IEICE TRANSACTIONS on Communications

VOL. E101-B NO. 9
SEPTEMBER 2018

A PUBLICATION OF THE COMMUNICATIONS SOCIETY
The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, TOKYO, 105-0011 JAPAN

PAPER

Efficient Approach for Mitigating Mobile Phishing Attacks

Hyungkyu LEE[†a)], ***Nonmember***, **Younho LEE**[††b)], ***Member***, **Changho SEO**[†††c)], ***and*** **Hyunsoo YOON**[†d)], ***Nonmembers***

SUMMARY We propose a method for efficiently detecting phishing attacks in mobile environments. When a user visits a website of a certain URL, the proposed method first compares the URL to a generated whitelist. If the URL is not in the whitelist, it detects if the site is a phishing site based on the results of Google search with a carefully refined URL. In addition, the phishing detection is performed only when the user provides input to the website, thereby reducing the frequency of invoking phishing detection to decrease the amount of power used. We implemented the proposed method and used 8315 phishing sites and the same number of legitimate websites for evaluating the performance of the proposed method. We achieved a phishing detection rate of 99.22% with 81.22% reduction in energy consumption as compared to existing approaches that also use search engine for phishing detection. Moreover, because the proposed method does not employ any other algorithm, software, or comparison group, the proposed method can be easily deployed.

key words: *mobile phishing, mobile phishing detection, phishing detection, mobile security, security*

1. Introduction

Mobile phishing is a phishing attack that occurs when a victim is using a mobile device. A phishing attack is a social engineering attack, where attackers can obtain any important private information of victims, such as credit card numbers and their PIN numbers, IDs and their passwords for important services, and their social security numbers, through fake websites of the attackers. APCERT Annual Report 2016 [1] says that 42.3% of the victims of cyber-attacks were plagued by phishing attacks in China.

Mobile phishing attacks are not currently in the spotlight because of other security vulnerabilities in the smartphone environment that enable attackers to implant their malicious codes into the smartphones of the victims in a considerably easier manner than through mobile phishing attacks. However, if the attacks become widespread, mobile security will be adversely affected. According to Lookout [2] and RSA Conference [3], the probability of success for a phishing attack on a mobile user is three times higher than that on a PC user.

The frequent use of smartphones is one of the main reasons users are vulnerable to mobile phishing. Unlike PCs, smartphones tend to be close to users at all times. According to Fluent [4], smartphones dominate other devices for email, with 2 in 3 Americans saying it's how they most often check their messages. Also, people who primarily check their emails as they arrive on their smartphones are twice more than people who check on their other devices. As emails are one of the most frequent methods used by phishing attackers, we can say users are more vulnerable to mobile phishing than to conventional phishing using PCs. Additionally, other methods for mobile phishing, including SMS ("smishing") and Social Network Service, increase the danger from mobile phishing attacks.

Secondly, small smartphone screens contribute to the vulnerability against phishing attacks. The most popular screen is 4.7-5.0 inch (as of 2017) [5], which is 1/5 the size of the average PC monitor. As a result, web pages for mobile devices are simplified, thereby containing only essential information. Mobile app developers also try their best to simplify app displays to fit the constraints of a small screen. Since the amount of information displayed on-screen is minimal, mimicking the display is easier, thereby simplifying a phishing attack. Moreover, this constraint leads to the prevalence of shortened URLs such as t.co and goo.gl, which hinders the ability of the users to use URLs for distinguishing phishing sites from legitimate sites.

Since the advent of phishing attacks, several studies have been conducted. The blacklist approach is one of the most common techniques in practice, and is used by almost all web browsers such as Microsoft Internet Explorer [6], Mozilla Firefox [7], and Google Chrome [8]. In this approach, users are warned if they access a site in the blacklist via a pop-up window or notification. One fundamental weakness in this approach is that, unless the blacklist is up-to-date, it will fail to identify some malicious websites while users are accessing them. As these malicious websites are considerably short-lived [9], they may disappear before they are reported. Another method is the heuristic approach, which analyzes the similarity of features (e.g., the content, position of images, and URL) between the website being checked

Manuscript received January 22, 2018.
Manuscript revised March 5, 2018.
Manuscript publicized March 23, 2018.

[†]The authors are with Division of Computer Science, School of Computing, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon, South Korea.
[††]The author is with ITM Division, Department of Industrial and Information Systems Engineering, SeoulTech, Nowon-gu, Seoul, South Korea.
[†††]The author is with Department of Applied Mathematics, Kongju National University, Gongju-si, Chungnam, South Korea.
a) E-mail: hklee@nslab.kaist.ac.kr
b) E-mail: younholee@seoultech.ac.kr (Corresponding author)
c) E-mail: chseo@kongju.ac.kr
d) E-mail: hyoon@kaist.ac.kr

DOI: 10.1587/transcom.2018EBP3020

and each of the websites popular for phishing targets. The ability to cope with zero-day attacks is a major advantage of this approach. In addition, it can achieve a high detection rate if the implementation employs good feature-extraction methods and classification algorithms. However, the methods of this type require a large amount of computation and communication resources; additionally, they suffer from relatively high false positive ratio as compared to the blacklist approach [10].

Unfortunately, all the approaches mentioned above are for PCs, and it is difficult to apply them to the mobile environment. The blacklist approach requires frequent updates of the list, thereby requiring heavy resource consumption. As phishing sites emerge endlessly [11], a local blacklist would occupy a large amount of storage space. It is also a weak defense against a zero-day attack. The heuristic approach requires various features of the target websites to be detected; however, mobile web pages are relatively simple. Moreover, heavy consumption is required for the extra softwares or algorithms such as optical character recognition (OCR) [12].

Thus, considerably few approaches have been proposed for anti-phishing in the mobile environment, thereby resulting in a situation where majority of the current apps that are capable of browsing the internet are not able to detect mobile phishing attacks. In our experiment, mobile web browsers could not detect even a single phishing site among the 8315 phishing sites tested, and this result is also verified by Tsalis et al's paper [13].

In this paper, we propose an anti-phishing approach for smartphones. Our techniques are threefold: First, we utilize the domains associated with the trustworthy apps in order to generate a mobile whitelist. We selected the top ranked apps by App Annie [14] as trustworthy apps because App Annie may be able to sustain its reputation unless it ensures the sanctity of the content hosted on its domain, as mentioned in [15]. Therefore, we can infer that the domains associated with them are not dangerous. This reduces the number of execution steps required for detecting phishing in the proposed scheme, thereby leading to lower consumption of battery power.

Second, we suggest a smart approach for deciding the mobile checkpoint (i.e., when the mobile phishing detection module is activated). Instead of checking every web page the user visits, only the web pages that take user input are checked. The nature of phishing attack sites requires the victim to first input valuable information. Thus, it is natural to treat web pages that do not require user inputs as no-phishing sites. We determined that an estimated 71.5% of web pages visited do not require text-based inputs from users[†]. Combining the first and second approaches decreases the amount of energy required. Our experiment confirmed that the above approaches decreased the power consumption by 81.22% on average, as compared to using an approach that does not consider this.

The third technique is a novel phishing detection method. Only a single query to the Google search engine is performed using a carefully crafted URL of the web page the user is visiting. The proposed method checks the domains of the URLs in the search results of Google and attempts to check if they are included in the list of the URLs in the websites that report phishing.

This approach is considerably simple, and it does not require complicated feature-extraction of the target website or complicated algorithms. Rather, it requires only one feature: checking the URL of a web page. Two network interactions are required. Based on the number of URLs in the search results of Google, and the resultant URLs from the phishing reporting website, we establish the following rules: in the case of zero search results, we conclude the target website is a zero-day phishing site. If one or more search results are observed, but the domains of all the resultant URLs are included in the phishing reporting website, the target is regarded as a phishing site. Except the above two cases, the target website is deemed a legitimate site. With such a simple approach, we can achieve considerably high phishing detection rates and a reasonably low false-positive rate. Additionally, because only a few network interactions are used, and no heavy computation or storage is required, the proposed approach requires considerably low power consumption, which is an essential feature for a mobile phishing approach.

In order to prove the feasibility of the proposed scheme, we installed the proposed method in a Google Nexus S phone by modifying the original source code for the phone and uploading the modified image into the phone. We used a phishing reporting website, PhishTank [18], which includes 8315 phishing sites collected within a month. The reported websites were found within 12 hours since their appearance. A total of 8315 target (legitimate) web pages were tested, using various character sets and languages.

In terms of the phishing detection rate, the proposed scheme (99.22%) was superior to the blacklist approach (54.91%), with the proposed scheme only incurring a 0.61% false positive rate. In terms of energy consumption, the proposed scheme required only an additional 15.22% power on average compared to the usage of no phishing detection. Because the proposed scheme does not use additional feature comparison, algorithms, and auxiliary software, we believe that the proposed scheme is superior to the heuristic approach in terms of power consumption.

As seen from above, our proposal is heavily dependent on Google search because we try to achieve the best performance with just one feature of the web page, URL. Thus, we assume that Google is a trustworthy and non-compromised third party. This premise leads to the additional advantages of the proposed scheme including effectiveness against zero-day and advanced phishing attacks such as cloaking, because it is aided by the Google search engine. It also functions without considering the character set used, since it does not

[†]This only deals with mobile search websites, which are the most-visited websites on smartphones [16]; additionally, 'Search' operation is expected to require more text-based inputs than other operations [17].

require any text information other than the URL.

The main contribution of this paper is the following:

- This research proposes an anti-phishing approach suitable for mobile environments, which utilizes the characteristics of mobile devices unlike PCs.
- It is shown the outstanding effectiveness of the proposed approach with the minimum feature use in web pages, just URLs in detecting phishing.
- It is shown the efficiency of our approach by figuring out the real power usage on a real smartphone when the proposed work is running.
- It is shown the analysis result of the experiments to show the effectiveness of the proposed work against zero-day phishing attacks which are most likely to occur in mobile environments.

The rest of this paper is organized as follows. In Sect. 2, we review the related work. In Sect. 3, we address the problem of existing anti-phishing approaches in concrete terms. In Sect. 4, we explain the operating steps of our proposed system, and the proposed scheme to detect mobile phishing using a search engine. Section 5 shows the experiments conducted for testing the accuracy and efficiency of our proposed method. We discuss and conclude our paper in Sects. 6 and 7.

2. Related Works

Numerous studies have been found in the anti-phishing area, including CANTINA [19], hybrid phish detection [20], GoldPhish [21], Huh et al's approach [22], Barraclough et al's approach [23], Thakur et al's approach [15], Abdelhamid et al's approach [24], Moghimi et al's approach [25], Abbasi et al's approach [26], Jain et al's approach [27], Srinivasa Rao et al's approach [28], and Raffetseder et al's approach [29].

CANTINA [19] focuses on the contents of web pages. It extracts keywords from the content in the target website using the TF-IDF algorithm [30]; then, it searches for legitimate websites containing the extracted keywords. Finally, it determines if the target website is phishing based on the domains contained in the search results. It additionally utilizes the age of the domain the target website belongs to, known images, and the degree of suspiciousness of the URL in order to reduce the false positive rate of the approach. This is the first approach in employing public search engines for determining phishing sites. Hybrid phish detection [20] enhanced CANTINA by the addition of a novel method called "identity-based detection component". It recognizes the structure of the target website's HTML documents with a DOM tree. Then, the method determines if it is a phishing site by extracting important features such as the title, copyright information, or keywords by TF-IDF. Even with a smaller number of features, this method was shown to perform better than CANTINA. This method also supports login form detection, which is used for identifying a phishing site by the knowledge that phishing sites normally require sensitive, private information such user ID and password. However, both it is not effective using CANTINA and hybrid phish detection for mobile devices, because TF-IDF does not function well in mobile environments, as we will show further in this paper. Moreover, CANTINA may miss many phishing sites, considering the current trend of phishing sites of using ordinary domain names similar to conventional websites.

GoldPhish [21] detects phishing sites based on images on the web pages without requiring the investigation of the HTML code. Text is extracted from the images using optical character recognition (OCR), and used for building features. The main advantage of this approach is that it can detect phishing sites when only images exist. However, extracting text from images requires heavy cost in resources [12]; additionally, if the images have non-English letters, they are considerably difficult to extract.

Huh et al's approach [22] uses a well-known search engine such as Yahoo, Bing and Google, and full URLs as a query. By applying a classification algorithm to the result, it generates a URL ranking. Based on that ranking, the method determines if the target website is a phishing site. This approach detects phishing more effectively with various search engines. Unfortunately, it used a considerably small sample set in the experiment, making it difficult to verify its efficacy. Barraclough et al's approach [23], Thakur et al's approach [15], Abdelhamid et al's approach [24], and Moghimi et al's approach [25] use machine learning or data mining algorithms such as neural network (NN), C4.5 decision tree, associative classification (AC), and Support Vector Machine (SVM), respectively. Abbasi et al's approach [26] uses a kernel-based method, the genre tree kernel. They increase the predictability of phishing with various algorithms or methods. They, however, require huge amount of training data in order to guarantee a sufficient level of accuracy.

Jain et al. [27] proposed whitelist-based approach aiming for fast access time and high detection rate. In Srinivasa Rao et al's approach [28], it is decided whether the visited web page is phishing or not, based on the result of submitting fake credentials to the web page. Both approaches use a hyperlink-checking method which checks if the domain of the current web page is one of the domains of its hyperlinks or not, which is one of the representative characteristics of phishing. However, if some attackers create the hyperlinks which have the same domain with phishing sites, both approaches have some troubles to detect phishing. Raffetseder et al's approach [29] uses a hooking technique for anti-phishing similar to the proposed method. However, the type of phishing it can detect is limited to JavaScript attacks, and also, the method was not designed to consider mobile environments.

There are few methods that are specific to anti-phishing in mobile environments. In Han et al's approach [31], genuine Login User Interfaces (LUIs) and IP addresses of websites are stored externally via Bluetooth. The method detects phishing by comparing the LUI of the target website with the stored LUIs. The cons of this approach is that all LUIs of the target websites should be stored in advance, so the URL of any website not stored in advance cannot be checked.

Niu et al. [32] have proven that phishing attacks can occur more easily in the mobile environment than in the desktop environment based on a vulnerability analysis of both environments. They suggested methods to reduce the possibility of success of phishing attacks, including anti-phishing proxy. However, this approach has weakness in that that the proxy must be checked to confirm if it is trustworthy or not. Felt et al [33] classified the methods for mobile phishing into four types based on a statistical analysis regarding the types of transferred controls. This classification reflects real-world phishing attacks well. Unfortunately, their work provided no detailed descriptions of the anti-phishing techniques. Vidas et al [34] analyzed the possibility of QR codes being used to access phishing sites. It is interesting that they focus on the QR code which is one of the new types of inputs for mobile devices. However, the research is limited in that people use various methods other than QR codes for accessing websites, and no anti-phishing method was provided.

Xu et al. [35] have found that some phishing attacks use notifications on various mobile platforms. They also proposed a mitigation method. ScreenPass [36] blocks password theft and software keyboard spoofing based on OCR. Marforio et al. [37] showed personalized security indicators which can help users to detect phishing in mobile platforms. Because the target attacks in these approaches are phishing attacks using malicious mobile apps that has the ability to provide spoofed personal information of the mobile phone holder to those methods, it is questionable that these approaches would function well against phishing attacks within normal apps that support conventional web browsing. Unlike these approaches, the proposed approach can cope with that situation.

Recently, Wu et al. [38] proposed an approach to combat both mobile web phishing and mobile app phishing, each of which has unique characteristics. In terms of efficiency, it only utilizes the search engine and the OCR algorithm. Unfortunately, because of the heavy energy consumption of the OCR algorithm [12], it is not suitable for mobile environments. MP-Shield [39] combines both blacklist technique and heuristic-based approach, implemented as a proxy service on top of the TCP/IP stack in mobile devices, with the aim of inspecting IP packets for phishing content. They suggested hybrid approach taking the advantages of the blacklist technique and the approach using machine learning. However, it requires data training for classifying web pages and relies on a set of machine learning algorithms that are implemented in the WEKA (Waikato Environment for Knowledge Analysis) framework, such as J48, BayesNet or SMO. NativeWrap [40] suggested a method of thwarting phishing attacks that occurs in mobile websites:making the apps with which the target mobile websites can access. Because users can access those mobile websites through the created apps, many attacks such as phishing can be prevented. Unfortunately, phishing attack detection is dependent on the users' attention.

3. Problem Definition

As we investigated in the previous section, there have been considerably few approaches dealing with mobile phishing attacks. Additionally, approaches for phishing in the desktop environment are often impossible to use in the mobile environment. On desktop PCs, users access web pages through web browsers, making it possible to use the blacklist or heuristic approach for anti-phishing by attaching an add-on that performs phishing detection when a novel web page is loaded into the web browser. However, in the mobile environment, any application can provide web browsing capabilities with the use of WebView [41] in Android or UIWebView [42] in iOS. Thus, the anti-phishing approach should function for every application in the mobile environment, requiring the help of all app developers, which is difficult to achieve. Another approach is to obtain support from the platforms. Unfortunately, Google is still not taking any action against mobile phishing attacks (as of Mar. 2017), and Apple only applies the blacklist approach to its Safari web browser; therefore, apps using UIWebView are not protected by Apple's phishing detection.

The second problem relates to the use of battery as a main source of power. In the desktop environment, anti-phishing approaches do not consider power consumption because an unlimited amount of power is assumed. However, any mobile anti-phishing approach should use small amount of power as possible since the supported power is considerably limited. This limitation causes a difficulty in the application of approaches, which check phishing whenever the user visits a web page. Furthermore, this shortage of power makes us difficult to utilize heuristic-based approaches, which requires checking as many features as possible, leading to heavy computation. Moreover, some effective heuristic-based approaches use extra softwares such as OCR, or search engines such as Google or Bing. To use them, a significant power consumption is mandatory, affecting the battery life of a mobile device to a great extent [12], [43]. In Huh et al's approach [22], at least 1857 mJ additional power for three HTTP communications with Google, Bing and Yahoo search engine are used for checking a target website in our experiment.

Third, the user interface of mobile devices contains less information than that of desktop devices. On average, a smartphone screen is considerably smaller than a desktop monitor. Thus, fewer features can be included on the smartphone screen. Moreover, some features cannot be extracted or have different characteristics from those in the desktop environment. For example, the term frequency of "facebook" on the Facebook login page is five when accessing the web page in the desktop environment. In the mobile environment, it is two. In another instance, the logo image is one of the representative internal features of the Taobao web page. However, this cannot be checked in the mobile environment, where the logo is invisible on the login page. Therefore, we can conclude that a mobile page tends to have less meaningful information for anti-phishing than a web page for desktop PCs, indicating that fewer phishing detection methods will function, causing the mobile environment to be more vulnerable to phishing attacks.

Therefore, from the previous arguments, mitigating mobile phishing attacks is a nontrivial problem, and any solution should require a small amount of power consumption. To achieve this, we should maintain both network communications and the number of features to a minimum with no extra software, while preserving a high phishing detection rate. The solution also should support zero-day attack mitigation, as it is currently performed in the existing heuristic-based approaches. Our research objective is to propose a method that meets all the above requirements.

4. Proposed Approach

In order to enhance the performance in terms of both power consumption and phishing detection rate, we combine three proposed techniques. Figure 1 shows an overview of the proposed approach. We should check the URL of the target web page as input, which will hereafter be called the target URL. The first step is to check whether the web page of the target URL takes the text-based input from the user. The proposed scheme does not run the main phishing detection method if the web page does not take the user input, since phishing requires acquiring user inputs. The next step is to check if the target URL is whitelisted. If the target URL is included in the whitelist, no other procedures are executed, and the target URL is regarded as a non-phishing site. The main phishing detection starts functioning only when the target URL is not filtered in the above two steps. Thus, considerable power can be saved. In the main phishing detection routine, round-trip communication and some considerably simple string-related operations are required. Therefore, it does not consume considerable power. In addition, the proposed method can detect zero-day attacks and advanced phishing attacks using Google Search, one of the most powerful search engines in the world. As we mentioned in Sect. 1, we assume that Google is a trustworthy and non-compromised third party, and cloud-based approaches are out of our scope in the paper. Below, we explain each step in detail.

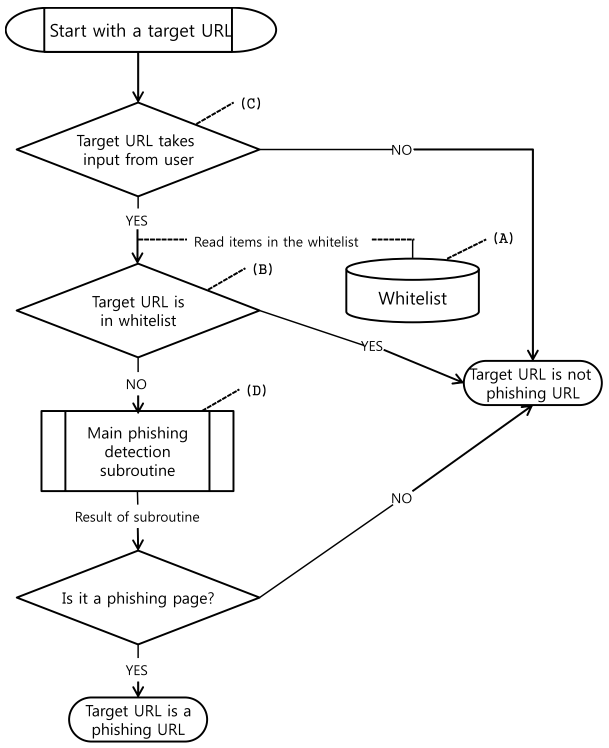

Start with a target URL
(C)
Target URL takes input from user
NO
YES
Read items in the whitelist
(A)
(B)
Whitelist
Target URL is in whitelist
YES
Target URL is not phishing URL
NO
(D)
Main phishing detection subroutine
NO
Result of subroutine
Is it a phishing page?
YES
Target URL is a phishing URL

Fig. 1 Overview of the proposed approach.

4.1 Whitelist Based Filtering

Algorithm 1 (A) Mobile whitelist construction

```
Require: Top ranked application by App Annie $app$
 1: procedure ConstructWhitelist($app$)
 2:     if $app$ has not sexual contents then
 3:         $d$ is the domain associated with $app$
 4:         if $d$ is not free hosting or free blogging domain then
 5:             if $d$ is not duplicated with any domain of whitelist then
 6:                 add $d$ to the whitelist
 7:             end if
 8:         end if
 9:     end if
10: end procedure
```

This step compares the input target URL with the trusted domain names stored in the whitelist. If there is a match between the URL and a trusted domain, we conclude that the target URL is not a phishing. To construct a trusted mobile whitelist, we investigated a list of top 16,000 applications: top 500 free apps and top 500 paid apps in the most popular 16 countries, on Google Play ranked by App Annie [14] on January, 2017. Of these, we filtered the apps that are duplicated and contained sexual contents. Finally, the remaining apps were only 9087. Based on these apps, we collected domains associated with these apps by investigating the domain used when they are handled on the web, the domain of the developer's company, or the domain of the first loaded URL if they have WebView format. Of these, we excluded free hosting or free blogging domains that might support phishing sites. Through these steps, we obtained a final list of 7278 domains, of which the mobile whitelist comprises.

4.2 Mobile Phishing Checkpoint: Checking If the Target URL Takes User Input

Unlike the web vulnerability checkers functioning in the PC environment, where most phishing detection is performed at every novel web page visit, the proposed approach delays checking as much as possible, until the user provides inputs to the web page. Owing to the nature of the attacks, phishing

Algorithm 2 (B) Whitelist based filtering

```
Require: URL $url$ of web page $p$
Ensure: $true$ – uncertain, $false$ – not phish
 1: procedure WhitelistFiltering($url$)
 2:     $d \leftarrow$ domain of $url$
 3:     if $d$ is in the whitelist then
 4:         return $false$
 5:     end if
 6:     return $true$
 7: end procedure
```

sites normally induce people to provide personal information. Therefore, in the proposed approach, the phishing detection starts only when the user touches the text input field of a web page. This is similar to SPS [44] because both of SPS and the proposed method believe that the text input is the source of phishing. However, the proposed method checks the possibility of phishing websites based on whether they get user inputs or not. On the other hand, SPS determines whether a site is phishing or not based on other information such the URL of it. Also, although SPS detect the text field by using HTTP headers or several HTML tags, the proposed method uses a special class of type “webtextview” that Android defines for the web object to take user inputs without burden to analyze HTTP headers or find HTML tags by parsing HTML codes. By checking that the web object of the associates is of type “webtextview” before the user opens the software keyboard, the proposed scheme can decide if the phishing detection mechanism should start execution.

Algorithm 3 (C) Mobile phishing checkpoint

```
Require: Touch text-input field
 1: procedure MobilePhishingCheckpoint
 2:     invoke IMM (Input Method Manager)
 3:     $v \leftarrow$ current instance of View class
 4:     $type \leftarrow$ type of $v$
 5:     if $type$ is ‘webtextview’ then
 6:         $url \leftarrow$ loaded URL of $v$
 7:         if WhitelistFiltering($url$) then
 8:             DetectPhish($url$)
 9:         end if
10:     end if
11:     open soft keyboard
12: end procedure
```

This approach has three advantages in the mobile environment. First, it saves battery power. Checking every web page leads to a large amount of energy consumption; therefore, we skip checking many pages. Furthermore, since mobile developers tend to limit the number of times the text-based input of the user is required (because of the small screen) [45], it can reduce power consumption requirements. Second, this approach can function in every browser-enabled application. As we mentioned in Sect. 3, anti-phishing software is usually attached as an add-on to a web browser. However, mobile devices like smartphones can browse web pages in any enabled application. Our approach is not dependent on a particular browser since it does not require any add-on or plug-in. Finally, this approach prevents phishing attacks by URL redirection. As we mentioned in Sect. 1, in the mobile environment, shortened URLs such as t.co and goo.gl are used prevalently because of the limited screen size. Even if the users cannot recognize the shortened phishing URLs, our approach can defend users from the phishing attacks by checking the URL of the redirected phishing pages that have text input fields.

4.3 Phishing Detection with Target URL, Google Search, and URL Scanning/Phishing Reporting Sites

If the target URL is not filtered in the two previous steps, the proposed method runs the phishing detection algorithm (Algorithm 4) with the target URL. The following algorithms show the details of the proposed phishing detection method.

4.3.1 Refine Algorithm

In Algorithm 5, the input target URL is refined in a manner that is similar to Huh et al’s approach [22]. This algorithm extracts the essential part of the input URL, allowing us to differentiate between legitimate and phishing pages. If the URL is entered into the Google search engine without manipulation, the false positive rate would increase, since legitimate web pages may not appear in search results because of relatively unimportant but unique parts of the target URL. The syntax of HTTP URLs follows the standard RFC 1738 [46] and RFC 3986 [47]. First, the algorithm deletes the strings that refer to the common protocol (“http” or “https”) in the scheme part and that refer to the lowest level domain (“www”) in the host part if exists. Finally, it deletes the optional parts such as query or fragment part attached to the input URL. Because our focus is the URLs for web pages, which begin with http or https, and all URLs’ structures follow the standard, we can say that Algorithm 5 works well with every possible URL.

For instance, in the case of the following target input URL, the Google search engine tends not to return the corresponding web page because the query part is typically unique to each search and may not appear frequently.

```
https://www.paypal.com/signin?
country.x=DE&locale.x=de_DE
```

$\Downarrow$

Eliminate ‘https://’

$\Downarrow$

```
www.paypal.com/signin?
country.x=DE&locale.x=de_DE
```

$\Downarrow$

Eliminate ‘www.’

$\Downarrow$

Algorithm 4 (D) Proposed mobile phishing detection method

```
Require: URL $url$ of web page $p$
Ensure: $true$ - phish; $false$ - not phish
 1: procedure DETECTPHISH($url$)
 2:     $url' \leftarrow$ REFINE($url$)                          ▷ 4.3.1
 3:     $rp \leftarrow$ SEARCHBYGOOGLE($url'$)                   ▷ 4.3.2
 4: ($rp$ is a search result page with query $url'$)

 5:     if $rp$ is zero search result then          ▷ Criterion 1 in 4.3.3
 6:         return $true$
 7:     end if

 8:     for $i \leftarrow$ each URL of matching documents in $rp$ do
 9:         $d \leftarrow$ domain of $i$
10:         if $d$ is URL Scanning / Phishing Reporting Domain
    then
11:                                                 ▷ Criterion 2 in 4.3.3
12:             continue
13:         else
14:             return $false$
15:         end if
16:     end for
17:     return $true$
18: end procedure
```

Algorithm 5 Refine

```
Require: URL $url$ of web page $p$
 1: procedure REFINE($url$)
 2:     if the protocol of $url$ is 'http' or 'https' then
 3:         $url \leftarrow$ eliminate the 'http://' or 'https://' of $url$
 4:     end if
 5:     if the lowest level domain of $url$ has a 'www' then
 6:         $url \leftarrow$ eliminate the 'www.' of $url$
 7:     end if
 8:     if $url$ has a query or a fragment part then
 9:         $url \leftarrow$ eliminate the query or the fragment part of $url$
10:     end if
11:     return $url$
12: end procedure
```

```
paypal.com/signin?
country.x=DE&locale.x=de_DE
```

⇓

Eliminate the query part

⇓

```
paypal.com/signin
```

However, if the REFINE algorithm cuts 'https://', 'www.' and the query part, and the refined URL (paypal.com/signin) is used for Google searching, then we can obtain the search result from the Google search engine.

4.3.2 SEARCHBYGOOGLE Algorithm

The routine queries the Google search engine with a refined URL url' that is a result of Algorithm 5 in Sect. 4.3.1. We use three parameters for Google search. First, we search for the URLs that contain url' by using the "as_epq" parameter. Using the parameter "q" as in a conventional search will make even legitimate web pages be included in the search result, as they insert the URLs of the legitimate web pages in their own URLs in order to pretend a legitimate web page. Second, we use 'hl=en' to obtain the contents of the web pages in the search result in English. Finally, we add 'nota=1' option in order to bring the source codes of the web pages in the search result: we can parse the resultant source codes for further analysis. Thus, we generate the following URL in order to make a query to the Google search engine.

```
http://www.google.com/search?as_epq=url'
&hl=en&nota=1
```

Algorithm 6 Search by Google

```
Require: the refined URL $url'$ by procedure $Refine$
1: procedure SEARCHBYGOOGLE($url'$)
2:     $googleurl$ ← "http://www.google.com/search?hl=en&
   nota=1&as_epq="
3:     $query \leftarrow url'$
4:     $searchurl \leftarrow googleurl + query$
5:     $result \leftarrow$ HTML code by HTTP connection to $searchurl$
6:     return $result$
7: end procedure
```

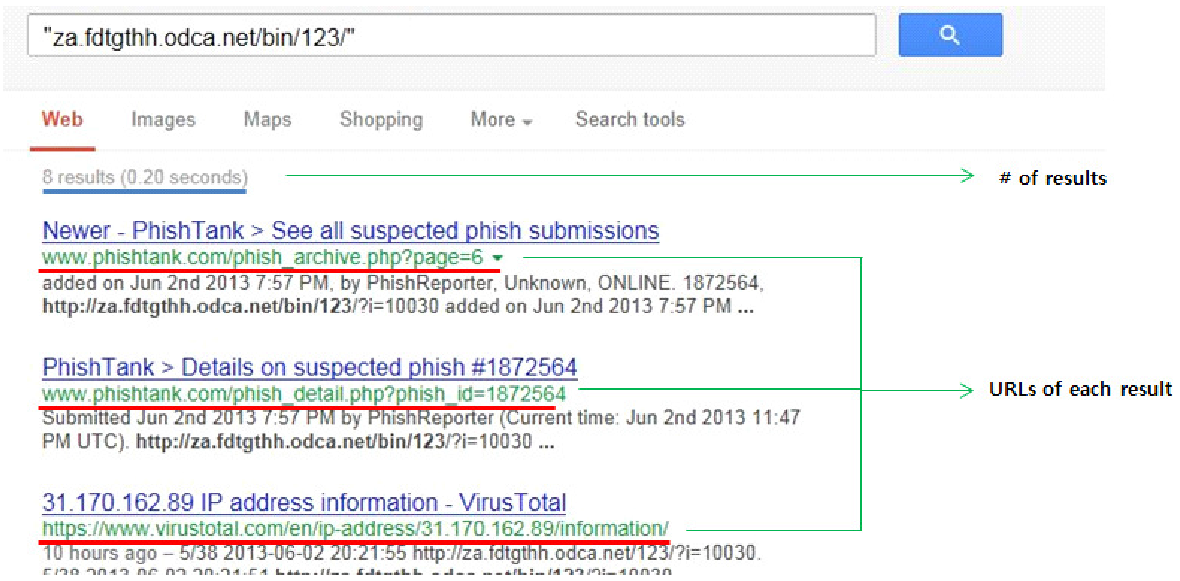
4.3.3 Phishing Detection with the Search Result and URL Scanning/Phishing Reporting Sites

Figure 2 shows an example where a refined URL, "za.fdtgthh.odca.net/bin/123/", is used as an input for the proposed SEARCHBYGOOGLE algorithm. We extract the number of results and the URLs of all matching documents to determine whether the target URL is a phishing site.

The following two criteria are used for the algorithm's decision:

- (Criterion 1) Zero-day phishing sites are not found in the search result.
- (Criterion 2) Anti-phishing websites can have the phishing URLs listed on their blacklists.

Criterion 1 is based on the fact that the phishing sites are difficult for search engines to crawl, because phishing sites are not contained in other web pages; no other web pages link to phishing sites. Phishing site developers normally ignore creating those sites because the URL of the phishing site is propagated using emails or SMSs. It is particularly useful to cope with zero-day phishing attacks, because it is highly unlikely the search engine has crawled the zero-day phishing sites which are new phishing sites that have not yet been discovered. Therefore, Criterion 1 can be used for distinguishing phishing sites. In CANTINA [19], the zero-means-phishing algorithm (ZMP) utilizes this criterion similarly as the proposed approach, but there are some differences between them. First, for ZMP, CANTINA



"za.fdtgthh.odca.net/bin/123/"

Web Images Maps Shopping More Search tools

8 results (0.20 seconds)

Newer - PhishTank > See all suspected phish submissions
www.phishtank.com/phish_archive.php?page=6
added on Jun 2nd 2013 7:57 PM, by PhishReporter, Unknown, ONLINE. 1872564, **http://za.fdtgthh.odca.net/bin/123/**?i=10030 added on Jun 2nd 2013 7:57 PM ...

PhishTank > Details on suspected phish #1872564
www.phishtank.com/phish_detail.php?phish_id=1872564
Submitted Jun 2nd 2013 7:57 PM by PhishReporter (Current time: Jun 2nd 2013 11:47 PM UTC). **http://za.fdtgthh.odca.net/bin/123/**?i=10030 ...

31.170.162.89 IP address information - VirusTotal
https://www.virustotal.com/en/ip-address/31.170.162.89/information/
10 hours ago – 5/38 2013-06-02 20:21:55 http://za.fdtgthh.odca.net/123/?i=10030.

Fig. 2 An example of executing SearchbyGoogle with the refined URL "za.fdtgthh.odca.net/bin/123/"

Table 1 Examples of URL scanning/phishing reporting web sites.

phishtank.com
blog.gmane.org/gmane.comp.security.phishings
clean-mx.com or clean-mx.de
virustotal.com
.
.
.

Table 2 Implementation environments.

Smartphone	Nexus S (crespo)
OS Version	Android 4.0.3 (Ice Cream Sandwich)
API Level	15
Emulator	DDMS
Battery Checking App	PowerTutor

needs to do compute-intensive operations such as TF-IDF operation to extract keywords used for search. On the other hand, the proposed scheme needs to do simple modification on the URL, which is much less costly than CANTINA's. Next, CANTINA does not say exactly why ZMP is working well except its experiment results, which means ZMP is a simply assumptive heuristic method without any logical reason. However, we show why Criterion 1 is reasonable as the above description and verify this by experiments that show the tendency in increasing phishing detection rate.

After a certain amount of time has passed, phishing reporting sites can be crawled because they maintain a list of links (URLs) to phishing sites. Table 1 shows a list of popular phishing-reporting web sites. We can filter the phishing URLs by using the list of phishing reporting web sites. In the case where the Google search result outputs a number of URLs, if all of them are contained in the phishing reporting web sites, we can determine that the target URL is a phishing site. Because the phishing reporting sites have links to the phishing sites, the Google search engine could crawl the phishing sites. However, the Google search engine quickly identifies and eliminates those phishing sites so that they cannot be searched even if they use an advanced technique such as cloaking, thereby helping our proposed scheme works better. Therefore, Criterion 2 can be used for distinguishing phishing sites.

5. Experiments and Analysis

We conducted experiments on a Google Nexus S running Android 4.0.3, using 8315 phishing sites and 8315 legitimate web pages. Then, we analyzed the results by comparing the accuracy, efficiency, and functionality of our proposed method with several existing anti-phishing approaches.

5.1 Implementation Environments

We implemented the proposed method and performed experiments after installing our implementation onto a flagship smartphone, a Google Nexus S. We modified the original Android 4.0.3 (Ice Cream Sandwich) source code for applying the proposed scheme. The performance of the scheme was checked with a DDMS emulator, which was connected to the device where the proposed scheme was installed. We modified the original source of the PowerTutor [48] application, which is intended predominantly for measuring the power consumption of the major components in a device. Instead, we used it for measuring the power consumption of the proposed scheme. Table 2 shows the implementation environment.

5.2 Phishing and Target Sites for Experiments

We have selected PhishTank [18], which is one of the best-

Table 3 Result of the experiment for evaluating detection accuracy.

		Experiment with phishing sites		Experiment with legitimate pages	
unit:(%)		True Positive	False Negative	True Negative	False Positive
Algorithm B		N/A	0	36.81	N/A
Algorithm C		N/A	N/A	0	N/A
Algorithm D	Criterion 1	49.63	0.78	62.58	0.61
	Criterion 2	49.59			0.00
Total		99.22	0.78	99.39	0.61

known websites used for the quick registration of novel phishing sites, in order to extract the phishing sites. Among the phishing sites, we selected those that are suspected as phishing but not clearly designated as phishing sites because they are included in the blacklist of web browsers quickly once they are designated as phishing sites. We only used the sites that were reported within the previous 12 hours and are manually verified as phishing sites among the suspected sites.

We conducted our experiment for 31 days with 8315 phishing sites that met the above requirements. Because the proposed scheme uses only URLs and does not use the contents, the sites that use various languages could be included in the 8315 sites. We also selected 8315 legitimate web pages whose domains are highly ranked in Alexa [49] where the ranking reflects the popularity of domain. The selected pages are 5116 main pages and 3199 login pages where all of both types of pages require text-based user inputs.

5.3 Accuracy Evaluation

We evaluated how well the proposed method detects phishing sites with the 8315 phishing sites and the 8315 legitimate web pages that were introduced in the above subsection.

Figure 3 shows the true positive rate of the proposed method in comparison with the blacklist method installed in Google Chrome web browsers. The detection rate of the proposed scheme is close to twice that of Google Chrome's method. All phishing sites are not in the whitelist and have text-based inputs so they are detected by two criteria. Among them, 49.63% are detected by Criterion 1, and 49.59% are detected based on the lists maintained by phishing reporting sites, Criterion 2. Also, as we define that 'zero day attack' is the phishing sites that Google chrome cannot detect in our experiment, all zero-day attacks, 45.09% of 8135 phishing sites, are detected by our proposal. 57.22% of the zero-day attacks are detected by Criterion 1, and others are detected by Criterion 2. Table 3 summarizes the result of the experiments related to accuracy evaluation. In the experiment with the legitimate pages, 36.81% of legitimate pages are filtered by our mobile whitelist, Algorithm B, and 62.58% are detected as legitimate by Algorithm D. None are detected by Algorithm C because all the legitimate pages of our dataset have text-based inputs. The false positive rate of the proposed scheme is 0.61%. Table 4 shows the accuracy comparison between the proposed method and other anti-phishing approaches. The proposed method is shown to have better true positive and false positive rates than most of the other approaches. GoldPhish shows the better false positive rate than the proposed method, but we question its accuracy because its dataset is too small (100 legitimate sites).

Figure 4 lists the eight main phishing reporting sites crawled by the proposed scheme and their contributions for detecting phishing sites in our experiment. PhishTank [18],

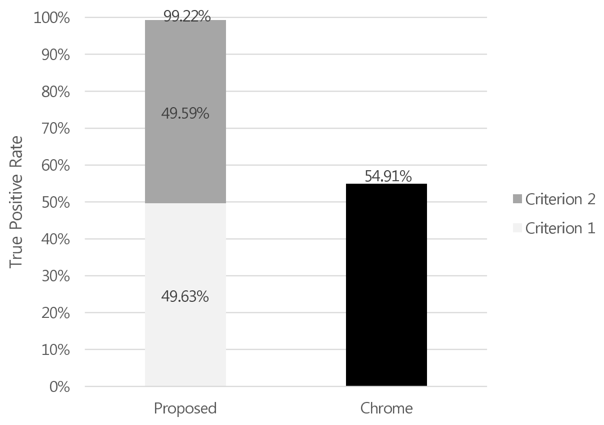


Fig. 3 Accuracy evaluation: true positive rate comparison with the proposed method and the blacklist approach installed in Chrome browser.

Table 4 Accuracy comparison with existing anti-phishing approaches.

	True Positive (%)	False Positive (%)
CANTINA [19]	90-97	1
Xiang's proposal [20]	90.06-93.31	1.95-2.26
GoldPhish [21]	98	0
Moghimi's proposal [25]	99.14	0.86
Our Approach	99.22	0.61

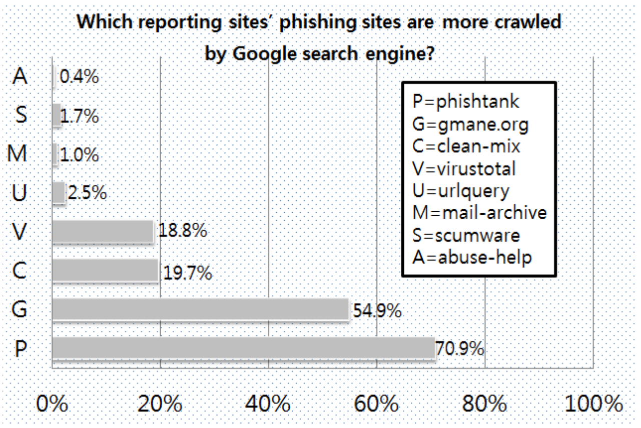


Fig. 4 The rate of each crawled phishing reporting sites containment of phishing sites on the total phishing sites used in the experiment.

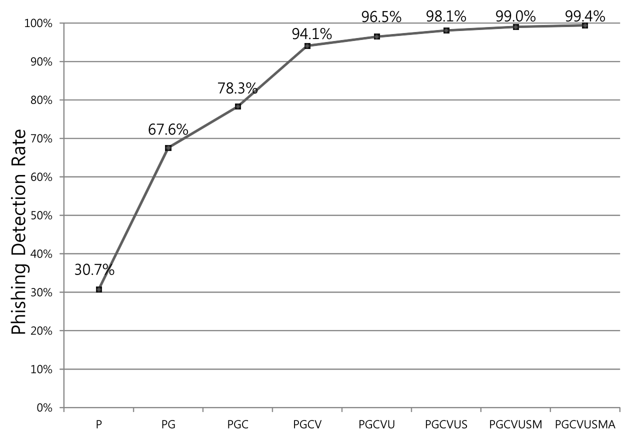


Fig. 5 The phishing detection rates on various combinations of phishing reporting sites.

phishwatch (gmane.org) [50], CLEAN MX [51], and VirusTotal [52] are the main contributors, as shown in Fig. 4. Fig. 5 shows how the detection rates change when multiple phishing reporting sites are used compared to the case of the experiment for accuracy evaluation, where total 21 phishing reporting web sites are used. The y-axis value in Fig. 5 refers to the relative ratio of phishing detection compared to the phishing detection rate 49.59% by Criterion 2 in Algorithm D in the experiment with the phishing sites for accuracy evaluation. The relative detection rate of a combination of four main contributors is 94.07%; when more phishing reporting sites are included, the detection rate shown by the proposed method increases slightly.

5.4 Energy Consumption

We analyze how much energy is consumed if the proposed scheme is employed. To generalize the analysis result, we assume the following two values:

- α: the required energy to perform whitelist-based filtering
- β: the required energy to perform the main phishing-detection routine

To obtain α and β, we performed an experiment for measuring the energy consumption. In the experiment, the smartphone was connected to one of the 8315 phishing sites through a web browser. While the connection was being established, we measured the amount of the energy consumed. Particularly, we measured the energy consumed in the first 8 seconds since the connection establishment began. We repeated this for 20 times. For comparison, we also measured the energy consumed when the proposed implementation did not function. Figure 6 shows the experiment result. In this experimental result, the required power to run the proposed method is 635.8 mJ in average, where α = 16.8 mJ and β = 619 mJ. By our analysis, the energy consumption for network interactions (HTTP) occupies the majority of β; the rest is considerably small to measure.

We can derive the following formula for calculating the average consumption overhead used by the proposed scheme:

$$0 * Pr[\mathsf{A}] + \alpha * Pr[\mathsf{B} \wedge \neg \mathsf{A}] + (\alpha + \beta) * (Pr[\neg \mathsf{A} \wedge \neg \mathsf{B}]) \quad (1)$$

where A is the event that the web page does not take inputs by users, and B is the event that the URL corresponding to the currently tested page is in the whitelist; those are independent from each other. As we mentioned in Sect. 4.2, we do not check web pages where users do not attempt to input any text. Thus, we assume the required energy for this is zero. In order to derive $Pr[\mathsf{B}]$, we should calculate the number of URLs of the legitimate web pages included in the whitelist. We already show the ratio in Table 3 so we can derive $Pr[\mathsf{B}]$ = 0.3681.

Now we derive $Pr[\mathsf{A}]$. Because it is considerably difficult to calculate $Pr[\mathsf{A}]$ in all cases, we focus on the worst case where $Pr[\mathsf{A}]$ is minimized. According to [17], people provide text-based inputs most frequently when they access mobile search services. Furthermore, because the websites for mobile search services are the most frequently visited websites by smartphone users [16], we calculate $Pr[\mathsf{A}]$ in the case when users access mobile search services.

To derive the realistic $Pr[\mathsf{A}]$, we refer to M. Kamvar et al's paper [53], which presents the results of a study that investigates the behavioral search characteristics of ordinary web search engine users. It provides how many pages of different types a user access in a single session when the user uses a mobile search service. Table 5 is the result from [53]. The number of queries can be mapped to the number of pages that require user inputs because users should provide text-based inputs to make a search query. The number of click-through per query and the number of "more search result" requests per query can be mapped to the number of web pages that users do not provide text-based inputs because only mouse-clicks are performed by users when they are visiting those web pages. With Table 5, we count the number of pages $n(P)$ the users visited during search session as follows:

$$n(P) = 1 + n(SRP) + n(CT) + n(MSR)$$

where 1 is the first search page; $n(SRP)$ is the number of

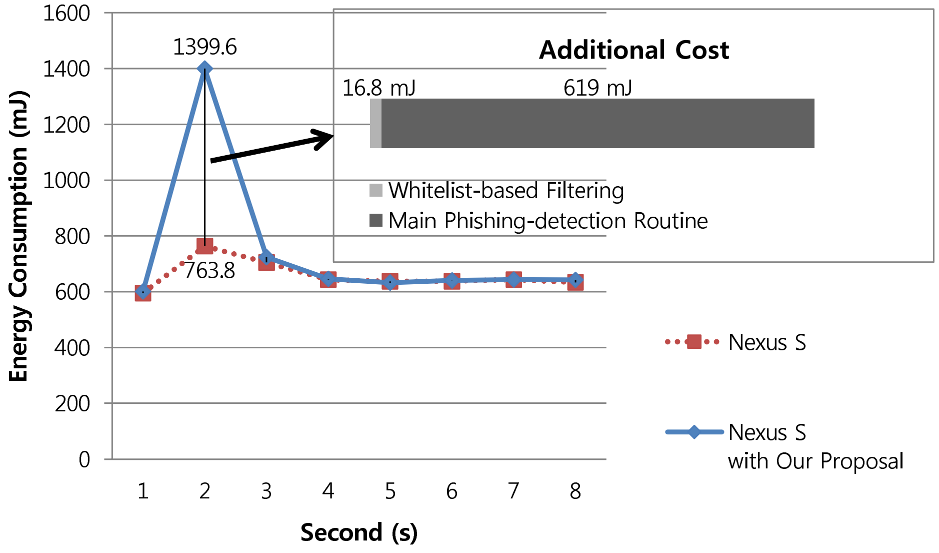


Fig. 6 Comparing the energy consumption between Nexus S and Nexus S with the proposed scheme.

Table 5 Minimum, average, maximum value for queries, click-through, and "more search result" requests.

	Minimum	Average	Maximum
The number of queries per session	1	1.6	43
The number of click-through per query	0	1.7	37
The number of "more search result" requests per query	0	0.187	82

Source : M. Kamvar et al., "A large scale study of wireless search behavior: Google mobile search"

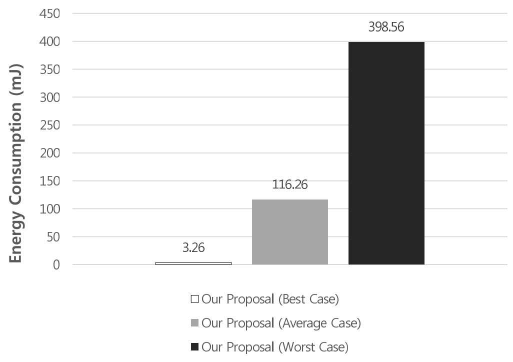


Fig. 7 Comparison of the total energy consumption in the proposed scheme in performing a search task.

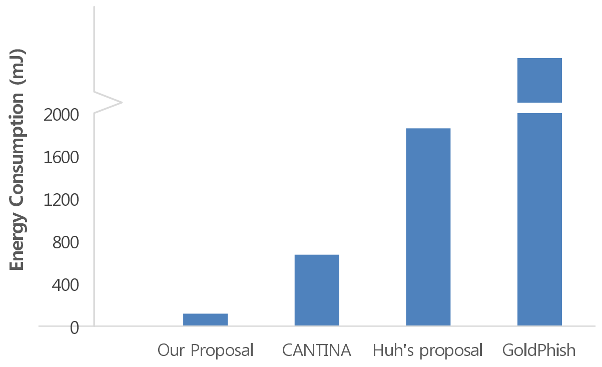


Fig. 8 Comparison of the estimated energy consumption with existing anti-phishing approaches using search engines.

search result pages, which is equal to the number of queries $n(Q)$; $n(CT)$ is the number of click-throughs; and $n(MSR)$ is the number of "more search result" requests.

Since the number of queries is equal to the number of pages where users provide input, we can derive $Pr[\mathrm{A}]$ as follows:

$$Pr[\mathrm{A}] = 1 - n(Q) \,/\, n(P)$$

Therefore, we can obtain $Pr[\mathrm{A}]$, which equals 0.992 for maximum, 0.715 for average, and 0.023 for minimum. Thus, the minimal additional energy consumption is 3.26 mJ (minimal), 116.26 mJ (average), or 398.56 mJ (maximum). Figure 7 shows all the energy consumption outlined above in our proposal. We prove that our proposed method is more efficient by showing the average additional consumption has decreased by 81.22% as compared to approaches that have a similar detection routine to our proposal using search engine, which consumes 619 mJ in the above analysis. Figure 8 shows the comparison of the estimated power consumption between the proposed method and the existing anti-phishing approaches using search engines. In case of CANTAINA, we estimated the power consumption which consists of Google search, which takes 619 mJ as our experiment, and WHOIS search, which takes 50.85 mJ according to our experiment. Thus, we can conclude that it requires at least additional 669.85 mJ that is much greater than our aver-

Table 6 Functionality comparison with existing anti-phishing approaches.

		Blacklist Technique [6]–[8]	CANTINA [19]	Xiang's proposal [20]	GoldPhish [21]	Moghimi's proposal [25]	Our Proposal
Zero-day Detection		Not detected	Detected	Detected	Detected	Detected	Detected
Costs	Comparison Group	Not required	Required	Not required	Not required	Not required	Not required
	Algorithm	Not used	TF-IDF	TF-IDF	Not used	SVM	Not used
	Training Data	Not required	Not required	Not required	Not required	Required	Not required
	Extra Software	Not used	Not used	NER	OCR	Not used	Not used

age consumption. In the case of Huh's proposal, it requires at least additional 1857 mJ because it uses three search engines such as Google, Bing, and Yahoo. In the case of GoldPhish, it requires at least additional 619 mJ for Google search and the extra power consumption for OCR. According to [12], the energy consumption of OCR application is more than 80,000 mJ.

5.5 Functionality Comparison

We now compare the proposed scheme with existing approaches in terms of functionality. A summary of this comparison is presented in Table 6. The blacklist technique [6]–[8] requires a small amount of energy consumption but cannot detect zero-day phishing attacks. CANTINA [19] requires the comparison group for phishing detection and uses TF-IDF, which does not function well in the mobile environment because less information is available as compared to the PC environment. Xiang's proposal [20] also uses TF-IDF algorithm, and requires extra software, Named Entity Recognizer (NER) such as Stanford NER [54]. Goldphish [21] requires extra software, OCR, which exhibits heavy energy consumption. Moghimi's proposal [25] requires data training with huge amount of training data for better accuracy, and machine learning algorithm, SVM. Unlike the other methods, the proposed scheme does not require any training data, additional algorithms and software, such as TF-IDF, SVM, NER, or OCR. In addition, it can detect zero-day attacks with minimal communication cost.

6. Discussion

We now discuss some issues related to the security and performance of the proposed approach.

6.1 Possible Response against the Proposed Approach by Attacker

First, an attacker can bypass our approach in the following method:

1. The attacker creates a normal page
2. The attacker waits until the page obtains some reputation
3. The page is crawled by Google search engine
4. The attacker changes the page into a phishing site

However, such an approach will not occur easily because time is a considerably valuable resource to phishers. If attackers use this method, they would have to spend time to create web pages with content that is interesting to many normal users, promote these pages so that other normal pages can link to it, and wait until the page gains popularity and is crawled by the Google search engine. Nevertheless, all the time and efforts involved in this strategy do not guarantee a high success rate of phishing, despite being able to bypass our approach. Meanwhile, smarter phishers may try to increase the success rate by creating more phishing sites in order to reach a higher volume of victims, repeatedly.

Second, an attacker can also try to bypass our approach in the following method:

1. The attacker finds a vulnerability at a legitimate page
2. The attacker compromises the legitimate page
3. The attacker changes the page into a phishing site with the same URL

This scenario may not work well if the legitimate site has much popularity. In this case, the website is normally administrated well in terms of security thus it is highly unlikely that the attacker is unable to find security vulnerabilities. In the other case, if the legitimate site is unpopular, it is possible that the attacker can inject malicious web pages to the website for phishing. However, by the result of our experiment with 8315 phishing sites, the attacker tends to put the company's name in the URL to improve the success rate of deceiving users a little. Then, the attacker creates new directory or file in the compromised site and puts the malicious web pages into them, and hence the URL is changed. In this case, our phishing mitigation method works well because the phishing URL is brand new: no result or only phishing reporting sites are given as a result of Google search if our method works with the URL, as described in Sect. 4.3.3. However, it is still possible that these attack scenarios succeed against the proposed method if the attacker creates the phishing site with the same URL of the legitimate page. We are fully aware that these attack scenarios are very important and we will overcome this problem in future work.

6.2 Attacks Using Google Docs

Some attacks show users the main site with advertisements or urgent contents, and then lead them to the site created by Google Docs, which is designed to be easy to obtain the inputs of users to retrieve their information. Our proposed method cannot detect these attacks since the top-level domain (TLD) of Google docs, "docs.google.com", is in the mobile whiteilst. We consider these attacks to be out of our scope because our definition of phishing mentioned in Sect. 1 does not include these attacks in that this attack does not masquerade as a trustworthy entity. They, however, could be included in phishing because they steal the private information of victims, and we intend to solve this problem in the future.

6.3 Attacks Using Multiple Redirection Pages

Redirection pages play a role to automatically send visitors to another page and hence they usually do not receive users' text-based inputs. Because the proposed method does not inspect the pages that have no text input fields, it has the limitation that when multiple redirection pages exist, the proposed method takes little more time to detect phishing than the blacklist technique like web browsers which can detect the redirection phishing pages immediately. However, multiple redirection phishing pages arrive at the redirected phishing page that has text input fields in the end and users try to input their information in that page, which is when the proposed method starts to inspect. Ultimately, from a security standpoint, the proposed method detects phishing attacks using multiple redirection pages, as does the blacklist technique, by checking the URL of the redirected phishing pages that have text input fields even though it is a little late. Even if users input their information in the redirection page that has text input fields, the proposed method also works because it runs only when users touches the text input field regardless of redirection. Therefore, the proposed method detects phishing attacks using multiple redirection pages.

6.4 Time Delay

The proposed scheme takes less than a second, and most of the checking time is spent on the Google search, according to our analysis. Google has also confirmed that the average response time on a search result is a fraction of a second [55]. Furthermore, the proposed scheme functions only when the mobile phishing checkpoint is invoked. To receive a text-based input, a mobile device should show users the software keyboard, which requires time to load. Thus, mobile users would hardly feel the time delay of the proposed scheme because they are already aware of the loading time of the software keyboard.

6.5 Compatibility

API Level 15 in our implementation environments may be a little concerning about compatibility. However, because we confirmed that all classes and functions used in our implementation still use in API Level 27, which is the latest version of Android API, we expect that the proposed method has no compatibility problem about any upper API Level. Also, as for the efficiency, we discovered the power consumption of HTTP network interactions in our implementation environments is almost the same as that in version 4.1.2 (Jelly Bean), which is the latest version of Nexus S: our observation found the difference of power consumption in those two environments was at most 5 mJ.

6.6 Automatic IP Block

Our approach might not work well if Google automatically blocks the access of the search engine from the device where our approach is working using the IP address of the device, due to large number of queries. We think this is not a big concern due to the following reasons: first, there is a little chance of the IP address block because we activate the Google search method less frequently by using mobile whitelist and mobile checkpoint. As mentioned in Table 5, the number of queries per session are only 1.6 on average. Also, due to the nature of mobile devices, the associated IP address to the device can change frequently. Second, there is a way to increase the available number of queries legitimately. We can perform queries through the Google Custom Search API, which supports 100 queries per a day for free. The number of possible queries can be increased with additional cost.

7. Conclusion and Future Work

In this paper, we proposed a high-performance and energy-efficient mobile phishing detection scheme for mobile devices, particularly smartphones, where reducing battery consumption is a crucial goal. To prevent mobile phishing, our method combines three techniques: an app-based whitelist, a mobile phishing checkpoint, and mobile phishing detection using a combination of Google search and websites that report phishing. The app-based whitelist reduces unnecessary URL checking by taking domains from the trusted apps. This technique offers a whitelist suitable for the mobile environment, not for the PC environment. The checkpoint on smartphones designates a point to check for phishing by considering the contrasting characteristics of smartphones and phishing, which are not considered in PC environment. This checkpoint can significantly minimize battery consumption compared to checking every visited page. Mobile phishing detection using only Google search results and the URL of the visited page is simple and efficient. This simple scheme requires only a few network interactions and demonstrates a high phishing detection rate, particularly with regard to zero-day attacks, the core of phishing attacks. In addition, our proposed method significantly enhances the energy efficiency compared to the existing PC-based anti-phishing schemes, considering the ratio of whitelist filtering and text-based input. Further, we prove that our method consumes less battery power than any other anti-phishing scheme. Therefore, our proposed method has the advantages of high phishing detection rates and considerably higher energy efficiency. While our proposal offers the best performance in terms of accuracy, its power consumption for HTTP connections is somewhat high. We will get over this limitation including delayed detection for multiple redirection phishing pages mentioned in Sect. 6.3. Also, we will get over attacks that change created normal pages crawled by Google search engine or compromised legitimate pages into phishing sites with the same URL mentioned in Sect. 6.1 and attacks that retrieve users' information using Google Docs Sect. 6.2 in future work.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2016R1C1B2011022, NRF-2016R1A4A1011761).

References

[1] "APCERT annual report 2016," http://www.apcert.org/documents/pdf/APCERT_Annual_Report_2016.pdf, 2017.
[2] "What is phishing?," https://www.lookout.com/know-your-mobile/what-is-phishing
[3] "Mobile security vulnerabilities are creating big problems," https://www.rsaconference.com/blogs/mobile-security-vulnerabilities-are-creating-big-problems, 2016.
[4] "The Inbox Report 2016: Consumer perceptions of email," http://www.fluentco.com/wp-content/uploads/2017/01/Fluent_InboxReport_2016.pdf, 2017.
[5] "Mobile web intelligence report january 2018," http://go.afiliastechnologies.com/mobile-web-intelligence-report-jan-2018/, 2018.
[6] "Internet explorer smartscreen filter," http://windows.microsoft.com/en-us/internet-explorer/use-smartscreen-filter
[7] "Mozilla support," https://support.mozilla.org/ko/kb/how-does-phishing-and-malware-protection-work
[8] "Chrome safe browsing," https://www.google.com/intl/ko/chrome/browser/features.html#security
[9] F.D.I. Corporation and U.S. of America, "Putting an end to account-hijacking identity theft," 2004.
[10] L.F. Cranor, S. Egelman, J.I. Hong, and Y. Zhang, "Phinding phish: An evaluation of anti-phishing toolbars.," Proc. 14th Annual Network and Distributed System Security Symposium (NDSS), ISOC, 2007.
[11] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," Proc. 16th international conference on World Wide Web (WWW), pp.649–656, ACM, 2007.
[12] Y.W. Kwon and E. Tilevich, "Energy-efficient and fault-tolerant distributed mobile execution," Proc. 32nd International Conference on Distributed Computing Systems (ICDCS), pp.586–595, IEEE, 2012.
[13] N. Tsalis, N. Virvilis, A. Mylonas, T. Apostolopoulos, and D. Gritzalis, "Browser blacklists: The utopia of phishing protection," International Conference on E-Business and Telecommunications, pp.278–293, Springer, 2014.
[14] "App Annie," https://www.appannie.com/
[15] T. Thakur and R. Verma, "Catching classical and hijack-based phishing attacks," International Conference on Information Systems Security, pp.318–337, Springer, 2014.
[16] "The mobile movement study," https://ssl.gstatic.com/think/docs/the-mobile-movement_research-studies.pdf, 2011.
[17] "2014 survey on the mobile internet usage executive summary," http://isis.kisa.or.kr/board/index.jsp?pageId=040300&itemId=329, 2014.
[18] "PhishTank," http://www.phishtank.com/
[19] Y. Zhang, J.I. Hong, and L.F. Cranor, "Cantina: A content-based approach to detecting phishing web sites," Proc. 16th international conference on World Wide Web (WWW), pp.639–648, ACM, 2007.
[20] G. Xiang and J.I. Hong, "A hybrid phish detection approach by identity discovery and keywords retrieval," Proc. 18th international conference on World Wide Web (WWW), pp.571–580, ACM, 2009.
[21] M. Dunlop, S. Groat, and D. Shelly, "Goldphish: Using images for content-based phishing analysis," Proc. 5th International Conference on Internet Monitoring and Protection (ICIMP), pp.123–128, IEEE, 2010.
[22] J.H. Huh and H. Kim, "Phishing detection with popular search engines: Simple and effective," International Symposium on Foundations and Practice of Security, pp.194–207, Springer, 2011.
[23] P. Barraclough, M.A. Hossain, M. Tahir, G. Sexton, and N. Aslam, "Intelligent phishing detection and protection scheme for online transactions," Expert Syst. Appl., vol.40, no.11, pp.4697–4706, 2013.
[24] N. Abdelhamid, A. Ayesh, and F. Thabtah, "Phishing detection based associative classification data mining," Expert Syst. Appl., vol.41, no.13, pp.5948–5959, 2014.
[25] M. Moghimi and A.Y. Varjani, "New rule-based phishing detection method," Expert Syst. Appl., vol.53, pp.231–242, 2016.
[26] A. Abbasi, F.â. Zahedi, D. Zeng, Y. Chen, H. Chen, and J.F. Nunamaker, Jr., "Enhancing predictive analytics for anti-phishing by exploiting website genre information," J. Manage. Inform. Syst., vol.31, no.4, pp.109–157, 2015.
[27] A.K. Jain and B. Gupta, "A novel approach to protect against phishing attacks at client side using auto-updated white-list," EURASIP J. Information Security, vol.2016, no.1, p.9, 2016.
[28] R. Srinivasa Rao and A.R. Pais, "Detecting phishing websites using automation of human behavior," Proc. 3rd ACM Workshop on Cyber-Physical System Security, pp.33–42, ACM, 2017.
[29] T. Raffetseder, E. Kirda, and C. Kruegel, "Building anti-phishing browser plug-ins: An experience report," Proc. 3rd International Workshop on Software Engineering for Secure Systems (SESS), p.6, IEEE Computer Society, 2007.
[30] G. Salton and M.J. McGill, Introduction to Modern Information Retrieval, McGraw-Hill, 1986.
[31] W. Han, Y. Wang, Y. Cao, J. Zhou, and L. Wang, "Anti-phishing by smart mobile device," IFIP International Conference on Network and Parallel Computing (NPC) Workshops, pp.295–302, IEEE, 2007.
[32] Y. Niu, F. Hsu, and H. Chen, "iPhish: Phishing vulnerabilities on consumer electronics," Proc. 1st Conference on Usability, Psychology, and Security (UPSEC), pp.1–8, 2008.
[33] A.P. Felt and D. Wagner, "Phishing on mobile devices," Web 2.0 Security and Privacy (W2SP) Workshop, 2011.
[34] T. Vidas, E. Owusu, S. Wang, C. Zeng, L.F. Cranor, and N. Christin, "QRishing: The susceptibility of smartphone users to QR code phishing attacks," International Conference on Financial Cryptography and Data Security, pp.52–69, Springer, 2013.
[35] Z. Xu and S. Zhu, "Abusing notification services on smartphones for phishing and spamming," Proc. 6th USENIX Conference on Offensive Technologies (WOOT), pp.1–11, 2012.
[36] D. Liu, E. Cuervo, V. Pistol, R. Scudellari, and L.P. Cox, "Screenpass: Secure password entry on touchscreen devices," Proc. 11th annual international conference on Mobile systems, applications, and services (MobiSys), pp.291–304, ACM, 2013.
[37] C. Marforio, R.J. Masti, C. Soriente, K. Kostiainen, and S. Capkun,

"Personalized security indicators to detect application phishing attacks in mobile platforms," arXiv preprint arXiv:1502.06824, 2015.
[38] L. Wu, X. Du, and J. Wu, "Effective defense schemes for phishing attacks on mobile computing platforms," IEEE Trans. Veh. Technol., vol.65, no.8, pp.6678–6691, 2016.
[39] G. Bottazzi, E. Casalicchio, D. Cingolani, F. Marturana, and M. Piu, "MP-Shield: A framework for phishing detection in mobile devices," International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing (CIT/IUCC/DASC/PICOM), pp.1977–1983, IEEE, 2015.
[40] A. Nadkarni, V. Tendulkar, and W. Enck, "NativeWrap: Ad hoc smartphone application creation for end users," Proc. 2014 ACM conference on Security and privacy in wireless & mobile networks, pp.13–24, ACM, 2014.
[41] "WebView class," http://developer.android.com/reference/android/webkit/WebView.html
[42] "UIWebView class reference," https://developer.apple.com/library/ios/documentation/uikit/reference/UIWebView_Class/Reference/Reference.html
[43] A. Carroll, G. Heiser, et al., "An analysis of power consumption in a smartphone," Proc. USENIX annual technical conference, p.21, Boston, MA, 2010.
[44] D. Miyamoto, H. Hazeyama, and Y. Kadobayashi, "SPS: A simple filtering algorithm to thwart phishing attacks," Asian Internet Engineering Conference, pp.195–209, Springer, 2005.
[45] "Forms on mobile devices: Modern solutions," https://www.smashingmagazine.com/2010/03/forms-on-mobile-devices-modern-solutions/, 2010.
[46] "RFC 1738," https://tools.ietf.org/html/rfc1738
[47] "RFC 3986," https://tools.ietf.org/html/rfc3986
[48] "Powertutor," http://ziyang.eecs.umich.edu/projects/powertutor/download.html, 2010.
[49] "Alexa," http://www.alexa.com/
[50] "phishwatch," http://blog.gmane.org/gmane.comp.security.phishings
[51] "CLEAN MX," http://www.clean-mx.com/
[52] "VirusTotal," http://www.virustotal.com/
[53] M. Kamvar and S. Baluja, "A large scale study of wireless search behavior: Google mobile search," Proc. SIGCHI conference on Human Factors in computing systems, pp.701–709, ACM, 2006.
[54] "Stanford named entity recognizer (version 1.1)," http://nlp.stanford.edu/software/CRF-NER.shtml
[55] "Google's Philosophy." https://sites.google.com/site/jurgensencompositionprojectweb/about/philosophy

Hyungkyu Lee is a Ph.D. candidate in Computer Science at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea. He received his B.S. degree in Computer Science from Sogang University, Seoul, South Korea, in 2004. He also received his M.S. degree in Computer Science from KAIST in 2009. His research interests include information security, phishing, mobile security, and botnet malware defense.

Younho Lee received the B.E., M.S., and Ph.D. degree in Computer Science from KAIST, Korea, in 2000, 2002, and 2006, respectively. He worked as a visiting postdoctoral researcher and research staff at the GeorgiaTech Information Security Center from 2007 to 2009. He is currently an associate professor in the ITM Programme, the department of Industrial and Information Systems Engineering, SeoulTech, Korea. His research interests include network security, applied cryptography, and data security.

Changho Seo is a professor at the Department of Applied Mathematics, Kongju National University. He has published over 50 papers and book chapters, and a number of books in the areas of information security. He received his undergraduate Mathematics at the Korea University, Korea. He earned his M.S. and Ph.D. in Cryptography from the Korea University, Korea. He has worked as a visiting researcher in ETRI. His areas of research include Cryptography algorithms, PKI and System security.

Hyunsoo Yoon received the B.S. degree in EE from the Seoul National University, Korea, in 1979, the M.S. degree in CS from KAIST, in 1981, and the Ph.D. degree in Computer and Information Science from the Ohio State University, Columbus, Ohio, in 1988. From 1988 to 1989, he was with the AT&T Bell Labs, as a member of technical staff. Since 1989 he has been a faculty member of the Dept. of EECS, KAIST. His research interests include parallel computer architecture, mobile communication, ad hoc networks, and information security.