recent years, phishing becomes an even bigger concern due to its prevalent usage in facilitating major data breaches [14], particularly the recent breaches in hospitals and health care companies [15, 16]. Besides, phishing plays an important role in many state-sponsored attacks. One of the recent examples is the spear phishing attack against John Podesta, the campaign manager of Hillary Clinton, during the US election in 2016 [12].

The research community has been studying phishing attacks from different aspects. While some existing works analyzed phishing emails [34], the vast majority focus on the *phishing websites* that are set up by attackers to trick users to reveal important information (*e.g.*, login credentials) [67, 71, 75, 79]. These phishing sites often impersonate other reputable entities to gain the victim's trust. More recently, researchers analyze phishing kits, the software packages for running phishing websites, to understand how phishing sites are deployed and operated [26, 33, 54]. However, these works only looked into the disconnected parts of phishing. There is a limited end-to-end understanding of the information flow after user credentials are leaked to the phishing sites.

On the other hand, online scan engines, designed to scan malware files and malicious websites, are critical tools for detecting phishing websites on the Internet [3, 4, 7, 8]. VirusTotal is one of the most popular scanning services that are widely used by researchers and industry

practitioners [8]. It works with more than 60 security vendors to aggregate their scanning results. Many recent works rely on VirusTotal's URL scanning API [24, 35, 51, 59, 60, 62, 67, 70, 77, 80] for data labelling. Unfortunately, VirusTotal works like a blackbox, and it is not well understood how VirusTotal and its vendors generate the labels for a given URL or file. This leads to critical questions: *are these labels even reliable? Are researchers using VirusTotal in the right way?*

In this thesis, we perform an empirical measurement to understand the attack and defense process of phishing, more specifically, the credential sharing process during phishing attack and the usage of online scan engine (VirusTotal) in phishing detection.

1.1 Credential Sharing during Phishing

To understand the data flow from attackers' perspective, we performed the measurement from August 2018 to January 2019 covering 179,865 phishing URLs. The client-side measurement covers 41,986 live phishing sites, and the server-side measurement is based on the analysis of 2,064 detected phishing kits. Our post-phishing exploitation analysis uses 100 honey accounts from Gmail and 50 accounts from ProtonMail for data collection. We explore how likely attackers would attempt to use the leaked password to further hijack the associated email account (in addition to the original online account). Our study leads to a number of key findings:

- *First*, we show that user credentials are shared in real time on both the client-side and the server-side. This easily exposes the stolen credentials to more malicious parties.
- *Second*, while the client-side sharing is not very common (about 5%), the third-party servers are often located in a different country (compared to the phishing server),

which may create difficulties to take them down. In particular, many "good" websites were used to receive stolen credentials (*e.g.*, Google Ads are used to track the phishing statistics for attackers).

- *Third*, server-side credential sharing is primarily done via emails. 20% of the phishing kits send the credentials to two or more email addresses. About 5% of the phishing kits contain backdoors that stealthily leak the credentials to third-parties.

1.2 Online URL Scan Engines towards Phishing

To figure out the data flow from defenders' perspective, we take the initial steps to explore the back-end process of how VirusTotal and its vendors generate the labels. We specifically look into how the URL scan API detects *phishing websites*. Focusing on phishing scan allows us to design more focused experiments. However, our methodology should be easily adapted to other applications (*e.g.*, file scan for malware analysis). We seek to explore (1) how VirusTotal works with 68 vendors to perform URL scanning and result updating; (2) how effective these scanners are in detecting simple and more advanced phishing sites; (3) how the scanners react to the dynamic changes of phishing sites. The goal is to provide insights to guide researchers and practitioners to better use VirusTotal.

To answer these questions, we set up a series of phishing websites of our own with freshly registered domains. By submitting the phishing URLs to various scan APIs (VirusTotal's APIs and some vendors' own APIs), we collect the incoming network traffic to our phishing sites. At the same time, we continuously query the labels for these URLs from VirusTotal over a month. We set up two types of phishing pages that impersonate PayPal and IRS (Internal Revenue Service) respectively. Across all the experiments, we have used 62 phishing websites in total. We have five key findings in this part:

- *First*, most vendors have trouble detecting the simple phishing sites we set up. Over multiple scans, only 15 vendors (out of 68) have detected at least one of the 36 simple phishing sites. The best vendor only detected 26 (out of 36) simple phishing sites.

- *Second*, the detection performance is drastically different for different phishing sites. The PayPal sites (as a popular target of phishing) can be detected quickly by more than 10 vendors during the first scan. However, the less common IRS sites cannot be detected by any of the 68 vendors using the VirusTotal scan API alone.

- *Third*, the scanning results of vendors are not updated to VirusTotal immediately after the scanning is finished. The delay is caused by the fact that VirusTotal only pulls the previous scanning results when a new scan request is submitted for the same URL. A user who simply calls the query/report API would not get the updated scanning results.

- *Fourth*, VirusTotal has inconsistent results with some of the vendors' own scan APIs. For example, the IRS sites can be detected by a few vendors' APIs but not the VirusTotal API. The result suggests that third-party vendors do not always give VirusTotal the scan permission or the most updated blacklists.

1.3 Contributions

This thesis makes four key contributions:

- *First*, we perform a large-scale empirical measurement on the information flow of credential sharing during phishing attacks as well as the quality of labels from VirusTotal for phishing detection.

- *Second*, we build a new measurement tool to automatically seed fake credentials to phishing sites to measure the information sharing in real time.
- *Third*, our measurements provide new insights into the credential sharing mechanisms (to third-parties) during the phishing process.
- *Fourth*, our work reveals the reliability of labels obtained from online scan engines, and measure the correlation between the labels from different vendors.

The content of this thesis has been published in Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security [56] and Proceedings of the Internet Measurement Conference [57].

Chapter 2

Background

2.1 Background of Phishing

Figure 2.1 shows the typical steps of a phishing attack. Attackers first need to trick users into visiting a phishing website. To gain the victim's trust, a phishing website often impersonates other reputable services. In step1, the victim user submits the login credential via the phishing page in the browser. After that, the information is then sent to the phishing server (step2.1). The phishing server either directly sends the collected credentials via emails to the attacker (step3.1), or the attacker will (manually) log into the phishing server to retrieve the information (step3.2). Once the login credentials are obtained by the attacker, they can proceed further with malicious activities against users or their organizations (*e.g.*, stealing data, compromising enterprise/government networks).

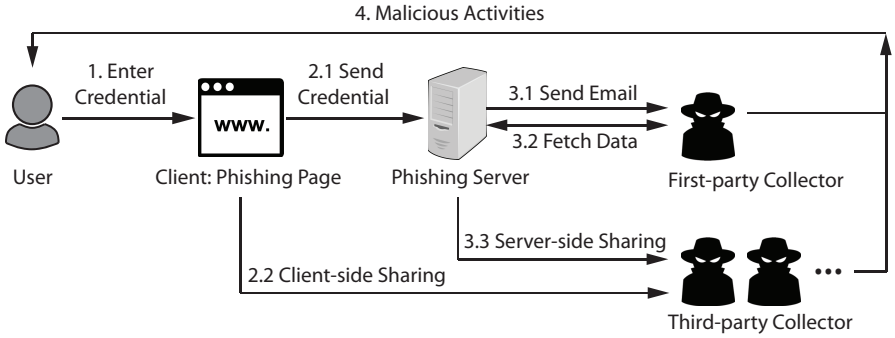

Figure 2.1: Phishing attack process.

2.1.1 Phishing Kits

Attackers often deploy phishing websites using a collection of software tools called phishing kits [26]. Phishing kits allow people with little technical skills to run phishing attacks. A typical kit contains a website component, and an information processing component. The website component contains the code, images, and other content to create a fake website. The information processing tool will automatically record and store the received information (password, login time, IP), and send the information to the attacker. Some phishing kits also contain a spamming tool, which can send spam emails to lead users to the phishing sites.

2.1.2 Third-party Information Sharing

During a phishing attack, it is possible that the user credentials are also shared to third-parties, in both the *client-side* and the *server-side*.

- **Client-side Third Parties.** Step`2.2` shows that client-side third-parties collect the user credential. In this case, the phishing server that directly hosts the phishing page is the first-party and any other servers that also collect the credential are third-parties. The information sharing happens in real time when the user clicks on the "submit" button.

- **Server-side Third Parties.** Step`3.3` represents the server-side third-parties. Certain phishing kits contain "back-doors" planted by other parties (*e.g.*, the phishing kit developer) [26]. After the login credentials are received by the phishing server, the information will be sent to the first-party (who deployed the phishing website), and also possibly to the third-party (who planted the back-door in the phishing kit).

2.2 Background of VirusTotal

2.2.1 VirusTotal APIs

VirusTotal is a popular service that scans malicious files and web URLs [8]. The URL scanning, in particular, aims to detect websites that deliver malware or perform phishing. As shown in Figure 2.2, VirusTotal works with 68 third-party security vendors (see the full list at [10]). After an URL is submitted to VirusTotal through the *scanning API*, VirusTotal pass the URL to these vendors (*i.e.*, anti-virus engines or online scanning services). The scanning results will be stored in the VirusTotal database.

VirusTotal provides another *querying API* (or report API) which allows people to query the VirusTotal database to check if an URL is malicious [11]. Given a URL, the API returns the labels from all the vendors that have previously scanned the URL (and the timestamp of the scanning). It is not uncommon for vendors to disagree with each other. For example, a URL might be labeled as "benign" by Google Safe Browsing, but is labeled as "malicious" by Kaspersky.

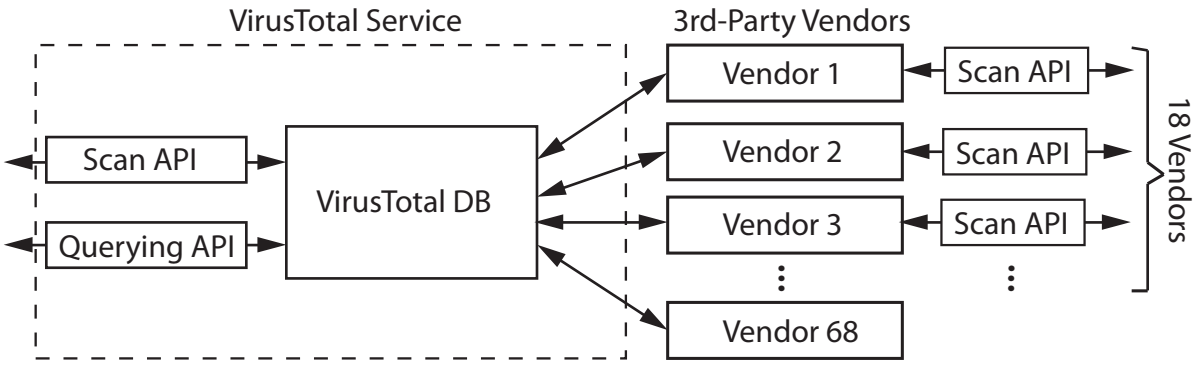

Figure 2.2: VirusTotal and third-party security vendors.

2.2.2 Third-party Vendors.

Among the 68 third-party vendors, we find that 18 of them also provide their own scanning APIs to the public. Table 2.1 lists the names of the 18 vendors. As shown in Figure 2.2, these 18 vendors can either be reached via the VirusTotal scanning APIs or via their own APIs directly. For a given vendor, it is not quite clear if the two APIs return consistent results.

Table 2.1: 18 VirusTotal vendors that provide their own scanning APIs.

CyberCrime, Dr.Web, Forcepoint, Fortinet,
Google Safe Browsing, Kaspersky, malwares.com, Netcraft,
NotMining, Phishtank, Quttera, scumware.org, securolytics,
Sucuri Site Check, URLQuery, ZeroCERT, ZeusTracker, zvelo

Chapter 3

Credential Sharing in Phishing Attack

In this chapter, we first describe our high-level methodology to track the information flow in each step in Figure 2.1, and then introduce the measurement tool designed by us. Next, we show the dataset collected over five months and the analysis results.

3.1 Methodology Overview

First, to track the information flow at step1, step2.1, and particularly step2.2, we design a measurement tool to automatically feed (fake) login credentials to real-world phishing websites via the login forms. The tool will also keep track any redirections and real-time credential.

Second, to infer the information flow of step3.1, step3.2, and step3.3, we try to obtain the phishing kits from phishing servers and analyze how the phishing kits work. We extract the email addresses that first-party attackers use to collect the user credentials. We also perform a dynamic analysis in a sandbox to identify potential backdoors planted by third-parties (§3.5).

3.2 Measurement Tool

Our tool is a web crawler implemented using Selenium[1]. It controls a headless ChromeDriver browser to complete a series of actions and records the network traffic in the ChromeDriver log.

3.2.1 Login Form Detection

We focus on phishing sites that collect login credentials, excluding those that collect other information such as credit card information or social security numbers. We detect the login form by looking for three fields: username, password, and the "submit" button. We look for related tags in HTML including `FORM` tags, `INPUT` tags and `BUTTON` tags. We also extract the form attributes such as `type`, `placeholder`, `name`, and `class`). We don't consider any read-only or invisible tags.

To make sure that the form is indeed a login form instead of other irrelevant forms (*e.g.*, searching bar, survey forms), we compile a list of login related keywords and search them within the form attributes. We select keywords manually analyzing the login forms of 500 randomly phishing websites. In total, we select 40 keywords including 14 keywords for username (*e.g.*, "user name", "id", "online id", "email", "email address"), 8 keywords for password (*e.g.*, "password", "passwd", "passcode"), and 18 keywords for the submit button (*e.g.*, "log in", "sign in", "submit"). The main challenge is that phishing websites often have unconventional designs, or even intentionally hide keywords to evade detection [67]. It is not always possible to locate all three fields. Below, we list the key problems and how to address them.

[1] https://www.seleniumhq.org/

- **Keywords in images**: The most common challenge is that attackers use an image to contain the "Login" keyword for the submit button, instead of placing the keyword to the `placeholder`. Our solution is to use the Tesseract Open Source OCR Engine[2] to extract the texts from images, and then perform the keyword search.

- **No FORM tags**: Phishing pages may intentionally leave out the `FORM` tags (to evade detection). Our solution is to search `INPUT` tags and keywords in the whole HTML page, instead of just within the `FORM` tags.

- **Two-step login**: In some phishing pages, users need to enter the username on the first page, and type in the password on the next page. Our tool can handle two-step login by tracking the log-in progress.

- **Previous unseen keywords**: the keywords may occasionally fail to match the corresponding input fields. To increase our success rate, we perform a simple inference based on the order of input fields. For example, if the username and button fields are matched, then we guess the unmatched input field in the middle is for the password.

3.2.2 Filling in the Fake Credential

After detecting the login form, our tool will automatically fill in the username and password fields and click the submit button. The username is an email address that belongs to us. The password is a random string of 8 characters which is uniquely created by us. The unique password is helpful later to detect the network requests that send out the password. This email address is never used to register any online account. The password is also not the real password for the email address. In this way, we make sure the leaked information would not affect any real users. We test the tool on 300 phishing sites (different from those that

[2] https://github.com/tesseract-ocr/tesseract

contributed the keywords). We show that the tool has a success rate of 90% to complete the login.

Here, we also want to make sure that using *fake credentials* does not affect our measurement result. We did a small experiment to see if the phishing site would react to real and fake password differently. We create 4 real accounts with PayPal, Microsoft, LinkedIn, and AT&T respectively. Then we select 60 live phishing websites from eCrimeX that impersonate these brands (15 websites per brand). We feed the real and fake passwords in separate runs, and find that the collected network traffic has no difference.

Table 3.1: Dataset summary.

Blacklist	Crawling Time Span	Dete. Time	# All	# Live	# w/ Form	# Success
OpenPhish	09/24/2018 - 01/03/2019	✓	75,687	44,553	24,202	19,720
eCrimeX	08/20/2018 - 01/03/2019	✓	65,465	33,319	21,161	19,172
PhishTank	09/24/2018 - 01/03/2019	✓	50,608	41,682	7,406	6,430
PhishBank	09/24/2018 - 01/03/2019	✓	3,093	2,027	1,010	864
Total	08/20/2018 - 01/03/2019	–	179,865	110,934	47,703	41,986

3.3 Data Collection

Using the measurement tool, we collect data from August 2018 to January 2019 by crawling 4 large phishing blacklists: PhishTank, PhishBank, eCrimeX, and OpenPhish. The detailed data statistics are shown in Table 3.1. For each phishing URL, all four blacklists share the timestamp when the phishing URL was reported/detected. Three of the blacklists also show the target brand (or website) that the phishing page is trying to impersonate. OpenPhish shares the target brand information only for the premium API (not the free-API we used). We notice that many phishing URLs become inaccessible quickly after they are blacklisted. To interact with the *live* phishing server, we build a crawler to fetch phishing URLs from the four blacklists every 30 minutes. Then we immediately use our measurement tool to load

the phishing page, feed the fake credential, and record the network traffic.

We also considered that situation where the phishing servers use cloaking techniques. More specifically, the phishing server may check the IP and User-Agent of the incoming request to see if the request is coming from a university, a security company, or a web crawler. In those cases, the phishing server may drop the request or return a benign page to avoid being detected. As such, we put our crawler behind web proxies and use a realistic User-Agent.

As shown in Table 3.1, we collected 190,087 unique phishing URLs (after removing duplicated URLs between the four blacklists). Among them, 68,751 (38.26%) are "dead", and the rest 110,934 (61.74%) are still alive. Figure 3.1 shows that the live pages are typically more recently-reported compared to the dead ones. 80% of the live pages were reported just 1 hour ago (by the time we visited the pages), while the dead pages were reported much earlier.

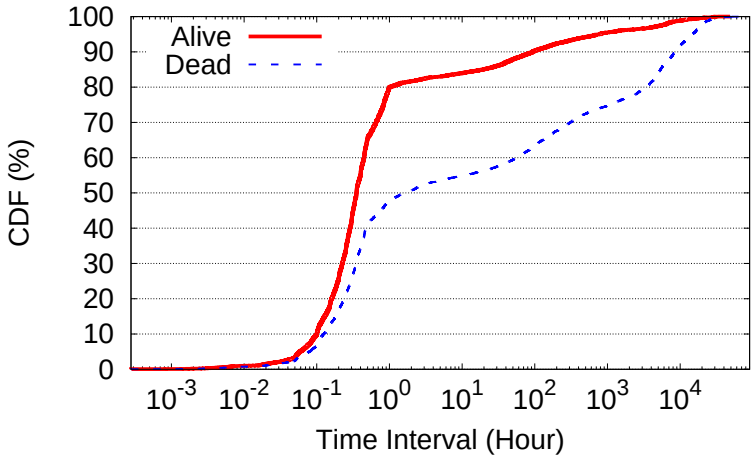

Figure 3.1: The gap between the time when a URL was blacklisted and the time when our crawler visited the URL.

3.3.1 Login Results

Not all the live URLs are still phishing pages. In fact, many of the live URLs have been reset to legitimate/blank pages. Among 110,934 (61.74%) live URLs, only 47,703 (26.55%)

still contain a login form. We use our measurement tool to feed the fake credentials to and record all the network traffic. Out of the 47,703 phishing sites, we successfully submitted the login form for 41,986 sites (88.01%), We manually checked the pages with failed logins. Some of the forms not only asked for username and password, but also required answering security questions by clicking a drop-down list. Other failure cases are caused by the special format constraints for the input data. We admit that there is still room for improving our measurement tool.

3.3.2 Identifying Relevant Network Traffic

Among all the network requests, we look for those that contain the seeded password. We consider both `POST` and `GET` HTTP/HTTPS requests. We expect that some phishing pages may encode or hash the credentials before transmission. As such, in addition to matching the plaintext, we also attempt to match the hashed/encoded versions of the password. We apply 31 hash/encoding function on the password and look for a match in the traffic (Table 3.2). After the filtering, we identified 41,986 network requests that contain the leaked password (either plaintext or hashed).

Table 3.2: Functions used to obfuscate login credentials.

Hash or encoding functions (31 in total)
MD2, MD4, MD5, RIPEMD, SHA1, SHA224, SHA256, SHA384, SHA512, SHA3_224, SHA3_256, SHA3_384, SHA3_512, blake2b, blake2s, crc32, adler32, murmurhash 3 32 bit, murmurhash 3 64 bit, murmurhash 3 128 bit, whirlpool, b16 encoding, b32 encoding, b64 encoding, b85 encoding, url encoding, gzip, zlib, bz2, yenc, entity

3.4 Client-Side Information Flow

In this section, we investigate the information flows of sending credentials from the client side. To identify HTTP requests containing user credentials, we follow the methodology discussed earlier in §3.3. Out of the 47,703 phishing sites with a login form, we are able to track credential information flow for 41,986 phishing sites.

3.4.1 Credential Sending Format

Recall that credential information could be transmitted in plaintext or using some encoding/hashing schemes (*e.g.*, MD5, SHA256). Table 3.3 shows statistics of different types of data formats used across phishing sites. Interestingly, most phishing sites (99%) use human interpretable formats (*i.e.*, either plaintext or URL encoding), and only a small fraction, 0.11% use other more advanced encoding schemes. This implies that most attackers did not try to obfuscate the information flow.

Table 3.3: Data format of credentials sent from the client-side.

Format	Plaintext	URL Encoding	Other Encoding
# Phishing sites	6,324 (15.06%)	35,616 (84.83%)	46 (0.11%)

3.4.2 Identifying Third-party Collectors

Any domain that collects credential information, and is not a direct phishing server domain, is considered to be a third-party collector. In total, we identify 694 third-party collector domains that include 1,021 URLs. These are entities that collect stolen credentials, and would be a vital component to target while building phishing defenses.

But do all phishing sites share credentials with third-party collectors? Table 3.4 shows

the distribution of phishing sites that share credentials with different number of third-party collectors. There are about 5% of phishing sites sharing credentials with third-party collectors from the *client side*. The percentage is not high, but there is a sizeable number. There are 2,019 phishing sites that interact with one or more third-party collectors. In fact, 56 phishing sites share with more than 2 third-party collectors.

Table 3.4: Distribution of third-party collectors. About 95% phishing sites don't have third-party collectors and they only send credentials to the original hosting domain.

# 3rd-parties	0	1	2	≥ 3
# Phish sites	39,967 (95.19%)	1,963 (4.68%)	48 (0.11%)	8 (0.02%)

3.4.3 Third-party Collectors vs. Phishing Sites

Next, we look at two aspects of third-party collectors that have implications for disrupting their network. *First*, do third-party collectors link with multiple phishing sites? If each third-party collector served a single phishing site, we would have to take down as many collector domains as the number of phishing sites. But we observe a different trend. Figure 3.2 shows the distribution of fraction of phishing sites covered by different external collectors. We find that the top 100 external collectors (out of 694) link with a majority, 68.76% of the phishing sites. Thus, even targeting a small fraction of external collectors can disrupt many phishing efforts.

Second, we further examine the geographical locations of third-party collectors. Third-party collectors are spread over 37 countries, but 42% of them are located in the U.S. When third-party collectors are based in a country different from the phishing site they link with, it would require different law enforcement efforts to take down their domains. We analyze the relative locations of phishing sites and their associated third-party collectors. Among 1,408

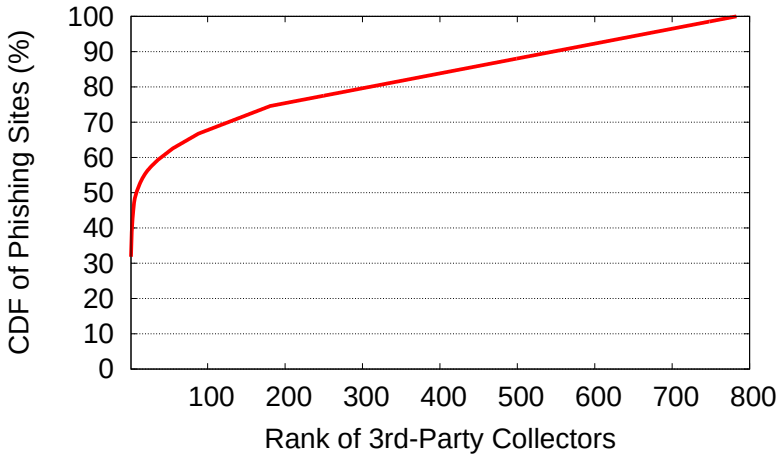

Figure 3.2: Distribution of fraction of phishing sites that connect to different third-party collectors. On the x-axis, third-party collectors are ranked based on # of phishing sites connected.

IP address pairs made of phishing sites, and their connected collector domains[3], 44% are co-located in the same country. A significant fraction of this number can be attributed to the U.S.—96% of co-located pairs are located within the U.S. The remaining 56% non-co-located pairs include phishing sites that are spread over 52 countries, and collectors over 37 countries. We also note that a significant fraction, 88% of non-co-located pairs involve phishing sites or collectors based in the U.S. The detailed breakdown for is shown in Figure 3.3. We only show the top 5 countries of phishing servers and third-party collectors and group the rest into "other". Overall, this means that a majority of pairs are not based in the same country, and this could raise challenges to disrupt their network.

3.4.4 How Reputed are Third-Party Collectors?

We investigate whether the third-party collectors are already known malicious entities or those with poor reputation.

[3]In total, there were 2,170 pairs, but we were unable to determine the geolocation for all of them.

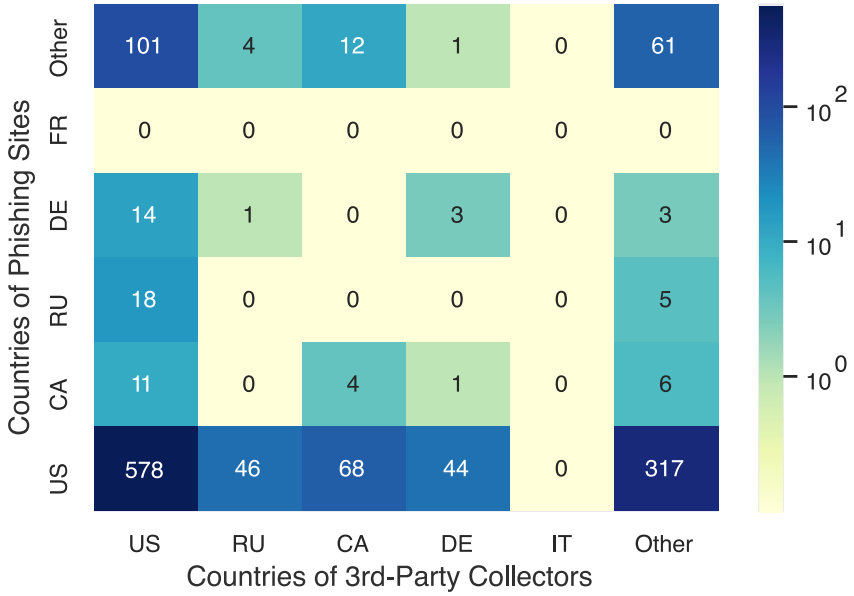

Figure 3.3: Countries of phishing sites and third-party collectors.

We start by analyzing the reputation of third-party collector domains using *The Talos IP and Domain Reputation Center (by Cisco)*[4]. The Talos IP and Domain Reputation Center is a real-time threat detection network. They provide a reputation score of "Good", "Neutral" and "Poor". Here "Good" means little or no threat activity has been observed. On the contrary, "Poor" indicates a problematic level of threat activity has been observed, while "Neutral" means the domain is within acceptable parameters. Note that "Neutral" is a common case for most domains, even well-known ones such as `facebook.com`. Among all 694 third-party collector domains, we obtain reports for 508 (73.20%) domains. We find that 14 of them are labeled "Good", 146 are "Poor" and the rest 348 are "Neutral".

We take a closer look at these scores—*First*, it is interesting to see that a significant fraction, 29% of domains already have poor reputation, but still managed to stay alive and form a collector network. *Second*, it is surprising to see 14 domains marked as "Good". We

[4]urlhttps://www.talosintelligence.com/

find these are indeed legitimate domains, *e.g.*, `delta.com`, `google.com`, `doubleclick.net`, `dropbox.com`. On examining the HTTP logs for these "Good" collectors, we find there are different reasons for them acting as third-party collectors. For example, certain phishing sites were sending the credentials to the legitimate sites that they were trying to impersonate (*e.g.*, `delta.com`). We suspect that they were trying to check the validity of credentials. Some good sites were collecting credentials because they were used by attackers as a web hosting service (*e.g.*, `dropbox.com`). Finally, popular ads platforms or tracking services such as Google Ads and `doubleclick.net` also received the stolen credentials. A close inspection shows that the phishing sites were connecting to these tracking services to keep track of the number of victims. While doing so, the stolen credential was "accidentally" placed within the referer URL of the request.

Only analyzing domain reputation does not provide the full picture. There can be legitimate domains that host malicious URLs. We leverage VirusTotal[5] to scan external collector URLs. VirusTotal has been widely used by the security community in prior work [50, 67]. For each submitted URL, VirusTotal provides a report from 66 diverse scanners that may classify it into one or more categories that indicate whether a URL is problematic, clean or unrated. Problematic categories include "Malware site", "Phishing site", "Malicious site", "Suspicious site", and "Spam site".

Figure 3.4 shows the distribution of collector URLs detected by VirusTotal scanners that fall into any one of the problematic categories. A small fraction, 16% of URLs are not flagged by any scanner, and will likely remain under the radar for a long time. On the other hand, a large majority, 84% of collector URLs are classified as problematic by at least one scanner. Table 3.5 shows a further breakdown of collector URLs that are flagged by at least one scanner. Interestingly, 81% of them are flagged as 'Phishing sites'. This suggests

[5] https://www.virustotal.com

the possibility of a network of phishing sites that exchange credential information with each other.

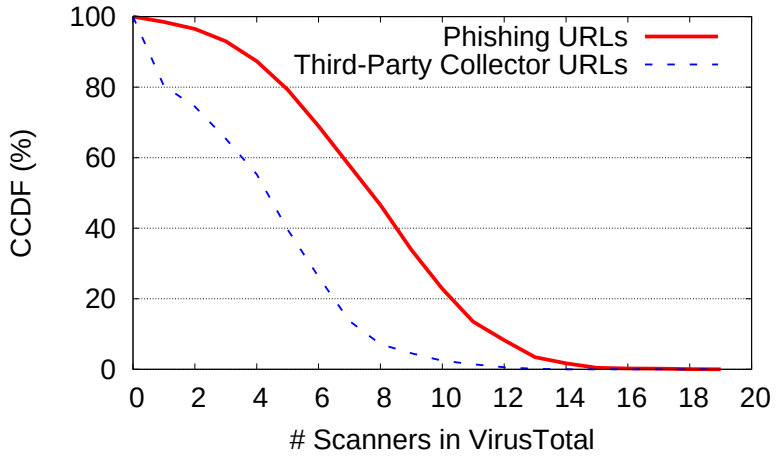

Figure 3.4: CCDF of Number of VirusTotal scanners that flagged the given URL as malicious. The majority of the third-party collectors are already flagged by VirusTotal scanners.

Table 3.5: Number of URLs detected by VirusTotal.

	# Phishing Sites w/ Third-party Collectors	# Third-party Collector URLs
Total	2,019	1,021
"Phishing Site"	1970 (97.57%)	823 (80.63%)
"Malicious Site"	1,840 (91.13%)	777 (76.10%)
"Malware Site"	239 (13.13%)	176 (17.24%)

To summarize, while a majority of third-party collector domains do not have a poor reputation, a large majority of their URLs are already known to be problematic, *e.g.*, for phishing. In spite of the poor URL reputation, it is surprising that these collector URLs are still alive. To understand the age of the collector domains, we examine WHOIS records to determine their domain registration dates. Figure 3.5 shows that the distribution of domain registration time of third-party collectors is quite close to that of the phishing servers. Many of the collector domains are actually aged domains. 20% of them were registered 10 years ago. About half of them were registered before 2016. This suggests that the collector network

has largely remained undisrupted.[6]

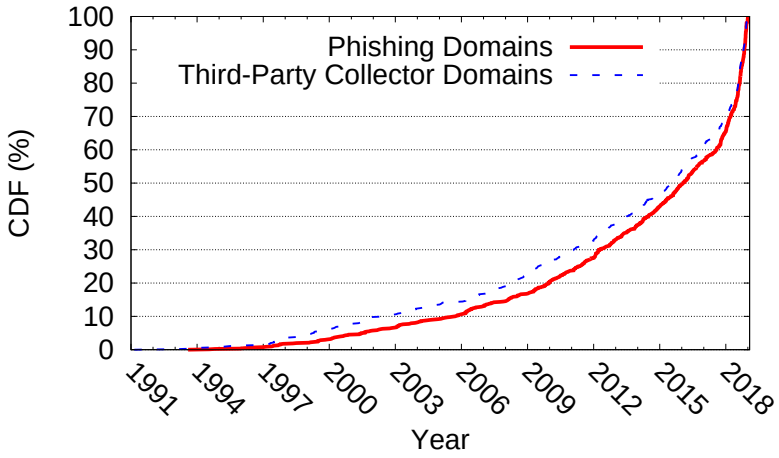

Figure 3.5: Registration time of phishing domains and third-party collector domains. Third-party collector domains have a similar distribution with phishing domains.

The top information collectors ranked by the number of phishing sites they serve is presented in Table 3.6. The largest information collectors here is "w32.info". This site was once hosting many phishing kits for downloading (not anymore). We confirm this by checking the achieved versions of this website[7]. It is possible that the kit developers were using this site to collect a copy of the stolen credentials from people who use their kits to perform phishing. We also notice that web hosting services or dynamic DNS services are often used to collect credentials for multiple collector URLs (possibly for different attackers). One interesting case is `ip-api.org`, a website that provides a lookup service for IP geolocations. 89 phishing websites were sending stolen credentials to this server via "`http://cdn.images.ip-api.org/s.png`". We suspect that this service might have been compromised.

[6]We removed known web hosting domains (as reported by Alexa top 1 Million) from this plot to avoid a possible wrong interpretation. Malicious collector URLs hosted on a legitimate webhosting service would show up as being long-lived, while the exact age of the URL would be hard to determine.

[7]https://web.archive.org/web/20151128133828/http://w32.info:80/

Table 3.6: Top 10 third-party collectors.

Rk.	Third-party Collector	Phish URLs	Domain Category	Collector URLs
1	w32.info	731	Infection source	1
2	jquerymobile.ga	168	Uncategorized	2
3	ip-api.org	89	Geolocation API	1
4	serveirc.com	57	Dynamic DNS	57
5	imgur-photobox.com	50	Uncategorized	1
6	000webhostapp.com	28	Web hosting	26
7	ptpjm.co.id	17	known infection	3
8	servehttp.com	16	Dynamic DNS	8
9	redirectme.net	16	Dynamic DNS	16
10	fitandfirmonline.com	14	Uncategorized	14

3.5 Server-Side Information Flow

In this section, we move to the server side to analyze the information flow of credential transmission. The challenge here is that we don't have internal access to the phishing servers. Our solution is based on the fact that some (careless) attackers may have left the phishing kit in publicly accessible locations on the phishing server [26]. As such, we attempt to retrieve these phishing kits and infer the server-side information flow by combining static and dynamic analysis.

3.5.1 Collecting Phishing Kits

We search for phishing kits on servers that host the phishing websites. Unlike §3.4, we inspect all 179,865 phishing URLs (*i.e.*, not just sites that were still alive) for possible phishing kits. The main reason is that even if a phishing site has been disabled[8], it is possible that phishing kits are still left accessible on the server [54].

[8]By disabled we mean the phishing site has been reset to a legitimate website by phisher or the web administrator.

Since we have no knowledge of possible file names to query for (on the phishing server), we start with phishing servers that enable directory listing to obtain a list of files available on the server. Prior work suggests that phishing kits are usually compressed/archive files (*e.g.*, zip, tar, rar) [26]. For each phishing site URL, we do the following steps: (1) Check if directory listing is available for each path segment in the URL (*i.e.*, separated by '/'). (2) If we find a directory listing, we download all compressed/archive files. (3) For each downloaded file, we decompress it and check the PHP/Python/Ruby/HTML files to make sure it is indeed a phishing kit. To further increase our chance to retrieve more phishing kits, we identify the most frequent 50 kit names (based on the first 1000 kits downloaded earlier). Then given a phishing URL, we exhaustively query each path segment for these 50 file names, in addition to checking the directory listing. This helps us to obtain kits from servers that disabled the directory listing.

We applied the above method to querying 179,865 phishing sites, and obtained 2,064 phishing kits in total. Compared to earlier work [13, 33], our hit rate for finding a phishing kit on phishing servers is lower—we observe a hit rate of 1.15%, compared to 11.8% in prior work. We suspect that phishers are being more careful, and avoid leaving publicly visible traces of their malicious activity.

3.5.2 Identifying Third-party Collectors

On the server side, the stolen credentials can be sent to third-parties in addition to the attacker who deployed the phishing kit. More specifically, prior work shows that phishing kits may contain backdoors [26] that allow third-parties to collect the stolen credentials. Often cases, the backdoors are stealthily inserted into the phishing kit code by the kit developers. When the kit is used by attackers to perform phishing, the kit developer also

receives a copy of the credentials.

To differentiate backdoor collectors, we conduct both dynamic and static analysis. The methodology is inspired by that in [26]. The assumption is that backdoors are usually planted stealthily, which are not directly visible in plaintext in the kit code. As such, we first apply *static analysis* by performing a text search within files in a kit to identify email addresses, and URL endpoints (for HTTP requests) that collect credentials. Then we put the phishing kit in a sandbox for a *dynamic analysis* to capture all the outbound HTTP and email traffic that transmit the stolen credentials. Any collector identified from dynamic analysis, but not identifiable via plain text search through static analysis, can be considered to be a backdoor collector (*i.e.*, the third-party). Note that throughout our dynamic analysis, we did not observe any outbound HTTP/HTTPS traffic from any phishing kits. For brevity, we only introduce the details of the email channel analysis below.

3.5.3 Static and Dynamic Analysis

Our *static analysis* is based on a simple method to extract the collectors in plaintext. The idea is to locate the `mail(to,subject,...,header)` function and identify their "to" and "header" variables. The "to" address is considered to be a collector on the server side. Out of 2,064 phishing kits in total, we successfully detected email addresses in 1,974 phishing kits. In total, we extracted 1,222 valid email addresses (as receivers).

For the *dynamic analysis*, we build up an Apache web server and upload all phishing kits to it. We record all the outbound traffic but block the corresponding ports (*e.g.*, port 25 for email) to avoid actually sending data to the attackers. For each phishing kit, since we do not know which files build the phishing pages, we run our tool described in §3.2 to detect login forms to locate the phishing page. Then like before, we use our measurement

tool to automatically fill in the username and password, and submit the information to the experimental server. To capture the server-side actions, we dump all the emails in the mail queue and all the HTTP logs.

We run the dynamic analysis on all of the 2,064 phishing kits. Using tools described in §3.2, we successfully logged into 1,181 (57%) phishing kits. Note that for 88 (9%) of these phishing kits, we did not find any outbound emails. It is possible that these attackers would rather log into the phishing server to retrieve the stolen credentials (step`3.2` in Figure 2.1). For the rest of the phishing kits, we search the leaked password in their outbound emails to make sure they are sending the stolen credentials. We only find 6 emails that did not contain the password (the emails were for status reports). For these 1,093 phishing kits, we compare the result of dynamic analysis and that of static analysis, and find 46 phishing kits with backdoor emails (4.2%).

3.5.4 Server-side Collectors

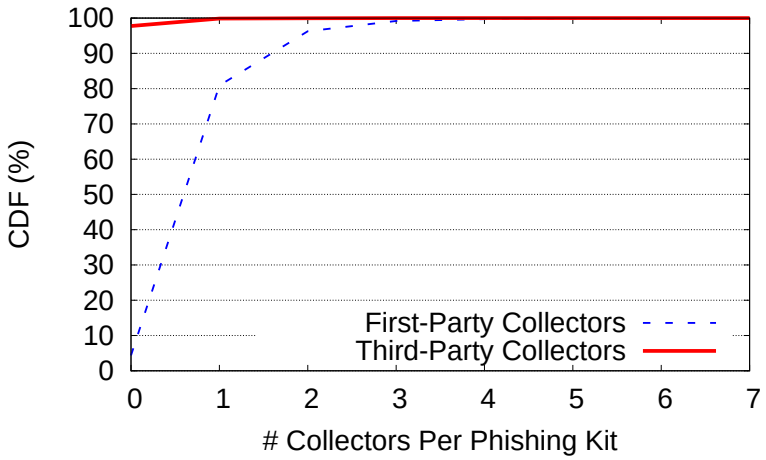

Figure 3.6: Number of server-side collectors per phishing kit.

Figure 3.6 shows the number of server-side collectors per phishing kit. Each collector is identified as a receiver email address. Most phishing kits (96%) do not have a backdoor

(third-party) collector. Among the 46 kits that have a backdoor, there is usually only one backdoor collector per kit. In total, there are 24 unique backdoor email addresses. Table 3.8 further displayed the top 5 third-party email addresses, ranked by the number of associated phishing kits. Some collectors (*e.g.*, `equallib12@gmail.com`) were embedded into multiple phishing kits.

Table 3.7: Top 5 first-party collectors on the server side.

Rk.	1st-parties	# Phishing Kits	# Domains
1	nosaplanter@gmail.com	76	10
2	chrismason601@gmail.com	27	6
3	mrgodwin2233@gmail.com	21	3
4	work-hard@dreambig.com	15	13
5	samzysoprano2@gmail.com	13	6

Table 3.8: Top 5 third-party collectors on the server side.

Rk.	3rd-parties	# Phishing Kits	# Domains
1	equallib12@gmail.com	10	6
2	hhforexxx@gmail.com	5	4
3	ebay1235x@gmail.com	4	2
4	sesurityas@yandex.com	3	2
5	boxnr1234@gmail.com	2	2

Regarding the first-party collectors, Figure 3.6 shows that most phishing kits have one first-party collector, but about 20% kits have more than one collectors. As shown in Table 3.7, some of the first-party collectors are associated with multiple kits, which indicates coordinated phishing campaigns, *i.e.*, one attacker deployed the kits onto multiple phishing servers.

Chapter 4

Online Scan Engine in Phishing Defense

In this chapter, we want to understand how VirusTotal and its vendors scan phishing URLs. We ask key questions regarding how the labels should be interpreted and used: (1) how effective are VirusTotal's vendors (scanners) in detecting basic phishing pages? (2) how quickly will the scanning results become available? (3) how consistent are the scanning results across vendors, and between vendor-APIs and VirusTotal API? (4) how quickly can VirusTotal react to phishing site changes such as take-down? (5) how much do basic obfuscation techniques help with evading the detection?

To answer the questions, we set up fresh phishing websites on newly registered web domains. Then by submitting the phishing URLs to VirusTotal, we collect incoming network traffic to the phishing servers and the VirusTotal's labeling results for these URLs. We have carefully designed the experiments to ensure research ethics.

4.1 Phishing Site Setups

4.1.1 Phishing Page Content

We create two phishing pages that mimic the login pages of PayPal [6] and IRS (Internal Revenue Service) [1]. PayPal is chosen for its popularity — more than 30% of phishing URLs at major blacklists are targeting PayPal [56]. IRS, as a comparison baseline, is not commonly targeted. We replicate the original sites of PayPal and IRS, and modify the login form so that login information will be sent to our servers. By default, we disable any form of cloaking for the phishing sites. Cloaking means a phishing site hides itself by showing a benign page when it recognizes the incoming request is from a known security firm [38, 53]. The `robots.txt` is also set to allow web crawlers to access the phishing page.

4.1.2 Domain Names

We register *fresh* domain names for our phishing sites. This is to make sure the domain names do not have any past history that may interfere with the measurement. To prevent innocent users from mistyping the domain names (*i.e.*, accidentally visiting our websites), we register long random strings as domain names (50 characters each) from NameSilo [5]. For example, one of the domain names is "`yzdfbltrok9m58cdl0lvjznzwjjcd2ihp5pgb295hfj5u42ff0.xyz`".

4.1.3 Web Hosting

We host the phishing websites at a web hosting service called Digital Ocean [2] on static IPs. Before the experiment, we made sure all the IPs and domain names are publicly accessible,

and are not blacklisted by any major blacklist. We have informed Digital Ocean of our research, and have received their consent.

4.2 Experiment Design

The experiments were conducted from March to April in 2019, including a *main* experiment and a *baseline* experiment.

4.2.1 Main Experiment

The main experiment is designed to measure (a) the phishing detection accuracy of VirusTotal and vendors; (b) the potential inconsistency between VirusTotal API and the vendors' APIs; (c) the reaction of VirusTotal to changes of phishing sites. Recall that there are 18 vendors that have their own scan APIs. To accurately capture their impact, we set up separate phishing sites (1 PayPal and 1 IRS) for each vendor (36 sites in total).

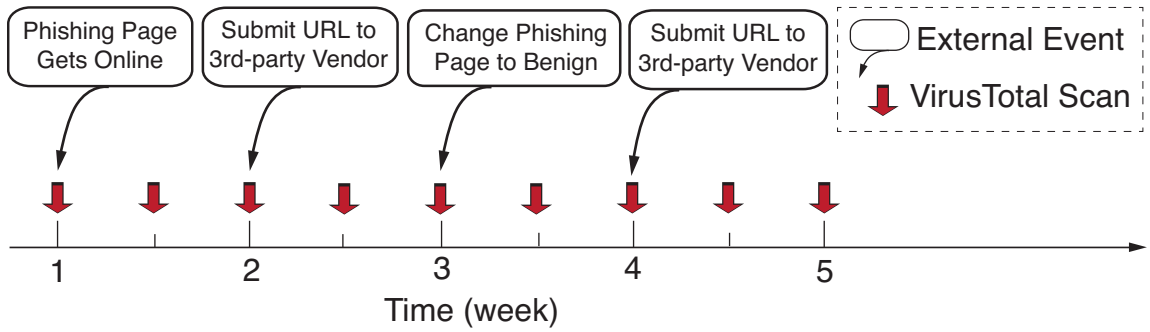

Figure 4.1: Illustration of the `main` experiment on a given phishing site. The third-party vendor is one of the 18 vendors that provide their own scan APIs.

For each phishing site, we conduct a 4-week experiment as illustrated in Figure 4.1. We periodically submit the phishing URL to VirusTotal's *scan API*. The VirusTotal scan API will trigger the scanning of (some of) the third-party vendors. VirusTotal scanning is conducted

twice a week on Mondays and Thursdays. At the same time, we schedule 4 external events (on the Mondays of each week):

1. Week1: We put the phishing site online.
2. Week2: We submit the phishing URL to one of the 18 vendors who have their own scan APIs.
3. Week3: We take down the phishing page, and replace it with a benign page (*i.e.*, a blank page).
4. Week4: We submit the phishing URL to the same third-party vendor as week2.

Note that (2) and (4) are designed to measure the consistency between VirusTotal scanning and the vendors' own scanning. Each phishing site is only submitted to one vendor API so that we can measure the differences between vendors.

During the experiment, we collect two types of data. First, we collect the *labels* for all the phishing URLs using VirusTotal's querying API. Note that after a URL is submitted for scanning, the scanning results (*i.e.*, labels) might not be immediately available in the VirusTotal database. So we crawl the labels every 60 minutes to track the fine-grained dynamic changes. Second, we log the incoming network traffic to all of the phishing servers.

4.2.2 Baseline Experiment

The baseline experiment is to measure the long-term reaction of VirusTotal after *a single VirusTotal scan*. We set 2 additional phishing sites (PayPal and IRS) and only submit the URLs to VirusTotal scan API for once at the beginning of the first week. Then we monitor incoming traffic to the phishing servers, and query the VirusTotal labels in the next 4 weeks.

4.2.3 Summary

In total, 38 websites are set up for our experiments (36 for main, 2 for baseline). There are 19 PayPal sites and 19 IRS sites. All the PayPal sites have identical web page content (hosted under different domain names). All the IRS sites share the same content (with different domain names).

4.3 Measurement Results

4.3.1 Delay of Label Updating

A closer examination of Figure 4.2 shows that VirusTotal has a delay of updating the labels to its database. More specifically, the x-axis in Figure 4.2 is the label querying time (label crawling is done every hour). We observe that only after the second VirusTotal scan will the first scan result get updated to VirusTotal database.

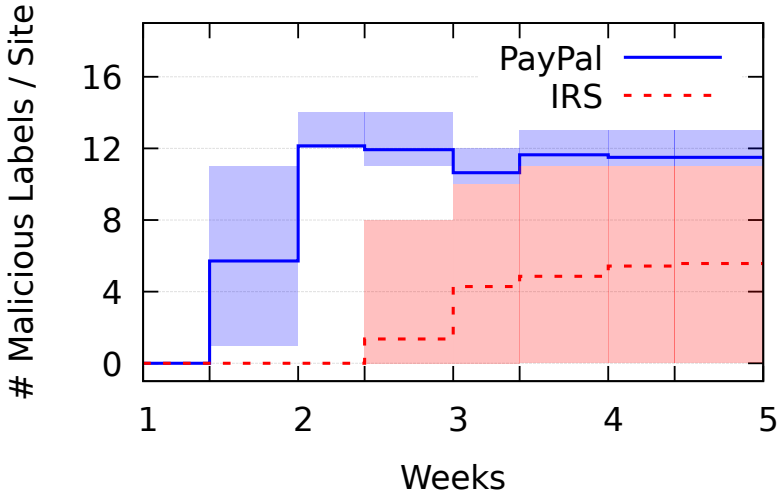

Figure 4.2: The average, maximum, and minimum number of malicious labels per site (main experiment).

For example, in the first week, we submit the PayPal URLs to VirusTotal on day-1. The querying API returns "benign" labels since these URLs were never scanned before by any vendor. Then after we submit the URLs again on day-4, the querying API starts to return "malicious" labels from some vendors. Based on the "scanning time" on the returned labels, we see that theses "malicious" labels are actually originated from the scan of *day-1*. This means, although some vendors have already detected the phishing page on day-1, the results would not be updated to VirusTotal database until the next scan request on day-4.

The result shows VirusTotal uses "pull" (instead of "push") to get scanning results from vendors. The pull action is only triggered by VirusTotal's scan API but not the querying API. Our baseline experiment further confirms that vendors do not proactively push new results to VirusTotal database. In the baseline setting, we only submit the URL to VirusTotal on day-1 without any further actions. By querying the labels for the next 4 weeks, we confirm that the scanning results are never updated back to VirusTotal. If a researcher only scans a URL once and queries the database afterward, she cannot get the updated labels. Instead, the researcher needs to perform *two* scans: one for URL scanning, and the other for triggering the database update.

4.3.2 PayPal vs. IRS

The VirusTotal scan during week-1 failed to detect any IRS website (Figure 4.2). After we directly submitted the IRS URLs to individual vendors, some of the IRS pages started to be flagged. On the contrary, all PayPal sites were flagged by at least 10 vendors during week-1. One hypotehsis is that PayPal phishing pages are more common, and thus the vendors' models (*e.g.*, classifiers) are better trained to detect them. To validate this hypothesis, more rigorous tests are needed by testing a wide range of phishing pages (content), which is part

of our future work.

4.3.3 VirusTotal vs. Vendors

The vendors' own scan APIs and VirusTotal's scan APIs do not always return consistent results. Note that when we use the vendor's API, the API returns the scanning result too. Unlike VirusTotal API, vendors' own APIs ask the user to wait until the scanning is finished so that the user gets the real scanning result. In Table 4.1, we show the vendor API results (on Monday of week 2), and the VirusTotal labels right before and after that (results for the Thursday of week 1 and the Thursday of week 2 respectively). We have considered the delay of label updating of VirusTotal and manually aligned the scan time accordingly.

Table 4.1: Inconsistent labels between VirusTotal scan and Vendor scan. "1" means malicious and "0" means benign.

Vendor Name	**Brand**	**VTotal Before**	**Vendor (week-2)**	**VTotal After**
Forcepoint	PayPal	**0**	**1**	**0**
Sucuri Site Check	PayPal	**0**	**1**	**0**
Quttera	PayPal	**0**	**1**	**0**
URLQuery	PayPal	**0**	**1**	**0**
ZeroCERT	PayPal	**0**	**1**	**0**
Fortinet	IRS	**0**	**1**	**0**
Google Safe Brows.	PayPal	**0**	**1**	**0**
	IRS	**0**	**1**	**0**
Netcraft	IRS	**0**	**1**	**1**

As shown in Table 4.1, there are in total 8 vendors that show inconsistent results. Most vendors have a "0-1-0" pattern for PayPal sites including `Forcepoint`, `Sucuri`, `Quttera`, `URLQuery`, `ZeroCERT`, and `Google Safe Browsing`. This means through VirusTotal scan, these vendors return the label "benign", even though their own scan APIs can detect that the page as "malicious". A possible explanation is that these vendors did not give VirusTotal

the permission to trigger their scanners. Instead, VirusTotal runs stripped-down versions of the scanners [9, 44], which cannot detect the phishing page.

For IRS pages, we show that `Fortinet`, `Google Safe Browsing`, and `Netcraft` have detected these IRS pages via their own scan APIs. However, only `Netcraft` has shared this result to VirusTotal after the scan. It should be noted that we have tried to analyze which scanners indeed visited the phishing sites. This attempt failed because scanners were actively hiding their identity by using proxies and cloud services. Overall, the result shows the VirusTotal does not always reflect the best detection capability of a vendor. If possible, researchers should cross-check the results with individual vendors' APIs.

4.3.4 Detection Accuracy of Vendors

In Table 4.2, we list all 15 vendors that detected at least one phishing site during the first two weeks (we took down the phishing pages after week-2). We show that even the best vendors cannot detect all phishing sites. The most effective vendors such as `Netcraft` flagged 14 (out of 18) PayPal pages and 12 (out of 18) IRS pages. It is not clear why some sites are not detected given that all 18 PayPal (IRS) sites have the identical content (except for using a different random string as the domain name). In addition, we observe that some of the vendors always flag *the same subset of phishing sites*. For example, `Netcraft`, `Emsisoft`, and `Fortinet` flagged the same 26 sites. Similarly, `Malwarebytes`, `BitDefender` and `ESET` flagged the same 15 sites. This indicates the possibility that certain vendors would copy (synchronize with) each other's blacklist. To validate this hypothesis, more rigorous experiment is needed in future work.

Table 4.2: A list of all the vendors that successfully detected the phishing pages (during the first 2 weeks).

Rank	Vendor	Total	PayPal	IRS
1	Netcraft	26	14	12
2	Emsisoft	26	14	12
3	Fortinet	26	14	12
4	Sophos	23	14	9
5	CRDF	17	14	3
6	Malwarebytes hpHosts	15	14	1
7	BitDefender	15	14	1
8	ESET	15	14	1
9	G-Data	14	14	0
10	Kaspersky	13	1	12
11	Phishtank	10	10	0
12	CyRadar	8	5	3
13	Avira	6	0	6
14	CLEAN MX	6	4	2
15	Trustwave	3	3	0

4.3.5 Reaction to Phishing Take-down

We observe that vendors do not quickly take a URL off the blacklist after the phishing site is taken down. On the Monday of week-3, we took down all the phishing pages and replaced them with benign pages. However, Figure 4.2 shows the number of malicious labels does not drop even after multiple re-scans.

After examining the results for each vendor, we find 4 vendors that flip some "malicious" labels to "benign" after the third week (for PayPal sites only). Figure 4.3 shows these 4 vendors and the number of phishing sites they flagged over time. `CyRadar` and `CLEAN MX` already started to flip their malicious labels in week-2 (before phishing take-down), which is not necessarily a reaction to the take-down. `Fortinet` flipped the label on one site in week-4. `Avira` is likely to be reacting to the take-down since it changed all "malicious" labels to "benign" right after the event. Interestingly, the labels were quickly reversed to

"malicious" in the next scan.

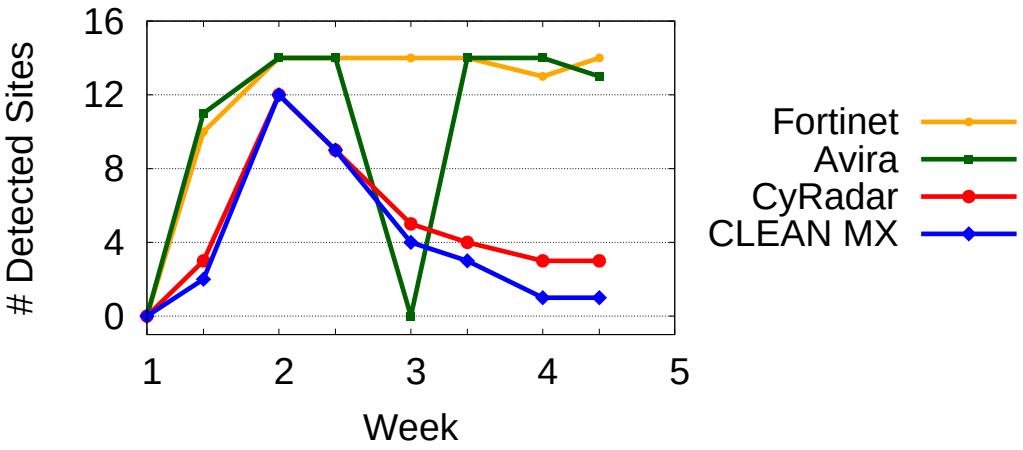

Figure 4.3: Four vendors show a sign of reaction to the phishing take-down (PayPal sites).

Chapter 5

Discussion

5.1 Implications of Results

Our measurement results have several key implications. *First*, credentials sharing happens throughout the phishing process at both client and server side, which exposes the stolen credentials to more malicious parties. The good news is that third-party sharing is not yet prevalent. *Second*, from the phisher's perspective, credential sharing can be both intended (*e.g.*, for validating the stolen credentials and tracking attack statistics) or unintended (*e.g.*, due to backdoors planted by phishing kit developers). *Third*, from the defender's perspective, client-side phishing efforts are easier to detect. We find that over 80% of client-side 3rd-party collectors are already flagged by VirusTotal. However, the problem is that they were not effectively taken down (they are usually in a different country compared to the phishing site). Nevertheless, defense schemes can still add these domains into local network blacklists to block credential sharing. *Fourth*, server-side efforts are harder to measure and disrupt. Web-hosting platforms can significantly contribute to phishing defenses by searching for phishing kits, and take action to block such sites, or issue a warning to the site moderator (in case they were compromised). *Fifth*, our experiments towards VirusTotal collectively involve 62 (38+24) phishing sites. We show that vendors have an uneven detection performance. In the main experiment, only 15 vendors have detected at least one site. Even the best vendor only detected 26 out of 36 sites, despite that these 36 sites have near-identical settings. Given

that vendors have an uneven capability, and their labels should not be treated equally when aggregating their results. Besides, we show the delays of label updating due to the non-proactive "pull" method of VirusTotal. We also illustrate the label inconsistency between VirusTotal scan and the vendors' own scans.

5.2 Using third-party Sharing Channel for Defense

We believe that third-party sharing (and backdoors) can also be used by defenders for good purposes. For example, for known third-party collectors (backdoor email addresses or client-side collectors), instead of directly shutting them down, the defenders (*e.g.*, law enforcement, service providers) may keep them alive but *take away the ownership from the malicious parties*. For example, Google can block the attacker from accessing the Gmail account that acts as the backdoor collector. Then Gmail's security team can keep this account alive as a vantage point to monitor the phishing activities from the same class of phishing kits. The benefit is that whenever the corresponding phishing kits are used to perform phishing in the wild, the defenders can directly pinpoint the location of the attackers (since the phishing kits will contact the backdoor collector). In addition, the defender will also receive a copy of the victim list, which allows defenders to take early actions to alert the victims.

5.3 Future Work about VirusTotal

During our experiments, we observe interesting phenomena that lead to new open questions. First, the vendors' models perform much better on PayPal pages than on IRS pages. Future work can further investigate the "fairness" of vendors' classifiers regarding their performance on more popular and less popular phishing brands. Second, we observe that some vendors

always detect the same subset of phishing sites (Table 4.2). If these vendors indeed fully synchronize their labels, then their labels are essentially redundant information. As such, these vendors should not be treated as independent vendors when aggregating their votes. Future work can further investigate the correlation of results between different vendors. Third, many vendors (*e.g.*, `Kaspersky`, `Bitdefender`, `Fortinet`) also provide API for file scanning to detect malware. File scan can be studied in a similar way, *e.g.*, submitting "ground-truth" malware and benign files to evaluate the quality of labels and the consistency between vendors and VirusTotal.

5.4 Limitations

Our study has a few limitations. *First*, while we obtain a complete view of client-side sharing, we still do not have the complete picture on the server-side. We only observe instantaneous sharing of credentials on the server-side, *i.e.*, as soon as the credentials are received by the server. This is a limitation because it is still possible that the server-side scripts may send credentials at a later point of time, *e.g.*, based on pre-set timers. Unfortunately, given the large number of phishing kits we need to test, we cannot monitor them for a long time. *Second*, our server-side analysis is based on the phishing kits—we have no information about phishing sites that do not leave kits publicly accessible. *Third*, we acknowledge that our dataset is biased due to the use of the four phishing blacklists which are skewed towards English speaking countries. However, our dataset still covers phishing sites that target major sectors and a broad set of brands. *Fourth*, when testing VirusTotal, the long domain names may affect the detection accuracy. However, we argue that the long domain names actually make the websites look suspicious, and thus make the detection easier. The fact that certain scanners still fail to detect the phishing sites further confirms the deficiency of

scanners. *Fifth*, the use of "fresh" domain names may also affect the detection performance of vendors, since certain vendors might use "infection vendors" as features (*e.g.*, reports from the victims of a phishing site). In practice, the vendors might perform better on phishing sites that already had victims.

Chapter 6

Related Work

6.1 Password Leakage

While existing works have studied password leakage [23] and password re-use [27, 64, 68], credentials sharing during the *phishing process* wasn't well understood. A related study [66] examined the potential victims of off-the-shelf keyloggers, phishing kits and previous data breaches. They explored how stolen passwords enabled attackers to hijack Gmail accounts.

6.2 Phishing Kit.

Zawoad et al. found 10% of phishing sites had evidence of using phishing kits [78]. Phishers' motivation and thought processes are inferred by analyzing phishing kits [13, 26, 45, 54]. Previous work has also sandboxed phishing kits to monitor their mechanisms and behavior of criminals [33]. Phishers usually use phishing kits to create a series of similar phishing pages [22].

6.3 Phishing Detection & Warning.

Content-based detection methods have been studied extensively. Cantina and Cantina+ [75, 79] base their detection on DOM and search engines information. Researchers also looked into other detection methods based on visual similarities [71], URL properties [21, 49, 67], OCR features [17, 31], and user behavior patterns [29, 63]. Going deeper, phishing hosts have also been extensively studied including compromised sites [28] and malicious web infrastructure [48]. Phishing emails are used to distribute phishing URLs. Phishers can use email spoofing techniques [36, 37] or email header injection [58] to deceive users. Other researchers looked into the effectiveness of phishing websites warning and prevention in web browsers [18, 32, 74]. A key novelty of our work is to track the information flow for credential sharing across different phases of phishing.

6.4 Using VirusTotal for Labeling

VirusTotal has been heavily used by the research community to label both malicious files [25, 40, 41, 43, 46, 47, 61, 65, 69, 73, 76] and suspicious IPs and URLs [24, 35, 51, 55, 59, 60, 62, 67, 70, 77, 80]. A closer examination shows that VirusTotal is used in different ways by researchers.

First, given that vendors often don't agree with each other (or some vendors have never scanned the URL), researchers need to aggregate the labels to determine if the URL is "malicious". Recall that given a URL, more than 60 labels are returned from the vendors via VirusTotal. We find that most papers define a *threshold* — if at least t vendors return a "malicious" label, then the URL is regarded as malicious. Most papers set $t = 1$ [24, 35, 51, 59, 60, 67, 80], while a few papers are more conservative by setting $t = 2$ or 3 [55, 62, 70].

Second, given a vendor, its internal model (*e.g.*, a machine learning classifier) may be updated over time, and thus the labels on URLs may also change over time. Kantchelian et. al investigate this issue for the file scan API [39], and show that one needs to wait for a while before the label gets stabilized. It is unknown if the same issue applies to URL scan.

6.5 Phishing Blacklist

Our work is also related to those that study phishing blacklists [18, 53, 72]. Phishing blacklists often have major delays in blocking new phishing sites [20, 30, 52], and suffer from incomplete coverage [19]. Different blacklists may return inconsistent results [42]. Our work aims to look deeper into the process of how URLs get blacklisted (*i.e.*, URL scanning) by VirusTotal and its vendors. The most relevant work to ours is [53]. The differences are two folds: First, [53] looks into the phishing blacklists used by different browsers (*e.g.*, Chrome, Safari), while we focus on how phishing blacklists are constructed by VirusTotal. Second, [53] focuses on the *cloaking techniques* used by phishing sites, while we focus on the performances of different vendors (scanners), and their consistency.

Chapter 7

Conclusions

In this thesis, we perform an empirical measurement on the information flows of credential sharing during phishing attacks and the label quality of online scan engines during phishing defenses. In Chapter 3, our analysis covers more than 179,000 phishing URLs (47,000 live phishing sites). We show that user credentials are shared in real-time to multiple parties at both the client side and the server side. Although third-party sharing exposes user credentials to even more malicious parties, we argue that defenders may make use of these channels to back-track phishing servers and alert phishing victims. In Chapter 4, we take the initial steps to explore how VirusTotal and its vendors assign labels. We also develop new methods to reliably aggregate different scanning results. As future work, we will continue working on the correlation between the labels from different vendors.

Bibliography

[1] IRS login page. https://sa.www4.irs.gov/ola/, .

[2] Digital ocean. https://www.digitalocean.com/, .

[3] Joe sandbox. https://www.joesecurity.org/, .

[4] Jotti's malware scan. https://virusscan.jotti.org/, .

[5] Namesilo. https://www.namesilo.com/, .

[6] Paypal login page. https://www.paypal.com/us/signin, .

[7] Virscan. http://VirSCAN.org, .

[8] Virustotal. https://www.virustotal.com/, .

[9] VirusTotal FAQ. https://support.virustotal.com/hc/en-us/articles/115002122285-AV-product-on-VirusTotal-detects-a-file-and-its-equivalent-commercial- .

[10] VirusTotal vendors. https://support.virustotal.com/hc/en-us/articles/115002146809-Contributors, .

[11] VirusTotal public API v2.0. https://www.virustotal.com/en/documentation/public-api/, .

[12] How John Podesta's Emails Were Hacked And How To Prevent It From Happening To You, 2016. https://www.forbes.com/sites/kevinmurnane/2016/10/21/how-john-podestas-emails-were-hacked-and-how-to-prevent-it-from-happening-to-you/.

[13] Phish in a barrel: Hunting and analyzing phishing kits at scale, 2017. https://duo.com/blog/phish-in-a-barrel-hunting-and-analyzing-phishing-kits-at-scale.

[14] Data Breach Investigations Report, 2018. https://enterprise.verizon.com/resources/reports/dbir/.

[15] UnityPoint Health Notifies 1.4M Patients of Data Breach Caused by Phishing Attack, 2018. https://www.healthcare-informatics.com/news-item/cybersecurity/unitypoint-health-notifies-14m-patients-data-breach-caused-phishing-attack.

[16] The biggest healthcare data breaches of 2018, 2019. https://www.healthcareitnews.com/projects/biggest-healthcare-data-breaches-2018-so-far.

[17] Sadia Afroz and Rachel Greenstadt. Phishzoo: Detecting phishing websites by looking at them. In *Proc. of ICSC*, 2011.

[18] Devdatta Akhawe and Adrienne Porter Felt. Alice in warningland: A large-scale field study of browser security warning effectiveness. In *Proc. of USENIX Security*, 2013.

[19] Simone Aonzo, Alessio Merlo, Giulio Tavella, and Yanick Fratantonio. Phishing attacks on modern android. In *Proc. of CCS*, 2018.

[20] Calvin Ardi and John Heidemann. Auntietuna: Personalized content-based phishing detection. In *NDSS Usable Security Workshop (USEC)*, 2016.

[21] Aaron Blum, Brad Wardman, Thamar Solorio, and Gary Warner. Lexical feature based phishing url detection using online learning. In *Proc. of AISec*, 2010.

[22] Jason Britt, Brad Wardman, Alan Sprague, and Gary Warner. Clustering potential phishing websites using deepmd5. In *Proc. of LEET*, 2012.

[23] Blake Butler, Brad Wardman, and Nate Pratt. Reaper: an automated, scalable solution for mass credential harvesting and osint. In *Proc. of eCrime*, 2016.

[24] Onur Catakoglu, Marco Balduzzi, and Davide Balzarotti. Automatic extraction of indicators of compromise for web applications. In *Proc. of WWW*, 2016.

[25] Binlin Cheng, Jiang Ming, Jianmin Fu, Guojun Peng, Ting Chen, Xiaosong Zhang, and Jean-Yves Marion. Towards paving the way for large-scale windows malware analysis: Generic binary unpacking with orders-of-magnitude performance boost. In *Proc. of CCS*, 2018.

[26] Marco Cova, Christopher Kruegel, and Giovanni Vigna. There is no free phish: An analysis of "free" and live phishing kits. In *WOOT*, 2008.

[27] Anupam Das, Joseph Bonneau, Matthew Caesar, Nikita Borisov, and XiaoFeng Wang. The tangled web of password reuse. In *Proc. of NDSS*, 2014.

[28] Joe DeBlasio, Stefan Savage, Geoffrey M Voelker, and Alex C Snoeren. Tripwire: Inferring internet site compromise. In *Proc. of IMC*, 2017.

[29] Xun Dong, John A Clark, and Jeremy L Jacob. User behaviour based phishing websites detection. In *Proc. of IMCSIT*, 2008.

[30] Zheng Dong, Apu Kapadia, Jim Blythe, and L Jean Camp. Beyond the lock icon: real-time detection of phishing websites using public key certificates. In *Proc. of eCrime*, 2015.

[31] Matthew Dunlop, Stephen Groat, and David Shelly. Goldphish: Using images for content-based phishing analysis. In *Proc. of ICIMP*, 2010.

[32] Serge Egelman, Lorrie Faith Cranor, and Jason Hong. You've been warned: an empirical study of the effectiveness of web browser phishing warnings. In *Proc. of CHI*, 2008.

[33] Xiao Han, Nizar Kheir, and Davide Balzarotti. Phisheye: Live monitoring of sandboxed phishing kits. In *Proc. of CCS*, 2016.

[34] Grant Ho, Aashish Sharma, Mobin Javed, Vern Paxson, and David Wagner. Detecting credential spearphishing in enterprise settings. In *Proc. of USENIX Security*, 2017.

[35] Geng Hong, Zhemin Yang, Sen Yang, Lei Zhang, Yuhong Nan, Zhibo Zhang, Min Yang, Yuan Zhang, Zhiyun Qian, and Haixin Duan. How you get shot in the back: A systematical study about cryptojacking in the real world. In *Proc. of CCS*, 2018.

[36] Hang Hu and Gang Wang. End-to-end measurements of email spoofing attacks. In *Proc. of USENIX Security*, 2018.

[37] Hang Hu, Peng Peng, and Gang Wang. Towards understanding the adoption of anti-spoofing protocols in email systems. In *Proc. of SecDev*, 2018.

[38] L. Invernizzi, K. Thomas, A. Kapravelos, O. Comanescu, J. Picod, and E. Bursztein. Cloak of visibility: Detecting when machines browse a different web. In *Proc. of IEEE S&P*, 2016.

[39] Alex Kantchelian, Michael Carl Tschantz, Sadia Afroz, Brad Miller, Vaishaal Shankar, Rekha Bachwani, Anthony D Joseph, and J Doug Tygar. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proc. of AISec*, 2015.

[40] Doowon Kim, Bum Jun Kwon, and Tudor Dumitraş. Certified malware: Measuring breaches of trust in the windows code-signing pki. In *Proc. of CCS*, 2017.

[41] Doowon Kim, Bum Jun Kwon, Kristián Kozák, Christopher Gates, and Tudor Dumitraș. The broken shield: Measuring revocation effectiveness in the windows code-signing pki. In *Proc. of USENIX Security*, 2018.

[42] Sabina Kleitman, Marvin KH Law, and Judy Kay. It's the deceiver and the receiver: Individual differences in phishing susceptibility and false positives with item profiling. *PLOS One*, 2018.

[43] David Korczynski and Heng Yin. Capturing malware propagations with code injections and code-reuse attacks. In *Proc. of CCS*, 2017.

[44] Bum Jun Kwon, Jayanta Mondal, Jiyong Jang, Leyla Bilge, and Tudor Dumitraş. The dropper effect: Insights into malware distribution with downloader graph analytics. In *Proc. of CCS*, 2015.

[45] Luda Lazar. Our analysis of 1,019 phishing kits. https://www.imperva.com/blog/our-analysis-of-1019-phishing-kits/, 2018.

[46] Chaz Lever, Platon Kotzias, Davide Balzarotti, Juan Caballero, and Manos Antonakakis. A lustrum of malware network communication: Evolution and insights. In *Proc. of IEEE S&P*, 2017.

[47] Bo Li, Phani Vadrevu, Kyu Hyung Lee, Roberto Perdisci, Jienan Liu, Babak Rahbarinia, Kang Li, and Manos Antonakakis. Jsgraph: Enabling reconstruction of web attacks via efficient tracking of live in-browser javascript executions. In *Proc. of NDSS*, 2018.

[48] Zhou Li, Sumayah Alrwais, Yinglian Xie, Fang Yu, and XiaoFeng Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *Proc. of IEEE S&P*, 2013.

[49] Justin Ma, Lawrence K Saul, Stefan Savage, and Geoffrey M Voelker. Learning to detect malicious urls. 2011.

[50] Rima Masri and Monther Aldwairi. Automated malicious advertisement detection using virustotal, urlvoid, and trendmicro. In *Proc. of ICICS*, 2017.

[51] Najmeh Miramirkhani, Timothy Barron, Michael Ferdman, and Nick Nikiforakis. Panning for gold.com: Understanding the dynamics of domain dropcatching. In *Proc. of WWW*, 2018.

[52] Ajaya Neupane, Nitesh Saxena, Keya Kuruvilla, Michael Georgescu, and Rajesh K Kana. Neural signatures of user-centered security: An fmri study of phishing, and malware warnings. In *Proc. of NDSS*, 2014.

[53] A. Oest, Y. Safaei, A. Doupé, G. Ahn, B. Wardman, and K. Tyers. Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists. In *Proc. of IEEE S&P*, 2019.

[54] 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 *Proc. of eCrime*, 2018.

[55] Alina Oprea, Zhou Li, Robin Norris, and Kevin Bowers. Made: Security analytics for enterprise threat detection. In *Proc. of ACSAC*, 2018.

[56] Peng Peng, Chao Xu, Luke Quinn, Hang Hu, Bimal Viswanath, and Gang Wang. What happens after you leak your password: Understanding credential sharing on phishing sites. In *Proc. of AsiaCCS*, 2019.

[57] Peng Peng, Limin Yang, Linhai Song, and Gang Wang. Opening the blackbox of virustotal: Analyzing online phishing scan engines. In *Proc. of IMC*, 2019.

[58] Sai Prashanth Chandramouli, Pierre-Marie Bajan, Christopher Kruegel, Giovanni Vigna, Ziming Zhao, Adam Doup, and Gail-Joon Ahn. Measuring e-mail header injections on the world wide web. In *Proc. of SAC*, 2018.

[59] Abbas Razaghpanah, Rishab Nithyanand, Narseo Vallina-Rodriguez, Srikanth Sundaresan, Mark Allman, Christian Kreibich, and Phillipa Gill. Apps, trackers, privacy, and regulators: A global study of the mobile tracking ecosystem. In *Proc. of NDSS*, 2018.

[60] Armin Sarabi and Mingyan Liu. Characterizing the internet host population using deep learning: A universal and lightweight numerical embedding. In *Proc. of IMC*, 2018.

[61] Edward J Schwartz, Cory F Cohen, Michael Duggan, Jeffrey Gennari, Jeffrey S Havrilla, and Charles Hines. Using logic programming to recover c++ classes and methods from compiled executables. In *Proc. of CCS*, 2018.

[62] Mahmood Sharif, Jumpei Urakawa, Nicolas Christin, Ayumu Kubota, and Akira Yamada. Predicting impending exposure to malicious content from user behavior. In *Proc. of CCS*, 2018.

[63] Routhu Srinivasa Rao and Alwyn R Pais. Detecting phishing websites using automation of human behavior. In *Proc. of the ACM Workshop on Cyber-Physical System Security*, 2017.

[64] Elizabeth Stobert and Robert Biddle. The password life cycle: user behaviour in managing passwords. In *Proc. of SOUPS*, 2014.

[65] Janos Szurdi and Nicolas Christin. Email typosquatting. In *Proc. of IMC*, 2017.

[66] Kurt Thomas, Frank Li, Ali Zand, Jacob Barrett, Juri Ranieri, Luca Invernizzi, Yarik Markov, Oxana Comanescu, Vijay Eranti, Angelika Moscicki, et al. Data breaches, phishing, or malware?: Understanding the risks of stolen credentials. In *Proc. of CCS*, 2017.

[67] Ke Tian, Steve TK Jan, Hang Hu, Danfeng Yao, and Gang Wang. Needle in a haystack: Tracking down elite phishing domains in the wild. In *Proc. of IMC*, 2018.

[68] Chun Wang, Steve TK Jan, Hang Hu, Douglas Bossart, and Gang Wang. The next domino to fall: Empirical analysis of user passwords across online services. In *Proc. of CODASPY*, 2018.

[69] Haoyu Wang, Zhe Liu, Jingyue Liang, Narseo Vallina-Rodriguez, Yao Guo, Li Li, Juan Tapiador, Jingcun Cao, and Guoai Xu. Beyond google play: A large-scale comparative study of chinese android app markets. In *Proc. of IMC*, 2018.

[70] Liang Wang, Antonio Nappa, Juan Caballero, Thomas Ristenpart, and Aditya Akella. Whowas: A platform for measuring web deployments on iaas clouds. In *Proc. of IMC*, 2014.

[71] Liu Wenyin, Guanglin Huang, Liu Xiaoyue, Zhang Min, and Xiaotie Deng. Detection of phishing webpages based on visual similarity. In *Proc. of WWW*, 2005.

[72] Colin Whittaker, Brian Ryner, and Marria Nazif. Large-scale automatic classification of phishing pages. In *Proc. of NDSS*, 2010.

[73] Michelle Y Wong and David Lie. Tackling runtime-based obfuscation in android with tiro. In *Proc. of USENIX Security*, 2018.

[74] Min Wu, Robert C Miller, and Simson L Garfinkel. Do security toolbars actually prevent phishing attacks? In *Proc. of CHI*, 2006.

[75] Guang Xiang, Jason Hong, Carolyn P Rose, and Lorrie Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *Proc. of TISSEC*, 2011.

[76] Dongpeng Xu, Jiang Ming, Yu Fu, and Dinghao Wu. Vmhunt: A verifiable approach to partially-virtualized binary code simplification. In *Proc. of CCS*, 2018.

[77] Zhaoyan Xu, Antonio Nappa, Robert Baykov, Guangliang Yang, Juan Caballero, and Guofei Gu. Autoprobe: Towards automatic active malicious server probing using dynamic binary analysis. In *Proc. of CCS*, 2014.

[78] Shams Zawoad, Amit Kumar Dutta, Alan Sprague, Ragib Hasan, Jason Britt, and Gary Warner. Phish-net: investigating phish clusters using drop email addresses. In *Proc. of eCRS*, 2013.

[79] Yue Zhang, Jason I Hong, and Lorrie F Cranor. Cantina: a content-based approach to detecting phishing web sites. In *Proc. of WWW*, 2007.

[80] Chaoshun Zuo and Zhiqiang Lin. Smartgen: Exposing server urls of mobile apps with selective symbolic execution. In *Proc. of WWW*, 2017.