Leaky Kits: The Increased Risk of Data Exposure from Phishing Kits

Bhaskar Tejaswi, Nayanamana Samarasinghe, Sajjad Pourali, Mohammad Mannan, Amr Youssef
Concordia University
Montreal, Canada
{b_tejasw,n_samara,s_poural,mmannan,youssef}@ciise.concordia.ca

Abstract—**Phishing kits allow adversaries with little or no technical experience to launch phishing websites in a short time. Past research has found such phishing kits that contain backdoors (e.g., obfuscated email addresses), which are intentionally added by the kit developers to obtain the phished data. In this work, we augment on prior research by exploring several ways in which security flaws in phishing kits make the victim data accessible to a wider set of adversaries beyond the kit deployers and kit developers. We implement an automated framework for kit collection and analysis, which includes a custom command-line PHP execution tool (for dynamic analysis) along with other open-source tools. Our analysis focuses on finding backdoors (e.g., obfuscated email address, command injection), measuring the extent of disclosure of sensitive information (e.g., via exposed plaintext files, hardcoded Telegram bot tokens, hardcoded admin console passwords) and detecting security vulnerabilities in phishing kits. We analyze 4238 distinct phishing kits (from a set of 26,281 compressed files collected from several sources over a span of 15 months), each having unique SHA-1 hash value. We found that 3.9% of the analyzed kits contained at least one form of backdoor. We also found hardcoded admin console passwords and API keys used to access third party services, in 8.3% and 16% of the analyzed kits, respectively. In addition, 15.8% of the analyzed kits wrote stolen information (PII) of users in plaintext files; 5.6% kits did not restrict external access to these plaintext files, leading to exposure of sensitive phished data (e.g., 178,504 passwords, 133,248 email addresses, 1253 credit card numbers). Furthermore, 11.7% of the analyzed kits contained hardcoded Telegram bots; we obtained invite links to join Telegram chats in 0.5% kits, and found them to expose chat messages containing sensitive PII information of victims (e.g., 73,342 passwords, 141,095 email addresses, 3584 credit card numbers). We also found that 64% of the kits are affected by security vulnerabilities (e.g., insecure file operations, SQL injection), which can be abused to further expose user data. We have open-sourced our framework and other artifacts to benefit future research.**

Index Terms—**Phishing websites, phishing kits**

I. Introduction

Phishing is a common technique used by adversaries to collect confidential information of users by mimicking legitimate websites. Phishing attacks are on the rise, and as per APWG's latest report [1], the second quarter of 2022 was the worst quarter for phishing that APWG has ever observed, with 27.6% attacks targeted at the financial sector (e.g., banks). Since phishing websites can be blocked or added to blocklists within a short time, cybercriminals have to generate these websites *quickly* and at *large scale*. This is achieved by the use of phishing kits that come with ready-made templates/scripts, enabling even non-tech-savvy adversaries (i.e., kit deployers) to launch phishing websites with minimal effort [2]. Such kits are sold online, or freely available through various distribution channels (e.g., darknet forums, Telegram channels).

Depending on how the kit is developed, the kit deployers obtain the stolen data (including sensitive information) entered by users (i.e., victims) of phishing websites via different means, e.g., email, plaintext files stored on the server, Telegram chat messages. Past work [3] has shown that the kit developers hide their email addresses in the kits and obtain copies of the phished data. Researchers have also found phishing kits that contain command injection backdoors [4], intentionally added by the kit developers to obtain shell access on web servers running the kits. However, such backdoors can be abused by any threat actor who is able to access the vulnerable URL endpoints. Apart from such intentionally added backdoors, the phishing kits may also contain several other security flaws. For instance, security vulnerabilities such as insecure file operations and SQL injection arise due to insecure coding practices (such as lack of input validation) adopted by kit developers. Any adversary can exploit these vulnerabilities to access the phished data. Phishing kits may also contain several hardcoded secrets (e.g., admin console passwords). If the kit deployers do not change the admin console passwords while setting up the kit, anyone with possession of the kit can log into the kit's admin panel and steal the victim data. Also, for the kits storing sensitive information in plaintext files, if the kit deployers do not restrict external access to those files, any adversary with the knowledge of the URL paths of such files can access the phished data. In effect, phishing kits used by (inexperienced) phishing deployers could result in the exposure of victim data to a much wider set of threat actors, as a result of added backdoors and other security flaws.

Apart from emails and plaintext files, Telegram bots are also used for data exfiltration in phishing kits. For example, a commercial phishing kit (*16Shop*) targeted Apple users by sending their sensitive information (including financial data) through a Telegram channel [5]. The authentication token of such a Telegram bot is typically hardcoded in the kit's source code. This enables anyone in possession of the kit to misuse this token to access the chat messages containing phished data. A large scale evaluation to measure the extent of data exposure resulting from such an abuse has not been done yet. Also, while past work [6]–[8] has demonstrated specific types of vulnerabilities in phishing kits and hosting infrastructure of corresponding phishing websites (e.g., [8] focused on file upload vulnerability), a comprehensive study of potential security vulnerabilities in phishing kits has so far not been conducted. In all, there is a lack of an automated evaluation framework, that presents an extensive analysis of phishing kits, with an aim of finding security flaws that may further dupe the victims of phishing attacks by exposing their data to a much wider set of threat actors.

In this work, we implement an automated framework for phishing kit collection and analysis, that achieves the following key goals: *(a)* finding backdoors in phishing kits (e.g., command injection, obfuscated email addresses); *(b)* measuring the extent of sensitive information exposure (via unrestricted plaintext files, hardcoded Telegram bots, hardcoded admin console passwords); and *(c)* analyzing security vulnerabilities in phishing kits. For collecting phishing kits, we automated the extraction of phishing website URLs from several sources (*PhishTank* [9], *OpenPhish* [10] and *ECX* [11]), and look for phishing kits hosted on these phishing sites, where the kit deployers have not removed or restricted access to the phishing kits. We also gather kits from other sources including GitHub repositories (*Phishing Kit Tracker* [12], *phishunt.io* [13]) and a public Telegram [14] channel. From these sources, we collect a total of 26,281 compressed files over a period of 15 months. We disregard files that are corrupted and those that do not contain any PHP/HTML source code files. This results in a total of 19,445 phishing kits, and after removing duplicates (files having the same SHA-1 hash value), our final dataset contains 4238 distinct phishing kits. We perform both static and dynamic analysis of the kits. To perform dynamic analysis, we implement a novel technique (using a custom PHP command line script) without having to host the respective kits on a web server (unlike past work). We analyze each kit for security flaws, and measure the data exposed as a result of such flaws.

Contributions and notable findings

1) We develop an automated framework for kit collection and evaluation, consisting of open-source tools along with an efficient custom command line tool developed to perform dynamic analysis of phishing kits. Using our framework, we identify possible channels through which confidential information can be exposed, e.g., obfuscated email addresses in source files of phishing kits, hardcoded authentication tokens of Telegram bots in phishing kits, unrestricted plaintext files on the server where the kit is hosted. In addition, we identify the threat actors (e.g., Telegram chat creators) behind the phishing kits.
2) We use our framework to perform a large scale evaluation of 4238 kits with distinct SHA-1 hash values, analyzing the ways in which victim data is exposed to multiple adversaries, and measuring the extent of the resultant data exposure.
3) 168/4238 (3.9%) phishing kits contained backdoors — obfuscated email addresses in 98/4238 (2.3%) kits, hidden Telegram bots in 12/4238 (0.2%) kits, and command injection in 62/4238 (1.5%) kits.
4) 670/4328 distinct kits deployed on 925 live phishing websites could create plaintext files on the web servers; on 297/925 (32.1%) live websites deployed using 239/670 (30.6%) kits, these plaintext files were found unrestricted on the web servers, thus exposing victim data, including: credit card numbers (1253), social security numbers (576), phone numbers (170), email address (133,248), and passwords (178,504).
5) We found 431 distinct hardcoded *Telegram bots* in 496/4328 (11.7%) phishing kits. 321/431 bots were found to be active. 19/321 Telegram bots returned invite links to chats containing sensitive information in chat messages — credit card numbers (3584), social security numbers (51), phone numbers (107), email address (141,095), and passwords (73,342).
6) We found 5227 hardcoded (non-obfuscated) and 50 obfuscated email addresses in 3441 and 98 distinct phishing kits, respectively; 1328/5227 (25.4%) hardcoded (non-obfuscated) email addresses and 18/50 (36%) obfuscated email addresses were used in two or more distinct phishing kits. The obfuscated email addresses are more likely backdoors added by kit developers.
7) 401 and 694 phishing kits had hardcoded passwords and API keys (used to access third party services), respectively. From 401 distinct kits, we obtained 275 unique passwords, which include 210 admin console passwords (used in 354 distinct kits).
8) 2698/4238 (64%) phishing kits have security vulnerabilities that exacerbate the security posture of the corresponding phishing websites, and expose stolen data. Such vulnerabilities include: command injection (62), insecure file operations (1590), and SQL injection (100).

To help future research, we have open-sourced our framework, collected phishing kits, and threat actor email addresses extracted from the kits. The repository can be accessed via the following URL: https://github.com/btcodes101/PhishingKitAnalysis. As a part of responsible disclosure, we have shared the details of all the Telegram bots and the corresponding chats found from our analysis with Telegram.

II. Related work

In this section, we discuss past work related to the collection of phishing kits, use of drop email addresses to expose confidential information, and vulnerabilities in phishing kits and its hosting infrastructure.

A. Identification and collection of phishing kits

Bijmans et al. [15] identified 70 distinct phishing kits by fingerprinting the kits based on their unique properties. The fingerprints created by them were used to identify 10 different families of phishing kits (e.g., uAdmin, tikkie, bonken, ics, livepanel). They used different properties to derive the fingerprint of the corresponding phishing kits — e.g., file name, full file path (from the root of the website), uncommon strings found on the main page of the phishing website. Kondracki et al. [16] implemented a fingerprinting tool to automate the discovery and analysis of three MITM phishing toolkits on the web. Using this tool, they discovered 1220 phishing websites (mimicking popular brands such as Google, Yahoo, Twitter and Facebook) that used MITM phishing toolkits. They also found that 56.3% of phishing websites created from MITM phishing toolkits are not detected by blocklists. Unlike ready to deploy kits, some of these phishing toolkits use command line tools to create phishing websites. Merlo et al. [17] obtained 20,871 phishing kits collected from forensic teams of private security firms over a period of three years. They implemented a tool for similarity analysis of source code of phishing kits, that can identify kits that are of a near copy of an already identified phishing kit. They found 90% of the analyzed phishing kits share 90% or more of their source code with other kits. However, their tool's source code and kit dataset is not released. In contrast, we build our dataset of phishing kits from publicly available resources, and our analysis framework would be open-sourced. Oest et al. [18] studied the *.htaccess* filtering techniques from 1794 live phishing kits (on 933 unique domains).

We used *Phishfinder* [19] to extract phishing kits (mostly built using PHP server side scripting language) hosted on phishing URLs, using feeds from *PhishTank* [9], *OpenPhish* [10] and APWG ecrime exchange (ECX) [11]. We also collected phishing kits uploaded to GitHub repositories such as *Phishing Kit Tracker* [12], *phishunt.io* [13].

B. Use of drop email address to send confidential information from phishing kits to its authors

Past work studied the use of phishing kits that sent confidential information to the developers of such kits. Cova et al. [20] analyzed 503 phishing kits obtained from live phishing websites and other channels (e.g., underground IRC channels, web forums). These phishing kits use different techniques to relay copies of phished data to the kit developers, by exploiting backdoors that are inserted to the phishing kit's source code (with obfuscation). In order to find backdoors, the developers run an automated analysis (e.g., hosting the kit locally and automating the simulation of form filling with Selenium [21]) to gather email addresses from emails triggered by the kit. Then, all email addresses that are hardcoded (in the kit) are removed. The remaining email addresses in the kit are those that are obfuscated. Thereafter, a manual analysis is performed to find obfuscation techniques used by the phishing kit developers. Lazar [22] examined 1019 phishing kits, sourced from TechHelpList (now inactive) and OpenPhish, and found 25% of those kits contained hidden email addresses. In contrast, in our study of 4238 kits obtained from various sources, we found hidden email addresses in a comparatively smaller set (2.3%) of kits. Zawoad et al. [23] proposed a clustering algorithm to identify related phishing websites created by common phishing kit developers (based on drop email addresses). They created clusters of kit developers and kit users, and used those clusters to determine most pervasive kit developers and kit deployers. Also, they identified the most dominant phishing campaigns (in terms of number of corresponding phishing websites, time span that phishing websites are alive), and the most active kit developers and kit users behind different clusters of phishing websites. We identify hardcoded information (admin console passwords, Telegram bot authentication tokens) in phishing kits that can be abused to obtain sensitive information stolen through the kits. We also measure the extent of sensitive information exposure corresponding to kits collected from live phishing sites, where the plaintext files having phished data are left unrestricted on the web servers.

C. Vulnerabilities in phishing kits and in the underlying deployment infrastructure

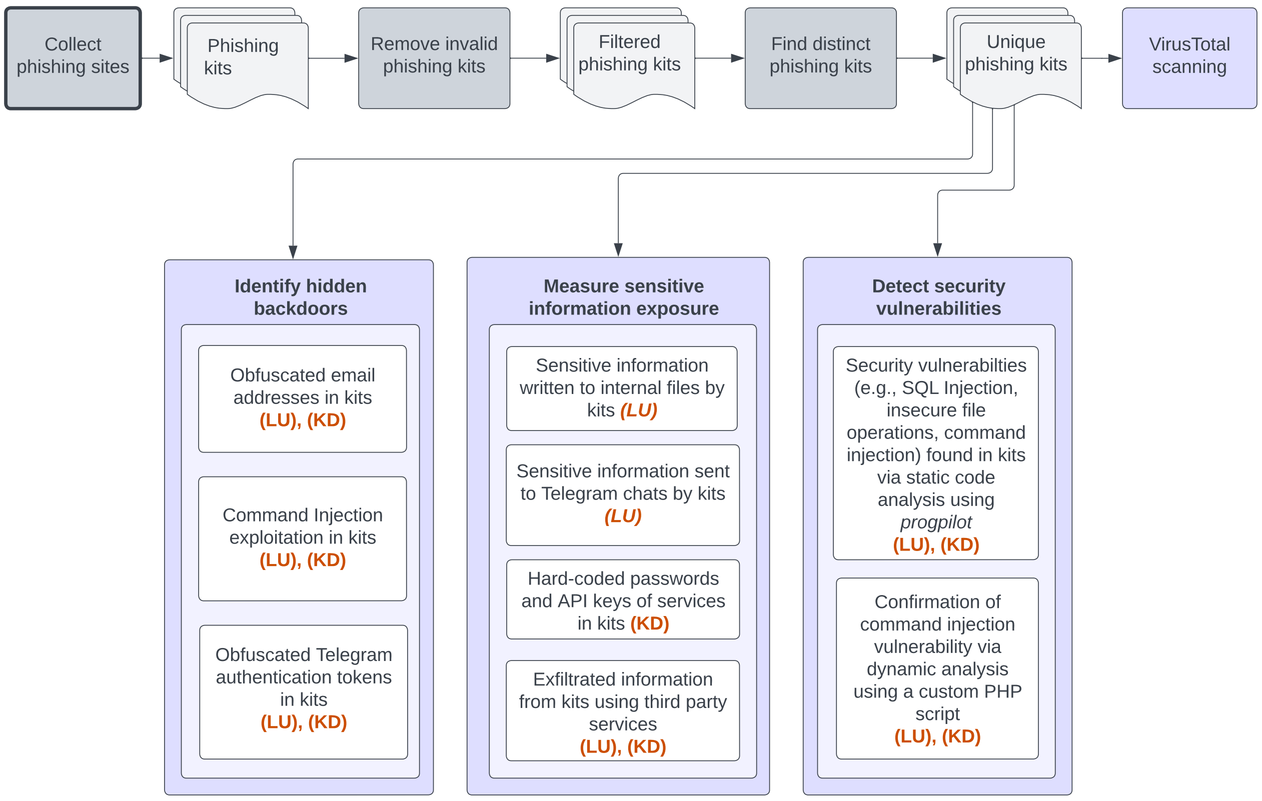
Han et al. [6] studied 643 distinct kits through a honeypot server (on Amazon EC2). The authors configured 18 vulnerable PHP files that allow attackers to upload files and execute shell commands. In addition, 98 of the analyzed kits contained code that communicates with remote machines under the control of phishers — to fetch information (e.g., blocklists) from the remote machines, and backdoors used to exfiltrate stolen information. Cashdollar [8], [24] analyzed phishing kits and observed file upload vulnerabilities using web shells that are uploaded to web servers, exposing those servers to further attacks. Wright [4] studied 3200 distinct phishing kits obtained from PhishTank [9] and OpenPhish [10], and identified PHP backdoor shells in 200 phishing kits; multiple phishing kits were reused on 30 different hosts. The author also analyzed common strings in phishing kit URLs, and found the corresponding phishing websites (hosting the kits) that are more likely to be exploited, are hosted on Wordpress content development platforms. We found 396 (12.8%, out of 3013) URLs hosting phishing kits were running from domains pertaining to Wordpress installations. Oest et al. [7] determined that phishing URLs deployed using phishing kits, are more likely to be on compromised infrastructure (if the phishing URLs are unintelligible, or contain random domains with deceptive path contents). We study 4238 distinct phishing kits and identify various vulnerabilities — e.g., SQL injection, command injection vulnerabilities.

III. Methodology

There are three goals of our analysis – to find backdoors in phishing kits, to measure the sensitive information exposed through phishing kits, and to find security vulnerabilities in phishing kits; see Figure 1 for an overview of our methodology. In this section, we discuss the technical details of our methodology for automated kit collection and analysis.

A. Phishing kit collection

We collect phishing URLs from *PhishTank* [9], *OpenPhish* [10] and *ECX* [11]. Then, for each of the live phishing URLs, we use the Phishfinder's [19] phishing kit collection



Collect phishing sites
Phishing kits
Remove invalid phishing kits
Filtered phishing kits
Find distinct phishing kits
Unique phishing kits
VirusTotal scanning
Identify hidden backdoors
Obfuscated email addresses in kits (LU), (KD)
Command Injection exploitation in kits (LU), (KD)
Obfuscated Telegram authentication tokens in kits (LU), (KD)
Measure sensitive information exposure
Sensitive information written to internal files by kits (LU)
Sensitive information sent to Telegram chats by kits (LU)
Hard-coded passwords and API keys of services in kits (KD)
Exfiltrated information from kits using third party services (LU), (KD)
Detect security vulnerabilities
Security vulnerabilties (e.g., SQL Injection, insecure file operations, command injection) found in kits via static code analysis using progpilot (LU), (KD)
Confirmation of command injection vulnerability via dynamic analysis using a custom PHP script (LU), (KD)

Fig. 1. Overall methodology; includes collecting phishing kits, filtering valid phishing kits, security analysis of phishing kits, and scanning the phishing kits with VirusTotal — legitimate users and kit deployers who are impacted by security issues in phishing kits are represented as **(LU)** and **(KD)**, respectively.

tool to download kits from live phishing websites by appending the compressed file extension[1] (i.e., zip, rar, 7z, tar) to each path segment of the phishing URL. We also gather kits from *phishunt.io* [13] Github repository, which stores kits obtained from live phishing websites, and provides the kits as well as the corresponding phishing URLs. We also obtain a set of phishing kits from *Phishing Kit Tracker* [12] Github repository. In contrast to *phishunt.io*, *Phishing Kit Tracker* only provides the phishing URL's domain instead of the complete URL. In addition, we collect distinct phishing kits from a public Telegram [14] channel. From all these sources, we collect a total of 4238 distinct phishing kits — see Table I.

B. Validation of collected kits

We first check if the phishing kits collected are packaged in a valid compressed file, by verifying whether the content of the compressed archive is extractable. If not, we record the file as faulty, and proceed to the next compressed file. For each phishing kit, we calculate the SHA-1 hash value, size (in bytes) and a list of extensions for files contained inside the compressed file. In the set of collected compressed files that are likely phishing kits, we find some of them do not pertain to phishing kits (e.g., zipped files containing only image and text files). We find valid compressed files that include PHP/HTML files are more likely phishing kits — a similar observation is made in [20]. Therefore, we filter out those compressed files that do not contain any PHP/HTML file.

[1]The tool only supports .zip extension. We modified the code slightly to support .rar, .7z and .tar extensions.

C. Scanning phishing kits with VirusTotal

We scan each distinct kit with VirusTotal [25] and check if the kit has been labeled as malicious. We also collect the suggested threat label (phishing, trojan) for each distinct kit. This classification is provided by VirusTotal on the basis of the scan results of different anti-virus engines [26].

D. Identifying backdoors

In this section, we discuss our methodology to find different types of backdoors in phishing kits.

Obfuscated email addresses in phishing kits. Phishing kit developers use source code obfuscation to exfiltrate stolen data to their email addresses. We dynamically analyze the source code of phishing kits, to find obfuscated email addresses included in phishing kits source files, that are used to siphon phished information to kit developers. Past studies [20] hosted phishing kits on isolated web servers, and automated the interaction with those kits to dynamically analyze them.

Source	# Total kits	# Duplicated kits	# Distinct faulty kits	# Distinct non-phishing kits	# Distinct kits
Live phishing websites	21776	19527	83	31	2249
Phishing Kit Tracker	4497	1682	111	408	2815
Telegram Channel	7	-	-	-	7

TABLE I
PHISHING KITS OBTAINED FROM DIFFERENT SOURCES — DISTINCT KITS ARE DETERMINED USING THE SHA-1 HASH VALUE. LIVE PHISHING SITES ARE FROM PHISHTANK [9], OPENPHISH [10], ECX [11] AND PHISHUNT [13]. A FAULTY KIT'S CONTENT IS NOT EXTRACTABLE FROM THE COMPRESSED FILE. NON-PHISHING KITS ARE THOSE WHICH DO NOT HAVE ANY PHP/HTML FILE. DISTINCT = TOTAL - DUPLICATED.

In contrast, we develop a custom PHP script (executed via command line) that converts all the PHP files within a phishing kit to output PHP files that can be executed on command line (so that we do not have to host the PHP files on a web server to dynamically analyze them). Our custom script dynamically analyzes phishing kits, and transforms server-side PHP code in PHP files included in it, to a version that can be executed via command line. The custom script converts the PHP code to a data structure that is in the form of an Abstract Syntax Tree (AST) [27] using *PHP-Parser* [28]. This technique allows to better process minified and obfuscated code, than simply applying regular expressions on such code (e.g., iterations using *for* loops in minified code can be better processed using AST data structures). The automated script performs the following to transform each PHP file to a version that can be run on command line (the transformed command line script also outputs the usage along with corresponding inputs that are passed as environment variables): (i) Recursively merge the source code of the main PHP page with that of multiple included PHP files (using *include* and *require* functions), where the end outcome is a single PHP file; (ii) Change server variables (e.g., client IP address) in the source code of the PHP file to hardcoded values; (iii) Change the inputs provided by *$_POST*, *$_GET* and *$_REQUEST* super global variables in the PHP source code to environment variables (i.e., *$_ENV*); (iv) Inject *file_put_contents* function to save emails and text files created from PHP source code. In addition, *PHPDeobfuscator* [29] is used to detect the possible obfuscated super global variables before the analysis/transformation.

Whenever our custom script finds the *file_put_contents* PHP function (in the source code of the PHP files) for the creation of a text file, the names of text files created are saved into a separate file. To find obfuscated email addresses in phishing kits, we configure the default *mail* function in PHP, to store the output in a file (*mail.out*) by editing the *sendmail_path* attribute in the *php.ini* configuration file. We extracted the obfuscated email addresses from phishing kits using the following steps: (*a*) Our analysis script reads all the PHP files in the kit, line by line, and gathers all the hardcoded email addresses from the PHP code. Then, we use *python-email-validator* [30] to filter out invalid email addresses; (*b*) We use our custom PHP script to execute each PHP file. The recipient email addresses are extracted from the triggered emails; (*c*) The difference between the list of email addresses obtained from the dynamic analysis (in step *b*), and the list of email addresses obtained by parsing the source code of PHP files (in step *a*), gives the list of obfuscated email addresses. For both hardcoded and obfuscated email addresses, we check if the email addresses have been disclosed via online paste services [31] using the *haveibeenpwned* (HIBP) API [32], and record the disclosed email addresses.

Identifying phishing kit clusters based on phishing kit developers. We extract 1780 (out of 4328) distinct phishing kits with email addresses (from dynamic analysis), to identify different categories of phishing clusters that are carried out using those phishing kits. These phishing kits include email addresses that pertain to kit developers. For this purpose, we collect data relating to the domain name of phishing site, name of phishing kit hosted on the phishing site, file extensions of source files (e.g., PHP, HTML) in the compressed phishing kit and the email addresses of kit developers. Then, we encode[2] and normalize [33] the values of data fields, to re-scale individual samples of the collected data. The normalized input data is applied to an unsupervised machine learning algorithm (hierarchical Agglomerative Clustering [34] technique), with appropriate parameter values (i.e., *euclidean* as the affinity and *ward* as the linkage), to cluster the phishing kit data based on kit developers.

Command injection. Adversaries can use command injection by running system commands on servers that host phishing websites, deployed via phishing kits, by exploiting the vulnerabilities in corresponding phishing sites.[3] The commands supplied through the phishing web application are executed with the privileges granted to the web application on the host operating system. Given the scope of our work, which focuses on the analysis of the kits and not the hosting servers, we only consider backdoors in phishing kits that leverage seemingly intentional command injection. We used *progpilot* [35] to examine the phishing kit source code to identify possible backdoors created using command injection. To confirm command injection vulnerability via dynamic analysis, we do the following steps — *(a)* we record the output of *id* command on our analysis VM; *(b)* we execute each PHP file in the phishing kit by providing *id* in parameter values, and record the output; *(c)* we check if the output of *(b)* contains the value recorded in *(a)*. With backdoors created by command injection, potential sensitive information can be exposed to adversaries.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

[3]https://owasp.org/www-community/attacks/Command_Injection

E. Sensitive information exposure via phishing kits

In this section, we discuss several means of sensitive information exposure through phishing kits.

Telegram Bots. Telegram [14] is a popular messaging application. Apart from the typical features such as sending and receiving text and multimedia messages, Telegram also provides several additional features — e.g., the ability to create third-party applications within Telegram, also known as bots. Past work [36], [37] reported the use of Telegram bots in phishing kits. We extend past work by obtaining more information on how Telegram bots are used in phishing kits (e.g., leveraging information available relating to Telegram chats/chat admins), and measuring the sensitive information exposed through these bots. Telegram offers APIs[4] for interacting with these bots. Each Telegram bot is given an authentication token at the time it is created. All queries to the Telegram APIs must be accompanied by the authentication token passed into the URL. Firstly, a bot needs to be created and added as a member to a Telegram chat (private[5], group[6], supergroup[7] or channel[8]). Subsequently, the phished data is submitted to the Telegram chat using the *sendMessage* bot API — i.e., *https://api.Telegram.org/⟨token⟩/sendMessage?chat_id=⟨chat_id⟩&text=⟨phished_data⟩*. The phished data can be viewed by the members of the chat. We analyze the phishing kits in our dataset and obtain information about these bots. The authentication tokens of Telegram bots are typically hardcoded in the phishing kit source code. Our analysis script reads the phishing kit source code and extracts all the lines containing *https://api.telegram.org*. We extract the Telegram bot tokens and chat identifiers from these lines via regular expression[9]. Thereafter, we gather further information (e.g., invite links to join Telegram channels/groups, messages exchanged in chats) about the extracted Telegram bots using the following Telegram APIs [38].

- **getMe**: Confirms if a given bot token is valid, in which case, it returns basic information about the bot — e.g., name, ID, privileges assigned to the bot (such as ability to join groups and read messages).
- **getMyCommands**: Returns the commands[10] supported by the bot.
- **getWebhookInfo**: Obtains webhooks configured in Telegram bots to relay the phished data. We check if these webhook URLs have been flagged as malicious using VirusTotal.
- **getUpdates**: Returns the information sent to the bot from the deployed phishing kit, during the past 24 hours, for the bots that do not have their webhooks configured.
- **getChat**: Confirms if a given chat ID is valid, in which case, this API returns information about the chat — e.g., chat name, ID, type (private, group, supergroup, channel), pinned messages and invite link.
- **getChatMemberCount**: Returns the number of members of a given chat.
- **getChatAdministrators**: Returns information about the admin users of a given chat (e.g., username, user ID).
- **createChatInviteLink**: Creates an invite link for the chat where the bot has the required permission. We create a test account on Telegram, join the chats through the invite links, and retrieve historical chat messages (possible in the case of channels) by using the export chat history functionality in Telegram's desktop application. We measure the extent of information exposure from the messages retrieved from the channels and responses from the *getUpdates* API using *PDSCAN* [39]. *PDSCAN* is a command line tool to scan files for unencrypted personal data (PII) — e.g., email addresses, phone numbers[11], credit card numbers. However, it does not detect passwords. We manually analyzed a few kits on a local setup to understand the possible formats in which passwords are sent from the kits (e.g., pass: 1234, Password: abcde, passe : 12345). Thereafter, we formed regular expressions to count the number of passwords. Thereafter, corresponding files are securely deleted using the *shred* [40] utility.

Other hardcoded secrets in phishing kits. We use the *Whispers* [41] tool to analyze phishing kit source code, to detect hardcoded sensitive information (if any). Apart from Telegram bot tokens, the kits contain several other hardcoded secrets such as API keys (for third party services), and passwords. Among the hardcoded passwords, we focus on the passwords used for admin consoles of phishing kits. Anyone having the admin console password could log into the admin console on a live phishing kit deployment and access the stolen data. In order to obtain admin panel passwords, we first identify passwords used for internal services on the basis of their corresponding variable names such as SMTP passwords (variable names such as *smtp_pass*, *SMTPPass*, *SMTP_PASSWD*, *smtp_password*) and database passwords (variable names such as *db_pass*, *DB_PASSWORD*, *dbpass*, *dbPassword*, *config_dbpasswd*). After removing the passwords used for internal services, we are left with the set of admin console passwords. We analyze the strength of these admin console passwords using *zxcvbn* [42], and also if these passwords have been exposed in a prior breach using *haveibeenpwned* (HIBP) [32] API.

Sensitive information written to files from phishing kits. Several phishing kits use plaintext files to store information stolen from victims. The default server configuration renders these files accessible to anyone without any restriction. There are two situations in which such files can be obtained: (i) If

[4]https://core.Telegram.org/bots/api

[5]Private chat refers to a direct chat between two users or between a user and a bot.

[6]Groups refers to a shared chat that can have up to 200 members.

[7]Supergroup refers to a shared chat that can have up to 200,000 members.

[8]Channels can have an unlimited amount of subscribers, and only admins have the right to post messages.

[9]Regular expression used for bot token — bot\d+:AA[a-zA-Z0-9-_]{20,60}, for chat identifier — chat_id=\s*[-@a-zA-Z0-9]*

[10]https://core.Telegram.org/api/bots/commands

[11]The tool supports detection of US phone numbers.

the phishing kit exists and access to it is not restricted, the exact location of the text files (typically at the same web app path or in a sub path) on the web server can be found and the corresponding plaintext files can be downloaded; (ii) If the phishing kit is removed from the server, but the directory containing sensitive plaintext files suffers from open directory listing, unrestricted files containing stolen data could still be downloaded. Our work focuses on analyzing the extent of exposure of stolen information from these plaintext files. For this purpose, we gather the corresponding files hosted on the live phishing websites, where the kit deployers have not restricted access to those files. Then we use *PDSCAN* [39] utility on the files extracted from the phishing websites to detect sensitive information (such as email addresses, phone numbers, credit card numbers). and use regular expressions to detect passwords. For ethical considerations, we measure the data exposure and delete the files using the *shred* [40] utility.

Exfiltration of information from phishing kits using third party services. Third party services (e.g., *Discord* [43], form submission services, SMS providers) are also used for data exfiltration from phishing kits. The kits that include these third party services are identified by applying the *Indicator Of Kit (IOK)* [44] rules to the kit source code, along with manual verification. Use of third party services increases the extent of exposure of stolen data.

F. Vulnerabilities in phishing kits

Phishing kits may contain vulnerabilities in their source code, which may be exploited to obtain victim data. For example, command injection allows an adversary to execute arbitrary commands on the web server's operating system. This would allow an attacker to access the web server and extract the content of the application source files, along with any information stolen through the phishing kit. With *progpilot* [35], we inspect the phishing kit's source code to identify security vulnerabilities (e.g., SQL injection, insecure file operations).

IV. Results

In this section, we summarize the main findings of our analysis of the collected phishing kits. The results presented here are based on experiments performed from April 8, 2021 up to July 15, 2022.

A. Exposure of sensitive information

In this section, we mention our findings on the exposure of sensitive information from phishing kits.

Telegram bots. We extracted 431 unique authentication tokens and 439 chat identifiers from Telegram bots in 496 distinct phishing kits. We extracted additional information about these bots using the Telegram APIs. We were able to get valid responses for 321 (74.5%, out of 431) of the Telegram bot tokens using the *getMe* Telegram API. The remaining 110 bot tokens resulted in an error message, which indicates that the authentication tokens had been changed for those bots. We found 3 Telegram bots provided commands for tasks relating to adding users. We were also able to get a valid response for 302 distinct chat IDs using the *getChat* Telegram API. The error responses for 34 chat IDs indicated that the corresponding chats had been deleted. One of the bots had been removed from the chat. These chat IDs belonged to 13 channels, 74 groups, 206 private chats and 8 supergroups. The *GetChatAdministrators*[12] API returned valid responses for 71 unique chat IDs, which revealed the identity of 57 chat creators and 3 chat administrators. In addition, one particular user was found to be the creator of a particular chat and also an administrator of another chat. We found a Telegram user who was a creator of 5 chats, along with 8 other Telegram users, who were creators of 2 chats each.

Using the *getChatMemberCount* API, we found that a majority of chat IDs (234 out of 302, 76.8%) of chats had two or less members; if only 2 members are in a chat, they are most likely the Telegram bot and the phisher — see Figure 2. We were able to obtain the invite links to join 11 channels, 5 groups and 4 supergroups. While the groups appeared to be dormant, the channels were actively used. We performed an automated analysis of all the chat messages obtained through the channels using PDSCAN [39] and regular expressions (for passwords), and found that these chats exposed sensitive information — e.g., credit card numbers (3584), social security numbers (51), phone numbers (107), email addresses (141,095) and passwords (73,342). The chats also contained several IP addresses (269,142). These are the IP addresses of the visitors of the phishing websites, that are recorded possibly to form blocklists for automated crawlers.

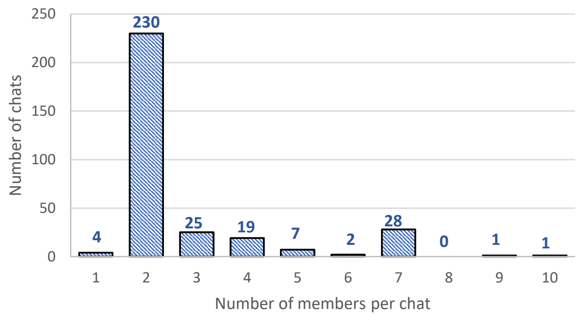


Fig. 2. Distribution of member statistics in Telegram chats used in phishing kits.

We also extracted the recent messages exchanged in chats used for phishing activities using the *getUpdates* API. Towards the end of our analysis, we called the *getUpdates* API for each of the collected bot token, which returned non-empty results for 46 different bots. From this analysis, we found different types of sensitive information — e.g., credit card numbers (75), email addresses (19), IP addresses (101). Furthermore, we obtained 3 webhook URLs configured for 38 bots. Two of the webhook URLs were flagged as malicious by at least one security vendor in *VirusTotal*. For those kits where the

[12]Note that this API does not return any user details for private chats. However, the API returns creator/admin user details for channels, groups and supergroups.

regular expressions returned invalid values for tokens/chat identifiers, we manually inspected the corresponding kits' source code. While doing so, we found 7 Telegram bots that were obfuscated and most likely to be added as backdoors in 12 phishing kits. The following techniques were used to hide these backdoors: *(a)* Telegram bot token was hidden in a GIF file, and the GIF file was included in the PHP source code; *(b)* Telegram bot link was included as an image source; *(c)* Telegram bot token was hidden using a combination of *base64_decode* and *strrev* functions along with PHP arrays.

PII disclosures from unrestricted plaintext files. Our custom PHP execution script, that was used to dynamically analyze phishing kits, identified the creation of plaintext files by 1344 distinct phishing kits; 670 of these kits pertained to live phishing websites. For 239 (35.7%, out of 670) distinct kits hosted on 297 live phishing sites, we could download the plaintext files. This indicated that the access to the plaintext files was not explicitly restricted on the web servers, providing unrestricted access to the phished data. Any adversary with the knowledge of the storage paths could access the information stored in the exposed plaintext files. We determined the count of the exposed sensitive information from these plaintext files using the *PDSCAN* [39] utility and regular expressions (for passwords) — credit card numbers (1253), email addresses (133,248), passwords (178,504), social security numbers (576) and phone numbers (170). Also, we found 17,595,190 IP addresses of victims/crawlers visiting the phishing websites. Improper access controls on the server side expose these files, rendering the corresponding stolen data accessible to multiple adversaries.

Common file names. From the dynamic analysis, we observed the creation of files in 1344 distinct phishing kits; 49.8% (670/1344) of these phishing kits were used in live phishing sites. We observed common names being used across these phishing kits for the plaintext files containing sensitive information — e.g., *login.txt*, *logs.txt*, *banks.txt* contained harvested information (e.g., login credentials, credit card numbers, phone numbers); *antibots.txt*, *whitelist.dat*, *blacklist.dat*, *visitors.txt* contained the IP addresses of visiting users. An adversary could utilize a list of such common file names and attempt to directly access unrestricted plaintext files on the phishing websites. However, unless the attackers have the kits used to deploy the phishing websites, they would also have to guess the URL paths where the plaintext files are located.

Third party services to exfiltrate information. Apart from known exfiltration methods e.g., email, storage in plaintext files and Telegram bots, we also found other exfiltration methods that involve the usage of third party services in a few phishing kits. We observed 3 kits in our dataset used *Discord* webhooks [45] for submitting stolen information to *Discord* channels. In one particular kit, we observed that *formsubmit.co*[13] is used as the sole method of data exfiltration. Using such forms of submission services, kit developers can create and deploy phishing kits, without setting up a backend server. Other exfiltration methods include SMS services, e.g., *textlocal* (8 kits), *Vonage* (19 kits) and *Twilio* (47 kits).

Open directory listings on phishing websites. While collecting phishing kits, we found open directory listings on 9575 corresponding phishing websites. From 356 (3.7%, out of 9575) phishing websites, we were able to retrieve plaintext files. Anyone who can locate the web app paths to open directory listings, will have access to the information stored in the plaintext files. In 262 (73.5%, out of 356) of these phishing websites, corresponding phishing kits had been removed. From our analysis, we found that most number of open directories exposing phishing kits hosted on phishing websites were from United States (6586 out of 9575, 68.8%).

B. Hardcoded secrets in phishing kit source code

In this section, we describe our findings on admin console passwords and API keys.

Admin console passwords. A phishing kit may provide the functionality for an admin user to view/manage the consolidated harvested information of a phishing site created using the kit (i.e., via the admin panel). The default password to log into these admin panels is often hardcoded in the phishing kits. With *Whispers* [41], after removing false positives, we obtained 275 unique passwords in 401 distinct phishing kits. After filtering the passwords for internal services, we obtained 210 unique admin console passwords that are hardcoded in 354 distinct kits. If a kit deployer uses the default hardcoded password for the admin console, an adversary can log into the admin console and steal victim data in following ways: *(a)* an attacker who is in the possession of the corresponding kit can obtain the hardcoded default password from the kit's source code; *(b)* an attacker who does not have the kit can launch a password brute force attack on the login page, which would likely succeed if the password is easily guessable. To find if the password brute force attacks like *(b)* could succeed, we calculated the password complexity score for each hardcoded admin console password using *zxcvbn* [42]. The passwords were easily guessable as the complexity score for 128 (out of 210, 60.9%) passwords were between 0 and 2. As reported from past breaches [32], 108 of these passwords (out of 210, 51.4%) were found to be reported in at least one password breach. Therefore, it is possible for potential adversaries to launch password guessing attacks on admin panels leveraging easily guessable and previously exposed passwords. Table II summarizes our analysis of admin console passwords' complexity and breach status.

API keys. We detected 622 API keys in 694 distinct phishing kits. These include API keys pertaining to various third party services — e.g., geolocation services (e.g., *ipinfo.io*[14], *ipstack.com*[15]), messaging services (e.g., Telegram, Twilio). These phishing kits also contained license keys of commercial phishing kits. Since API keys of these services are hardcoded

[13] *Formsubmit.co* is a form endpoint that is used to connect forms in web pages (without requiring to setup a backend service).

[14] https://ipinfo.io/

[15] https://ipstack.com/

Password complexity	# Distinct phishing kits	# Unique passwords	# Breached unique passwords
0	133	40	39
1	168	67	52
2	45	21	5
3	44	26	7
4	68	56	6

TABLE II
ANALYSIS OF ADMIN CONSOLE PASSWORDS FOR COMPLEXITY AND PAST BREACHES

in the kit's source code, anyone with access to these kits can extract these API keys. Third party services (e.g., *ipinfo*, *ipstack*) are used in phishing kits to identify visitor IP addresses to avoid detection (e.g., if the visitor is a security crawler). These third party services often allocate a usage quota depending on the subscription level (e.g., 50,000 API requests for a month are allocated for the free plan of *ipinfo*[16]). Overuse of API requests that use a specific hardcoded API key, by anyone who has access to these phishing kits, can disable the target kit's evasion mechanism. This will expose the corresponding phishing website to security crawlers/analysts.[17]

C. Security vulnerabilities in phishing kits

We analyzed phishing kits using *progploit* to find security vulnerabilities — see Table III for a summary of number of affected phishing kits/corresponding vulnerabilities, and Figure 3 for an overview of the number of vulnerabilities/number of affected phishing kits. 63.6% (2698/4238) phishing kits have at least one security vulnerability — kits with highest number of vulnerabilities include *wp-admin.zip* (10), *BOARDIRAN v10.5.zip*(9), *allenc.zip* (8), *upandruning.zip* (8), *alldo.zip* (8). From the context of corresponding phishing websites deployed by phishing kits, some of these vulnerabilities (e.g., SQL injection, insecure file operations, command injection) can be exploited by malicious actors to steal phished data. Among these, command injection vulnerability is often added as a backdoor, while others are unintentionally introduced in phishing kits due to insecure coding practices that are followed by kit developers. In this section, we present the results from our static source code analysis and dynamic analysis script.

Command injection. We found command injection vulnerability in 62 (1.5%, out of 4238) phishing kits. Command injection vulnerability can allow an attacker to supply system commands via web requests, which are then executed on the corresponding hosting server, with the system privilege granted to the corresponding phishing web application (deployed using the vulnerable kit) on the web server. From our analysis of the code, this vulnerability seems to be typically added intentionally by the phishing kit developers to maintain persistent access to the server hosting the phishing site and the phished data. We use dynamic analysis to determine the exploitability of the command injection vulnerability in phishing kits. We confirmed that command injection can be successfully exploited in 44 phishing kits, and after a manual review of the corresponding code segments in kits, we found that all of these were intentionally added as backdoors, and did not serve any other purpose in these kits. These 44 (70.9%, out of 62) kits that have the command injection backdoor contained *useragent* variable used to execute commands on the web server, that hosts these kits.

[16]https://ipinfo.io/developers#rate-limits
[17]Such an attack would not have an impact if the kit is using other evasion techniques in conjunction to using the third party APIs.

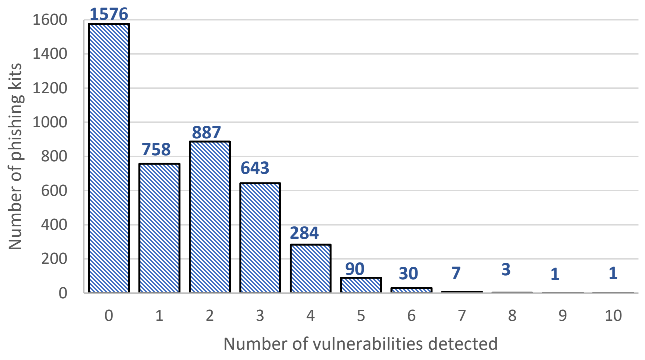


Fig. 3. Security vulnerabilities in phishing kits.

Security vulnerability name	# Phishing kits
Insecure File Operations	1590
Cross-Site Scripting	1417
File Disclosure	1060
Security Misconfiguration	744
Header Injection	682
Code Injection	517
SQL Injection	100
Command Injection	62
Session Fixation	46

TABLE III
SECURITY FLAWS/VULNERABILITIES IN PHISHING KITS

Insecure file operations. We found insecure file operations[18] in 1590 (37.5%, out of 4238) of the analyzed phishing kits. With insecure file operations, a phishing kit could let an attacker upload a PHP shell via the corresponding phishing website. Thereafter, the attacker would be able to execute system commands on the server hosting the phishing website; see [8] on a potential server take over, using a PHP shell and improperly secured scripts. An attacker could also abuse the file removal scripts in the phishing kits to remove arbitrary files on the hosting server via directory traversal.

SQL injection. We also observed SQL injection in 100 (2.3%, out of 4238) of the analyzed phishing kits. SQL injection is only relevant for those phishing kits that use a database for storage. The affected kit allows any visitor of the corresponding phishing website to submit database queries via form fields, that are executed on the database. Therefore, the specific database can be abused for malicious purposes — i.e., stealing and deleting (partially or entirely) phished data from the database.

[18]*Insecure file operations* is categorized as *idor* in progpilot scan results.

D. Characteristics of email addresses in phishing kits

We obtained two types of email addresses in phishing kits: obfuscated email addresses (dynamically extracted) and hardcoded (statically extracted).

Obfuscated email addresses. Phishing kit developers hide email addresses using various obfuscation techniques within the phishing kit's source code. These obfuscated email addresses are a backdoor to siphon phished data to the kit developers. We found 49 hidden email addresses (in 97 distinct phishing kits); see Table V. We manually examined the affected kits and observed the following techniques used to obfuscate email addresses: *(a)* phishing kit developers hide email addresses within the source code using PHP arrays. In this case, the *mail()* PHP function is written twice, to send the harvested phished information to the attacker deploying the kit, and to the kit developer (using the hidden email addresses); *(b)* email addresses are obfuscated by embedding them in GIF files and including the GIF file in the PHP source code; *(c)* obfuscated email addresses are included in a JavaScript file using *base64* encoding and arrays. We observed that these backdoors are hidden in PHP files (e.g., *antibots.php*, *bt.php*, *blocker.php*) contained in phishing kits that are meant to provide protection against crawling bots. Such a backdoor provides the kit developer a copy of phished data from all the live deployments of the phishing kit.

Phishing kit clustering based on kit developers. Using the hierarchical Agglomerative Clustering unsupervised machine learning algorithm, we clustered the data samples that contained the email addresses of kit developers, phishing domain, kit name and file extension names of kit content, into 8 clusters of phishing campaigns. We normalized the individual data samples to have unit norm [33]. Figure 4 shows the scatter plot of these 8 clusters. We observed clusters where the same phishing kit file, with different email addresses in each of the kits, are hosted on various phishing domains; e.g., In cluster 1, separate copies of *DHL.zip* phishing kit file, that contain *gang19goalz@gmail.com* and *info@yourcoolsite.com* email addresses, in each of the kits, are hosted on *lupusinspirations.com* and *support-security.paypal.com.komamen.com* phishing domains, respectively. We also observed, the same email address appeared in multiple phishing kits; e.g., in cluster 5, *wirez@googledocs.org* email address was included in *docu.zip* and *linkedin.com.zip* phishing kits.

Common Email Addresses. We obtained 5967 hardcoded email addresses from 3549 distinct kits, out of which 5227 email addresses (from 3441 distinct kits) were labeled as valid by *python-email-validator*. We found 50 obfuscated email addresses from 98 distinct kits, all of which were labeled as valid. For each type of email address, We found 1328 (25.4%, out of 5227) hardcoded and 18 (36%, out of 50) obfuscated email addresses were used in two or more phishing kits. In addition, 1011 (18.2%, out of 5227) hardcoded and 5 (10%, out of 50) obfuscated email addresses were found in online pastes (as confirmed from HIBP) — see Table IV. The 50 obfuscated email addresses were in 98 distinct phishing kits, and 45 of those email addresses were not reported in the past; see Table V for the top frequently observed obfuscated email addresses.

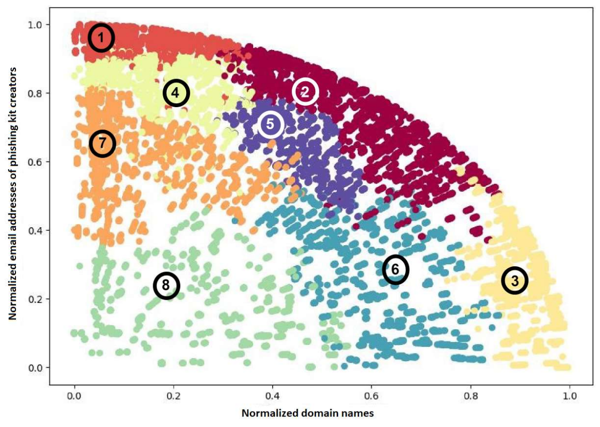


Fig. 4. Phishing kit clusters based on kit developers.

Email Providers. We obtained 5273 valid distinct email addresses, extracted from 3555 phishing kits, out of which 3445 of them belonged to 6 popular email providers (i.e., *Gmail*, *Yandex*, *Yahoo*, *Protonmail*, *Hotmail* and *Outlook*) along with 123 email addresses with other providers (*Aol* (49), *Mail.com* (36), *Icloud* (16), *Zoho* (22)) — see Fig 5 for a comparison among these email providers. However, we also found 1705 email addresses were hosted on non-public, custom email domains.

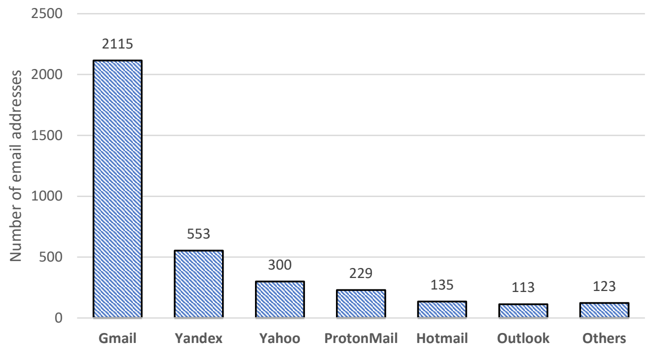


Fig. 5. Distribution of phishing email addresses across service providers.

EA type	# Total EA	# EA in more than 1 kit	# EA in online pastes
Hardcoded	5227	1328	1011
Obfuscated	50	18	5

TABLE IV
TYPE OF COMMON EMAIL ADDRESSES IN PHISHING KITS — EMAIL ADDRESSES = EA.

E. Other observations on phishing kits

VirusTotal scanning of phishing kits. 814/4238 phishing kits were not flagged as malicious by any scanning engine on

Email address	# Phishing kits	# Past breaches
cntr.ii2t@gmail.com	26	3
solustn@gmail.com	6	0
vairus.oh@gmail.com	5	13
hector.dexter1@gmail.com	5	3
dioscalco4@gmail.com	5	0
updates@ourtimewhorers.com	4	2

TABLE V
FREQUENTLY OBSERVED OBFUSCATED EMAIL ADDRESSES IN PHISHING KITS.

VirusTotal, also see [46]. The phishing kits we analyzed, were labeled into different categories by VirusTotal — 2325/4238 as phishing, 830/4238 as trojan, 32/4238 phishing kits were assigned to other labeled categories (e.g., scam, hacktool, spam), and 1051/4238 phishing kits were not labeled.

Malicious files in phishing kits. We examined extensions types (e.g., PHP, HTML) of files within each compressed phishing kit. Common file extensions in phishing kits included PHP, HTML, PNG, CSS, TXT, JS, ICO, GIF and SVG. Apart from these, we also found EXE files in 24 kits (0.6%). We obtained 7 unique EXE files from these kits — i.e., *adb-setup-1.3.exe* (1 kit), *Bomboflix1.1 [Private Edition].exe* (1 kit), *Setup.exe* (1 kit), *Firefox Installer (1).exe* (4 kits), *rundll32.exe* (12 kits), *hiddeninput.exe* (4 kits) and *NuGet.exe* (1 kit). We scanned these EXE files with VirusTotal. The first 3 of them were marked as malicious by at least 4 security vendors (in VirusTotal). *rundll32.exe*[19] is a process that serves as a backdoor, allowing access to user's machine from remote locations, in order to steal sensitive information (e.g., passwords). *Firefox Installer (1).exe* appeared to be a repackaged Firefox installer with malicious dynamic link libraries (DLL).

Organizations targeted by websites created using phishing kits. Feeds of phishing websites from *ECX*, include the brands of the mimicked phishing websites. 206 (4.8%, out of 4328) phishing kits were collected using *ECX*. Using the brand information in *ECX* phishing feeds, we extracted the top 10 organizations pertaining to these phishing kits. Most of these kits deployed phishing websites related to software (36), law enforcement (30), financial services (23) and shopping (19) — see Table VI.

Reuse of phishing kits. We consider a phishing kit is reused, if a kit with the same SHA-1 hash value is hosted on more than one website. We used this approach to measure kits downloaded from live phishing websites. We observed 444 phishing kits were hosted on at least two websites. The maximum number of phishing sites deployed using a particular phishing kit is 20. However, 74.1% (1667 out of 2249) of the phishing kits obtained from live phishing websites were hosted only on one website. The reuse of phishing kits increase return on investment for adversaries who don't have to spend time to recreate phishing sites by mimicking popular websites.

[19]https://www.processlibrary.com/en/directory/files/rundll32/25747/

Organization	Service	# Phishing kits
Microsoft	Software	36
National Police Agency (Japan)	Law enforcement	30
Made-in-China.com	Shopping	19
Banco do Brasil	Financial	10
Linkedin	Social network	8
Societe Generale	Financial	7
BECU	Financial	6
DHL	Logistics	6
Chronopost	Logistics	4
Comcast Cable	Telecommunication	4

TABLE VI
TOP 10 ORGANIZATIONS TARGETED BY PHISHING KITS.

V. ETHICAL CONSIDERATIONS AND LIMITATIONS

In our experiments, we capture leaked sensitive information that are of different types (e.g., email addresses, credit card numbers, phone numbers). As per the guidelines from our university's research ethics unit, we appropriately dispose the captured sensitive information of victims, by using the *shred* utility. We also responsibly disclosed details of Telegram bots used in the analyzed phishing kits to Telegram, for them to take appropriate action.

Although we use known sources of phishing URLs and kits to prepare our dataset, some of the compressed files remaining after our initial filtering may not correspond to real phishing kits. For example, one of the collected files, *MadelineProto-master.zip*, contained the source code for Telegram's MTProto client. Another one, *login.html.zip*, was an incomplete kit as several dependencies were missing from the compressed kit. In this regard, we tried using a tool [47] for phishing kit detection, which indicated that 160/4238 phishing kits in our dataset were not real phishing kits. However, upon manual review of the tool results, we found several instances of both false positives and false negatives. The results pertaining to extracted obfuscated email addresses and generated plaintext files (during our dynamic analysis) are a lower bound, as our framework may not traverse all execution paths in the PHP source code. Our findings on sensitive information exposed as a result of security flaws in phishing kits is also a lower bound. Furthermore, we do not have access to phishing kits hosted on websites on restricted paths. As such, we cannot measure the sensitive information exposed on the corresponding live phishing sites (unless the website has open directory listing). If the authentication tokens of Telegram bots are changed or the Telegram bot is removed from the chat, it will impact our analysis of the exposure of sensitive information through Telegram bots. We use a source code scanning tool (*progpilot*) for detecting security vulnerabilities, which could report false positives in its detection results. Furthermore, some of the vulnerabilities detected by this tool may not be exploitable. However, the analysis from *progpilot* provides an overview of insecure coding practices adopted by kit developers.

VI. Discussion

Security issues in phishing sites that lead into privacy issues. Security issues (e.g., insecure file operations, SQL injection) in phishing sites deployed using phishing kits, will increase the exposure of victim data to multiple threat actors. We found 37.5% of analyzed phishing kits were vulnerable to insecure file operations, that allow an attacker to upload PHP shells to corresponding phishing sites; 2.3% of analyzed phishing kits were prone to SQL injection attacks, that allow stealing phished data stored in databases of corresponding phishing sites. Phishing kits are accessible to everyone, if the path on phishing websites is unrestricted (i.e. open directory listing). These unrestricted paths of phishing websites may contain files with sensitive information, even when the corresponding phishing kit is not accessible from the same paths.

Novel types of backdoors. In the past, traditional forms of backdoors used by phishing kit owners include drop email addresses [48] that are used to expose sensitive information collected through user interactions. Attackers are now increasingly using social networking services, to expand the available backdoor channels (e.g., using Telegram bots) in order to expose sensitive user information to adversaries. We found 7 Telegram bots that served as backdoors were obfuscated (in 12 phishing kits) using various techniques (e.g., hidden in GIF files included in PHP source code).

Exposure of threat actor information. Phishing kits can be used to collect information about the threat actors behind them. Apart from email addresses, our work also reveals the potential of hardcoded Telegram bots in phishing kits in revealing information about other participants involved in phishing (e.g., creators/administrators of chats where the phished information is posted). We found 61 Telegram users were creators/administrators of 71 chats that were used for phishing. Necessary measures, such as legal actions, addition to blacklists and termination of services, can be taken by law enforcement and service providers by leveraging the exposed threat actor information.

Anti-evasion in phishing kits. Phishing kits use various evasion techniques [49], [50], including the use of third parties (e.g., *ipinfo.io*) to hide its presence for non-human users (e.g., security crawlers). We observed *ipinfo.io* third party service used in 388 phishing kits to avoid detection by security crawlers. The API key used for *ipinfo.io* is hardcoded in the kit. Anyone with access to the kit can obtain this key and use it multiple times such that it exceeds the allowed usage rate limit, effectively disabling this evasion mechanism in all phishing websites deployed using the kit.

Enabling notification to phishing victims. We do not retain the obtained victim data (including credit card numbers, login credentials, social security numbers, phone numbers) for ethical and legal reasons. However, using our framework, such data can be used by law enforcement and service providers to notify the victims whose data have been stolen and exposed on the internet, to limit its misuse by adversaries.

VII. Conclusion

We design and implement a framework for automated collection and security analysis of phishing kits to understand additional user-data exposure risks due to the use of these kits. We used our framework to collect and evaluate a large dataset of phishing kits. We found backdoors, measured the sensitive data exposure, and detected security vulnerabilities in the collected kits. Due to the security flaws in phishing kits and the mistakes made by kit deployers while setting up these kits, the victim data indeed becomes accessible to a wider set of adversaries, beyond the kit deployers and kit developers. We hope our framework would be useful for security professionals and researchers to efficiently detect security flaws in phishing kits and their deployments—to quickly restrict data exposure caused by these kits.

Acknowledgment

This work is partially supported by the Canadian Internet Registration Authority (CIRA). We are also grateful to the anonymous reviewers of APWG eCrime 2022 for their insightful comments and suggestions. We are also thankful to APWG for granting us access to the eCX phishing dataset.

References

[1] APWG, "Phishing activity trends report," 2022, https://docs.apwg.org/reports/apwg_trends_report_q2_2022.pdf.

[2] Kaspersky, "Quick, cheap and dangerous: how scammers are creating thousands of fake pages using phishing kits," 2022, https://www.kaspersky.com/about/press-releases/2022_quick-cheap-and-dangerous-how-scammers-are-creating-thousands-of-fake-pages-using-phishing-kits.

[3] T. Moore and R. Clayton, "Discovering phishing dropboxes using email metadata," in *APWG eCrime Researchers Summit 2012)*, Las Croaba, Finland, Oct. 2012.

[4] J. Wright, "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.

[5] Akamai, "16Shop: Commercial Phishing Kit Has A Hidden Backdoor," 2019, https://www.akamai.com/blog/security/16shop-commercial-phishing-kit-has-a-hidden-backdoor.

[6] X. Han, N. Kheir, and D. Balzarotti, "Phisheye: Live monitoring of sandboxed phishing kits," in *ACM Conference on Computer and Communications Security (CCS'16)*, Vienna, Austria, Oct. 2016.

[7] T. Moore and R. Clayton, "Examining the impact of website take-down on phishing," in *APWG Symposium on Electronic Crime Research (eCrime'07)*, Pennsylvania , CA, USA, Oct. 2007.

[8] L. Cashdollar, "Identifying vulnerabilities in phishing kits," 2019, https://www.akamai.com/blog/security/identifying-vulnerabilities-in-phishing-kits.

[9] PhishTank, "Phishtank," 2022, https://phishtank.org/.

[10] OpenPhish, "Openphish," 2022, https://openphish.com/.

[11] APWG, "The APWG ecrime exchange (ECX)," 2022, https://apwg.org/ecx/.

[12] Pavlovic, Alen, "Introducing phishing kit tracker," 2020, https://marcoramilli.com/2020/07/16/introducing-phishingkittracker/.

[13] phishunt.io, "phishunt.io," 2021, https://phishunt.io/.

[14] Telegram, "Telegram Messenger," 2021, https://telegram.org/.

[15] H. Bijmans, T. Booij, A. Schwedersky, A. Nedgabat, and R. van Wegberg, "Catching phishers by their bait: Investigating the dutch phishing landscape through phishing kit detection," in *USENIX Security Symposium (USENIX Security'21)*, Online, Aug. 2021.

[16] B. Kondracki, B. A. Azad, O. Starov, and N. Nikiforakis, "Catching transparent phish: Analyzing and detecting MITM phishing toolkits," in *ACM Conference on Computer and Communications Security (CCS'2021)*, Online, Nov. 2021.

[17] E. Merlo, M. Margier, G.-V. Jourdan, and I.-V. Onut, “Phishing kits source code similarity distribution: A case study,” in *Software Analysis, Evolution and Reengineering (SANER’2022)*, Honolulu, HI, USA, Mar. 2022.

[18] A. Oest, Y. Safei, A. Doupé, G.-J. Ahn, B. Wardman, and G. Warner, “Inside a phisher’s mind: Understanding the anti-phishing ecosystem through phishing kit analysis,” in *APWG Symposium on Electronic Crime Research (eCrime’18)*, San Diego, CA, USA, May 2018.

[19] phishfinder, “phishfinder,” 2019, https://github.com/cybercdh/phishfinder.

[20] M. Cova, C. Kruegel, and G. Vigna, “There is no free phish: An analysis of” free” and live phishing kits.” in *USENIX Workshop on Offensive Technologies (WOOT’08)*, San Jose, CA, USA, Jul. 2008.

[21] Selenium.dev, “Selenium,” 2022, https://www.selenium.dev/.

[22] L. Lazar, “Our analysis of 1,019 phishing kits,” 2018, https://www.imperva.com/blog/our-analysis-of-1019-phishing-kits/.

[23] S. Zawoad, A. K. Dutta, A. Sprague, R. Hasan, J. Britt, and G. Warner, “Phish-net: investigating phish clusters using drop email addresses,” in *APWG eCrime Researchers Summit 2013*, San Francisco , CA, USA, Sep. 2013.

[24] S. Gatlan, “Phishing kits add more vulnerabilities to hacked servers,” 2019, https://www.bleepingcomputer.com/news/security/phishing-kits-add-more-vulnerabilities-to-hacked-servers/.

[25] VirusTotal, “Virustotal,” 2022, https://www.virustotal.com/gui/home/upload.

[26] VirusTotal, “Threat classification,” https://developers.virustotal.com/reference/popular_threat_classification.

[27] Wikipedia, “Abstract syntax tree,” 2022, https://en.wikipedia.org/wiki/Abstract_syntax_tree.

[28] PHP-Parser, “PHP Parser,” 2022, https://github.com/nikic/PHP-Parser.

[29] Github.com, “Phpdeobfuscator,” 2022, https://github.com/simon816/PHPDeobfuscator.

[30] python-email validator, “email-validator: Validate Email Addresses,” 2022, https://github.com/JoshData/python-email-validator.

[31] T. Hunt, “Introducing paste searches and monitoring for “have i been pwned?”,” 2014, https://www.troyhunt.com/introducing-paste-searches-and/.

[32] Haveibeenpwned.com, “Have I been pwned,” 2021, https://haveibeenpwned.com/API/v3.

[33] Scikit-learn, “sklearn.preprocessing.normalize,” 2022, https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.normalize.html.

[34] scikit-learn, “sklearn.cluster.AgglomerativeClustering,” 2022, https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html.

[35] Progpilot, “Progpilot,” 2020, https://github.com/designsecurity/progpilot.

[36] Forcepoint, “Forcepoint,” 2019, https://www.forcepoint.com/blog/x-labs/tapping-telegram-bots.

[37] Group-ib, “Send to saved messages: Cybercriminals use telegram bots and Google forms to automate phishing,” 2021, https://www.group-ib.com/media/phishing-automation/.

[38] Telegram, “Telegram Bot APIs,” 2021, https://core.telegram.org/bots/api.

[39] PDSCAN, “Pdscan,” 2019, https://github.com/ankane/pdscan.

[40] Linux.die.net, “Shred,” 2010, https://linux.die.net/man/1/shred.

[41] Whispers, “Whispers,” 2021, https://github.com/Skyscanner/whispers.

[42] D. L. Wheeler, “zxcvbn: Low-Budget password strength estimation,” in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, Aug. 2016, pp. 157–173. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/wheeler

[43] Discord, “Discord,” 2022, https://discord.com/.

[44] P. Report, “Iok rules,” 2022, https://phish.report/IOK/indicators.

[45] Discord, “Intro to Webhooks,” 2022, https://support.discord.com/hc/en-us/articles/228383668-Intro-to-Webhooks.

[46] P. Peng, L. Yang, L. Song, and G. Wang, “Opening the blackbox of Virustotal: Analyzing online phishing scan engines,” in *IMC’19*, Amsterdam, Netherlands, Oct. 2019.

[47] K. Hunter, “Github.com,” https://github.com/SteveD3/kit_hunter.

[48] H. McCalley, B. Wardman, and G. Warner, “Analysis of back-doored phishing kits,” in *International Conference on Digital Forensics (IFIP’2011)*, Orlando, FL, USA, Jan. 2011.

[49] A. Oest, Y. Safaei, A. Doupé, G.-J. Ahn, B. Wardman, and K. Tyers, “Phishfarm: A scalable framework for measuring the effectiveness of evasion techniques against browser phishing blacklists,” in *IEEE Symposium on Security and Privacy (SP’2019)*, San Francisco , CA, USA, May 2019.

[50] BleepingComputer, “Phishing kits constantly evolve to evade security software,” 2022, https://www.bleepingcomputer.com/news/security/phishing-kits-constantly-evolve-to-evade-security-software/.