MobiFish: A Lightweight Anti-Phishing Scheme for Mobile Phones

Longfei Wu, Xiaojiang Du, and Jie Wu
Dept. of Computer and Information Science
Temple University
Philadelphia, Pennsylvania 19122
{longfei.wu, dux, jiewu}@temple.edu

***Abstract*—Recent years have witnessed the increasing threat of phishing attacks on mobile platforms. In fact, mobile phishing is more dangerous due to the limitations of mobile phones and mobile user habits. Existing schemes designed for phishing attacks on computers/laptops cannot effectively address phishing attacks on mobile devices. This paper presents MobiFish, a novel automated lightweight anti-phishing scheme for mobile platforms. MobiFish verifies the validity of web pages and applications (Apps) by comparing the actual identity to the identity claimed by the web pages and Apps. MobiFish has been implemented on the Nexus 4 smartphone running the Android 4.2 operating system. We experimentally evaluate the performance of MobiFish with 100 phishing URLs and corresponding legitimate URLs, as well as fake Facebook Apps. The result shows that MobiFish is very effective in detecting phishing attacks on mobile phones.**

***Keywords*—Mobile phones; phishing attack; security; Android**

I. Introduction

Phishing attacks aim to steal private information such as usernames, passwords, and credit card details by way of impersonating a legitimate entity. Although security researchers have proposed many anti-phishing schemes, the threat of phishing attacks is not well mitigated. On the one hand, lots of phishing sites expire and revive rapidly. According to the Anti-Phishing Working Group (APWG), the average time that a phishing site stays online is 4.5 days [1]. Cranor et al. even found sometimes it is on the order of hours [2]. On the other hand, Phishing attackers keep improving their techniques so that their new attacks are able to circumvent existing anti-phishing tools.

Mobile phishing is an emerging threat targeting mobile users of financial institutions, online shopping, and social networking companies. In 2012, researchers from Trend Micro found 4,000 phishing URLs designed for mobile web pages [3]. Although this number takes up less than 1% of all collected phishing URLs, it highlights that mobile platforms has become new targets of phishing attacks. The trend of launching phishing attacks on mobile phones may be attributed to the hardware limitations such as the small screen size, and the inconvenience of user input and application switching. Besides, mobile users could also be spoofed by conventional phishing web pages when browsing with their phones.

Almost all phishing attacks on PC are in the form of bogus websites. Nowadays, with browsers powerful enough to support all kinds of Internet services, people are accustomed to online banking, online shopping, and online socializing. They are familiar with being requested and providing private information and credentials to websites. However, browsers have many fancy features and convenient functions abandoned or truncated during their adaptation to hardware-constrained mobile platforms; this results in an in unpleasant experience for users. To improve their services, most well-known enterprises have published mobile applications (Apps) for major mobile platforms. This sheds new light on phishing scams: some phishing attackers develop fake applications or repackage legitimate applications, and then upload these phishing applications to unofficial app markets: victim users' credentials will be sent to the phishing server. Phishing applications are even harder to detect than phishing web pages since for web pages we are able to judge the destination of form-data from HTML source code (*action* attribute in *form*). But for mobile apps, there is no way to check whether user credentials are sent to the legitimate authentication server or phisher' s server. Hence, phishing attacks on mobile phones are more complicated than those on PCs.

Current phishing detection schemes can be roughly divided into two categories: heuristics-based schemes and blacklist-based schemes. Blacklist-based schemes can only detect phishing sites that are in the blacklist but cannot detect zero-day phishing attacks that have appeared for days or even hours. It is possible that new phishing sites may have already stolen user credentials or even expired before being added into the blacklist. Heuristics-based schemes largely depend on features extracted from URL and HTML source code, and then other techniques such as machine learning are used to determine the validity. However, we find that the features extracted from HTML source code could be inaccurate, and phishing sites can easily bypass those heuristics. Furthermore, there is no off-the-shelf tool to detect phishing Apps in mobile devices. Therefore, it is important to design effective phishing defense schemes for both conventional web pages (for PCs), mobile web pages, and mobile Apps (for mobile devices).

In this paper, we propose a novel lightweight anti-phishing scheme for mobile devices – MobiFish, which is capable of defending against phishing attacks on mobile web pages and Apps. MobiFish aims to solve the essential problem of identity masquerade without reliance on HTML source code, search engine, or machine learning techniques. We employ the optical character recognition (OCR) technique to extract text from the screenshot of a login interface, which achieves better performance on mobile phones than on PCs. We are able to find the claimed identity from the extracted text, and the actual identity from the URL of a web page or remote server (for mobile apps). If these two identities are different, our tool sends a warning to the user. Our tool could block the credential transmission in phishing applications as well.

Our contributions are summarized as follows:

– We propose MobiFish, a novel automated lightweight anti-phishing scheme for mobile phones.

– We find the weakness of previous heuristics-based security schemes for conventional web phishing, and we propose a lightweight detecting strategy that utilizes optical character recognition (OCR).

– We implement MobiFish on a Google Nexus 4 smartphone running the Android 4.2 operating system.

– We evaluate the effectiveness of MobiFish with 100 phishing URLs and corresponding legitimate URLs, as well as "Facebook" phishing Apps. The results show that MobiFish can effectively defend mobile phishing attacks.

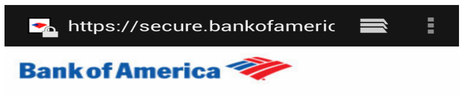
The rest of the paper is organized as follows. Section II describes background information of mobile phishing. Section III provides an overview of the MobiFish scheme. Section IV presents the details of MobiFish. Section V describes our evaluation methodology and results. Section VI discusses related works. Section VII concludes the paper.

II. Background of Mobile Phishing

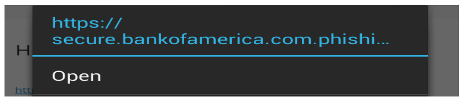
A. User Interface of Mobile Phone

To accommodate the small screen size, browsers on mobile phones have to remove or change some features, including security functionalities. The simplified version of user interface may help phishing sites to bypass user inspection.

Due to the small size of phone screens, most mobile browsers have to remove the status bar and hide the URL bar once the web pages finish loading. Even during the loading process, long URLs are truncated to fit the browser frame. Since the ability to read and verify URLs is crucial in detecting phishing attacks, partial URL (especially URL displayed with partial domain name) would certainly increase the risk of being spoofed by phishing attacks. For example, Figure 1(a) shows the URL bar with only a partial domain name when loading the "Bank of America" site. This could lead to a successful phishing attack if users are convinced by the partial URL and submit their credentials, while the full URL turns out to be "*https://secure.bankofameric.com*" or "*https://secure.bankofamerica.com.phishing.com*". Such tricks would fail if the entire URL (or at least domain name) is displayed. One possible way by which a user can view the complete URL is to click the address bar and manually scroll all the way to the end. Another way is to view the actual destination of a link, which can be invoked by holding the link for about two seconds. Though the destination URL is also partially presented as in Figure 1(b), it can display the domain name with as many as 31 characters, instead of 19 characters in URL bar. Since the full domain names of the login pages are no longer than 30 characters for most legitimate sites, checking the destination allows users to detect phishing sites more quickly.

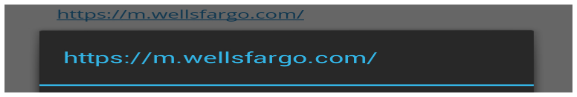


(a) Display of Partial URL

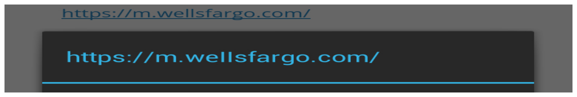


(b) Display of Link Destination

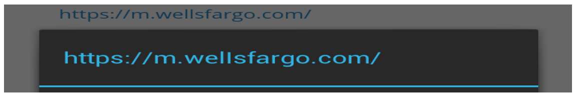
Fig. 1. Display of URL in Mobile Browser (Android)



(a) Wells Fargo Domain Name



(b) Wells Fargo Domain Name with 'l' Replaced by Capital 'i'



(c) Wells Fargo Domain Name with 'l' and Capital 'i'

Fig. 2. URL Letter Replacement

Besides, for certain legitimate sites, their domain name could be easily mimicked by replacing letters. For example, it is hard to distinguish 'l' from capital 'i' because mobile browsers display them both in vertical slash shape (e.g., Figure 2(a) and 2(b)). In Figure 2(c), we list both 'l' and capital 'i' together and find that their small difference in height is difficult to discern by human eyes. For this kind of letter replacement phishing attack, even attentive and observant users who always check the entire URL (domain name) are likely to be fooled.

To sum up, the limitations in mobile browsers considerably increase users' vulnerability to mobile phishing attacks.

B. Mobile Application Phishing Attacks

Phishing attacks based on applications are quite uncommon in PCs, but are disturbing problems on mobile platforms. According to Felt et al. [4], the four specific phishing attack models with prevalence level "common" and accuracy level "perfect" are all associated with mobile applications that impersonate legitimate apps. Application-oriented phishing attacks can be further categorized into two types based on the way they launch: Some phishing applications attempt to hijack existing legitimate apps. These phishing apps keep performing task polling, and launch themselves as long as they detect the launch of target apps. As the result, the fake login interface layers over the top of the real one, and the phishing app pretends to be the target app. Mobile users do not know what has happened since everything is accomplished during a single window switching process. One possible way to solve this is to check the identity of foreground app from the multitasking list menu, though normally no user does that. Another type of phishing app directly appears as the target app. This may occur when a user downloads fake apps from unofficial app markets.

Despite the various methods of stealing user credentials, the essential attack pattern must end with the transmission of credentials to the attacker. Hence, runtime monitoring and blocking the transmission of phishing application can effectively defeat the attack.

C. *Mobile User Habits and Preferences*

Mobile users' habits and preferences further ease the mobile phishing attacks. During the past few years, touch screen smartphones have become dominant in the mobile phone market. However, typing on a virtual keyboard is not as easy as on a physical keyboard due to lower input accuracy, particularly when walking or sitting in a moving vehicle. Because of that, it is tempting to follow links in web pages or e-mails rather than typing the links manually. Another factor is that in smartphones, switching among applications or even shifting to other pages within a browser, is more complicated and tedious than when performed on a PC. Users who value convenience usually prefer to follow links from other applications [5].

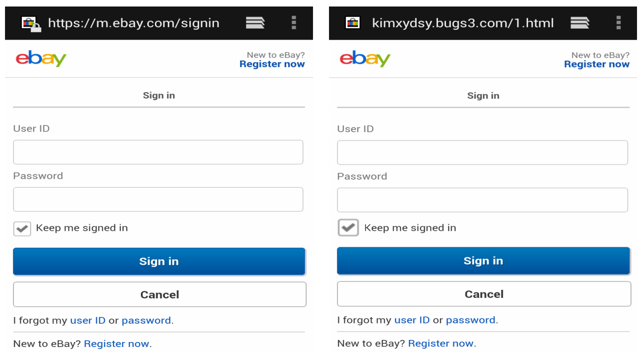
In addition, phishing attacks can succeed because users become accustomed to entering their credentials in familiar, repeated login interfaces. If users frequently encounter legitimate links whose targets prompt them for private data, then users get used to reflexively supplying the requested data [6].

III. Overview of MobiFish Scheme

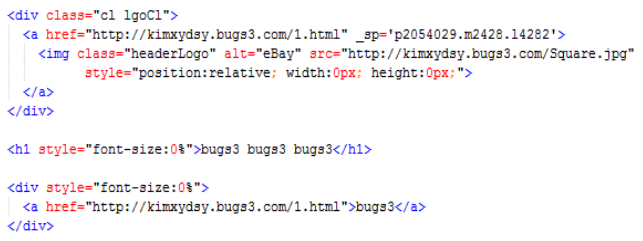
A. *Motivation*

Phishing attackers take fancy tactics to direct victims to their phishing sites or applications, which masquerade as trustworthy entities. The key of solving the phishing problem is to find the discrepancy between the identity it claims and the identity it actually has. However, none of the two major existing methods for phishing detection – blacklist and heuristics, are designed based on the key evidence of identity difference. Blacklist method is to search a suspicious site in a list of reported phishing sites. Although it can achieve high accuracy at the cost of human verification, the delay in updating the blacklist would greatly degrade its effectiveness. Specifically, blacklist-based methods cannot defend new phishing sites that have not been listed, such as zero-day phishing attacks. Heuristic detection methods are based on features extracted from URL and HTML source code, and often work with the assistance of search engine or machine learning techniques. These features are summarized from reported phishing sites. However, phishing sites may not have the features at all because each feature may only appear in some of the phishing samples. This means that carefully constructed phishing sites that remove all suspicious features are able to bypass the heuristic detection methods.

In addition, we find that information extracted from HTML source code may not always represent the web page displayed to users. This is because attackers can add text, images, and links into HTML source code, and simultaneously can make "undesirable" content disappear from a web page by simply changing its size or covering it with other images. Therefore, features like word frequency, brand name, and company logo could easily be manipulated. For example, Figure 3(a) shows the real Ebay mobile login page. We copy the code of the original site and migrate it to our web page. We also upload the image components to our website and change the links to the corresponding places in our website, especially its form-data submission URL. Then the code segment in Figure 3(c) is added into the source code. However, the tampered web page (Figure 3(b)) looks exactly the same as the official Ebay site. No user will suspect its validity without looking at its URL. In this manner, phishers can insert as many "bugs3" as needed into the HTML source code to obfuscate conventional identity extractors. The large number of "bugs3"s extracted are able to convince the identity extractor that this web page claims to be "bugs3" instead of "Ebay". As a result, anti-phishing heuristics would fail since the phishing web page indeed belongs to "bugs3.com" domain. Moreover, the title of a web page is not visible unless the user clicks the page menu icon and switches to an overview of the opened page list, which means the title could also be replaced by "bugs3" to enhance the consistency of HTML source code. As discussed above, we show that HTML source code is not a reliable way to find the claimed identity of a phishing site. As an alternative, we should focus on the screen presented to mobile users since users are directly spoofed by what they see.


https://m.ebay.com/signin
ebay
New to eBay?
Register now
Sign in
User ID
Password
Keep me signed in
Sign in
Cancel
I forgot my user ID or password.
New to eBay? Register now.
kimxydsy.bugs3.com/1.html
ebay
New to eBay?
Register now
Sign in
User ID
Password
Keep me signed in
Sign in
Cancel
I forgot my user ID or password.
New to eBay? Register now.

(a) Ebay Official Login Page (b) Ebay Phishing Login Page



```
<div class="cl lgoCl">
  <a href="http://kimxydsy.bugs3.com/1.html" _sp='p2054029.m2428.14282'>
    <img class="headerLogo" alt="eBay" src="http://kimxydsy.bugs3.com/Square.jpg"
          style="position:relative; width:0px; height:0px;">
  </a>
</div>

<h1 style="font-size:0%">bugs3 bugs3 bugs3</h1>

<div style="font-size:0%">
  <a href="http://kimxydsy.bugs3.com/1.html">bugs3</a>
</div>
```

(c) Inserted HTML Source Code

Fig. 3. Comparison of Real and Fake Ebay Login Page

Besides, existing anti-phishing schemes cannot detect phishing applications. Thus, there is a strong need for an effective defense scheme against phishing attacks on mobile platforms.

B. *Identity Extraction*

As discussed above, the screen presented to mobile users should be the exact place where the claimed identity is extract-

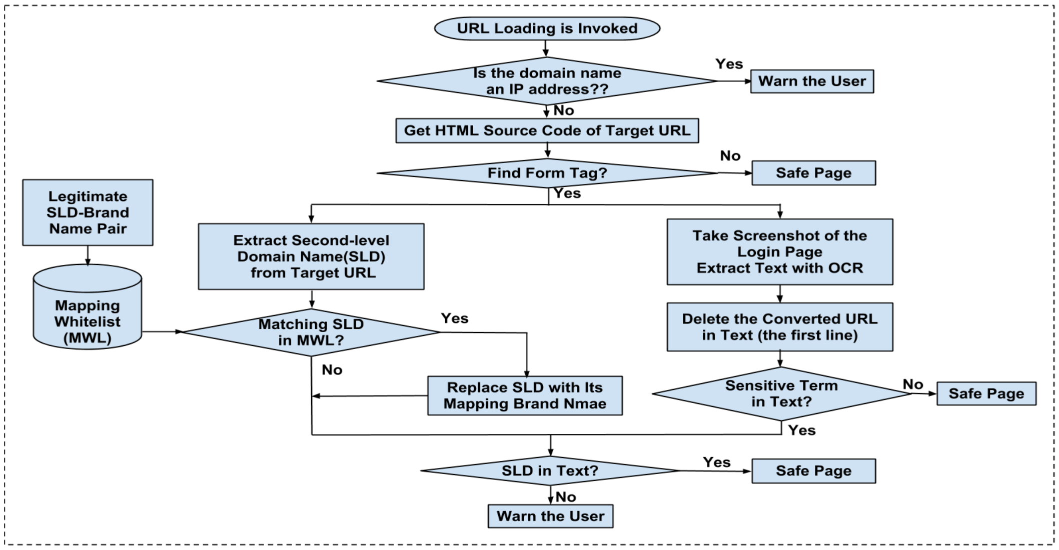

URL Loading is Invoked
Is the domain name an IP address??
Yes
Warn the User
No
Get HTML Source Code of Target URL
Find Form Tag?
No
Safe Page
Yes
Legitimate SLD-Brand Name Pair
Extract Second-level Domain Name(SLD) from Target URL
Take Screenshot of the Login Page Extract Text with OCR
Mapping Whitelist (MWL)
Matching SLD in MWL?
Yes
No
Delete the Converted URL in Text (the first line)
Replace SLD with Its Mapping Brand Nmae
Sensitive Term in Text?
No
Safe Page
Yes
SLD in Text?
Yes
Safe Page
No
Warn the User

Fig. 4. The Work Flow of WebFish

ed from. It turns out that a good way to capture phone screen is to take a screenshot. There are two common observations that lead us to believe that screenshot can work well in identity extraction and verification. The first is that most login interfaces of legitimate mobile sites and apps are very simple. The entire login page, or the majority of the page, can be captured in one screenshot. Another observation is that most well-known enterprises use brand name as the second-level domain name (SLD) of their official websites. This can be best illustrated by Bank of America (BoA). BoA uses the entire brand name as SLD despite its length. In special cases that brand names are not exactly the same as SLD, e.g., “AT&T” contains symbol in the brand name, all content-based schemes will fail due to the mismatch of brand name “AT&T” and SLD “att” (for legitimate URLs, special symbols are usually not included in domain name). However, such inconsistency can be easily solved with a mapping whitelist, in which brand name “AT&T” is directly mapped to SLD “att”, and vice versa.

Screenshot can be used for phishing detection in both web pages and applications. In the phishing web page detection, we have to know the content in the screen displayed to users. Similarly, for the detection of phishing applications, there is no way to acquire any information related to the content of login interface other than taking screenshots. Then, to obtain claimed identity, OCR technique is utilized to convert the screenshot into text. The actual identity of a mobile web page can be obtained from the SLD. However, for mobile application, the actual identity cannot be captured until authentication or transmission of user credentials happens.

C. OCR Techniques

Optical character recognition (OCR) is the mechanical or electronic conversion of an image to machine-encoded text. We believe that the OCR technique could achieve better performance on mobile phones because mobile phones have smaller screen size and relatively higher pixel density. We deploy the OCR technique into mobile platforms and show its performance and effectiveness through real experiments. The tool we use is Tesseract [7], which is one of the most accurate open source OCR engines and can support over 60 languages. Our testing uses a Thinkpad T420 laptop (2.40GHz, 4GB RAM) with pixel density of 131 dpi and a Google Nexus 4 smartphone (1.5GHz, 2GB RAM) with 320 dpi.

We open the Ebay mobile login page in both mobile and PC browsers, and each captures a screenshot. Tesseract is used to extract text from the screenshot taken on mobile phone while Microsoft Office Document Imaging (MODI) is used to process the screenshot of PC browser (this tool is used in a previous anti-phishing scheme [8]). We find that Tesseract extracts all words correctly from the mobile screenshot, except the “sign in” in dark blue button (Figure 3(a)). The extraction result on PC is worse as it not only missed the dark blue button, but also missed the word “ebay” in the logo. Besides, the OCR extraction on mobile phones only takes 1.6 seconds, while on PC the processing time is 4.5 seconds.

IV. The MobiFish Scheme

In this section, we present an automated lightweight scheme for mobile phishing defense named MobiFish. MobiFish consists of two independent components designed for mobile web pages and mobile applications, called WebFish and AppFish respectively.

A. The WebFish Scheme

The work flow of WebFish is given in Figure 4. As we can see, the defense scheme is initiated with URL loading. When a browser attempts to load a web page, WebFish first scans its URL to see whether the domain name is an IP address. Legitimate websites always use domain names as verification of their identities while phishers are likely to list IP address in URL to disguise their fake identities. Next, WebFish obtains the HTML source code of the loading page and checks if there is any form in that page. The existence of form is important since phishers also need a form with *input* tag which allows user to enter (confidential) information and then submit. WebFish checks forms to speed up the detection, but the core

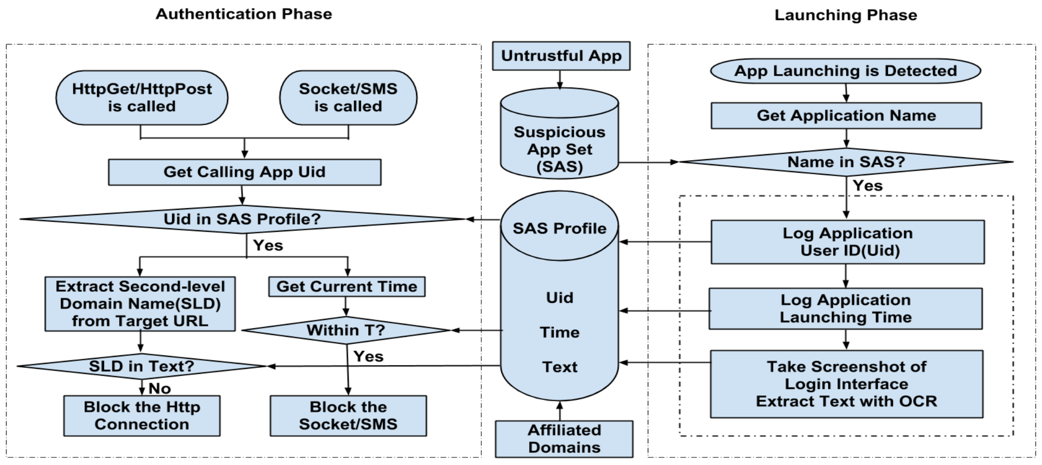


Fig. 5. The Work Flow of AppFish

module of identity extraction does not rely on any part of HTML source code. If a form is found, WebFish starts the identity extraction and verification. On one hand, it extracts second-level domain name (SLD) from URL, which represents the actual identity of the site. Then the SLD is indexed in the Mapping White-List (MWL). If it matches any of the SLD–Brand names in the MWL, the original SLD is replaced by the corresponding brand name. On the other hand, it calls the *screencap* native function to take a screenshot of the login page and extract the text with the OCR tool. Note that the URL shown in the URL bar may also be captured in the screenshot, and it should be removed from text since it reflects the actual identity of the site. The existance of a URL bar in the screenshot can be determined by whether the first line contains one of the top-level domain names (e.g. gov, edu, com, and org). To reduce the false-positive rate, we further search for sensitive terms such as “username”, “password”, and “credit card number” in the text. If not found, the form may be just used for search or general data input purposes and the page is marked as safe. Otherwise, the last step is to search the SLD in text. If not found, it is probably a phishing site. WebFish shows a notification window to the user indicating the high possibility of a phishing attack, along with the URL of the suspected web page.

Our design is based on the assumption that if the domain name of the phishing site appears in the fake login page of legitimate sites, users can immediately discern the difference and check the URL to verify the validity of this web page. This is reasonable since as far as we know, no phishing site uses common terms in login page like “sign”, “username”, “password”, or “welcome” exactly as a second-level domain name (SLD). Legitimate mobile login pages are made very simple and clear. It is highly unlikely for these well constructed and maintained web pages to have strange words appear on them. Thus, users would become alerted if a web page contains text different from the brand name or common login terms (such as welcome). If the attacker adds the phishing domain name in tiny font size to prevent the user from noticing, then the OCR is not able to recognize it either, and WebFish will still detect it as a phishing site. The key feature for WebFish to detect a phishing URL is that the SLD is not among the text extracted from the screenshot of the login page.

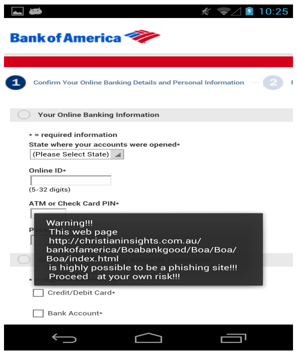
B. *The AppFish Scheme*

The work flow of AppFish is shown in Figure 5. AppFish maintains a database called suspicious app set (SAS), which contains profiles of untrusted apps including user ID (Uid), launching time, and screenshot text. Users can add the apps they suspect into SAS, and only apps listed in SAS are under the monitoring of AppFish. These apps are characterized as

1. Specified for one company. This is to ensure that the app only contacts the company’s official sites or affiliated (partners) servers. The domain name of the collaborators are collected and added into the SAS profile in advance. Having multiple domains often happens to websites that need extra storage. For example, we find in our testing that Facebook always requests data from domains like *fbcdn.net* and *akamaihd.net*. This is because Facebook uses them as a content delivery network (CDN). The substantial amount of photos generated by Facebook users are uploaded to *akamaihd.net* instead of *facebook.com*. Whenever a user want to view a photo, the request is actually sent to the nearest akamaihd server.

2. Have the user login. There are lots of apps that do not need users to login, like browsers and apps for news, music, maps, etc. In these cases, phishing attacks would not happen at all. For browsers, web page login is protected by WebFish.

The AppFish defense scheme works in two phases: launching phase and authentication phase. In launching phase, AppFish obtains the name of each launching application and searches for it in SAS. If found, the logging process begins, in which AppFish takes a screenshot of the login interface and extracts the text with the OCR tool. Then, the text along with the application Uid and launching time are logged into the profile of that app. After the user has entered the credentials and clicks the “sign in” button, the authentication phase begins. Legitimate applications (like Facebook and Twitter) usually send the user’s credentials to a remote server for authentication via HttpGet/HttpPost. Once the credentials are verified, the application loads data belonging to the account. On the contrary, phishing apps are not able to load user data,


10:25

Bank of America

1 Confirm Your Online Banking Details and Personal Information 2

Your Online Banking Information

* = required information

State where your accounts were opened*

(Please Select State)

Online ID*

(5-32 digits)

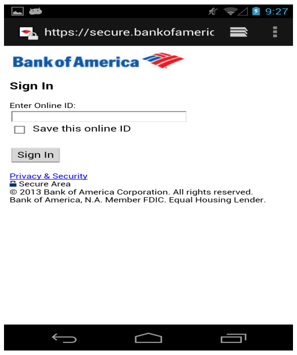
ATM or Check Card PIN*

Warning!!!
This web page
http://christianinsights.com.au/bankofamerica/Boabankgood/Boa/Boa/Boa/index.html
is highly possible to be a phishing site!!!
Proceed at your own risk!!!

Credit/Debit Card*

Bank Account*

(a) BOA Phishing Login Page


9:27

https://secure.bankofameric

Bank of America

Sign In

Enter Online ID:

Save this online ID

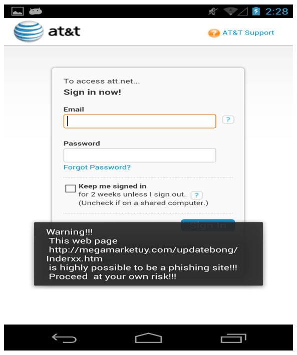
Sign In

Privacy & Security

Secure Area

© 2013 Bank of America Corporation. All rights reserved.
Bank of America, N.A. Member FDIC. Equal Housing Lender.

(b) BOA Official Login Page


2:28

at&t AT&T Support

To access att.net...

Sign in now!

Email

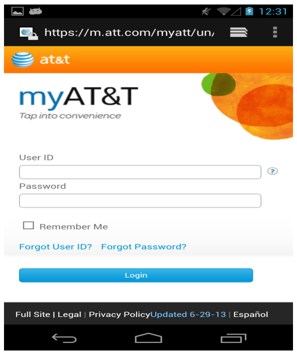
Password

Forgot Password?

Keep me signed in
for 2 weeks unless I sign out.
(Uncheck if on a shared computer.)

Warning!!!
This web page
http://megamarketuy.com/updatebong/Inderxx.htm
is highly possible to be a phishing site!!!
Proceed at your own risk!!!

(c) AT&T Phishing Login Page


12:31

https://m.att.com/myatt/un/

at&t

myAT&T
Tap into convenience

User ID

Password

Remember Me

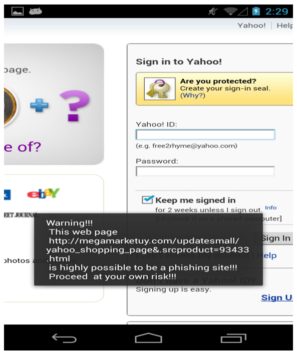
Forgot User ID? Forgot Password?

Login

Full Site | Legal | Privacy PolicyUpdated 6-29-13 | Español

(d) AT&T Official Login Page

Fig. 6. Experiments with WebFish on Mobile Phishing Login Page


2:29

Yahoo! | Help

Sign in to Yahoo!

Are you protected?
Create your sign-in seal.
(Why?)

Yahoo! ID:

(e.g. free2rhyme@yahoo.com)

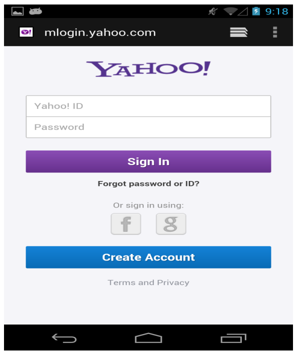
Password:

Keep me signed in
for 2 weeks unless I sign out. Info

Warning!!!
This web page
http://megamarketuy.com/updatesmall/yahoo_shopping_page&.srcproduct=93433.html
is highly possible to be a phishing site!!!
Proceed at your own risk!!!

Signing up is easy.

(a) Yahoo Phishing Login Page


9:18

mlogin.yahoo.com

YAHOO!

Yahoo! ID

Password

Sign In

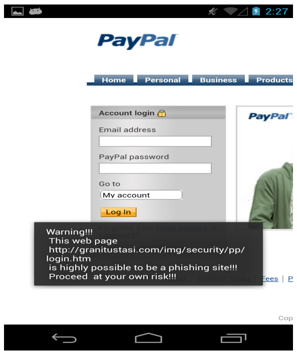
Forgot password or ID?

Or sign in using:

Create Account

Terms and Privacy

(b) Yahoo Official Login Page


2:27

PayPal

Home | Personal | Business | Products

Account login

Email address

PayPal password

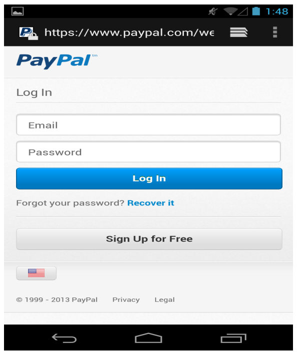
Go to

My account

Log In

Warning!!!
This web page
http://granitustasi.com/img/security/pp/login.htm
is highly possible to be a phishing site!!!
Proceed at your own risk!!!

(c) PayPal Phishing Login Page


1:48

https://www.paypal.com/we

PayPal

Log In

Email

Password

Log In

Forgot your password? Recover it

Sign Up for Free

© 1999 - 2013 PayPal Privacy Legal

(d) PayPal Official Login Page

Fig. 7. Experiments with WebFish on Conventional Phishing Login Page

and the only trick they can play is to tell the user that he or she has entered the wrong password. However, after two or three times, most users will suspect the validity of the application and uninstall it. Hence, a phishing app is able to send out the user credentials only during the period from submission to uninstallation. Appfish monitors the possible paths for a phishing app to transmit data to outside world, which include HttpGet/HttpPost, socket, SMS and email (email is also based on socket). If an application calls any of these ways to send out information, AppFish checks whether it is one of the suspects in SAS. If confirmed, http connections (HttpGet/HttpPost) are filtered while other communications (socket/sms) are blocked. For all URLs the suspicious app requests to connect with, AppFish ensures that the SLD name appears among the text or affiliated domain names stored in SAS profile. The idea is the same as that in WebFish. Meanwhile, socket and SMS function could be blocked for certain amount of time T, which should be long enough for the user to discover and uninstall the malicious app. Thus, for phishing applications, they will not be able to send out credentials before being removed by user.

Note that we have to compare the SLD to the text extracted from the login page. The reason why extracted text (instead of the application name) is used is that: for phishing attacks based on task interception, phishers could develop an app with the same name as the SLD of the phishing server, and can pop up fake a login page. For instance, mobile user downloads a phishing app named "abc" due to its tempting fancy functions. However, this phishing app could pop up a fake Facebook login interface as soon as the legitimate Facebook launches. Once the user is spoofed, the app "abc" immediately sends the credentials to the phishing server "abc.com". In this example, the foreground (fake) application name is the same as the SLD of phishing server but instead, we would not find "abc" in the phishing login interface.

V. PERFORMANCE EVALUATION

We implement MobiFish on a Google Nexus 4 smartphone running the Android 4.2 operating system. We modify the source code of the Android system so that it is able to support MobiFish. The MobiFish scheme may be applied to other mobile platforms as well. To evaluate the performance of MobiFish, we conduct experiments for WebFish and AppFish separately.

A. *Experiments with WebFish*

In the process of evaluating WebFish, we were not able to collect enough phishing URLs specified for mobile web pages. Instead, we randomly picked up 100 phishing URLs from *PhishTank.com* in 2013. Although all the phishing URLs have been blocked by PC browsers like Chrome, they are accessible through mobile browsers (including both Android's built-in browser and Chrome for Android). This fact highlights the significance of WebFish, which can provide web phishing defense for mobile platforms. In our experiments, WebFish can effectively mark all the phishing URLs as dangerous, and can warn the user. Figures 6(a) and 6(c) show that WebFish displays alertness because it is not able to find the SLD inside the mobile phishing login pages of "*Bank of America*" and "*AT&T*".

Meanwhile, we have two observations for the conventional (PC) phishing web pages. First, a large number of them are in high similarity to their legitimate counterparts, and the input forms in phishing login pages are often surrounded by brand names or company logos, as legitimate login pages do. Second, when loading large conventional web pages, mobile browsers often display the area that contains the input form instead of displaying an overview of the entire web page. Figure 7(a) and 7(c) show the phishing login pages of Yahoo and Paypal. As we can see, the brand name Yahoo and Paypal logo appear more than once in the input form area that is presented to user. Both of them are reported as phishing sites since WebFish cannot find the SLD in the screenshot.

To evaluate the performance of WebFish on legitimate web pages, we use the URLs of the corresponding official login web pages for comparison. Figures 6(b), 6(d), 7(b) and 7(d) are the official mobile login pages; WebFish successfully verifies the validity of these pages and no warning is generated. WebFish's ability to verify the legitimate *AT&T* web page shows that the Mapping White-List (MWL) scheme works for company websites with different brand name and SLD. The 19 corresponding legitimate mobile login web pages can demonstrate WebFish's performance on legitimate web pages. Table I summarizes the testing results of phishing URLs (Column 1, 2, 3) and legitimate URLs (Column 4, 5). The "MWL" behind legitimate SLD name means that MWL is used to convert the SLD to the brand name.

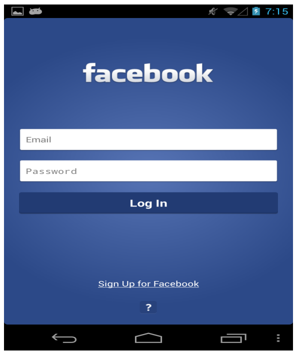
TABLE I. SUMMARY OF TESTED URLS

Website	Phishing Samples	Phishing Feature Found	Legitimate Mobile Login SLD	Verification of Legitimate URLs
Amazon	6	100%	amazon	✓
AOL	3	100%	aol	✓
Apple	5	100%	apple	✓
AT&T	2	100%	att (MWL)	✓
Bank of America	10	100%	bankofamerica	✓
Barclays	4	100%	barclays	✓
Chase	5	100%	chase	✓
Citi	4	100%	citibank (MWL)	✓
Ebay	8	100%	ebay	✓
Facebook	5	100%	facebook	✓
Hotmail	2	100%	live (MWL)	✓
HSBC	8	100%	hsbc	✓
Microsoft	1	100%	live (MWL)	✓
NAB	1	100%	nab	✓
NatWest	3	100%	nwolb (MWL)	✓
PayPal	12	100%	paypal	✓
Vodafone	4	100%	vodafone	✓
Wells Fargo	7	100%	wellsfargo	✓
Yahoo	10	100%	yahoo	✓
Total:	100	100%	Tot: 19	100%

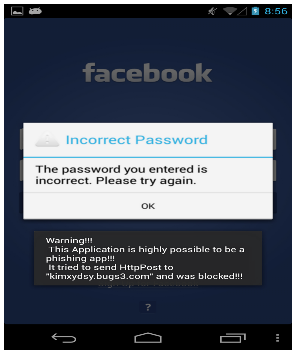
Table I shows that: (1) WebFish is able to find key features of phishing web pages for 100% tested phishing URLs; and (2) WebFish achieves 100% verification rate of legitimate URLs. The results demonstrate the effectiveness of WebFish in detecting mobile phishing sites.

B. Experiments with AppFish

When we conduct experiments with AppFish, there are only a few reported phishing applications and none of them is available online. To test the effectiveness of AppFish, we develop two sample phishing applications. Figure 8(a) shows the login interface of the fake Facebook apps developed by us. Most users would not be able to discern its being differemt from legitimate Facebook. Our first phishing application runs by itself and pretends to be the real Facebook app. The second one runs when it detects that the real Facebook app is launching, and it pops up a fake login interface to cover the real Facebook app. After a user clicks the "Log in" button, the fake apps send the credentials to us by HttpGet, HttpPost, socket, SMS, and email, respectively. Meanwhile, a notice window is displayed, informing the user of an incorrect password, and prompts for another try. When AppFish runs, it is able to block all the requests and warn users about the phishing attempts. Figure 8(b) (lower part) shows the warning generated by AppFish for the phishing attack.



(a) Fake Facebook App



Incorrect Password

The password you entered is incorrect. Please try again.

Warning!!!
This Application is highly possible to be a phishing app!!!
It tried to send HttpPost to "kimxydsy.bugs3.com" and was blocked!!!

(b) Warning of Phishing App

Fig. 8. Experiments with AppFish on Mobile Applications

VI. RELATED WORK

Web users have been suffering from phishing attacks since their first appearance in 2003. Researchers have proposed many solutions (such as alert protection and phishing detection) to defend against phishing attacks.

Alert protection is a simple notification when a user is entering sensitive information. Kirda et al. proposed AntiPhish [9], which tracks the sensitive information of a user and generates warnings whenever the user attempts to give away this information to a website that is considered untrusted. However, this scheme cannot automatically check and detect phishing attacks. Instead, users have to judge by themselves after being warned.

In addition, many phishing detection tools have been designed for phishing on PC web pages. Based on the methodology used, they can be generally categorized into two groups: heuristics schemes and blacklist schemes. Heuristics schemes outperform blacklist schemes since they can deal with new phishing sites without the need of waiting for update. Usually, heuristics schemes for phishing detection utilize other techniques such as machine learning techniques [10], [11], [12] and search engine [12], [13]. CANTINA [13] is a content-based approach to detecting phishing websites, and it adopts TF-IDF information retrieval algorithms. Garera et al. [10] proposed a heuristics-based scheme which identified several generic features of phishing URLs, and used these features in a logistic regression classifier. CANTINA+ [12] is a comprehensive feature-based solution for phishing web page which combines machine learning and search engine techniques. However, existing heuristics used in phishing detection are all based on features extracted from HTML source code. As we showed in section III, HTML source code should not be trusted since it may not reflect the actual content presented to users.

Based on the assumption that the most spoofing phishing sites are those whose visual appearances look identical or very similar to authentic sites [14], [15], several similarity based phishing detection approaches are proposed. SpoofGuard [16] uses URLs, images, links, and domain names to check the similarity between a given page and the pages previously stored. Afroz et al. proposed PhishZoo [17] that uses the profiles of trusted websites' appearances built with fuzzy hashing techniques to detect phishing. PhishZoo makes profiles of sites that consist of fuzzy hashes of several common content elements (e.g. URL, images, most used texts, HTML codes, script files, etc.), which are related to their structure and appearance. They further enhanced their phishing detection scheme by adding displayed images into profiles and utilizing SIFT image-matching algorithm [18]. However, similarity-based approaches cannot detect phishing sites with different appearances.

GoldPhish [8] utilizes optical character recognition (OCR) technique for phishing detection on PC browsers. OCR is used to extract text from images found on web pages, such as the company logo, and then it is compared to the top ranked domains from Google' s search service. However, OCR performance on PC is demonstrated to be limited in both speed and accuracy. And our lightweight scheme works with mobile browser and does not depend on external search engine.

Mobile Phishing is emerging as a significant threat for mobile users. iPhish [5] discusses the weaknesses caused by the hardware limitation of mobile devices. Felt et al. [4] examined the mobile phishing threats by detailing several phishing attack models during control transfers. In terms of solutions for mobile phishing, we only found one piece of work proposed by Jie et al. [19], in which they load hooks into iOS so that the system interrupts the user when sensitive information is being entered into applications not in the whitelist, and prompts the user to decide whether to continue or not. However, this idea is quite similar to AntiPhish [9], which only provides a warning, rather than detection and defense.

VII. CONCLUSION

In this paper, we studied the important issue of mobile phishing detection. We proposed MobiFish, a novel mobile phishing defense scheme. We identified the weaknesses of the heuristics-based anti-phishing schemes that highly rely on the HTML source code of the web page. MobiFish resolves this issue by using OCR, which can accurately extract text from the screenshot of a mobile login interface such that the claimed identity can be verified. Compared to existing OCR-based anti-phishing schemes (designed for PC only), Mobifish is lightweight as it works without using external search engines or machine learning algorithms. We implemented MobiFish on a Google Nexus 4 smartphone running the Android 4.2 operating system. Our experimental evaluation demonstrated that MobiFish can effectively detect and defend against mobile phishing attacks.

ACKNOWLEDGMENT

This research was supported in part by the US National Science Foundation (NSF) under grants CNS-0963578, CNS-1022552, and CNS-1065444.

REFERENCES

[1] Anti-Phishing Working Group, "Phishing activity trends report," http://www.antiphishing.org/reports/apwg_report_june_06.pdf, 2006.

[2] L. F. Cranor, S. Egelman, J. I. Hong, and Y. Zhang, "Phinding phish: Evaluating anti-phishing tools," *In Proceedings of The 14th Annual Network and Distributed System Security Symposium (NDSS '07)*, February, 2007.

[3] Trend Micro, "Mobile phishing: A problem on the horizon," 2012.

[4] A. P. Felt and D. Wagner, "Phishing on mobile devices," *In Proceedings of W2SP'11: WEB 2.0 Security and Privacy*, 2011.

[5] Y. Niu, F. Hsu, and H. Chen, "iphish: phishing vulnerabilities on consumer electronics," in *Proceedings of the 1st Conference on Usability, Psychology, and Security*, ser. UPSEC'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 10:1–10:8.

[6] C. Karlof, J. D. Tygar, and D. Wagner, "Conditioned-safe ceremonies and a user study of an application to web authentication," in *Proceedings of the 5th Symposium on Usable Privacy and Security*, ser. SOUPS '09. New York, NY, USA: ACM, 2009, pp. 38:1–38:1.

[7] Tesseract OCR, http://code.google.com/p/tesseract-ocr/.

[8] M. Dunlop, S. Groat, and D. Shelly, "Goldphish: Using images for content-based phishing analysis," in *Internet Monitoring and Protection (ICIMP), 2010 Fifth International Conference on*, 2010, pp. 123–128.

[9] E. Kirda and C. Kruegel, "Protecting users against phishing attacks with antiphish," in *Proceedings of the 29th Annual International Computer Software and Applications Conference - Volume 01*, ser. COMPSAC '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 517–524.

[10] S. Garera, N. Provos, M. Chew, and A. D. Rubin, "A framework for detection and measurement of phishing attacks," in *Proceedings of the 2007 ACM workshop on Recurring Malcode*, ser. WORM '07. New York, NY, USA: ACM, 2007, pp. 1–8.

[11] Y. Pan and X. Ding, "Anomaly based web phishing page detection," in *Proceedings of the 22nd Annual Computer Security Applications Conference*, ser. ACSAC '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 381–392.

[12] G. Xiang, J. Hong, C. P. Rose, and L. Cranor, "Cantina+: A feature-rich machine learning framework for detecting phishing web sites," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 2, pp. 21:1–21:28, Sep. 2011.

[13] Y. Zhang, J. I. Hong, and L. F. Cranor, "Cantina: a content-based approach to detecting phishing web sites," in *Proceedings of the 16th international conference on World Wide Web*, ser. WWW '07. New York, NY, USA: ACM, 2007, pp. 639–648.

[14] J. S. Downs, M. B. Holbrook, and L. F. Cranor, "Decision strategies and susceptibility to phishing," in *Proceedings of the second symposium on Usable privacy and security*, ser. SOUPS '06. New York, NY, USA: ACM, 2006, pp. 79–90.

[15] M. Jakobsson, A. Tsow, A. Shah, E. Blevis, and Y.-K. Lim, "What instills trust? a qualitative study of phishing," in *Proceedings of the 11th International Conference on Financial cryptography and 1st International conference on Usable Security*, ser. FC'07/USEC'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 356–361.

[16] N. Chou, R. Ledesma, Y. Teraguchi, and J. C. Mitchell, "Client-side defense against web-based identity theft," *In Proceedings of 11th Annual Network and Distributed System Security Symposium (NDSS '04)*, February, 2004.

[17] S. Afroz and R. Greenstadt, "Phishzoo: An automated web phishing detection approach based on profiling and fuzzy matching," Drexel University, Tech. Rep., 03 2009.

[18] ——, "Phishzoo: Detecting phishing websites by looking at them," in *Semantic Computing (ICSC), 2011 Fifth IEEE International Conference on*, 2011, pp. 368–375.

[19] J. Hou and Q. Yang, "Defense against mobile phishing attack," Computer Security Course Project, http://www-personal.umich.edu/ỹangqi/pivot/mobile_phishing_defense.pdf, 2012.