Toward a Better Understanding of Mobile Users’ Behavior: A Web Session Repair Scheme

FABIEN CHARMET, TOMOHIRO MORIKAWA, AND TAKESHI TAKAHASHI, (Member, IEEE)

1National Institute of Information and Communications Technology, Koganei, Tokyo 184-8795, Japan
2Graduate School of Information Science, University of Hyogo, Hyogo 650-0047, Japan

Corresponding author: Fabien Charmet (fabien.charmet@nict.go.jp)

This work was supported in part by the National Institute of Information and Communications Technology (NICT) and in part by the University of Hyogo.

ABSTRACT Using mobile devices to browse the Internet has become increasingly popular over the years. However, the risk of being exposed to malicious content, such as online scams or malware installations, has also increased significantly. In this study, we collected smartphone data from volunteer users by monitoring their use of the Web and the applications they install on their devices. However, the collected data is sometimes incomplete due to the technical limitations of mobile devices. Thus, we propose a data repair scheme to restore incomplete data by inferring missing attributes. Here, the restored data represent the browsing history of a mobile user, which can be used to determine if and how the user has been the victim of web or mobile-specific attacks to compromise their sensitive data. The accuracy of the proposed data repair scheme was evaluated using a machine learning algorithm, and the results demonstrate that the proposed scheme properly reconstructed a user’s browsing history data with an accuracy of 95%. The usability of the repaired data is demonstrated by a practical use case. The user’s browsing history was correlated with other types of data, such as received SMSs and the applications installed by the user. The results demonstrate that a user can fall victim to SMS-based phishing (SMShing) attacks, where the attacker sends an SMS message to a user to trick them install a malicious application. We also present a case of a social engineering attack, where the victim was manipulated to provide their Amazon credentials and credit card details.

INDEX TERMS Mobile security, machine learning, data collection, attack detection.

I. INTRODUCTION

Mobile communication devices, e.g., smartphones, are indispensable in daily life, and the number of smartphone is estimated to be greater than seven billion worldwide [1]. Among such devices, the Android operating system (OS) occupies the largest market share [2]. Unfortunately, mobile users are vulnerable to a wider range of attacks compared to PC users due to the nature of the devices. For example, mobile users can receive SMS messages and phone calls from attackers and scammers. In recent years, the number of SMS-based phishing (SMShing) attacks has been increasing consistently [3]. In addition, various malware can be installed on unsuspecting users’ phones. Such malware may come from various application markets, including the official Google Play Store [4].

Few studies have focused on the characteristics of browsing patterns from mobile users. In [5], the HTTP logs of mobile devices were analyzed and used to provide a statistical analysis of the malicious websites visited by users, and a predictive model was developed to determine the probability of a user being exposed to malicious content within a given period. However, web-browsing activities from mobile devices are interwoven with other activities, e.g., receiving an SMS message or installing an application, and such activities generate logs that can be analyzed to improve the security of mobile users. Accessing this type of information remains a complex task because most of it cannot be obtained outside a mobile device. Note that the HTTP logs provided by mobile...
network carriers may only present a small picture of a mobile user’s habits (due to their Wi-Fi use), which increases the difficulty of recomposing the user’s complete web history.

Based on the above, we aim to address the following three issues in this paper.

- **Efficient data collection**: Good understanding of mobile device usage comes from the ability to gather sufficient and diverse data to represent a user’s behaviors and patterns. This requires circumventing the limitations of the Android OS. In addition, data collection processes must respect the user’s privacy needs, e.g., avoiding collecting banking information or confidential emails. Thus, collecting appropriate data and protecting users are crucial factors.

- **Comprehensive trace repair**: Data collection often generates incorrect or incomplete entries, which is partly due to the lack of resources available to the device collecting the data. Another cause is proactive avoidance of confidential data collection. When the collected data are exploited by investigators, these issues must not hinder the investigators’ understanding. Thus, accurate repair of the data collected by a sensor is required to conduct a comprehensive analysis of the user’s behaviors and patterns.

- **Pattern analysis**: Understanding how mobile users are exposed to malicious content is the first step in developing mechanisms to protecting users. The different traces generated by users collectively contain the patterns we seek to identify; thus, these traces must be analyzed manually to provide the initial interpretation of a given event.

In this study, data from cell phones were collected using a custom-made sensor [6] installed by volunteer users. This sensor is based on the Android OS’ accessibility service [7], and it collects data from multiple sources and addresses issues caused by the sandboxing of applications of the Android OS. While this sensor was developed specifically for the Android OS, we believe that other mobile OSs (e.g., iOS) are susceptible to the limitations we observed throughout this study. The user-side data collection process employed in this study allows us to exploit multiple aspects of cell phone usage that cannot be captured by mobile network carriers. However, the collection process is limited by the computing power of the device and the privacy requirements of both the users and our research ethics standards. When the sensor collects incomplete malicious information, it may become impossible to identify and understand the corresponding security incident. Thus, we propose a data repair scheme that can infer the missing properties of the collected data and generate traces (hereafter referred to as browsing sessions). These browsing sessions are then used to compose each user’s web history.

In addition, the usability of the repaired browsing sessions is demonstrated in a practical use case. Here, by correlating the browsing sessions with the reception of SMSs and the installation of malicious applications, we determine when a user has installed malware due to an SMShing attack. We also find traces of a user falling victim to a social engineering attack to reveal their credit card details and Amazon credentials. We believe that our findings can help other researchers and serve as a basis for the design of detection and prevention systems for mobile users. Possible applications include data augmentation for security analysts, basic forensic capabilities for end users on their own devices, and data mining results for academic researchers. Our primary contributions are summarized as follows.

1) Various data are collected from mobile devices using a custom-made sensor application. These data include web-browsing history information, web links obtained from received SMSs, installed and uninstalled applications, and information about applications currently running in the foreground on the device.

2) We propose a data repair scheme, and its usability to reconstruct the collected information is demonstrated in a practical use case. In contrast to the work described in [8], the data collection process employed in this study does not include a repair application programming interface (API); thus, other features are exploited. This scheme differs from the existing methods because mobile devices (in contrast to PC browsers) cannot access typical information such as referrer headers or a source tab ID. Instead, we augment the dataset with additional features to perform the data repair process. Specifically, the collected data are augmented by including similarity and technical features related to the visited websites.

3) A machine learning classifier is designed to evaluate the accuracy of the proposed data repair scheme, and experimental results demonstrate that 95% classification accuracy was achieved. Here, the training data were obtained by correcting a subset of the reconstructed sessions manually and determining whether each entry fits the current session. In addition, the training data were generated to facilitate more complex future interpretations of the results obtained by the proposed data repair scheme.

4) A case study was conducted to demonstrate the usability of the repaired data. Here, the browsing sessions were correlated with other elements collected by the sensor to investigate several security incidents. Traces showing users being manipulated by malicious webpages or by successful SMShing attacks to compromise sensitive data are presented. For example, an SMShing attack prompts a user to click on a malicious link and install a malicious application. We also demonstrate how a user may fall victim to a phishing website by exposing their credit card information.

To the best of our knowledge, this study represents the first work on reconstructing web-browsing sessions using collected user data. The goal of this study is to improve security analyses by correlating information about malicious
webpage visits with SMS attacks and application installation, which has typically been unavailable to researchers.

The remainder of this paper is organized as follows. Related work is reviewed in Section II. The dataset and proposed data repair scheme are introduced in Section III. The machine learning model designed to evaluate the accuracy of the proposed session-repair scheme is presented in Section IV. The findings of our investigation of the dataset are presented in Section V. A discussion of the limitations and hypotheses of this study is presented in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORK
Malicious page detection is a well-investigated topic in the research community. A classic way to prevent users from accessing malicious pages is the implementation of blacklists. Typically, a third-party maintains a database of URLs considered malicious and distributes this database to end users who can access it when browsing the Internet. Google Safe Browsing (GSB) [16] is a representative example of a blacklist system. However, such implementation is expensive in terms of maintenance and manual labor because the target URLs are frequently submitted and evaluated manually by users. PhishTank [17] is an open community that provides free access to a list of phishing URLs. To circumvent the disadvantages of manual labor, heuristic methods have been developed to automate the characterization of malicious pages. A previous study [18] analyzed a malicious JavaScript code and characterized it according to its activation conditions. The results demonstrate that environment-dependent codes are frequently malicious.

Machine learning models have also been prevalent due to the large URL databases that are currently available. A preliminary study is Prophiler [19], where classifiers were trained to filter benign webpages from malicious webpages. Here, if a webpage poses a certain risk, it is forwarded to a subsequent and more time-consuming analysis tool. As a result, Prophiler effectively reduces the number of pages that must be analyzed by costly tools. In [19], lexical features from URLs, JavaScript code included in the webpages, and information extracted from DNS records were exploited. In addition, Verma et al. [20] explored both batch and online algorithms to characterize various URLs. Here, legitimate URLs were provided by DMOZ [21], and malicious URLs were obtained from PhishTank and the Anti-Phishing Working Group [22]. However, the features used in this study were limited to various N-grams extracted from the URLs. This hypothesis arises from the fact that phishing URLs trick users by only switching a few characters, which may be overlooked by the human eye. Verma et al. employed multiple flavors of the perceptron [23]. In [24], Ma et al. explored a broader range of features. Beginning with a simple set of lexical features, they augmented it by including host-based features, e.g., WHOIS registration date information and DSN records etc.. Here, three different models were evaluated, i.e., naive Bayes, support vector machine, and logistic regression models, and a comparative analysis of multiple features that were generated automatically and several expert-selected features was provided. The results demonstrated that an increased number of features, particularly those derived from WHOIS information, improved classification performance significantly.

User behaviors, including their knowledge about mobile security, have also been investigated [5], [8], [9], [25], [26], [27], [28], [29]. Some of these studies included participants who consented to install a sensor for monitoring purposes. The investigation of the user security awareness and expertise has demonstrate that high expertise does not necessarily correlate positively with computer security.

In addition, surveys can suffer from insufficient sample sizes or low response rates [5], [26]. A seminal work on user behavior prediction was proposed by Canali et al. [9], where the web histories of 100,000 users obtained from data provided by a major antivirus software were analyzed. Then, a set of features that can be used by machine learning models was extracted. This study achieved a reasonable level of accuracy (up to 87%) in determining which users were likely to be the victims of web attacks.

Sharif et al. [5] proposed an extensive analysis of HTTP logs from a mobile network carrier. This log analysis process was combined with a mobile user survey, and a prediction model was designed to determine whether a user is likely to be exposed to malicious content over a long period. However, the performance of this model was slightly worse than that reported in [9]. Sharif et al. [5] claimed that, in contrast to the findings reported in [9], they used a more limited set of features, which is also less computationally intensive. Finally, they built a classifier that predicts the probability of a user being exposed to malicious content within a 30-second time frame.

In addition, a study based on chain-redirection reconstruction has been reported [10], and this study is similar to our approach. In [10], crowdsourcing was used to examine how users go through redirection chains before reaching the final webpage. Then, individual chains were combined into redirection graphs, which were then used to distinguish legitimate webpages from malicious webpages. Although that study exploited the referrer’s header to construct the corresponding graph, the necessary information was inaccessible and could not be used in our current study, which is explained in further detail later in the paper.

Another study by Takahashi et al. [8] used a Chrome extension [6] developed to monitor the browsing behavior of PC users. This extension records each tab opened by a user and captures page redirections and bookmark usage. Using these data, the authors concluded that the predominant source of user compromise is actually caused by bookmarks. Indeed, users regularly visit malicious websites, e.g., illegal streaming services, and add such sites to their bookmarks. The authors also designed a graph analysis tool that considers the transitions between legitimate and malicious domains, and they implemented a filtering solution to block
malicious domains not listed on commercial blacklists. They also employed a dataset that is similar to the dataset used in our study. The major difference is that we cannot capture tab and page transition data as accurately as they did, and our study focuses on mobile users, whereas their study focused on PC users.

In [11], Bank et al. collected the browsing history of 24,198 users to build Markov chains representing the risk of exposure to malicious content. Here, volunteer users installed a browser extension to monitor their activity, similar to [8]. The collected dataset also reports similar features, e.g., the previous website or the requested page. Bank et al. reported that they had to partially repair their data, specifically when the previous_site attribute was blank. However, they considered that this was either the beginning of a browsing session or the opening of a new tab in the browser. In this study, the maliciousness of pages was determined using GSB, and the authors consider two unique scenarios, i.e., using truncated sessions or using complete sessions. Truncated sessions stop at the first malicious page encountered by the user. The results demonstrated that while the complete session yielded the best predictions in terms of user exposure, the truncated sessions also performed relatively well. Note that the dataset used in that was also extremely imbalanced (1:540 malicious to benign) because it was labeled with GSB five years after its original creation.

Similar to [9], Levesque et al. [12] attempted to link the sociodemographic factors of users with their behavior and tendencies to access malicious content. Their user study involved 50 users recruited from the university hosting Levesque et al.. While they acknowledged that this is a small sample size, they argued that their user-side collection process can leverage insights that were inaccessible in previous studies. Specifically, they ensured that the users were all distinct and represented a diverse population in terms of age, gender, occupation, etc. The scope of this study is also broader because it included both web-browsing history and downloaded malware. Their predictive model employed a multilayer perceptron to predict the likelihood of a user being exposed to malicious content.

Kovacs [14] investigated a problem similar to that considered in our study. Rather than repairing incomplete browsing session, they predicted the time spent on each website and which tab was in focus during a given browsing session. They also exploited the Google API through a browser extension to collect data from volunteer users. The features used in that study were to be divided into three categories, i.e., time-based, number of visits, and the referrer id. In addition, they employed a random forest classifier model and achieved approximately 80% classification accuracy. Note that this study exists beyond the cybersecurity domain; however, the methodology used yielded valuable insights relative to browsing session reconstruction.

In opposition to previous works, Szurdi et al. [13] emulated users to analyze the behaviors of malicious websites. They implemented a platform to generate browsing patterns involving various devices, browsers, whether a proxy is employed, etc. This study also represents one of the few studies that targeted mobile users (both emulated and a real human agent). Their platform included multiple evasion mechanisms designed to avoid the countermeasures set up by malicious websites. They collected a total dataset of 2 TB of data, and they discussed an automated labeling scheme based on HTTP features to determine whether a given webpage is malicious or benign.

Other studies have focused on other related issues. For example, Bhagavatula et al. [15] investigated whether users read articles about major security incidents, e.g., WannaCry or the Equifax breach. Similar to the work of Levesque et al. [12], Bhagavatula et al. [15] conducted a demography analysis of their userbase (303 participants over 44 months). Their findings demonstrated that users are not actively engaged in researching security incidents, despite the fact that they might be affected directly by such events, e.g., out of the 59 users potentially affected by the Equifax breach, only 15 of these users actually read about the event. Bhagavatula et al. also emphasized the need to better disseminate information related to security incidents and how to attract the reader’s attention.

Other than the studies listed in Table 1, various studies from adjacent domains have investigated related concepts, e.g., domain generation algorithms (DGA). Specifically, the task of detecting DGA-generated domains is similar to the detection of phishing URLs. For example, in [30], Morbidoni et al. computed (1,2,3)-grams of domain names generated using different algorithms. Their goal was to allow a deep learning

| Reference | Device type | Data collection location | Data collection tool | Data reconstruction | Contribution |
|-----------|-------------|-------------------------|---------------------|--------------------|--------------|
| [9]       | PC          | User side               | Antivirus software  | Not required       | Prediction to exposure |
| [5]       | PC          | Network side            | Network Log collection | Not required       | Prediction to exposure |
| [10]      | PC          | User side               | Antivirus software  | Not required       | Redirection chain analysis |
| [8]       | PC          | User side               | Custom sensor       | Limited            | Prediction to exposure |
| [11]      | PC          | User Side               | Custom sensor       | Not required       | Prediction to exposure |
| [12]      | PC          | User Side               | Custom sensor       | Required           | Security awareness analysis |
| [13]      | PC / Mobile | User Side               | Custom sensor       | Not required       | Security analysis of malicious websites |
| [14]      | PC          | User Side               | Custom sensor       | Required           | Browsing time and active tab prediction |
| [15]      | PC          | User Side               | Custom sensor       | Not required       | Security awareness analysis |
| Proposed scheme | Mobile | User side               | Custom sensor       | Required           | User compromise extraction |
model to differentiate benign domain names from malicious domain names efficiently using a small dataset. In [31], Lazar et al. deployed an analysis platform in a DNS architecture provided by IBM, and their corresponding experiment represented a rare example where a real-world production environment was used for data collection. They attempted to link DNS communications with specific malware attack campaigns. Here, they used DNS communication and VirusTotal to help label their data, and then they applied clustering algorithms to the labeled data. By grouping domains with similar communications, they found that they could effectively determine the type of malware using these domains. Liang et al. took an approach similar to [30] by considering the length of domain names. For ultra-short domains, they employed an attention mechanism, and for moderate length domain names they extracted n-grams features and a Convolutional Neural Network. Finally, for extra-long domains, they employed handcrafted-features combined with a random forest classifier. It appears that a common limitation in DGA research lies in the data collection process, and only Lazar et al. were able to deploy a large infrastructure.

III. METHODOLOGY
In this section, we describe the steps of the methodology used in this study. First, we present the proposed data collection process, and then we discuss the ethical considerations of our methodology in detail. Finally, we present the proposed data repair scheme.

A. USER-SIDE DATA COLLECTION PROCESS
Section II described previous studies that collected data from either the user side or from the network carrier side. External data collection processes will arguably always provide fewer details about the end user’s behavior. Independent of the technical limitations of the collection process, data collected on the user’s side will always raise ethical and privacy concerns. In this study, users provided consent after being thoroughly informed about the purpose of the sensor and our investigation. Specific details are provided in Section III-B. Due to privacy and legal requirements, the dataset used in this study cannot be disclosed in any form.

A sensor developed by [6] was installed on the mobile device of each participant. The sensor collected various types of data, which are described in the following. The data required for this study were collected from March 2020 to February 2021. Here, 2,125 volunteer users participated, as shown in Figure 1. Among the participants, 107 users who were exposed to malicious content provided entries flagged by GSB as malicious. Thus, we limited the scope of our study to those 107 users. Therefore, approximately 5% of the users in the dataset were exposed to malicious content. While this is approximately half of the percentage used in [5], the population size of our dataset is only 10% that of [5]. The recruitment campaign was run similarly to the one run in [8]. We argue that this is a sufficient number of users for the purposes of this study because it is similar to the population size in similar works, e.g., 185 users in [14], and 50 users in [12]. This is because our population size is only 10% of that used in [5] and because our recruitment campaign targeted users who were more tech-savvy (similar to [8]) than the general population.

The data collected by the sensor included web accesses, application installations and uninstalls, received SMSs, changes in the foreground application, authorization for installation of unknown applications, and other physical device information data (the latter were not used in this study). Some of the collected data contained attributes that can be exploited by GSB to detect potential threats. The information collected for each data type is described as follows.

1) Web accesses
   - User id: The unique identifier of the user generating the entry
   - URL: The URL collected from the address bar\(^1\)
   - Access type: Origin of the browser window: Chrome, Custom tabs [32], or Unknown if the sensor failed to collect the information.
   - Malicious: Relevant threat information if GSB flagged the URL as malicious
   - Tab hash: Unique identifier for the window
   - Multiple redirect: Boolean representing whether the current URL was reached through multiple redirections
   - Date: Time when the user visited the website

2) Application installation or uninstallation
   - User id: Unique identifier of the user generating the entry
   - Package name: Name of the application
   - Removed: Boolean indicating whether the app was installed or uninstalled

3) SMS
   - User id: Unique identifier of the user generating the entry

\(^1\) The sensor also collected the default text when opening a new window e.g., “Search or type web address” or “about:blank.”
privacy. We also accepted the terms and conditions associated with the use of the mobile sensor [6]. All collected data conformed to these terms and conditions, stipulating that the data collected by the sensor were used only for research purposes (i.e., to detect and prevent access to malicious URLs). All users that were required to install this browser extension agreed to these terms and conditions.

The collected data included privacy-related details; thus, they were used under strict restrictions. Any personally identifiable user information was deleted or coded before the records were stored on the servers. The user ID recorded in the log was an internal number unique to each user that could not be used to reveal any personally identifiable information. In addition, raw URLs could not be shared by external parties. Thus, we did not use VirusTotal [33], which requires the submission of raw URLs. Instead, we used GSB to evaluate the maliciousness of URLs because this tool does not require the upload of raw URLs. We also deleted all records of users who requested them to be deleted.

The logs used in the analysis were stored on a server in a secure facility. Only registered users from registered machines with adequate security measures were permitted to access the log data. In addition, no permission to copy the raw data outside the machine was given. Thus, all analyses were conducted on secure servers, and only the aggregated results were exported from the secure server for further analysis.

C. SESSION-REPAIR SCHEME

Here, we discuss the algorithms, assumptions, and metrics employed to repair the browsing session. As described in Section I, some entries in the dataset were incomplete due to several limitations. When reconstructing browsing sessions, the most important field is the tab hash field. In the following, entries with a missing tab hash value are referred to as orphan entries.

A simple approach to reconstructing browsing session is to group entries with the same hash value and sort them in chronological order. Algorithm 1 describes a naïve implementation of a browsing session-repair algorithm. This algorithm begins by initializing a dictionary containing a list of sessions for each user. Then, for each user, a dictionary with the tab hash as a key and entries that match the key are initialized. Next, a loop for the corresponding user is created.

1) ASSUMPTIONS

The following assumptions were considered in the design of our metrics and the corresponding repair algorithm.
Algorithm 1 Naive Session-Repair Algorithm

Data: Log entries of the browsing history
Result: Log entries of each browsing session for each user

Initialize an empty user dictionary user_dict
for user u in data do
    Initialize an empty dictionary as session_dict
    for d in data[u] do
        tabhash ← d['tab hash']
        session_dict[tabhash].insert(d)
    end
    user_dict[u] = session_dict
end
return user_dict

1) **Contextual coherence:** It was assumed that entries are typically related to either one or both of their neighbor entries in a browsing session. It seems unlikely that each successive page visited by a user has no contextual relation with the previous page (e.g., a user never visits two pages of the same website consecutively).

2) **URL similarity:** The next entry’s URL to be inserted should be similar to either one or both of the URLs corresponding to the previous and/or next entry’s URLs. This assumption allowed us to represent continuity when browsing the same website. However, this may not be true for the transition to another website. Potential solutions to this problem are discussed in Section VI.

3) **Chronological dependence:** We assumed that a user continuously uses a cell phone when browsing the Internet. Thus, if two browsing entries are close chronologically, it is likely that they belong to the same browsing session. Note that this assumption does not cover cases where the same browser tab may be reused multiple times or if there is a long break between the two visits. We propose solutions to partially address this issue.

2) **SESSION-REPAIR SCORE**

In this section, we describe the evaluation score designed for the proposed repair algorithm. The purpose of this score is to evaluate the probability of a specific entry being related to the URL similarity of its chronological neighbors but is inversely proportional to the temporal distance between its neighbors.

Here, the temporal distance between entries \( x_1 \) and \( x_2 \) is denoted \( \text{dur}(x_1, x_2) \), and the similarity ratio between the URLs of entries \( x_1 \) and \( x_2 \) is denoted \( \text{sim}(x_1, x_2) \). In addition, \( \text{prev}_{\text{entry}}(x) \) (respectively \( \text{next}_{\text{entry}}(x) \)) represents the entry that chronologically precedes (respectively succeeds) entry \( x \) in the current browsing session. The following formula is proposed to evaluate a specific entry \( x \):

\[
\text{score}(x) = \frac{1 + \text{sim}(x, \text{prev}_{\text{entry}}(x)) + \text{sim}(x, \text{next}_{\text{entry}}(x))}{\text{dur}(x, \text{prev}_{\text{entry}}(x)) + \text{dur}(x, \text{next}_{\text{entry}}(x))}
\]  

Algorithm 2 Proposed Repair Scheme Algorithm

Data: Log entries of the browsing history for user \( u \)
Result: Browsing sessions including orphan entries

session_dict ← Algorithm 1
orphan_list ← getOrphans(data)
for o in orphan_list do
    available_session ← list()
    session_found ← False
    for s in session_dict do
        if o['access type'] == e['access type'] or o['access type'] == "Unknown" then
            available_session.insert(s)
            session_found ← True
        end
    end
    if session_found == True then
        tempscore ← 0
        tempsession ← Ø
        finalsession ← Ø
        for s in available_session do
            session, score ← compute_score(x, p)
            if score > tempscore then
                tempscore ← score
                tempsession ← s
                finalsession ← session
        end
    end
    session_dict[tempsession] ← finalsession
end
return session_dict

We evaluated several metrics related to the similarity of URLs between each web access [34], [35], [36], [37], i.e., the Levenshtein distance [38], the \( \text{Sørensen-Dice} \) coefficient [39], the Jaccard index [40], and the SequenceMatcher [41] function from the difflib Python library, which is based on the Ratcliff and Obershelp algorithm [42].

3) **REPAIR ALGORITHMS**

We designed an algorithm to repair each user’s browsing sessions. Here, the first step is to extract the existing browsing sessions using the tab hash field (Algorithm 1) and retrieve orphan entries for further processing. Then, for each orphan entry, the corresponding sessions that the entry may fit in are retrieved. The score for each session is computed, and the orphan entry is inserted into the session with the highest score. This process is shown in Algorithm 2, and the computation of the corresponding score is described in Algorithm 3.

Algorithm 2 begins by collecting the reconstructed sessions acquired by Algorithm 1 and the orphan sessions. Then, a Boolean session_found that serves as an end condition for
Algorithm 3 Score Computation

**Data:** $e$ is the orphan entry, session $s$ to check  
**Result:** score as defined in Section III-C2, updated session $s$

$s$.insert($e$)  
index $\leftarrow$ $s$.index($e$)  
if index $==$ 0 then  
prev_entry $\leftarrow$ $e$['URL']  
before $\leftarrow$ $e$['date']
else  
prev_entry $\leftarrow$ $s$[(index - 1)][‘URL’]  
before $\leftarrow$ $s$[(index - 1)][‘date’]
if index $== s$.length()-1 then  
next_entry $\leftarrow$ $e$['URL']  
after $\leftarrow$ $e$['date']
else  
next_entry $\leftarrow$ $s$[(index + 1)][‘URL’]  
after $\leftarrow$ $s$[(index + 1)][‘date’]  
t $\leftarrow$ $e$['date’]  
$u$ $\leftarrow$ $e$[‘URL’]  
score $\leftarrow$ $1 + \text{sim}(u, \text{prev}_\text{entry}) + \text{sim}(u, \text{next}_\text{entry})$  
$1 + |t - \text{before}| + |\text{after} - t|$  
return score, $s$

**TABLE 2. Manual evaluation summary.**

| Dataset description | 8553544 |
|---------------------|---------|
| Dataset size (# of entries) | 8553544 |
| Dataset size (GB) | 1.2 |
| Number of malicious sessions | 964 |

**Manual correction**

| Number of participants | 1 |
| Time spent (# hours) | 2 |
| Number of corrected entries | 2852 |
| Malicious-to-legalitimate session ratio | 1:1 |

During the evaluation, we assumed that entries having URLs in the same domain and a small-time difference were inserted coherently. If the domains did not match, we attempted to identify whether the domain belongs to an ad provider because the entry may be generated by a redirection or a pop up. If it does not, we checked the website using a search engine to determine if the topic is coherent with the neighboring entries. Note that some URLs were no longer valid; thus, they could not be verified directly. However, they could still be investigated using traffic analysis or malware detection websites. It was also assumed that an adult website is a plausible source for redirecting malicious content. Here, each entry not labeled correct is considered incorrect by the binary classifier. However, in future, we intend to use a ternary classification to reconstruct browsing sessions not recorded by the sensor, as described in Section VI. The data corresponding to the manual evaluation of the proposed repair scheme are summarized in Table 2. Some excerpts generated by the proposed repair scheme were selected due to their relevance to our study (Tables 9 and 10). We extracted several features from the browsing sessions, and these features can be divided into similarity features and technical features. Here, the similarity features represent the chronological and URL similarities between the current entry and its neighbors in the session, and the technical features represent the specificity of an entry. The features used by the classifier are described in Table 3.

**B. DEVELOPING THE CLASSIFIER**

We encountered two limitations during the development of the classifier. The first is the limited amount of data...
available for training. As stated previously, all training data were obtained by manually correcting the repair algorithm. We maintained a 1:1 ratio for malicious and legitimate classes when creating the training dataset, and we corrected approximately 0.3% of the total dataset, as shown in Table 2. The second limitation is related to the number of features available to construct the dataset. Here, the objective was to evaluate how well an entry fits inside a browsing session; however, the collected data do not provide many features that can be exploited for this task.

We evaluated multiple classifiers: support vector machine (SVM), logistic regression (LogReg), gradient boosting (XGB), stochastic gradient descent (SGD), random forest (RF), and K-nearest neighbors (KNN). The results are shown in Table 4, where the base dataset contains all features and data entries, the “No outlier” dataset removed all statistical outliers, the pruned dataset removed features with low importance, and “Pruned No outlier” represents a combination of the previous two methods. As can be seen, the KNN and XGB classifiers outperformed the other algorithms. Note that the XGB algorithm was optimized using a random grid search. Then, a voting classifier was employed to combine the KNN and optimized XGB results using the weighted mean. The voting classifier was also optimized using random grid search to determine the optimal ratio for the weights.

**TABLE 3. Classifier features.**

| Feature name | Description |
|--------------|-------------|
| before_j, after_j | Jaccard indexes for both entries |
| before_s, after_s | Sorensen-Dice coefficients for both entries |
| before_l, after_l | Levenshtein distances for both entries |
| before_d, after_d | SequenceMatcher differential for both entries |
| before_t, after_t | Time differences for both entries |

**TABLE 4. Accuracy of classifiers for various datasets.**

| Dataset | SVM | LogReg | SGD | RF |
|---------|-----|--------|-----|----|
| Base    | 0.9058 | 0.8643 | 0.8649 | 0.8941 |
| Base - No SMOTE | 0.8682 | 0.8327 | 0.8327 | 0.8129 |
| Pruned | 0.8419 | 0.8495 | 0.8412 | 0.8163 |
| Pruned - No SMOTE | 0.8305 | 0.8257 | 0.8194 | 0.8460 |
| No outlier | 0.7528 | 0.8405 | 0.8374 | 0.9229 |
| No outlier - No SMOTE | 0.8683 | 0.8527 | 0.8527 | 0.8529 |
| Pruned No Outlier | 0.8537 | 0.8105 | 0.8175 | 0.8525 |
| Pruned No Outlier - No SMOTE | 0.8662 | 0.8357 | 0.8379 | 0.8489 |

**TABLE 5. Evaluation of proposed repair scheme.**

| Similarity metric | Accuracy | Duration (~2M rows) | Number of browsing sessions |
|-------------------|----------|----------------------|-----------------------------|
| Jaccard           | 0.8550   | 46s                  | 371263                      |
| Sorensen           | 0.8530   | 46s                  | 371363                      |
| Levenshtein        | 0.8517   | 55s                  | 371392                      |
| SequenceMatcher    | 0.8509   | 30s                  | 371184                      |

The class instances of the dataset were balanced using the synthetic minority oversampling technique (SMOTE). The imbalanced-learn Python library [43] was used. The evaluation of the classifiers was complemented using both feature selection and outlier removal. Here, feature selection was implemented using extremely randomized trees [44], and features below an arbitrary threshold of 0.01 were removed. These outliers were removed from the dataset using the isolation forest algorithm [45]. The scores of the classifiers were averaged over 30 iterations to consider randomness in the training process.

**C. EXPERIMENTAL RESULTS**

The classifier selected for the evaluation was an optimized XGB algorithm combined with the KNN classifier using the dataset, even though the pruned dataset obtained better scores (Table 4). In fact, the pruned dataset only offered a 0.21% improvement, which can be attributed to the low volume of training data. As shown in Table 5, the proposed repair scheme achieved 95% accuracy, where all metrics performed similarly. We considered this to be an acceptable accuracy level to proceed with the case study (Section V). The dataset was selected using the Jaccard index because of its best performance.

The feature pruning process removed the type of threat and the multiple redirect features from the web accesses. These features are not expected to impact the reconstruction process because they are relevant to security incidents. However, the access type was retained, which is reasonable because there should be continuity in access types within a single web-browsing session. For the sake of comparison, Table 4 includes the performance of classifiers without resampling of the imbalanced class using SMOTE. During training, most samples were classified as inserted correctly, which improved the performance of even the worst performing classifiers. This is easily explainable because most training samples belonged to the majority class; thus, the classifiers would overfit. Similarly, an analysis of the outliers demonstrated that most of them belong to the majority class. Note that these outliers did not include the synthetic data generated using SMOTE, and approximately half of the outliers were incomplete entries reconstructed by the proposed scheme. Thus, half of the outliers belonged to complete data that was collected without issue.

The Jaccard index was the most accurate similarity metric among the compared metrics; however, its accuracy was less than 1% better than the worst performing metric, i.e., the SequenceMatcher metric. In addition, the SequenceMatcher metric was the slowest to compute (by a factor of five compared to all other metrics). This was not an issue for the amount of data processed in the current study; however, it could become problematic if the scope of the evaluation was extended to also consider users who did not trigger the GSB sensor. This represents a factor of 20 per run of the algorithm. Finally, we observed that the number of browsing sessions...
was quite similar among the metrics, which implies a certain degree of stability during the splitting of the browsing session. The results are summarized in Table 5.

V. CASE STUDY
In the previous section, we discussed how the proposed scheme was employed to repair multiple browsing sessions containing security incidents that otherwise would not have been detected (Tables 8, 9 and 10). Here, we further analyze and discuss how some attacks can reach a user’s cell phone by inspecting browsing sessions, and we correlate web sessions with other elements to outline security incidents. The following discussion is divided into two primary topics, first we discuss examples of how users are originally compromised by mobile attacks (SMSShing or web-related malicious content), and then we highlight specific security incidents detected in our dataset by demonstrating how web sessions can be correlated with SMS or applications installations.

A. ENTRY POINTS
The objective was to identify the starting point of a browsing session that led a web user to malicious content. The sessions reconstructed using Algorithm 2 were analyzed, and we found that the browsing sessions can be long (e.g., they could last over a couple of days). It is highly unlikely for a user to browse the Internet for multiple days without stopping; thus, we assumed that the same browser tab was used multiple times. Some of these long sessions could be split using certain strings captured by the sensors as delimiters. For example, long sessions could be split using strings corresponding to home pages or the opening of a new window, e.g., “Search or type web address” or “about:blank”. These sentences may appear when a user opens a new tab or presses the home button. During our investigation, we considered the fact that some users have changed their homepage. In fact, multiple sessions contained visits to the Yahoo domain (*.yahoo.co.jp), and 17611 sessions started on a Yahoo page. We believe that more homepage addresses could be discovered by conducting more extensive statistical and contextual analyses.

There are two main ways for a cell phone user to compromise their sensitive data, i.e., web content and SMS attacks. Whenever a user receives an SMS, the sensor parsed the content of the message, and if the text contains one or more URLs, they would be recorded. Thus, if the user clicked one of these URLs, an extra entry was generated by the sensor when the browser accessed the website. However, a common SMSShing attack employs shortened URLs to trick the user into clicking it. In addition, the multiple redirections that may occur could also evade the sensor, thereby making it difficult to detect the actual consequences of an attack. Note that the ability to correctly capture multiple redirections is an ongoing aspect of the sensor’s development.

As described in Section III-A, the web accesses captured by the sensor can be performed using Chrome, Custom Tabs [32], or Unknown. Thus, both the Chrome browser and mobile applications are monitored by the sensor. Table 6 summarizes the tab distributions for both legitimate and malicious accesses. As can be seen, Chrome accesses are nearly equally represented among all entries and malicious entries, whereas SMSs represent only a small fraction (11%) of malicious accesses in the entire dataset.

GSB identifies three types of threats in the dataset, i.e., Malware, Social_Engineering, and Unwanted_Software. Table 7 summarizes the distribution between threats and legitimate entries for both web accesses and links in the SMS messages. We found that no SMS message triggered the Unwanted_Software flag; however, the data sample was too small to be sure that SMS messages are not commonly used to share such software. In addition, malware represents the majority of malicious content found in SMS messages. In terms of web entries, we found that social engineering content makes up the majority of attacks, which is in opposition to the trend observed with SMSs. As will be discussed in Section V-B, if a user is going to install malware received through an SMS, they will typically do that shortly after receiving the message.

B. CORRELATING WEB SESSIONS
In this section, we discuss how different dataset elements are correlated to investigate the attacks made on users. In [5], the authors stated that they could not evaluate the actual infection rate due to the limitations of their dataset. Thanks to the user-side collection process used to construct our dataset, it is possible to overcome some of these limitations. Note that we do not intend to present a detection methodology for mobile attacks. The purpose of the following is to complement the discussion about the impacts of attacks on cell phone users. This decision was motivated by the fact that only GSB was used by the sensor to classify the entries as malicious or legitimate.

1) SOCIAL ENGINEERING ATTACKS
Social engineering attacks cover any attack that targets a user’s personal information using deception. Here, we considered attacks coming from either an SMS or typical web-browsing activities. Note that the technical and ethical
TABLE 8. Malware installation after SMShing attack.

| SMS Link                | Threat          | App name               | Δ, 0? |
|-------------------------|-----------------|------------------------|-------|
| tvw8ikzje dresserdns.org| MALWARE         | com.jsyw.muhk          | 22.0  |
| knkhe.com               | SOCIAL_ENGINEERING | com.jsyw.xrfg          | 15.0  |
| djimuqncv.dussdons.org  | SOCIAL_ENGINEERING | sqyx.kxqen.mahy       | 51.0  |

dlimitations of this study (as described in Section VI-E) did not allow collection of form data transmitted by users. Thus, we could not verify which data may have been stolen from the user. Depending on the construction of a social engineering attack, the risk of a user compromising their sensitive data increases if they visit multiple pages identified by GSB as “SOCIAL_ENGINEERING” from the same domain. A browsing session can also be inspected manually by an expert to assess the potential severity of such attacks. Table 10 shows an excerpt from the proposed session-repair scheme. Here, a user visited what appears to be a counterfeit of the Amazon website and reached a page where credit card information has been entered.

2) MALWARE INSTALLATION
Exposure to malware is difficult to evaluate using only log analysis. We traced the installation of malware by identifying installation events shortly after a user visits a malicious link (see the Δ, column in Table 8). Despite the limited amount of available samples in the dataset, we found that users generally take less than one minute to open a malicious SMS and install the malware.

3) ANTIVIRUSES
Interventions by mobile phone antivirus software can be detected by monitoring a user’s browsing session. Table 9 shows that a user was redirected to a localhost page using the Kaspersky (KIS) antivirus, thereby preventing the user from reaching a malicious website. An analysis of the dataset did not show that any antivirus application was active when browsing malicious websites, which can be explained by the fact that redirection to a warning page is automatic and does not incur a change in focus from the browser to the antivirus software.

VI. DISCUSSION
In this section, we discuss the different assumptions and limitations encountered during the repair and exploration of the dataset.

A. LIMITATIONS REGARDING THE GENERATION OF TRAINING DATA
The ability to automatically label the training data of the proposed repair scheme suggests that the scheme already includes a baseline it can use for reference. The raw data do not provide such baseline, and during our reflection on this issue, we concluded that a form of unsupervised learning would be required to cluster entries. However, whether those clusters would capture the correctness of the proposed repair scheme would still require manual intervention to evaluate the clusters. Thus, we evaluated the correctness of the proposed repair scheme directly.

B. LIMITATIONS REGARDING MISSING INFORMATION
The main limitation of the proposed repair scheme is that it does not differentiate between incorrectly inserted single entries and the set of entries belonging to a different browsing session that was not captured. The reasons for missing information in this other session may vary. For example, the user may have not authorized the sensor to record this information or the sensor failed to capture the event. Thus, we may consider orphan entries as a block of entries rather than inserting them individually in multiple sessions. This could produce more meaningful sessions with less noise during the data repair process.

C. ASSUMPTION ON URL SIMILARITY
In Section III-C1, it was assumed that the similarity between two URLs was sufficient to assume continuity in a browsing session. Similarly, two similar URLs may indicate that the corresponding webpages may include similar topics. However, this assumption should be reinforced using third-party tools, e.g., SimilarWeb [46], that provide contextual analyses and related key words to achieve better evaluations of the semantic similarity between webpages.

D. DEFINITION OF MALICIOUS CONTENT
The scope of the current study was limited to defining malicious content based on the output of GSB. However, GSB may not be the optimal choice for detecting malicious content. In fact, it has been pointed out [5] that GSB can take some time to include a reference in their database. Confidentiality issues were also outlined using VirusTotal to analyze the content due to the permanent recording of the submitted data. However, VirusTotal can be used to analyze a URL’s domain without including specifics, e.g., subdomains or parameters in the URL. We used VirusTotal manually to verify the web domain in an SMS link and found that a user had installed malware, which was then used to remotely control their smartphone.

E. LIMITATIONS REGARDING THE DATASET
The dataset constructed in this study also has some limitations. First, the total number of users was lower compared to that reported in [5]. In addition, a smaller percentage of users were victims of mobile attacks. Note that we obtained a response rate similar to those reported in [8]. Our user base also suffered from the same bias, as the communication campaign for the mobile sensor was similar that run by [8]. In addition, some domains were intentionally not recorded by the sensor (e.g., accesses made to bank-related websites or email accounts). Privacy concerns were also considered carefully throughout the course of this study, and users may arbitrarily decide to exclude specific domains from the scope of the sensor.
TABLE 9. Antivirus software intercepting a malicious webpage.

| Access type | URL                                         | GSB Threat       | Date              |
|-------------|---------------------------------------------|------------------|-------------------|
| Chrome      | rekutan.<DELETED>.top/rms/nid/index.html    | SOCIAL_ENGINEERING | 2020-09-29 08:31:41 |
| Chrome      | rekutan.<DELETED>.asia                      | SOCIAL_ENGINEERING | 2020-09-29 08:31:43 |
| Chrome      | 127.0.0.1:40425/kis/blocked.html             | SOCIAL_ENGINEERING | 2020-09-29 08:31:43 |
| Chrome      | 127.0.0.1:40425/kis/permission_denied.html   | SOCIAL_ENGINEERING | 2020-09-29 08:31:44 |

TABLE 10. Successful phishing attack.

| Access type | URL                                           | GSB Threat       | Date              |
|-------------|-----------------------------------------------|------------------|-------------------|
| SMS         | d-amazon.top                                  | SOCIAL_ENGINEERING | 2020-09-26 13:50:19 |
| Chrome      | d-amazon.top                                  | SOCIAL_ENGINEERING | 2020-09-26 13:50:36 |
| Chrome      | d-amazon.top/amazon/                          | SOCIAL_ENGINEERING | 2020-09-26 13:50:37 |
| Chrome      | d-amazon.top/amazon/login/login.php           | SOCIAL_ENGINEERING | 2020-09-26 13:50:39 |
| Chrome      | d-amazon.top/amazon/login/login.php?id=<USER_DELETED> | SOCIAL_ENGINEERING | 2020-09-26 13:50:43 |
| Chrome      | d-amazon.top/amazon/update_billing/update_billing | SOCIAL_ENGINEERING | 2020-09-26 13:50:45 |
| Chrome      | d-amazon.top/amazon/login/login.php?id=<USER_DELETED> | SOCIAL_ENGINEERING | 2020-09-26 13:50:47 |
| Chrome      | d-amazon.top/amazon/update_billing/update_billing | SOCIAL_ENGINEERING | 2020-09-26 13:51:19 |
| Chrome      | d-amazon.top/amazon/Update_Your_Card/Card.php?id=<USER_DELETED> | SOCIAL_ENGINEERING | 2020-09-26 13:51:20 |
| Chrome      | d-amazon.top/amazon/Update_Your_Card/Card.php | SOCIAL_ENGINEERING | 2020-09-26 13:53:45 |

The sensor itself also suffers from some technical limitations. The Android accessibility service can only capture information that is visible to users. Thus, it may be impossible to record the current URL because it might not be displayed to the user, thereby leading to an empty entry in the logs. In addition, the computational power required to accurately record each entry may be at fault because some entries may not have been recorded or may have been recorded twice.

F. CURRENT LIMITATIONS OF THE PROPOSED APPROACH

Using the proposed approach, browsing sessions were reconstructed, and the dataset was investigated manually to identify existing markers that indicate the beginning of a new session. However, it appears that these markers are insufficient to divide long sessions. Although the median numbers shown in Table 5 are coherent with what can be expected of mobile browsing (several seconds per page), there was sufficient data to increase the average numbers within the range of several hours, which is less realistic.

Another limitation is the lack of referrer information when a user navigates through a link. If the domain remains the same, the assumption that a user clicked somewhere on the webpage is true. In contrast, if the user was redirected to a partner site or if a pop up was loaded and the user reached a completely different page, the repair process will be less accurate. Thus, the ability to identify context regarding each website could help determine the type of entries the proposed repair scheme can process.

VII. CONCLUSION

In this study, a sensor was deployed on cell phones to collect Internet browsing behavior and application usage. Data are sometimes improperly collected; thus, a repair scheme was also developed to infer missing attributes of web-browsing elements. The performance of the proposed scheme was evaluated using a machine learning classifier, and the results demonstrated that the proposed scheme obtained an accuracy of 95%. The classifier was trained by manually correcting the samples of the reconstructed data, which allowed us to gain insights about how to improve the proposed repair scheme and the security of mobile users. More precisely, we found that some limitations in the data collection process may prevent us from obtaining insights into a user’s behavior, which should be investigated further in future work. Finally, we demonstrated the usability of the proposed repair scheme by performing an exploratory data analysis, and we outlined risky behaviors leading to successful attacks, e.g., malware installation via an SMSShing attack and the leakage of credit card details via a phishing website. This analysis was made possible by using the proposed repair scheme and recovering missing entries in the web-browsing session information.

REFERENCES

[1] Number of Mobile Users. Accessed: Aug. 2, 2022. [Online]. Available: https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/
[2] Mobile OS Share Worldwide. Accessed: Aug. 2, 2022. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide
[3] E. O. Yeboah-Boateng, E. Osei, and P. M. Amanor, “Phishing, SMiShing & vishing: An assessment of threats against mobile devices,” J. Emerg. Trends Comput. Inf. Sci., vol. 5, no. 1, pp. 297–307, 2014.
[4] K. Chen, P. Wang, Y. Lee, X. Wang, N. Zhang, H. Huang, W. Zou, and P. Liu, “Finding unknown malice in 10 seconds: Mass vetting for new threats at the Google-play scale,” in Proc. 24th USENIX Secur. Symp. (USENIX Secur.), 2015, pp. 659–674.
[5] M. Sharif, J. Urakawa, A. Kubota, and A. Yamada, “Predicting impending exposure to malicious content from user behavior,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2018, pp. 1487–1501.
[6] Warprive Project. Accessed: Aug. 22, 2022. [Online]. Available: https://warprive-project.git/
[7] Android Accessibility Service. Accessed: Aug. 2, 2022. [Online]. Available: https://developer.android.com/reference/android/accessibilityservice/AccessibilityService
[8] T. Takahashi, C. Kruegel, G. Vigna, K. Yoshioka, and D. Inoue, “Tracing and analyzing web access paths based on user-side data collection: How do users reach malicious URLs?”, in Proc. 23rd Int. Symp. Res. Attacks, Intrusions Defenses (RAID), 2020, pp. 93–106.
[9] D. Canali, L. Bilge, and D. Balzarotti, “On the effectiveness of risk prediction based on users browsing behavior,” in Proc. 9th ACM Symp. Inf., Comput. Commun. Secur., 2014, pp. 171–182.
[10] G. Stringhini, C. Kruegel, and G. Vigna, “Shady paths: Leveraging surfing crowds to detect malicious web pages,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2013, pp. 133–144.

[11] D. Bank and D. Yamin, “Predicting cyber-infections via web-browsing patterns,” Tech. Rep., 2020, doi: 10.2139/ssrn.3757877. [Online]. Available: https://ssrn.com/abstract=3757877

[12] F. L. Lévesque, J. M. Fernandez, and A. Somayaji, “Risk prediction of malware victimization based on user behavior,” in Proc. 9th Int. Conf. Malicious and Unwanted Softw., Amer. (MALWARE), Oct. 2014, pp. 128–134.

[13] J. Szurdi, M. Luo, B. Kondracki, N. Nikiforakis, and N. Christin, “Where are you taking me? Understanding abusive traffic distribution systems,” in Proc. Web Conf., 2021, pp. 361–3624.

[14] G. Kovacs, “Reconstructing detailed browsing activities from browser history,” 2021, arXiv:2102.03742.

[15] S. Bhagavatula, L. Bauer, and A. Kapadia, “What breach? Measuring online awareness of security incidents by studying real-world browsing behavior,” in Proc. Eur. Symp. Usable Secur., 2021, pp. 180–199.

[16] Google Safe Browsing. Accessed: Aug. 22, 2022. [Online]. Available: https://safebrowsing.google.com/

[17] PhishTank. Accessed: Aug. 22, 2022. [Online]. Available: http://phishTank.org/

[18] C. Kolbitsch, B. Livshits, B. Zorn, and C. Seifert, “Rozzle: De-cloaking Internet malware,” in Proc. IEEE Symp. Secur. Privacy, May 2012, pp. 443–457.

[19] D. Canali, M. Cova, G. Vigna, and C. Kruegel, “Prophiler: A fast filter for the large-scale detection of malicious web pages,” in Proc. 20th Int. Conf. World Wide Web, 2011, pp. 197–206.

[20] R. Verma and A. Das, “What’s in a URL: Fast feature extraction and malicious URL detection,” in Proc. 3rd ACM Int. Workshop Secur. Privacy Anal., 2017, pp. 55–63.

[21] DMOZ Open-Directory Project. Accessed: Aug. 2, 2022. [Online]. Available: http://dmoz.org/

[22] Anti-Phishing Work Group. Accessed: Aug. 2, 2022. [Online]. Available: http://apwg.org/

[23] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain,” Psychol. Rev., vol. 65, no. 6, p. 386, 1958.

[24] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, “Beyond blacklists: Learning to detect malicious web sites from suspicious URLs,” in Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2009, pp. 1245–1254.

[25] M. A. Harris, K. Patten, and E. Ragan, “The need for BYOD mobile device security awareness and training,” in Proc. 19th Americas Conf. Inf. Syst. (AMCIS), vol. 5, Jan. 2013, pp. 3441–3451.

[26] F. Alotaibi, S. Furnell, I. Stengel, and M. Papadaki, “A survey of cybersecurity awareness in Saudi Arabia,” in Proc. 11th Int. Conf. Internet Technol. Secured Trans. (ICTIST), Dec. 2016, pp. 154–158.

[27] R. Bitton, A. Finkelshtein, L. Sidi, R. Puzis, L. Rokach, and A. Shabtai, “Taxonomy of mobile users’ security awareness,” Comput. Secur., vol. 73, pp. 266–293, Mar. 2018.

[28] A. Forget, S. Pearman, J. Thomas, A. Acquisti, N. Christin, L. F. Cranor, S. Egelmann, M. Harbach, and R. Telang, “Do or do not, there is no try: User engagement may not improve security outcomes,” in Proc. 12th Symp. Usable Privacy Secur. (SOPUS), 2016, pp. 97–111.

[29] F. L. Levesque, J. Nsniemja, J. M. Fernandez, S. Chiasson, and A. Somayaji, “A clinical study of risk factors related to malware infections,” in Proc. ACM SIGSAC Conf. Comput. Commun. Secur., 2013, pp. 97–108.

[30] C. Mombidzwa, L. Spalazzi, A. Teti, and A. Cucchiarelli, “Leveraging n-gram neural embeddings to improve deep learning DGA detection,” in Proc. 57th ACM/SIGAPP Symp. Appl. Comput., 2022, pp. 995–1004.

[31] D. Lazar, K. Cohen, A. Freund, A. Bartik, and A. Ron, “IMDoC: Identification of malicious domain campaigns via DNS and communicating files,” IEEE Access, vol. 9, pp. 45242–45258, 2021.

[32] Custom Tabs. Accessed: Aug. 2, 2022. [Online]. Available: https://developer.chrome.com/docs/android/custom-tabs/overview/

[33] VirusTotal. Accessed: Aug. 2, 2022. [Online]. Available: https://virustotal.com/

[34] S. Kim, J. Kim, and B. B. Kang, “Malicious URL protection based on attackers’ habitual behavioral analysis,” Comput. Secur., vol. 77, pp. 790–806, Aug. 2018.

[35] M. Zounia and B. Ouatj, “A novel lightweight URL phishing detection system using SVM and similarity index,” Hum.-Centric Comput. Inf. Sci., vol. 7, no. 1, pp. 1–13, 2017.

[36] J. Nowak, T. Holotyak, M. Korytkowski, R. Scherer, and S. Voloshynovskiy, “Fingerprinting of URL logs: Continuous user authentication from behavioural patterns,” in Proc. Int. Conf. Comput. Sci. Cham, Switzerland: Springer, 2020, pp. 184–195.

[37] U. Meyer and V. Drury, “Certified phishing: Taking a look at public key certificates of phishing websites,” in Proc. 15th Symp. Usable Privacy Secur. (SOPUS), 2019, pp. 211–223.

[38] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Sov. Phys.-Dokl., vol. 10, no. 8, pp. 707–710, 1966.

[39] T. A. Sotenes, “A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on Danish commons,” Biol. Skr., vol. 5, pp. 1–34, May 1948.

[40] P. Jaccard, “Distribution de la flore Alpine dans le Bassin des Dranses et dans quelques regions voisines,” Bull. Soc. Vaudoise Sci. Nat., vol. 37, pp. 241–272, Jan. 1901.

**DiffLib SequenceMatcher. Accessed: Aug. 2, 2022. [Online]. Available: https://docs.python.org/3/library/difflib.html

[41] J. W. Ratcliffe and D. E. Metzener, “Pattern-matching-the gestalt approach,” Dr Dobbs J., vol. 13, no. 7, p. 46, 1988.

[42] Imbalanced Learn. Accessed: Aug. 2, 2022. [Online]. Available: https://imbalanced-learn.org/stable/

[43] P. Geurts, D. Ernst, and L. Wehenkel, “Extremely randomized trees,” Mach. Learn., vol. 63, pp. 3–42, Apr. 2006.

[44] F. L. Liu, K. M. Ting, and Z.-H. Zhou, “Isolation forest,” in Proc. 8th IEEE Int. Conf. Data Mining, Dec. 2008, pp. 413–422.

[45] Similarweb, Website Analysis. Accessed: Aug. 2, 2022. [Online]. Available: https://www.similarweb.com/fr/