IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON PRIVACY PRESERVATION FOR LARGE-SCALE USER DATA IN SOCIAL NETWORKS

Received July 25, 2017, accepted August 11, 2017, date of publication August 23, 2017, date of current version September 19, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2743528

Phishing-Alarm: Robust and Efficient Phishing Detection via Page Component Similarity

JIAN MAO[1], (Member, IEEE), WENQIAN TIAN[1], PEI LI[1], TAO WEI[2], (Member, IEEE), AND ZHENKAI LIANG[3], (Member, IEEE)

[1]School of Electronic and Information Engineering, Beihang University, Beijing 100191, China
[2]Baidu USA LLC, Sunnyvale, CA 94089, USA
[3]School of Computing, National University of Singapore, Singapore 117417

Corresponding author: Jian Mao (maojian@buaa.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402029, in part by the National Natural Science Foundation of China under Grant 61370190 and Grant 61379002, and in part by the Singapore Ministry of Education, NUS, under Grant R-252-000-539-112.

ABSTRACT Social networks have become one of the most popular platforms for users to interact with each other. Given the huge amount of sensitive data available in social network platforms, user privacy protection on social networks has become one of the most urgent research issues. As a traditional information stealing technique, phishing attacks still work in their way to cause a lot of privacy violation incidents. In a Web-based phishing attack, an attacker sets up scam Web pages (pretending to be an important Website such as a social network portal) to lure users to input their private information, such as passwords, social security numbers, credit card numbers, and so on. In fact, the appearance of Web pages is among the most important factors in deceiving users, and thus, the similarity among Web pages is a critical metric for detecting phishing Websites. In this paper, we present a new solution, called Phishing-Alarm, to detect phishing attacks using features that are hard to evade by attackers. In particular, we present an algorithm to quantify the suspiciousness ratings of Web pages based on the similarity of visual appearance between the Web pages. Since cascading style sheet (CSS) is the technique to specify page layout across browser implementations, our approach uses CSS as the basis to accurately quantify the visual similarity of each page element. As page elements do not have the same influence to pages, we base our rating method on weighted page-component similarity. We prototyped our approach in the Google Chrome browser. Our large-scale evaluation using real-world websites shows the effectiveness of our approach. The proof of concept implementation verifies the correctness and accuracy of our approach with a relatively low performance overhead.

INDEX TERMS Phishing, Web security, browser, privacy protection.

I. INTRODUCTION

Social networks have become one of the most popular platforms for users to interact with each other. Users use social networks to communicate and share. Given the huge amount of social data available in social network platforms, user privacy protection has become one of the most urgent research issues. As a traditional information stealing technique, phishing attacks still work in their way that caused a lot of privacy violation incidents.

Phishing is a form of social engineering in which an attacker mimics electronic communications to lure users to provide their confidential information. Such communications are usually through emails that trick users to visit fraudulent web sites that in turn collect users' private information, such as passwords, credit card numbers, and social security numbers. Originated in the early 1990's on the America Online (AOL) network [16], phishing attacks have been evolving with increasing sophistication. For example, targeted phishing attacks customize the phishing pages for a particular user or user group with their personal information, such as photos of their family members. In synchronized phishing attacks, attackers use victim credentials immediately after they are collected, which allows attackers to authenticate to systems protected by token-based two-factor authentications [34].

In a web-based phishing attack, an attacker sets up phishing web pages (pretending to be an important web site such as a social network portal) to lure victim users to input their private information, such as, passwords, social security numbers, and credit card numbers, etc. The attacker usually sends

emails or publishes web links on social networks that trick users to visit the phishing pages. As social networks become a convenient platform to initial social engineering attacks, phishing attacks are on the rise. According to a PhishMe report [1], the number of phishing emails has exceeded six million in the first quarter of 2016, more than 6-fold increase compared to the last quarter of 2015.

To detect phishing attacks, one class of solutions is based on analysis of URLs of phishing pages [6], [20]. Meanwhile, additional features of a web page are suggested to be included in anti-phishing mechanisms [12], [15], [35], e.g., registration dates, life time, or lock sign of SSL-enabled web sites, etc. Though solutions based on URL information can be easily integrated into browsers, attackers have flexibility in changing URL features to evade detection, as these features are not the key features in deceiving users.

One key feature of phishing pages is that they usually maintain the similar visual appearance as their target pages. Therefore, another class of solutions is based on page content similarity. With the help of search engines, researchers developed techniques comparing the similarity of texts on pages between the current web page and other known web pages [31], [46]. However, attackers can add noises to their web page texts by embedding invisible web contents to bypass these solutions. Other solutions [3], [8], [21] compare images of rendered pages to evaluate their visual similarity. However, these approaches are not efficient due to the effort needed to render the page. In addition, the effectiveness of the detection is significantly influenced by the differences of browser rendering mechanisms. In our earlier investigation [25], we note that page contents and page layouts are fundamental features that determine web pages' appearance, so we use the Cascading Style Sheets (CSS) of web pages to represent these features. However, not all page elements and CSS style rules are of the same importance in deceiving the users. The insignificant elements and style rules can be leveraged by attackers to bypass detection.

In this paper, we propose a new technique, called Phishing-Alarm, to efficiently and accurately detect phishing pages according to the similarity of page visual features.[1] As the visual appearance of a web page is jointly determined by its elements and its CSS rules, our approach first evaluates the impact these elements and rules have on the final page appearance to users, and selects the *effective* features as the basis of our approach. In this way, our decision will not be affected by elements that have little influence on the visual appearance. Based on these effective features, we develop a new algorithm to effectively measure the similarity of page components between a suspicious page and its potential target. We detect phishing pages based on the similarity.

We prototyped our approach as an extension to the Google Chrome browser. We evaluated it using more than 7,000 phishing pages collected from *phishtank.com*. The experiment results show the advantages of Phishing-Alarm over existing solutions in accuracy and performance.

In summary, we made the following contributions in this paper:

- We propose a robust solution to identify phishing pages according to the visual similarity of web page components, which are difficult to be evaded by attackers.
- We develop techniques to select the effective features on a web page, and propose an efficient method for page similarity detection according to these features.
- We prototyped our approach and evaluated it using a large set of phishing pages. The results illustrate that our approach is efficient and effective.

II. RELATED WORK

A. BLACK/WHITE-LIST BASED DETECTION

Blacklist/whitelist-based detection is the most widely deployed anti-phishing techniques used in browsers, like Google's safe browsing API [13] and Mozillia's Firefox [29]. Bayesian networks are used in anti-phishing toolbars [6], [20] together with black-list databases of phishing sites. Anti-Phishing Whitelist (APWL) [38] allows each user to establish a personal whitelist and refuses any request that sending sensitive data to websites beyond the whitelist. Automated individual white-list (AIWL) [14] employs a Naive Bayesian classifier to automatically maintain an individual white-list of a user. Belabed et al. [2] proposed a personalized whitelist approach combined with a support vector machine (SVM) classifier. The phishing pages that are not blocked by the whitelist are passed to SVM for further analysis. Blacklist/whitelist-based detection is easy to implement, therefore it is widely employed by browser toolbars or extensions. Nevertheless Blacklist and whitelist based detection both suffer from high false negative due to the short lifetime of phishing web pages.

The main weakness of the blacklist/whitelist-based solutions is in that they are not effective on pages that is previous unseen, and need to be constantly updated to maintain a good accuracy.

B. URL BASED DETECTION

URL-based detection techniques analyze URL features of web pages to filter out suspicious malicious websites. Ma et al. [23], [24] use online learning and statistical methods to discover the lexical and host-based properties of malicious website URLs. Khonji et al. [18] present a modified variant of a website classification technique to filter phishing URLs in e-mails. Their previous work lexically analyzes URL tokens to increase prediction accuracy [17]. Mohammad et al. [28] extract 17 features of websites (e.g. URL length, specific URL symbols, domain name, domain year, etc.) and manually set a rule for each feature to determine whether a link is malicious. Zhang et al. [44] introduced a statistical machine learning classifier to detect the phishing sites based on some new aspects of the common features that appear in the phishing URLs. Nguyen et al. [30] proposed a neuro-fuzzy model

[1]This version of paper is extended from the conference version [26] of our work

that combines neural networks and fuzzy systems without using rule sets for phishing detection by analyzing features of URL. Zouina and Outtaj [47] presented a lightweight phishing detection approach completely based on the URL. The proposed system uses only six URL features to perform the recognition. The mentioned features are the URL size, the number of hyphens, the number of dots, the number of numeric characters plus a discrete variable that correspond to the presence of an IP address in the URL and finally the similarity index.

Lee et al. [19] exploit a linear chain CRF model to study users' web browsing behaviors faced phishing situations and then make behavioral prediction for context-aware phishing detection. Experiments were made to show good performance for prediction and blocking of phishing threats from user behaviors. Wu et al. [41] present an automated lightweight anti-phishing method for mobile phones, MobiFish. In this scheme, actual identity is compared to the claimed identity of webpages and applications.

C. CONTENT BASED DETECTION

Eric et al. proposed a phishing detection scheme [27] based on the visual similarity between a page and other target pages. The features used include: text and styles, images in the page, and the overall visual appearance of the page. Chen et al. [4] proposes an approach for detecting visual similarity between two web pages using algorithmic complexity theory. *CANTINA* [46] detects phishing web sites based on page contents, using term frequency-inverse document frequency (TF-IDF) combined with other heuristics. The browser plugin *SpoofGuard* [5] checks domain name, URL, link and image check to determine whether a page is phishing. Search engines are also used to help check similarity to popular pages [31], [46]. The above approaches are not resilient to evasions, where attackers can change the contents used by the above solution, but still can lure the victim users.

Content-based approaches generally extract content features of web pages to identify suspicious websites. To deal with such evasion attempts, some solutions [3], [8], [21] compare images of rendered pages to evaluate their visual similarity. *GoldPhish* [11] captures an image of the page, employs optical character recognition (OCR) to convert the image into text (especially the company logo), leverages the Google PageRank algorithm to retrieve the top ranked domains from a search engine and compares them with the current page. Pan and Ding [32] extract textual clues from the DOM tree of websites to detect anomalies in DOM objects and HTTP transactions in the page based on the fact that phishing pages activate more abnormal behaviors compared to benign pages. Zhang et al. [45] extract spatial layout characteristics as rectangle blocks from a given page and compute spatial layout similarity between the current page and the real one based on spatial characteristic matching algorithms. Furthermore, they leverage an R-tree index algorithm to query similar-looking web pages in a spatial feature library and thus determine whether a web page is imitating another one. Wardman et al. [39] proposed a cadre of file matching algorithms to calculate file similarity between two pages so that they can effectively filter out potential phishing web pages.

D. ACTIVE RESPONSE TO PHISHING

Yue et al. [43] developed an active solution to respond to phishing attacks with "bogus bites." It actively provides bogus credentials to a suspected phishing site. It conceals credentials that can trigger response actions after being used, among bogus credentials. In this way, a legitimate web site can quickly identify stolen credentials.

Phoolproof phishing prevention [33] proposed a two-factor authentication system to defeat phishing attackers. The system uses mobile phone as the trusted device to provide mutual authentication. *Phishpin* [36] is an identity-based anti-phishing approach proposed by Tout et al. The proposed scheme is a cryptography-based system that requires the verification of and online entity's certificate. It generates browser-side OTP based on the server-side OTP to enforce the authenticity of online entities, and thus can prevent attackers from masquerading as legitimate users.

E. PHISHING DETECTION USING OTHER FEATURES

According to the features of the X.509 certificates of the websites, Dong et al. proposed a machine-learning based approach to detect phishing webpages [9], [10].

WOT [40] and *iTrustPage* [15], [35] decide whether a page if a phishing page based on page reputation, which are either reported by the anti-phishing community or derived from other web pages. The basis of such approaches, user opinion, may be biased.

Liu et al. [22] develop a method that is based on the semantic link network (SLN) of web pages. It constructs the SLN from known malicious pages and pages related to them. It uses SLN to discover implicit relations among web pages to identify phishing pages.

Compared to solutions in this category, Phishing-Alarm is based on the fundamental visual features of web pages. These features are not easily changed by attacker without impacting the effectiveness of phishing attacks.

III. OVERVIEW

In this section, we introduce the background of web pages' visual representation, and give an overview of our solution.

A. CASCADING STYLE SHEETS (CSS)

The visual appearance of a web page is decided by its page layout and contents. To achieve a consistent appearance across all variants of web browsers, web developers use Cascading Style Sheets (CSS) as the standard technique to represent the layout of web pages. CSS includes a series of rules that specifies the visual properties of web page elements. The browsers retrieve the CSS specification of the web pages and render them accordingly.

Figure 1 gives an example of the CSS rule definition. A CSS rule consists of two parts [37], a *selector*, and a series of *declaration*s, in which the *selector* is an HTML elements

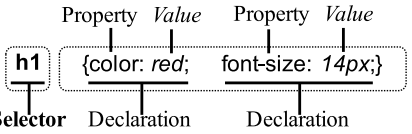


FIGURE 1. CSS rule representation.

and the *declaration* includes two components, *property* and *value*.

CSS has several types of selectors, e.g., tag selectors, id selectors, .class selectors and other selectors (for example, some attribute selectors). In a CSS rule, properties define the attributes of the specific elements, e.g., property color, font-size, font-family, border, margin, padding corresponding to the *paragraph* element. And the visual appearance of an element is specified by the *value* of the corresponding *property*. For instance, as shown in List 1, a developer can set the value of color property as "`#ff0000`"(red).

```
div { font-size: 2px; }
.c1 {color: #ff0000; }
#id1 {background-color:black; }
.c1 p{width:100px; }
```

Listing 1. A simple CSS rule set example.

B. MOTIVATION

To deceive the victim users effectively, attackers need to try the best to mimic the target web pages. According to our analysis of over 7,000 samples in `phishtank.com`, there are usually two typical approaches of using CSS to develop phishing pages.

- Attackers directly use the target web page's CSS link or copy and paste the target page's CSS content to a new created CSS file. Figure 3 present a phishing page whose CSS structure is copied from the CSS file of the web page shown in Figure 2. Most of the phishing samples from *phishtank.com* are created in this way.
- Attackers write a completely different CSS file that achieves the similar visual effect as the target web page as shown in Figure 4. However, a social network or financial web site usually contains a large amount of elements, resulting in a complicated CSS rule structure. Obviously, completely rewriting different CSS files that shows the similar visual appearance will cause extremely high overhead for attackers to develop such phishing pages. To be more efficient, attackers usually reuse some CSS properties defined in the original CSS files; therefore, the phishing page and its target page usually have same property sets in their CSS structures after being parsed by browsers.

Based on above observation, the basic idea of our approach is to detect phishing web pages based on the similarity of web pages' visual appearance (specifically, the visual similarity between the suspicious page and popular financial web pages), which is considered as the fundamental feature of the phishing attack.

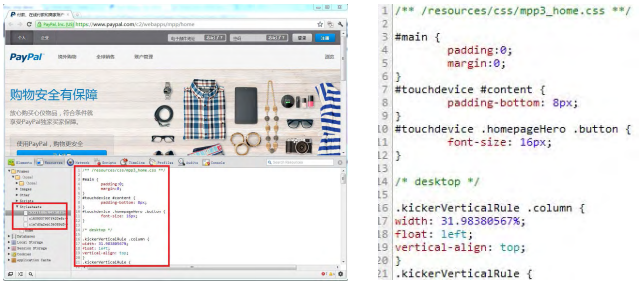


FIGURE 2. Original site.

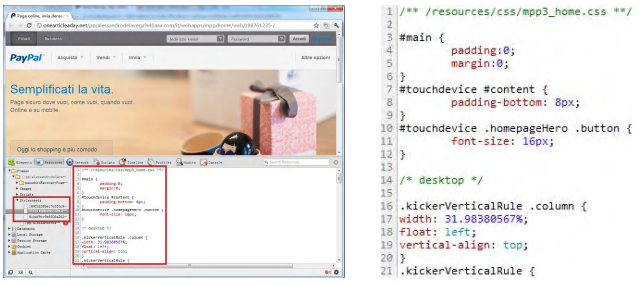


FIGURE 3. Phishing site (coping CSS document of the original site).

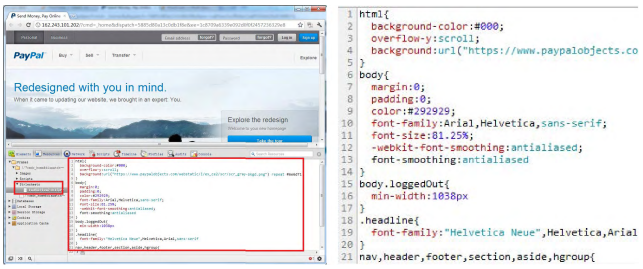


FIGURE 4. Phishing site (rewriting CSS document).

Furthering our discussion in Section II, there are two types of page-similarity-based phishing detection approaches that are widely studied: approaches based on text contents and approaches based on rendered pages. Text-based mechanisms [31], [46] use web pages' keyword frequency or sensitive text matching ratio as the metrics to detect the phishing pages. However, attackers may evade these approaches easily using the images to replace the corresponding web pages' content fragments, and attackers may also insert the invisible contents (e.g., by setting the contents the same color as the web pages' background). Both attacks can disable the text-based detection without affecting the visual layout of the phishing web pages. Rendered-page-based mechanisms [3], [8], [21] evaluate the pages' similarity by comparing the pixels of the rendered pages (the suspicious web page and the benign one). Unfortunately, these methods introduce high additional performance cost during image extraction.

To address the problems above, our approach extracts the static features of the web pages' layout structures rather than dynamically rendering the web pages. We extract and quantify the fundamental features in CSS files according to which

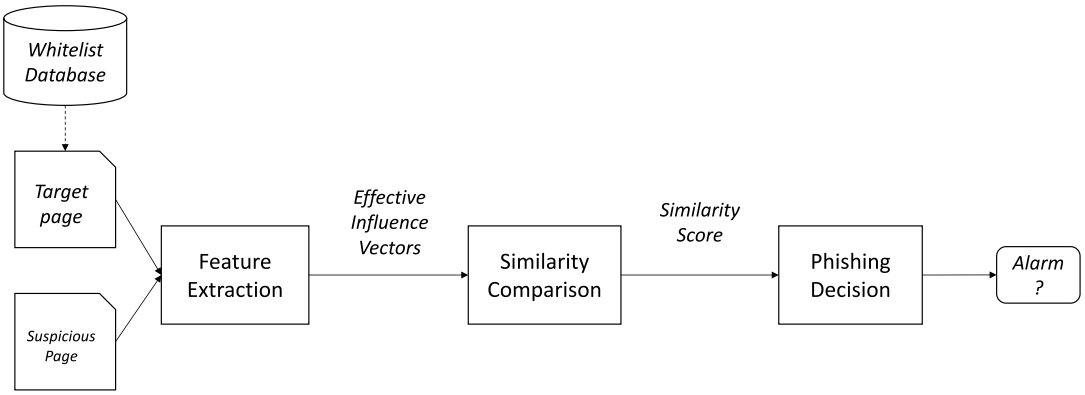


FIGURE 5. Overview of Phishing-Alarm.

we evaluate the similarity of web pages. We have validated this observation in a preliminary investigation [25].

C. PAGE FEATURE REPRESENTATION

CSS rules specify the layout/display features of page components. To decide page similarity, we need to represent the rules from another angle: the influence of page elements on page layout and appearance. For this purpose, we convert the captured CSS structure into a normalized representation, which we call *influence vector*. The normalized representation of the CSS structure is specified in Defintion 1.

Definition 1 (Influence Vector): Given the following CSS rule set of a web page,

$$\{\ldots, [Selector_i\{\ldots; [Property_j : Value_k; \ldots], \ldots\}], \ldots\},$$

its normalized specification, **influence vector**, is defined as

$$\{\ldots, [Property_j : [\ldots; Value_j^k : [\ldots, Selector_i^{j,k}; \ldots], \ldots], \ldots, \}.$$

According to the Definition 1, an influence vector mainly consists of two parts: a *property*, and one or more *declaration*s, each of which consists of a *value* and one or more *selector*s.

In addition, we classify the selectors into four categories *Tag, ID, Class* and *Others*. For example, selector *p*, *div* belong to the *Tag* category. Class selectors belong to the *Class* category; ID selectors belong to the *ID* category and the other selectors belong to the *Others* category.

More specifically, given a CSS rule set below,

$$\begin{aligned}
&div && \{padding : 2px;\},\\
&p && \{padding : 3px; color : \#ff0000\},\\
&.class1 && \{padding : 2px; color : \#ff0000\},\\
&.class2 && \{padding : 3px\},\\
&\#id1 && \{padding : 2px; color : \#ff0000\},\\
&\#id2 && \{padding : 3px; color : \#00ff00\}.
\end{aligned}$$

According to the definition of the influence vector, we get two following influence vectors from the original CSS rule set.

$$padding : \begin{bmatrix} \{\text{“}2px\text{”} : [\text{“}div\text{”}, \text{“}.class1\text{”}, \text{“}\#id1\text{”}]\}, \\ \{\text{“}3px\text{”} : [\text{“}p\text{”}, \text{“}.class2\text{”}, \text{“}\#id2\text{”}]\}. \end{bmatrix},$$

$$color : \begin{bmatrix} \{\text{“}\#ff0000\text{”} : [\text{“}p\text{”}, \text{“}.class1\text{”}, \text{“}\#id1\text{”}]\}, \\ \{\text{“}\#00ff00\text{”} : [\text{“}.class2\text{”}, \text{“}\#id2\text{”}, \text{“}\#id3\text{”}]\} \end{bmatrix}.$$

In the first influence vector the *Values* related to *Property* “`padding`” are “`2px`” and “`3px`”. While, “`div`”, “`.class2`”, and “`#id1`” are corresponding selectors of *Values* “`2px`”.

D. SYSTEM ARCHITECTURE

We show the architecture of Phishing-Alarm in Figure 5. Our approach works in three phases: *feature extraction*, *similarity computation*, and *phishing decision*.

In the first phase, feature extraction, given a suspicious page P_s, we extract its CSS structure $CSS(Sus)$, and covert it into the infection vector to represent the static feature of page P_s's visual layout. We also maintain a white-list database, which contains popular web pages targeted by phishing attacks. We extract the page features from the database using the same method.

In the second phase, based on the influence vectors, we match the similarity between the suspicious page and the pages in the whitelist database.

Finally, we make the decision by comparing the pages' similarity scores to a preset threshold ϵ. If the similarity scores are beyond ϵ and there exist other clues indicating that two testing pages are different, the suspicious page will be considered as a phishing page.

IV. PAGE COMPONENT SIMILARITY

In this section, we present our algorithm to measure page component similarity. This is the key part of our approach.

A. EFFECTIVE CSS FEATURE EXTRACTION

As we have noted, phishing pages typically keep its CSS style similar to their target pages. Based on this observation, a straightforward approach to detect phishing pages is to compare all CSS rules of two web pages and

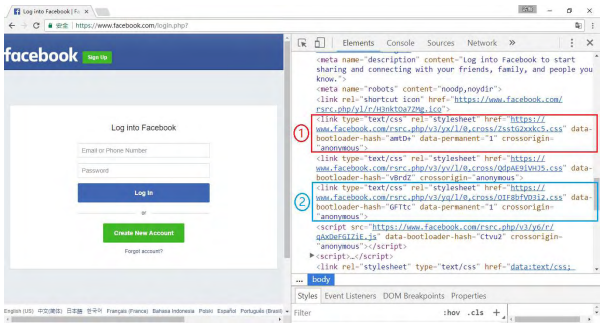


FIGURE 6. Original Facebook web page.

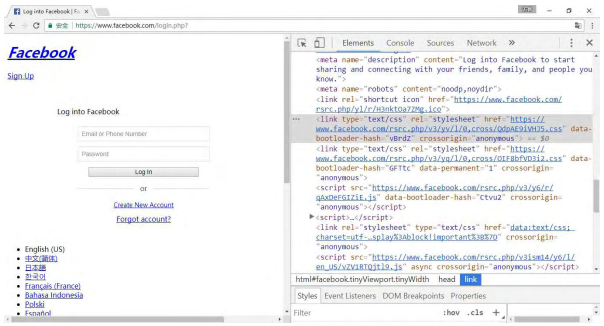


FIGURE 7. Facebook web page after removing part ①.

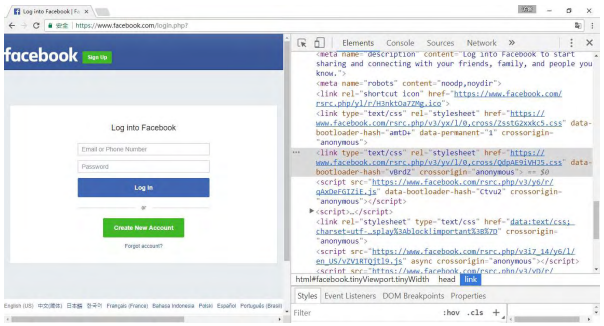


FIGURE 8. Facebook web page after removing part ②.

calculate the similarity rate according to the number of matched selectors, as in our preliminary work [25]. However, this method can be evaded by attackers. For example, Figure 6 shows the Facebook's login page and its corresponding CSS file. If we delete the CSS rules wrapped in the first block (red block), it results in a significant visual appearance change as shown in Figure 7. If we delete the CSS rules enclosed in the second block (blue block), it will not cause too much changes of the web page layout, as shown in Figure 8. Therefore, not all page elements and CSS rules have the same influence on page appearance.

As a result, attackers can easily bypass the straightforward detection methods by leveraging the components with little influence to page appearance, using the following techniques:

- Adding elements whose areas are too small to affect the appearance of the web page, as shown in Figure 9.

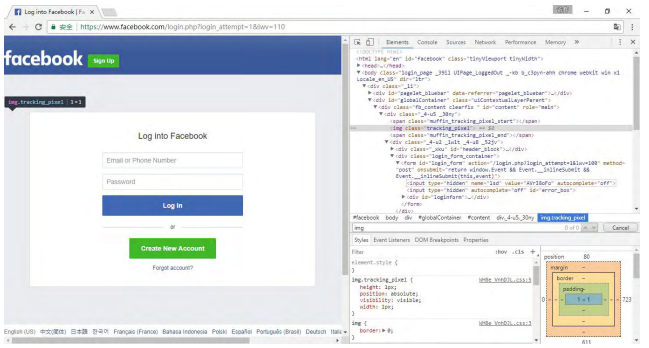


FIGURE 9. Small-sized element.

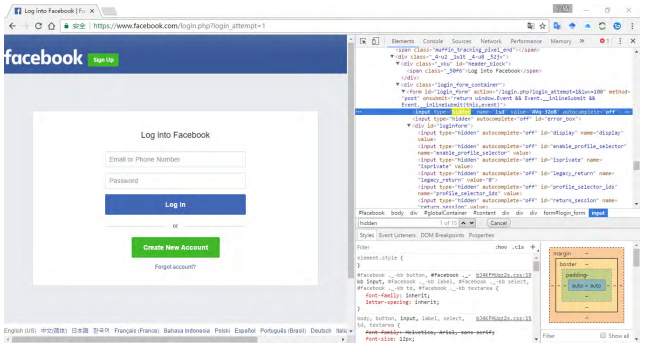


FIGURE 10. Element with hidden property.

A CSS sample is shown as:

$$
\begin{aligned}
&\textit{img.tracking\_pixel}\{\\
&\textbf{height}: \textbf{1px};\\
&\textit{position}: \textit{absolute};\\
&\textit{visibility}: \textit{visible};\\
&\textbf{width}: \textbf{1px};\\
&\}
\end{aligned}
$$

- Adding elements whose visibility property is `hidden`, which does not affect the appearance of the web page, as in Figure 10. A CSS sample is shown as:

$$
\begin{aligned}
&< \textit{inputtype} = \text{“}\textbf{hidden}\text{”} \textit{name} = \text{“}\textit{lsd}\text{”}\\
&\textit{value} = \text{“}\textit{AVrI8oFo}\text{”} \textit{autocomplete} = \text{“}\textit{off}\text{”} >;
\end{aligned}
$$

- Adding elements whose display property is `none`, which do not affect the appearance of the web page, as shown in Figure 11. A CSS content sample is shown as:

$$
\begin{aligned}
&\textit{script}\{\\
&\textbf{display}: \textbf{none};\\
&\}
\end{aligned}
$$

Therefore, we should only focus on the effective features that have actual influence on the web page appearance. According to our analysis, the size of a page element can be used to quantify its impact on the page layout.

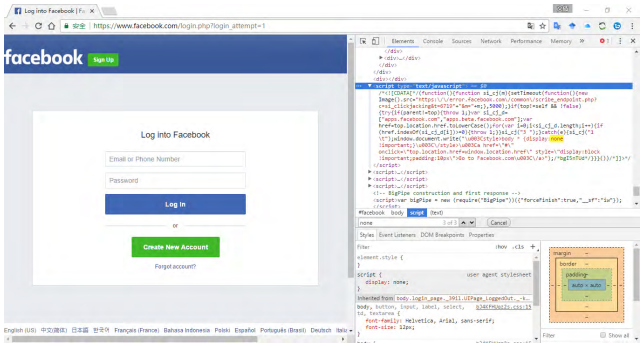


FIGURE 11. Element with non display property.

As the impact of a page element is jointly decided by the element's contents and the CSS rules, our solution is based on both page contents and CSS rules. In particular, we need to decide the rules that effectively have significant impact on page appearance.

Therefore, we evaluate the influence of CSS rules on page appearance by joining the page contents and CSS rules. In effect, the following types of CSS rules have little impact on page appearance: CSS rules that do not have matching page elements, and CSS rules for page elements with minor or no visual presence. In particular, they include:

Case I: the area of element is too small, occupying only a few pixels.

Case II: the *visibility* property of element is *hidden*.

Case III: the *display* property of element is *none*.

B. PAGE SIMILARITY CALCULATION

The *similarity score* and the related concepts that we use to evaluate the pages' similarity are defined as follows.

Given the CSS rule set of a web page X, CSS(X), let $\{p_j\}_{1\le j\le N_P}$ represent the property set in CSS(X), where N_P is the number of the properties in CSS(X). For $1 \le j \le N_P$, let $\{v_j^k\}_{1\le k\le N_V^j}$ represent the value set of the property p_j, where N_V^j is the number of optional values of the property p_j. For $1 \le j \le N_P, 1 \le k \le N_V^j$, let $\{s_l^{jk}\}_{1\le l\le N_S^{jk}}$ represent the set of selectors that have property p_j with value v_k, where N_S^{jk} is the number of the chosen selectors. And we use I_l^{jk} ($j \in [1..N_p], k \in [1..N_V^j], l \in [1..N_S^{jk}]$) represent the area of selector s_l^{jk} (the l-th selector that has property p_j with value v_k), in the web page X.

The *Complexity Score* of a web page is a score to describe how complicated the layout of web page is.

Definition 1 (Complexity Score): Given the comparison-unit representation of a web page X, the complexity of the web page X is

$$C(X) = \sum_{j=1}^{N_P} \sum_{k=1}^{N_V^j} \sum_{l=1}^{N_S^{jk}} I_l^{jk} \tag{1}$$

Given two CSS rule sets of two web pages, CSS(X) and CSS(Y), let $\{p_j\}_{1\le j\le M_P}$ represent the common property set in CSS(X) and CSS(Y), where M_P is the number of the matched properties. For $1 \le j \le M_P$, let $\{v_j^k\}_{1\le k\le M_V^j}$ represent the value set of the property p_j in the common property set, where M_V^j is the number of common values of the property p_j. For $1 \le j \le M_P, 1 \le k \le M_V^j$, let $\{s_r^{jk}\}_{1\le r\le M_{S_x}^{jk}}$ represent the set of selectors in CSS (X) that have property p_j with value v_k and $\{s_t^{jk}\}_{1\le t\le M_{S_y}^{jk}}$ represent the set of selectors in CSS (Y) that have property p_j with value v_k, where $M_{S_x}^{jk}$, $M_{S_y}^{jk}$ are the number of the chosen selectors respectively. $I_r^{jk}(X)$ ($j \in [1..M_p], k \in [1..M_V^j], r \in [1..M_{S_x}^{jk}]$) and $I_t^{jk}(Y)$ ($j \in [1..M_p], k \in [1..M_V^j], t \in [1..M_{S_y}^{jk}]$) represent the area of selector s_r^{jk} and s_t^{jk} in web page X and Y respectively.

The *Match Score* is a metric that measures how much visual appearance two pages have in common.

Definition 2 (Match Score): Given two web pages X and Y, the *Match Score* of X and Y labeled as $M(X, Y)$ is

$$M(X,Y) = \sum_{j=1}^{M_P} \sum_{k=1}^{M_V^j} min\Big(\sum_{r=1}^{M_{S_x}^{jk}} I_r^{jk}(X), \sum_{t=1}^{M_{S_y}^{jk}} I_t^{jk}(Y)\Big) \tag{2}$$

The *Similarity Score* is a metric that measures how similar one page is to another.

Definition 3 (Similarity Score): Given two web pages X and Y, the *Similarity Score* of X and Y labeled as $S(X, Y)$ is

$$S(X,Y) = \frac{M(X,Y)}{C(X) + C(Y) - M(X,Y)} \tag{3}$$

To evaluate the similarity of two web pages according to the definitions above, we propose an algorithm that takes the visual characteristics of two pages as inputs and outputs their visual similarity score. The visual characteristics of website include the CSS rules in every stylesheets and the structure of web page, which is defined as the DOM tree of the page's body. As shown in Algorithm 1, it first extracts the visual characteristics from a suspicious page and a legitimate page, identifies the effective CSS rules, and computes the similar rate of the two web pages.

V. IMPLEMENTATION

We have implemented our algorithm as a Google Chrome browser extension and the system configuration is illustrated in Table 1. As shown in Figure 12, the extension consists of three modules: *Pre-Processor*, *Similarity Checker* and *Target List*.

The Pre-Processor contains three components: *CSS Extractor*, *DOM Extractor* and *Visual Characteristics Filter*. CSS Extractor extracts internal CSS rules directly from the code of web page, and downloads CSS rules in external stylesheets from online servers. DOM Extractor is in charge of copying the structure of page's body, and acquires the area as well as the value of *display* and *visibility* property of each page element. Then Visual Characteristics Filter uses information from both two extractors to exclude CSS rules that have no significant visual influence.

Algorithm 1 Phishing Detection Scheme Based on Page Visual Similarity

let P be a suspicious web-page;
let L be the corresponding target web-page;
let ϵ be a preset similarity threshold
let `InfluenceVec()` be the *influence-vector* of web page.
let `FilUnit()` be the filtered *influence-vector* of web page.
Phase I:Extracting and Filtering.
Function `Extract()` **is**
    input : a suspicious page P
    output: the filtered *influence-vector* of P, *FilUnit*(P)
    get the *CSS* text of P, `CSS(P)`;
    compute `InfluenceVec(P)`;
    `/* convert CSS(P) into influence-vector */`
    compute `FilUnit(P)`;
    `/* delete unnecessary CSS from InfluenceVec(P) */`
    Return `FilUnit(P)`
Phase II: Similarity Computing.
Function `Similarity(A,B)` **is**
    input : `FilUnit(P)`, `FilUnit(L)`
    `/* FilUnit(L) is pre-provided */`
    output: similarity score $S(P, L)$
    compute the complexity score of the P, $C(P)$;
    compute the complexity score of page L, $C(L)$;
    compute the match score of page L and P, $M(P, L)$;
    compute the similarity score between page L and P, $S(P, L)$;
    Return $S(P, L)$
Phase III:Making Decision.
Function `Decision(`$S(P, L)$`)` **is**
    input : $S(P, L)$
    output: the classification result of P
    if $S(P, L) > \epsilon$ **then**
        the page A is similar to the target page B;
        display "warning" and the similarity score $S(P, L)$;
    else
        display similarity score *Sim*(A, B)

TABLE 1. System configuration.

CPU	Intel Core i5-3210M, 2.5GHz
Memory	4G-RAM
OS	Windows 10 (64-bit)
Browser	Google Chrome 58.0.3029.110 (64-bit)

Finally, all the rest of CSS rules will be converted into the comparison unit representation and sent to the similarity checker.

Target List stores the comparison units and the URLs of a set of legitimate websites. We include legitimate web pages

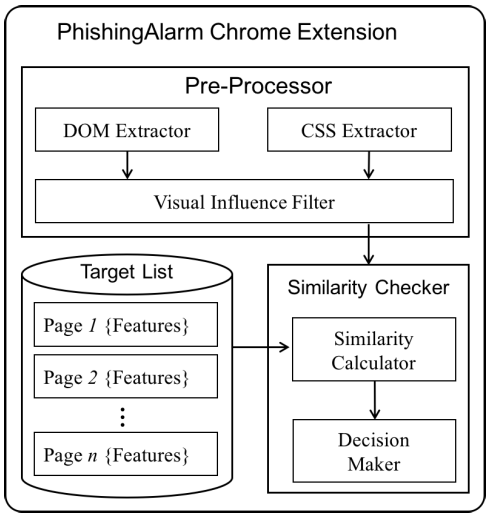


FIGURE 12. Overall architecture of PhishingAlarm.

TABLE 2. Experiment data.

Target list		246 whitelisted webpages	
Sample resource		*phishtank.com*	
		Threshold selecting	Evaluation
phishing sample	Raw	6192	3115
	Valid	1258	289
		Set I: 547 / Set II: 711	

Set I : to evaluate similarity w.r.t. their targets
Set II : to evaluate similarity w.r.t. non-targets

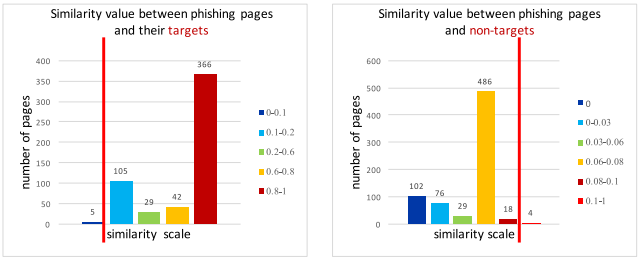


FIGURE 13. Threshold selection.

that are most likely to be attacked by phishing attackers into the target list.

Similarity Checker includes a *Similarity Calculator*, which computes the similarity value between two pages, and a *Decision Maker*, which decides whether the suspicious page is phishing or not. Similarity Calculator computes the similarity value between suspicious page and legitimate pages from Target List, one pair at a time. The results are sent to Decision Maker. If any of similarity value is beyond the preset threshold, the suspicious page is classified as phishing.

VI. EVALUATION AND ANALYSIS

We collected a large number of phishing samples to train the similarity threshold and evaluated the correctness of Phishing-Alarm.

Experiment Setup: We collected 9,307 verified phishing websites from *PhishTank.com* as the experiment sample set,

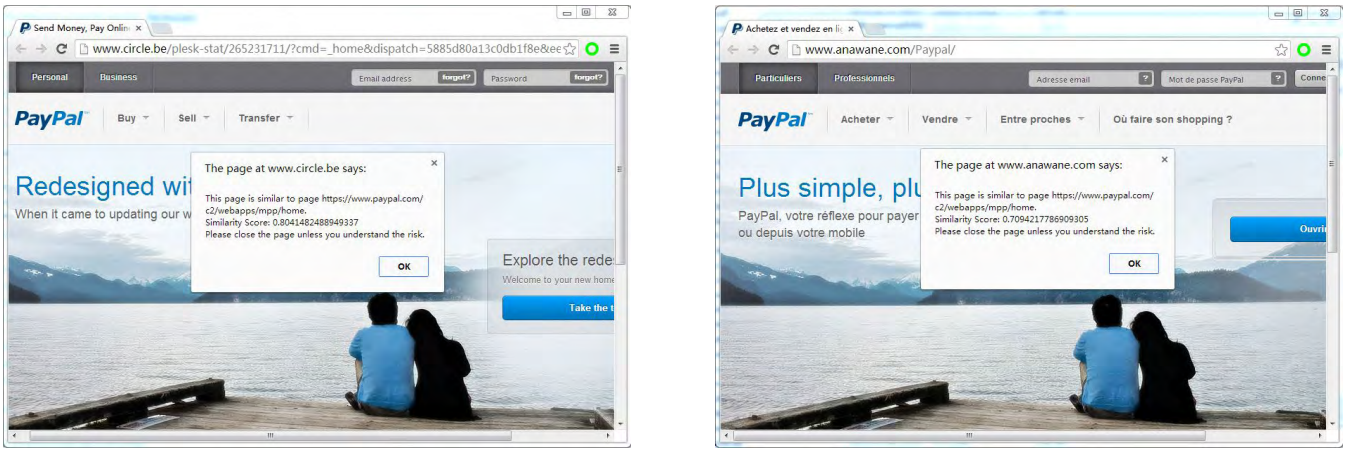
TABLE 3. Similarity Comparison Results of two Categories. (a) Similarity between phishing pages and their target. (b) Similarity between phishing pages and non-targets.

Similarity p	Number	Ratio
$0 \leq p < 0.1$	5	0.92%
$0.1 \leq p < 0.2$	105	19.20%
$0.2 \leq p < 0.6$	29	5.30%
$0.6 \leq p < 0.8$	42	7.67%
$0.8 \leq p < 1$	366	66.91%

(a)

Similarity p	Number	Ratio
$p = 0$	102	14.27%
$0 \leq p < 0.03$	76	10.62%
$0.03 \leq p < 0.06$	29	4.06%
$0.06 \leq p < 0.08$	486	67.97%
$0.08 \leq p < 0.1$	18	2.52%
$p > 0.1$	4	0.56%

(b)



(a) (b)

FIGURE 14. Language-independence phishing detection. (a) English phishing site. (b) French phishing site.

which consists of phishing pages targeting Paypal, eBay, Apple, and other popular website. We used 6,192 pages from them as the sample set for similarity threshold training, and 3,115 of them as the sample set for correctness assessment. To make sure whether these pages are appropriate for our experiment, we manually checked all these collected sample combined with page element extraction operation. There are 4,934 pages within the training set that cannot be accessed or showed as blank pages, and 2,826 pages of evaluation set had the same problem as well. So we excluded these invalid web pages from our experiment. Besides, we use 246 legitimate pages to test the false positive rate of Phishing-Alarm. The experiment data settings are shown in Table 2.

A. SIMILARITY THRESHOLD TRAINING

We selected 547 samples from the 1,258 valid (active) pages to calculate the similarity values between them and their corresponding target pages. The results are illustrated in Table 3(a). 66.91% of the phishing samples have a similarity value over 0.8 and less than 1% of them have a similarity value under 0.1. We use the rest 711 pages in the training sample set to compute the similarity values between these phishing samples and non-targeted websites. For instance, for a phishing page of AOL, we evaluate the similarity rate between it and a non-AOL page. The experiment results are shown in Table 3(b), in which 99.44% testing samples have less than 0.1 similarity rate with non-targeted websites. The similarity rate distributions of the two category testing results are illustrated in Figure 13.

Furthermore, we manually analyzed the five pages with similarity value less than 0.1 from Table 3(a). Three of them are visually different from their reported target pages and can be easily distinguished by users. Intuitively, a proper threshold should keep the number of misclassified pages as low as possible. According to our experiment results (shown in Figure 13), the value 0.1 satisfies this condition well, so we select 0.1 as the similarity threshold for Phishing-Alarm to determine wether the two tested pages are (visually) similar or not.

B. RESULT ANALYSIS

We use three basic metrics, *precision*, *recall*, and *F1*, to describe the detection performance of Phishing-Alarm. The precision rate and recall rate (as shown in Equation (4)(5)) measure the percentage of web pages correctly classified as phishing pages.

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

TABLE 4. The precision, recall, and F1 score of Phishing-Alarm and other approaches.

Approaches	Precision	Recall	F1
CANTINA [46]	94.2%	97.0%	0.956
CANTINA+ [42]	97.5%	93.47%	0.963
Corbetta et.al [7]	95.3%	73.08%	0.827
Belabed et.al [2]	96.6%	98.0%	0.973
Zhang et.al [45]	91.0%	91.90%	0.915
CASTLE [31]	99.5%	98.50%	0.990
Phishing-Alarm	100%	97.92%	0.990

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

True positive (TP) and false negative (FN) are the number of correctly and incorrectly classified phishing pages respectively. False positive (FP) is the number of legitimate pages misclassified as phishing pages. Besides, we use F1-measure (Equation (6)) as a metric to evaluate our approach as well.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

We used the 289 valid web pages from evaluation set to carry out a real-time detection. 283 out of 289 phishing sites were correctly detected by Phishing-Alarm, while six of them successfully bypassed it. According to our manual analysis, three of these six misclassified phishing pages are obviously different from the visual appearance of their target pages, and can be easily recognized by users. From the experiment results, the TP of Phishing-Alarm is 283 and FN is 6. We also use Phishing-Alarm to check the 246 legitimate websites and none of them were classified as phishing. as a result, the FP is 0.

Based on the data above, the *precision* rate of Phishing-Alarm is 100% and *recall* is 97.92%. According to the equation (6), the *F1* score is 98.95%.

Table 4 illustrates the three metrics of Phishing-Alarm and six other approaches (CANTINA [46], CANTINA+ [42], Corbetta et al. [7], Belabed et al. [2], Zhang et al [45] and CASTLE [31]) based on the evaluation reported in the papers. CASTLE has a higher recall than Phishing-Alarm, but Phishing-Alarm achieves the highest precision. Both of them have an F1 score as 0.99, higher than five other approaches.

Because the detection process of our approach relies on the fundamental features of the page's visual layout (e.g., CSS rules), Phishing-Alarm has another advantage that its detection performance is independent from the language of the pages. We made the evaluation and the detection results of English and French phishing pages are shown in Figure 14.

C. DISCUSSION

In our offline test and analysis, we have an interesting finding that some phishing web pages have relatively low similarity rates with their target pages, but they get high similarity rates with some specific phishing pages that have common victim target pages, which means phishing patterns can be extracted from similar phishing pages and used to prompt the efficiency of Phishing-Alarm. Motivated by this, we manually selected 841 valid Paypal phishing web pages and 18 Apple phishing pages from our samples, and test them using Phishing-Alarm. We found that 45 Paypal phishing pages and 10 Apple phishing pages has the similarity rates under less than 0.1. We exclude 7 Paypal phishing samples that are not visually similar as Paypal and can be distinguished. The remained 48 false negative samples are taken as inputs to carry out patter extraction and two CSS patterns are obtained corresponding to Paypal and Apple web pages respectively.

VII. CONCLUSION

Phishing is a popular social engineering attack technique for attackers to obtain victim users' sensitive information, e.g., username with passwords, credit card numbers, social security numbers, etc. In this paper, we propose a robust phishing detection approach, Phishing-Alarm, based on CSS features of web pages. We develop techniques to identify effective CSS features, as well as algorithms to efficiently evaluate page similarity. We prototyped Phishing-Alarm as an extension to the Google Chrome browser and demonstrated its effectiveness in evaluation using real-world phishing samples.

REFERENCES

[1] (2016). *PhishMe Q1 2016 Malware Review*. [Online]. Available: https://phishme.com/project/phishme-q1-2016-malware-review/

[2] A. Belabed, E. Aimeur, and A. Chikh, "A personalized whitelist approach for phishing webpage detection," in *Proc. 7th Int. Conf. Availability, Rel. Security (ARES)*, Aug. 2012, pp. 249–254.

[3] Y. Cao, W. Han, and Y. Le, "Anti-phishing based on automated individual white-list," in *Proc. 4th ACM Workshop Digit. Identity Manage.*, 2008, pp. 51–60.

[4] T.-C. Chen, S. Dick, and J. Miller, "Detecting visually similar Web pages: Application to phishing detection," *ACM Trans. Internet Technol.*, vol. 10, no. 2, pp. 1–38, May 2010.

[5] N. Chou, R. Ledesma, Y. Teraguchi, D. Boneh, and J. C. Mitchell, "Client-side defense against Web-based identity theft," in *Proc. 11th Annu. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2004, pp. 1–16

[6] C. Inc. (Aug. 2016). *Couldmark Toolbar*. [Online]. Available: http://www.cloudmark.com/desktop/ie-toolbar

[7] J. Corbetta, L. Invernizzi, C. Kruegel, and G. Vigna, "Eyes of a human, eyes of a program: Leveraging different views of the Web for analysis and detection," in *Proceedings of Research in Attacks, Intrusions and Defenses (RAID)*. Gothenburg, Sweden: Springer, 2014.

[8] X. Deng, G. Huang, and A. Y. Fu, "An antiphishing strategy based on visual similarity assessment," *Internet Comput.*, vol. 10, no. 2, pp. 58–65, 2006.

[9] Z. Dong, K. Kane, and L. J. Camp, "Phishing in smooth waters: The state of banking certificates in the US," in *Proc. Res. Conf. Commun., Inf. Internet Policy (TPRC)*, 2014, p. 16.

[10] Z. Dong, A. Kapadia, J. Blythe, and L. J. Camp, "Beyond the lock icon: Real-time detection of phishing websites using public key certificates," in *Proc. 10th Symp. Electron. Crime Res.(eCrime)*, May 2015, pp. 1–12.

[11] M. Dunlop, S. Groat, and D. Shelly, "GoldPhish: Using images for content-based phishing analysis," in *Proc. 5th Int. Conf. Internet Monitor. Protection (ICIMP)*, May 2010, pp. 123–128.

[12] I. Fette, N. Sadeh, and A. Tomasic, "Learning to detect phishing emails," in *Proc. Int. World Wide Web Conf. (WWW)*, May 2007, pp. 649–656.

[13] G. Developers. (2011). *Google Safe Browsing*. [Online]. Available: http://code.google.com/intl/fr/apis/safebrowsing/

[14] W. Han, Y. Cao, E. Bertino, and J. Yong, "Using automated individual white-list to protect Web digital identities," *Expert Syst. Appl.*, vol. 39, no. 15, pp. 11861–11869, 2012.
[15] *iTrustPage*. Accessed on Nov. 1, 2013. [Online]. Available: http://www.cs.toronto.edu/ronda/itrustpage/
[16] M. Jakobsson and S. Myers, Eds., *Phishing and Countermeasures: Understanding the Increasing Problem of Electronic Identity Theft*. Hoboken, NJ, USA: Wiley, 2006.
[17] M. Khonji, Y. Iraqi, and A. Jones, "Lexical URL analysis for discriminating phishing and legitimate websites," in *Proc. 8th Annu. Collaboration, Electron. Messaging, Anti-Abuse Spam Conf.*, 2011, pp. 109–115.
[18] M. Khonji, Y. Iraqi, and A. Jones, "Enhancing phishing e-mail classifiers: A lexical url analysis approach," *Int. J. Inf. Security Res. (IJISR)*, vol. 2, pp. 1–2, Mar. 2012.
[19] L.-H. Lee, K.-C. Lee, Y.-C. Juan, H.-H. Chen, and Y.-H. Tseng, "Users' behavioral prediction for phishing detection," in *Proc. 23rd Int. Conf. World Wide Web*, 2014, pp. 337–338.
[20] P. Likarish, E. Jung, D. Dunbar, T. E. Hansen, and J. P. Hourcade, "B-APT: Bayesian anti-phishing Toolbar," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2008, pp. 1745–1749.
[21] W. Liu and X. Deng, "Detecting phishing Web pages with visual similarity assessment based on earth mover's distance," *IEEE Trans. Depend. Sec. Comput.*, vol. 3, no. 4, pp. 301–311, Apr. 2006.
[22] W. Liu, N. Fang, X. Quan, B. Qiu, and G. Liu, "Discovering phishing target based on semantic link network," *Future Generat. Comput. Syst.*, vol. 26, no. 3, pp. 381–388, 2010.
[23] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 1245–1254.
[24] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. 26th Annu. Int. Conf. Mach. Learn.*, 2009, pp. 681–688.
[25] J. Mao, P. Li, T. Li, T. Wei, and Z. Liang, "BaitAlarm: Detecting phishing sites using similarity in fundamental visual features," in *Proc. 5th Int. Conf. Intell. Netw. Collaborative Syst. (INCoS)*, 2013, pp. 790–795.
[26] J. Mao, W. Tian, P. Li, T. Wei, and Z. Liang, "Phishing website detection based on effective CSS features of Web pages," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl. (WASA)*, 2017, pp. 804–815.
[27] E. Medvet, E. Kirda, and C. Kruegel, "Visual-similarity-based phishing detection," in *Proc. SecureComm*, Sep. 2008, p. 22.
[28] R. Mohammad, F. Thabtah, and L. McCluskey, "An assessment of features related to phishing websites using an automated technique," in *Proc. Int. Conf. Internet Technol. Secured Trans.*, 2012, pp. 492–497.
[29] Mozilla. (2011). *Phishing Protection*. [Online]. Available: http://www.mozilla.com/en-US/firefox/phishingprotection/
[30] L. A. T. Nguyen, H. K. Nguyen, and B. L. To, "An efficient approach based on neuro-fuzzy for phishing detection," *J. Autom. Control Eng.*, vol. 4, no. 2, pp. 519–525, 2016.
[31] A. Nourian, S. Ishtiaq, and M. Maheswaran, "CASTLE: A social framework for collaborative anti-phishing databases,"in *Proc. ACM Trans. Internet Technol.*, 2009, pp. 1–10.
[32] Y. Pan and X. Ding, "Anomaly based Web phishing page detection," in *Proc. 22nd Annu. Comput. Security Appl. Conf.*, 2006, pp. 381–392.
[33] B. Parno, C. Kuo, and A. Perrig, "Phoolproof phishing prevention," in *Proc. Int. Conf. Financial Cryptography Data Security*, Feb. 2006, pp. 1–19.
[34] (Feb. 2011). *Phishing Bank of China Internet Banking Page*. [Online]. Available: http://news.sohu.com/20110124/n279045084.shtml/
[35] T. Ronda, S. Saroiu, and A. Wolman, "itrustpage: A user-assisted anti-phishing tool," in *Proc. Eurosys*, Apr. 2008, pp. 261–272.
[36] H. Tout and W. Hafner, "Phishpin: An identity-based anti-phishing approach," in *Proc. Int. Conf. Comput. Sci. Eng.*, Oct. 2009, pp. 347–352.
[37] W3CSchool. *CSS Tutorial*. Accessed on Aug. 1, 2017. [Online]. Available: http://www.w3schools.com/css/
[38] Y. Wang, R. Agrawal, and B.-Y. Choi, "Light weight anti-phishing with user whitelisting in a Web browser," in *Proc. IEEE Region Conf.*, Apr. 2008, pp. 1–4.
[39] B. Wardman, T. Stallings, G. Warner, and A. Skjellum, "High-performance content-based phishing attack detection," in *Proc. eCrime Res. Summit*, San Diego, CA, USA, Nov. 2011, pp. 1–9.
[40] WOT. (2011). *Web of Trust*. [Online]. Available: http://www.antiphishing.org/reports/apwg_trends_report_h2_2011.pdf
[41] L. Wu, X. Du, and J. Wu, "MobiFish: A lightweight anti-phishing scheme for mobile phones," in *Proc. ICCCN*, 2014, pp. 1–8.
[42] 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. Security*, vol. 14, no. 2, p. 21, 2011.
[43] C. Yue and H. Wang, "BogusBiter: A transparent protection against phishing attacks," *ACM Trans. Internet Technol.*, vol. 10, pp. 1–30, Dec. 2009.
[44] J. Zhang, Y. Pan, Z. Wang, and B. Liu, "URL based gateway side phishing detection method," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 268–275.
[45] W. Zhang, H. Lu, B. Xu, and H. Yang, "Web phishing detection based on page spatial layout similarity," *Informatica*, vol. 37, no. 3, pp. 231–244, 2013.
[46] Y. Zhang, J. I. Hong, and L. F. Cranor, "CANTINA: A content-based approach to detecting phishing Web sites," in *Proc. Int. World Wide Web Conf. (WWW)*, May 2007, pp. 639–648.
[47] M. Zouina and B. Outtaj, "A novel lightweight URL phishing detection system using SVM and similarity index," *Human-Centric Comput. Inf. Sci.*, vol. 7, no. 1, p. 17, 2017.

JIAN MAO received the B.S. and Ph.D. degrees from Xidian University, Shanxi, China. She is currently an Assistant Professor with the School of Electronic and Information Engineering, Beihang University, Beijing, China. Her research interests include cloud security, Web security, and mobile security.

WENQIAN TIAN received the B.S. degree in electronic and information engineering from Beihang University, Beijing, China, in 2016, where she is currently pursuing the master's degree in electronic and information engineering. Her research interests include Web security, mobile security, and privacy analysis.

PEI LI received the B.S. and M.S. degrees in electronic and information engineering from Beihang University, Beijing, China, in 2012 and 2015, respectively. Her research interests include Web security and mobile security.

TAO WEI received the B.S. and Ph.D. degrees from Peking University, Beijing, China, in 1997 and 2007, respectively. Prior to joining Baidu, he was an Associate Professor with Peking University. He is currently the Head of the Baidu X-Lab. His research interests include software security, Web security, mobile security, and program analysis.

ZHENKAI LIANG received the B.S. degree from Peking University, Beijing, China, in 1999, and the Ph.D. degree from Stony Brook University in 2006. His research interests include software security, Web security, and mobile security.