Detecting Mobile Malicious Webpages in Real Time

Chaitrali Amrutkar, Young Seuk Kim, and Patrick Traynor, *Senior Member, IEEE*

Abstract—Mobile specific webpages differ significantly from their desktop counterparts in content, layout and functionality. Accordingly, existing techniques to detect malicious websites are unlikely to work for such webpages. In this paper, we design and implement kAYO, a mechanism that distinguishes between malicious and benign *mobile* webpages. kAYO makes this determination based on static features of a webpage ranging from the number of iframes to the presence of known fraudulent phone numbers. First, we experimentally demonstrate the need for mobile specific techniques and then identify a range of new static features that highly correlate with mobile malicious webpages. We then apply kAYO to a dataset of over 350,000 known benign and malicious mobile webpages and demonstrate 90% accuracy in classification. Moreover, we discover, characterize and report a number of webpages missed by Google Safe Browsing and VirusTotal, but detected by kAYO. Finally, we build a browser extension using kAYO to protect users from malicious mobile websites in real-time. In doing so, we provide the first static analysis technique to detect malicious mobile webpages.

Index Terms—Mobile security, webpages, web browsers, machine learning.

1 Introduction

Mobile devices are increasingly being used to access the web. However, in spite of significant advances in processor power and bandwidth, the browsing experience on mobile devices is considerably different. These differences can largely be attributed to the dramatic reduction of screen size, which impacts the content, functionality and layout of mobile webpages.

Content, functionality and layout have regularly been used to perform static analysis to determine maliciousness in the desktop space [20], [37], [51]. Features such as the frequency of iframes and the number of redirections have traditionally served as strong indicators of malicious intent. Due to the significant changes made to accommodate mobile devices, such assertions may no longer be true. For example, whereas such behavior would be flagged as suspicious in the desktop setting, many popular benign mobile webpages require multiple redirections before users gain access to content. Previous techniques also fail to consider mobile specific webpage elements such as calls to mobile APIs. For instance, links that spawn the phone's dialer (and the reputation of the number itself) can provide strong evidence of the intent of the page. New tools are therefore necessary to identify malicious pages in the mobile web.

In this paper, we present kAYO[1], a fast and reliable static analysis technique to detect malicious *mobile* webpages. kAYO uses static features of mobile webpages derived from their HTML and JavaScript content, URL and advanced mobile specific capabilities. We first experimentally demonstrate that the distributions of identical static features when extracted from desktop and mobile webpages vary dramatically. We then collect over 350,000 mobile benign and malicious webpages over a period of three months. We then use a binomial classification technique to develop a model for kAYO to provide 90% accuracy and 89% true positive rate. kAYO's performance matches or exceeds that of existing static techniques used in the desktop space. kAYO also detects a number of malicious mobile webpages not precisely detected by existing techniques such as VirusTotal and Google Safe Browsing. Finally, we discuss the limitations of existing tools to detect mobile malicious webpages and build a browser extension based on kAYO that provides real-time feedback to mobile browser users.

We make the following contributions:

- **Experimentally demonstrate the differences in the "security features" of desktop and mobile webpages:** We experimentally demonstrate that the distributions of static features used in existing techniques (e.g., the number of redirections) are different when measured on mobile and desktop webpages. Moreover, we illustrate that certain features are inversely correlated or unrelated to or non-indicative to a webpage being malicious when extracted from each space. The results of our experiments demonstrate the need for mobile specific techniques for detecting malicious webpages.
- **Design and implement a classifier for malicious and benign mobile webpages:** We collect over 350,000 benign and malicious mobile webpages. We then identify *new* static features from these webpages that distinguish between mobile benign and malicious webpages. kAYO provides 90% accuracy

- *Chaitrali Amrutkar is with Google. This work was completed while she was a graduate student at the Georgia Tech Information Security Center (GTISC), Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail:chaitrali.amrutkar@gmail.com*
- *Young Seuk Kim is with PWC. He completed this work at GTISC, Georgia Institute of Technology, Atlanta, GA 30332, USA. E-mail: ykim320@gmail.com*
- *Patrick Traynor is with the Florida Institute for Cyber Security, University of Florida, Gainesville, FL 32611, USA. E-mail:traynor@cise.ufl.edu*

1. Our technique is called "kAYO" (knockout in boxing terminology) because it knocks out malicious mobile webpages.

in classification and shows improvement of two orders of magnitude in the speed of feature extraction over similar existing techniques. We further empirically demonstrate the significance of kAYO's features. Finally, we also identify 173 mobile webpages implementing cross-channel attacks, which attempt to induce mobile users to call numbers associated with known fraud campaigns.

- **Implement a browser extension based on kAYO:** To the best of our knowledge kAYO is the first technique that detects mobile specific malicious webpages by static analysis. Existing tools such as Google Safe Browsing are not enabled on the mobile versions of browsers, thereby precluding mobile users. Moreover, the mobile specific design of kAYO enables detection of malicious mobile webpages missed by existing techniques. Finally, our survey of existing extensions on Firefox desktop browser suggests that there is a paucity of tools that help users identify mobile malicious webpages. To fill this void, we build a Firefox mobile browser extension using kAYO, which informs users about the maliciousness of the webpages they intend to visit in real-time. We plan to make the extension publicly available post publication.

We note that we define maliciousness broadly, as is done in the prior literature on the static detection in the desktop space [20], [37], [51]. However, because drive-by-downloads are not at all common in the mobile space at the time of writing, the overwhelming majority of detected pages are related to phishing.

2 Related Work

Content-based and in-depth inspection techniques to detect malicious websites: Dynamic approaches using virtual machines [45], [51] and honeyclient systems [32], [42], [46] provide deeper visibility into the behavior of a webpage. Therefore, such systems have a very low false positive rate and are more accurate. However, downloading and executing each webpage impacts performance and hinders scalability of dynamic approaches. This performance penalty can be avoided by using static approaches. Static approaches rely on the structural and lexical properties of a webpage and do not execute the content of the webpage. One such technique of detecting malicious URLs is using statistical methods for URL classification based on a URL's lexical and host-based properties [28], [30], [35], [39]. However, URL-based techniques usually suffer from high false positive rates. Using HTML and JavaScript features extracted from a webpage in addition to URL classification helps address this drawback and provides better results [20], [41], [55], [59]. Static approaches avoid performance penalty of dynamic approaches. Additionally, using fast and reliable static approaches to detect benign webpages can avoid expensive in-depth analysis of all webpages.

Differences between mobile and desktop websites: All these approaches for malicious webpage detection have focused on websites built for desktop browsers in the past. Mobile browsers have been shown to differ from their desktop counterparts in terms of security [13], [14]. Although differences in mobile and desktop websites have been observed before [19], it is unclear how these differences impact security. Furthermore, the threats on mobile and desktop websites are somewhat different [26]. Static analysis techniques using features of desktop webpages have been primarily studied for drive-by-downloads on desktop websites [20], [51], whereas, the biggest threat on the mobile web at present is believed to be phishing [18]. Efforts in mitigating phishing attacks on desktop websites include isolating browser applications of different trust level [29], email filtering [28], using content-based features [55], [59] and blacklists [38]. The best-known non-proprietary content-based approach to detect phishing webpages is Cantina [59]. Cantina suffers from performance problems due to the time lag involved in querying the Google search engine. Moreover, Cantina does not work well on webpages written in languages other than English. Finally, existing techniques do not account for new mobile threats such as known fraud phone numbers that attempt to trigger the dialer on the phone. Consequently, whether existing static analysis techniques to detect malicious desktop websites will work well on mobile websites is yet to be explored.

Mobile application security: Significant work has been done in the past few years on the security of mobile applications. Static feature extraction, especially with respect to permissions, has been one of the most important early areas of research [21], [23], [25], [27]. Such techniques have led to dramatically more rapid detection of malicious applications across a range of marketplaces.

DNS based approaches to detect malicious domains: A popular approach in detecting malicious activity on the web is by leveraging distinguishing features between malicious and benign DNS usage. Both passive DNS monitoring [15], [17], [49], [58] and active DNS probing methods [31], [33] have been used to identify malicious domains. While some of these efforts focused solely on detecting fast flux service networks [31], [47], [48], [54], another [17] can also detect domains implementing phishing and drive-by-downloads. DNS based mechanisms do not provide deeper understanding of the specific activity implemented by a webpage or domain.

3 Motivation

Static analysis techniques to detect malicious websites often use features of a webpage such as HTML, JavaScript and characteristics of the URL. Usually, these features are fed to machine learning techniques to classify benign and malicious webpages. These techniques are predicated on the assumption that the features are distributed differently across benign and malicious webpages. Accordingly, any changes in the distribution of static features in benign and/or malicious webpages impacts classification results of static analysis techniques. While

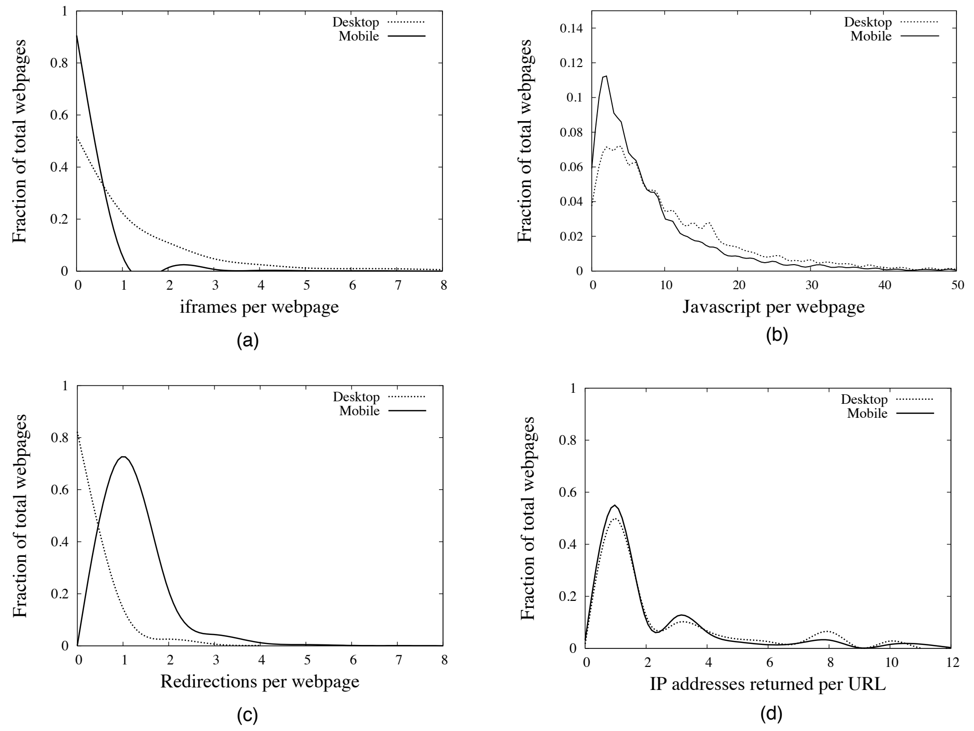


Fig. 1: Normalized density curves of static features. There is substantial difference between the distributions of the number of (a) iframes, (b) Javascript and (c) redirections when measured on mobile and desktop versions of the same websites, whereas, the distribution of the number of (d) IP addresses is similar.

successful, these static analysis techniques have been used *exclusively* for desktop webpages [20], [51], [59].

Mobile websites are significantly different from their desktop counterparts in content, functionality and layout. Consequently, existing tools using static features to detect malicious desktop webpages are unlikely to work for mobile webpages. We explain four factors that motivate building separate static analysis techniques to detect malicious mobile webpages.

1) Differences in content: Mobile websites are often simpler than their desktop counterparts. Therefore, the distribution of content-based static features (such as the number of JavaScripts) on mobile webpages differs from that of desktop webpages. For example, Figure 1 (a) and Figure 1 (b) show the normalized density of the number of iframes and the number of Javascript found in mobile[2] and the corresponding desktop versions of the top-level webpage of the 10,000 most popular websites from Alexa [11]. Approximately 90% of mobile webpages do not have any iframes, whereas the corresponding desktop webpages have multiple iframes. Desktop webpages have more Javascripts than mobile webpages.

Due to the simplicity of mobile webpages, the majority of other content related static features used in existing techniques including, the number of images, page length, the number of hidden elements, and the number of elements with a small area also differ in magnitude in mobile and desktop webpages.

2) Infrastructure: Website providers use JavaScript or user agent strings to identify and then redirect mobile users to a mobile specific version. Figure 1 (c) shows the normalized density of the number of redirection steps taken by the desktop and mobile versions of the top 10,000 websites on Alexa before landing on the final URL[3]. Even the most popular mobile websites show multiple redirects, which has *traditionally* been a property of desktop websites hosting malware [51]. However, multiple redirects does not necessarily indicate bad behavior for mobile websites due to the characteristics of their hosting infrastructure.

We note that not all static features used in existing techniques differ when measured on mobile and desktop webpages. For example, the number of IP addresses returned by DNS servers for mobile and desktop versions of the same sites are comparable. Mobile websites appear to share their hosting infrastructure with the corresponding desktop websites [36]. We used seven public (Google, OpenDNS, UltraDNS, Norton, DynDNS, Level3, and Scrubit) DNS servers to obtain the IP addresses returned in the DNS A records of mobile and corresponding desktop URLs of Alexa top 10,000 websites. As seen in Figure 1 (d), the distributions of the number of IP addresses returned by the seven DNS servers are similar

2. We describe the method used to define and identify mobile webpages in detail in Section 4.2.

3. We use the term *final URL* to denote the URL that is rendered in the browser after redirections (if any) from the seed URL. The final URL may change based on the browser's user agent.

for mobile and desktop websites.

3) Impact of screen size: The screen size of a mobile phone is significantly smaller that that of a desktop computer. Therefore, a mobile user only sees a part of the URL of a webpage. Intuitively, the author of a mobile phishing webpage may only need to include misleading words at the beginning of the URL and a short URL might suffice to trick a user.

4) Mobile specific functionality: Mobile websites enable access to a user's personal information and advanced capabilities of mobile devices through web APIs. Existing static analysis techniques do not consider these mobile specific functionalities in their feature set. We argue and later demonstrate that accounting for the mobile specific functionalities helps identify new threats specific to the mobile web. For example, the presence of a known 'bank' fraud number on a website might indicate that the webpage is a phishing webpage imitating the same bank [16].

Limitations of existing techniques: These discrepancies between mobile and desktop webpages demand investigation. Existing static analysis techniques and tools for detecting malicious webpages are focused on desktop webpages. *Therefore, they are unable to detect mobile specific threats with high accuracy.*[4] Secondly, several webpages built specifically for mobile, return empty pages when rendered in a desktop browser. Thus, even existing dynamic analysis techniques that execute websites in desktop browsers on virtual machines, are ineffective on such mobile websites. Finally, signature based tools such as Google Safe Browsing currently only work with desktop browsers. We manually visited five mobile specific known malicious webpages collected from PhishTank [6], from the Google Chrome mobile browser. We observed that these webpages are flagged as malicious on the Chrome desktop browser, but not on the Chrome mobile browser whose users are the real targets of the mobile malicious webpages. Although enabling Google Safe Browsing in mobile Chrome is an engineering effort, we argue and later demonstrate that a mobile specific static technique can also detect new threats previously unseen by such services.

Goals: Considering the limitations of existing techniques, the goals of this work are three-fold. First, identifying relevant static features from mobile specific webpages in the wild. Second, implementing a fast and reliable static analysis technique to detect malicious mobile webpages in real-time. And finally, developing a mobile browser extension that will inspect mobile webpages in real-time and provide feedback to the user.

4 Methodology

Our objective is to design and develop a technique to identify mobile specific malicious webpages in real-time. We extract static features from a webpage and make predictions about its potential maliciousness. We first discuss the feature set used in kAYO followed by the collection process of the dataset.

4.1 kAYO Feature Set

A webpage has several components including HTML and JavaScript code, images, the URL, and the header. Mobile specific webpages also access applications running on a user's device using web APIs (e.g., the dialer). We extract structural, lexical and quantitative properties of such components to generate kAYO's feature set. We focus on extracting mobile relevant features that take minimal extraction time. Our hypothesis is that such features are strong indicators of whether a webpage has been built for assisting a user in their web browsing experience or for malicious purposes.

Our feature set consists of 44 features, 11 of which are new and not previously identified or used. We describe these new features in detail. A subset of features in kAYO have been used by other authors in static inspection of desktop webpages in the past.[5] However, it is important to note that these features in mobile webpages and desktop webpages differ in magnitude (e.g., number of iframes) and show varying correlation with the nature of the webpage (i.e., malicious/benign).

We divide kAYO's 44 features into four classes: mobile specific-, JavaScript, HTML and URL features. To the best of our knowledge, we are the first to use these mobile specific features, and do not claim novelty on using subsets of other previously identified features. Table 1 summarizes the 8 mobile, 10 JavaScript, 14 HTML and 12 URL features. We empirically illustrate the effectiveness of each of the features in Section 5.2.

4.1.1 Mobile specific features

We collect eight mobile specific features to capture the advanced capabilities of mobile webpages. Mobile websites enable access to personal data from a user's phone, an experience not offered by desktop websites. For example, mobile web APIs such as `tel:` and `sms:` spawn the dialer and the SMS applications respectively on a mobile device. In order to characterize the behavior of mobile API calls, we extracted the number of API calls `tel:`, `sms:`, `smsto:`, `mms:` and `mmsto:` from each mobile webpage. We further extracted the target phone numbers from these API calls. We ran the commercially available Pindrop Security Phone Reputation System (PRS) [7] on each phone number. Based on the results of the PRS, we gave the score of 1/0 (known fraud/benign) to each phone number scraped from the mobile API calls, and added the score as a feature in kAYO. We only extracted phone numbers with API prefixes that could trigger an application installed on a user's phone. We

4. We demonstrate this experimentally in Section 5.

5. A subset of kAYO's features was selected from prior literature on desktop webpage classification using static features. Only the features deemed to be the most critical and definitively applicable to mobile webpages as shown by manual analysis, were selected based on the authors' experience and knowledge of the area.

Category	Features	Total # of features
Mobile specific	# of API calls to tel:, sms:, smsto:, mms:, mmsto:, geolocation; # of apk, # of ipa	8
JavaScript	presence of JS, noscript, internal JS, external JS, embedded JS; # of JS, noscript, internal JS, external JS, embedded JS	10
HTML	presence of internal links, external links, images; # of internal links, external links, images # of cookies from header, secure and HTTPOnly cookies, presence of redirections and iframes, # of redirects and iframes, whether webpage served over SSL, % of white spaces in the HTMl content	14
URL	# of misleading words in the URL such as *login* and *bank* , length of URL # of forward slashes and question marks, digits, dots, hyphens and underscores, # of equal signs and ampersand, subdomains, two letter subdomains, semicolons, presence of subdomain, % of digits in hostname	12
	Total:	44

TABLE 1: The 44 features of kAYO from four categories. The significance of both new mobile and prior features res is evaluated in Section 5.2.

did not consider phone number strings simply listed on webpages without an API prefix.

We argue that due to the popularity of application markets such as Google play and iTunes, a website hosting its own mobile application binary (e.g., *.apk* or *.ipa* files) possibly suggests bad behavior. If we found more than a threshold (in the few hundreds) of apk/ipa files on the same webpage, we assumed that the webpage was a third-party app store (of which there are many) and was unlikely to be malicious.

4.1.2 JavaScript features

JavaScript enables client-side user interaction, asynchronous communication with servers, and modification of the DOM objects of webpages on the fly. We extract 10 features that capture the JavaScript relevant static behavior of a webpage, two of which are new. All the features are faster to extract than the features based on JavaScript deobfuscation.

JavaScript found on malicious webpages can be obfuscated. Instead of deobfuscating every JavaScript, we extract simple JavaScript related features from a webpage. The primary reason in choosing this approach is that a large number of benign webpages include potentially dangerous JavaScript code as shown by Yue et al. [57]. For example, 44.4% of the top 6,805 websites from Alexa use the potentially dangerous `eval` function. These observations invalidate the assumption made in existing techniques [20]; that potentially dangerous JavaScript keywords are more frequently used in malicious webpages. Secondly, external JavaScript can be very large, sometimes of the order of a few megabytes. Our goal is to build a real-time browser extension based on kAYO. Accordingly, we avoided using features that would slow down the feature extraction process.

We argue that benign webpage writers take effort to provide good user experience, whereas the goal for malicious webpage authors is to trick users into performing unintentional actions with minimal effort. We therefore examine whether a webpage has `noscript` content and measure the number of `noscript`. Intuitively, a benign webpage writer will have more `noscript` in the code to ensure good experience even for a security savvy user. We add these two newly identified features to our set.

Webpages generally include three types of JavaScript: internal, external and embedded. An internal JavaScript is one hosted on the same domain as that of its parent webpage, whereas, an external JavaScript's domain is different from its host's domain. Since mobile webpages are often simpler than desktop webpages and phishing is the biggest threat on mobile webpages at present, we expect that benign webpages will include more external JavaScript for advertisements and analytics purposes, whereas malicious webpages will have a lower number of external JavaScript. Accordingly, we determine whether a webpage holds external and internal JavaScript, and then extract the number of internal and external JavaScript from a webpage. Unlike internal and external JavaScript, embedded JavaScript code is contained in the webpage. If the number of lines of JavaScript is relatively small, a webpage with embedded JavaScript loads faster than pages that must reference external code. This is because, as the web browser loads the page and encounters the reference to the external code, it must make a separate request to the web server to fetch the code. Webpages built for performance often use a number of embedded JavaScript. Performance is critical in the mobile web since it impacts revenue and user interest [50]. Therefore, we determine whether a webpage hosts embedded JavaScript and then calculate the number of embedded JavaScript in a webpage. Our hypothesis is that on average, benign webpages will have more embedded JavaScript. Finally, we determine whether JavaScript is present at all on a webpage, and measure the total number of JavaScript on the webpage including embedded, internal and external. Note that we believe that this feature is indicative, but not an alarm by itself as malicious pages could also seek to gain revenue from advertisements.

4.1.3 HTML features

We extract 14 features in total from the HTML code of each webpage. Popular webpages include a number of images, and internal and external HTML links for better user experience. For example, the top-level page of *m.cnn.com* includes links to other news articles published by CNN (internal HTML links), advertisements for a local restaurant (external HTML link) and images related to the latest breaking news. Accordingly, we first determine whether a webpage has any images, internal and external HTML links. We then extract the number of internal links, external links and images from a webpage as features of kAYO.

Malicious webpages (especially those implementing drive-by-downloads and clickjacking) include links to bad content in iframes [51]. Recall that the distribution of iframes on mobile webpages is different as compared to that on desktop webpages. However, we do not rule out the possibility of a mobile malicious webpage including malicious content in iframes and consider the presence and number of iframes in a webpage as features in kAYO. Past research also shows that malicious websites take several redirections before leading the user to the target webpage to avoid DNS based detection [51]. Recall that mobile webpages generally take at least one or more redirections since both desktop and mobile versions of the webpage share hosting infrastructure. Therefore, we determine whether a webpage was redirected and then measure the number of redirections the user experiences before landing on the final URL. Finally, we extract other features such as the percentage of white spaces in the HTML content, the number of cookies from the header, the number of secure and HTTPOnly cookies, and whether the webpage is served over an SSL connection. Readers are encouraged to refer to prior literature [20], [41], [59] for more information on the usefulness of these HTML features.

4.1.4 URL features

Structural and lexical properties of a URL have been used to differentiate between malicious and benign webpages. However, using only URL features for such differentiation leads to a high false positive rate. We extract 12 URL features in total.

Authors of phishing webpages often exploit the familiarity of users to a webpage [22] by including words in the URL that can mislead a user into believing that the phishing webpage is the legitimate webpage. Words such as *login* and *bank* are commonly used in the URL of the login webpage for benign websites that are highly prone to imitation. Only a part of the URL is visible to the user of a mobile phone due to the small screen [13]. Therefore, intuitively, the author of a phishing webpage will include misleading words at the beginning of the URL. We consider the presence of such words in the URL as a new feature in kAYO.

A significant number of phishing domain names are simply IP addresses of machines hosting them [28], [40]. Therefore, we calculated the number of digits in a URL and the percentage of digits in the hostname. Phishing webpage developers usually create a number of subdomains to include deceptive keywords such as `paypal` as a subdomain. This might increase the length of phishing URLs [40]. Therefore, we include the length of a URL, whether the URL contains a subdomain, the number of subdomains, and the number of dots

Mobile Webpage Indicators	
Top Level Domain	.mobi
Subdomain	m., mobile., touch., 3g., sp., s., mini., mobileweb., t.
URL Path Prefix	/mobile, /mobileweb, /m, /mobi, /?m=1, /mobil, /m_home

TABLE 2: Indicators of mobile specific webpages extracted by manual analysis of the top-level mobile and desktop webpages of the 1,000 most popular websites on Alexa. We identified one top-level domain (TLD), nine subdomains and seven URL path prefixes.

as features. Our URL feature set also contains the number of semicolons, equal signs and ampersand symbols, hyphens and underscores, forward slashes and question marks. Interested readers are referred to prior literature [28], [34], [39] for details on the importance of these URL features.

Note that the HTML, JavaScript and URL features are not specific to mobile and can be used for analyzing desktop webpages as well. However, the mobile features derived from mobile applications such as dialer and SMS do not apply to desktop webpages.

4.2 Data Collection

Our data gathering process included accumulating labeled benign and malicious mobile specific webpages. First, we describe an experiment that identifies and defines 'mobile specific webpages'. We then conduct the data collection process over three months in 2013. We use these crawls specifically because they are close to the publication of the related work, making them as close to equivalent as possible.

Identification of mobile specific webpages: We crawled the top-level webpage of the 1,000 most popular websites from Alexa.com [11] using the Android mobile and desktop Internet Explorer (IE) browsers. We used Android mobile version 4.0 and IE desktop version 9.0 for Windows 7. We then manually analyzed each pair of final URLs for the same seed URL when crawled from each browser. Before classifying a URL as mobile specific, we confirmed that the final URLs for desktop and mobile were different for the same seed URL. We also compared the contents of each pair of desktop and mobile webpages, and ensured that the two contents were different. We ignored all the seed URLs that led to an identical final URL when crawled from the desktop and the mobile browser. Our analysis identified nine subdomains (e.g., m.) and seven URL path prefixes (e.g., /mobile) in the URLs of popular websites to represent their mobile specific webpages. Additionally, we considered all URLs with the '.mobi' Top Level Domain (TLD) to be mobile sites [12]. We defined a mobile specific webpage as one containing any of these 17 mobile indicators in the URL and showing

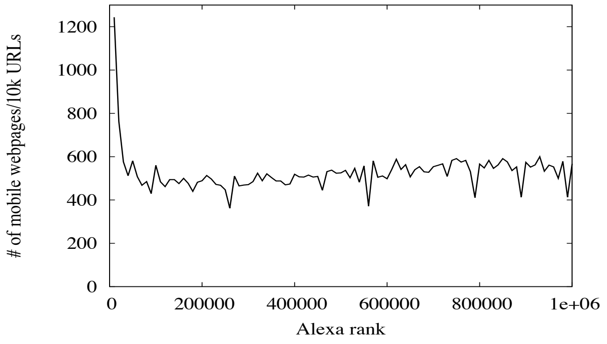


Fig. 2: Number of mobile specific websites found in every 10,000 websites in the Alexa top 1M.

differences in content from the corresponding desktop webpage. Table 2 summarizes the mobile indicators.

Building the dataset: To generate training data for our model, we statically crawled the top-level webpage of the top 1,000,000 most popular websites from Alexa from an Android mobile browser. We then extracted the mobile specific webpages using the algorithm described above. Figure 2 shows the number of top-level mobile specific webpages found in the dataset. 1,244 out of the first 10,000 most popular websites offer a mobile specific version and 763 maintain mobile specific webpages in the 10,000-20,000 range. From 20,000 onwards upto one million, the number of mobile specific webpages found using our algorithm is largely constant. We observed that 485 out of the top one million Alexa websites have the '.mobi' TLD. Using the 17 mobile indicators defined in Table 2, we collected 53,638 mobile specific URLs at the top-level by statically crawling each website in Alexa from an Android mobile browser. We then crawled each of the 53,638 mobile specific websites two levels deep. Interestingly, we found links to several non-mobile URLs on the mobile specific webpages. We discarded non-mobile webpages and were left with 295,512 mobile URLs at depth two. In total, we derived 349,150 mobile URLs from the Alexa one million websites.

Gathering data for malicious mobile URLs was challenging since the mobile web is still evolving and new threats are emerging. We monitored several public blacklists [2], [3], [5] continually for three months and extracted mobile specific URLs from the blacklists. We set up a continuous feed from two public blacklists and crawled newly uploaded malicious URLs every two seconds. We also monitored PhishTank's [6] online dataset for mobile specific phishing URLs. After monitoring these sources for three months, we gathered data from 531 top-level and 4,681 depth two mobile specific malicious URLs. Note that our dataset also contains mobile URLs that were submitted to the blacklists before 2013, but were live at the time of crawling.

We established ground truth of the labels (malicious/benign) of webpages in our dataset by using VirusTotal [9] and Google Safe Browsing [10]. The Google Safe Browsing tool performs both static and dynamic analysis on webpages [51]. It first discards benign webpages identified using static analysis and then performs dynamic analysis on the webpages tagged as malicious. VirusTotal queries 41 different malware detection tools based on dynamic analysis, crowd sourcing and signatures. To be conservative, we labeled a URL as malicious only when Google Safe Browsing tagged a URL as malicious, or four or more tools queried by VirusTotal labeled the URL as malicious. We also performed manual inspection if necessary. For example, the URLs from PhishTank are crowdsourced, and Google Safe Browsing and VirusTotal do not detect all valid URLs from PhishTank as malicious. We manually visited such URLs to ensure that they are phishing webpages. Our final dataset consisted of 349,137 benign URLs and 5,231 malicious URLs. We used this dataset to train kAYO's model. We note that we waited for a number of months to determine if many of our pages were ever classified by this engine, so as to give other detection tools time to discover the candidate sites [43]. Finally, we used the lifetime of pages that were clear scams (e.g., banking pages that disappeared within 24 hours) to judge a small subset of pages.

5 Implementation and Evaluation

We describe the machine learning techniques we considered to tackle the problem of classifying mobile specific webpages as malicious or benign. We then discuss the strengths and weaknesses of each classification technique, and the process for selecting the best model for kAYO. We build and evaluate our chosen model for accuracy, false positive rate and true positive rate. Finally, we compare kAYO to existing techniques and empirically demonstrate the significance of kAYO's features. We note that where automated analysis is possible, we use our full datasets; however, as is commonly done in the research community, we use randomly selected subsets of our data when extensive manual analysis and verification is required.

5.1 Model Selection and Implementation

We treated the problem of detecting malicious webpages as a binary classification problem. We considered each known benign mobile webpage as a negative sample and each known malicious mobile webpage as a positive sample. We considered a wide range of popular binary classification techniques in machine learning, but for space discuss three popular options: Support Vector Machines (SVM), naïve Bayes and logistic regression.

Support Vector Machines (SVM) is a popular binary classifier. However, it works well only on a few thousand samples of data. Due to the scaling problem of SVMs and our large dataset, SVM was not the best choice for kAYO.

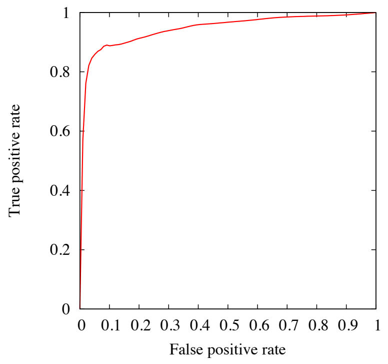


Fig. 3: The final ROC curve for kAYO's logistic regression model with regularization.

Naïve Bayes is generally used when the values of different features are mutually independent. Many features that we extracted were mutually dependent. For example, the number of scripts in a webpage was dependent on the number of internal, external and embedded JavaScript in the webpage, which were three other features of our model. Since the assumptions required for optimal performance of naïve Bayes did not hold for our dataset, we could not use the naïve Bayes classifier.
Logistic Regression is a scalable classification technique and makes no assumption about the distribution of values of the features. Therefore, this technique was the best fit for our dataset. We used the binomial variation of logistic regression to model kAYO and employed $\ell 1$-regularization to avoid overfitting of the data.

We used the *scrapy* [8] web scraping framework to crawl the collected mobile URLs. We then built a parser for extracting features discussed in Section 4.1 from each input webpage dynamically. The crawler and feature extraction scripts were implemented in Python. We used logistic regression on the extracted features for training and testing. We programmed the logistic regression model in the numerical computing language Octave [1]. We tested the model on a machine with quad core 3.4 GHz Intel Core i7 processor and 16 GB memory.

5.2 Evaluation

Our dataset contained 349,137 benign URLs and 5,231 malicious URLs. We divided our dataset into three subsets: training, cross-validation and test. We first randomly shuffled the data and set aside 10% of the data as the test set. The remaining 90% of data was used for training and 10-fold cross-validation.

For each validation round we calculated the accuracy, the false positive rate and the true positive rate on the validation set. We further used $\ell 1$-regularization to avoid overfitting. We varied the regularization parameter from 0 to 1,000 in the intervals of 10 and chose the best parameter. We then plotted a ROC curve by taking the mean of all false positive rates[6] and false negative rates[7] output from every cross-validation step, and found the best threshold for differentiating between malicious and benign data. Figure 3 shows the final ROC curve.

Factor	**Cantina** [56], [59]	**kAYO**
Designed to detect	Phishing	Mobile web threats
Detects pages written in	English-only	Any language
Avg. feature extraction time	**2.82 sec**	**0.016 sec**
Evaluation set size (# of webpages)	**200**	**34914**
True positive rate	97%	89%
False positive rate	6%	8%
External dependencies	Depends on Google search	None
Detects pages missed by Google Safe Browsing?	No	Yes

TABLE 3: Comparison of kAYO with Cantina, a technique using static webpage features to detect phishing webpages in real-time. kAYO's evaluation set size is over two orders of magnitude larger than that of Cantina. Moreover, kAYO's feature extraction process is two orders of magnitude faster than Cantina. Cantina's functionality is dependent on external tools unlike kAYO and Cantina works well only on webpages written in the English language. kAYO does not have these drawbacks.

kAYO provided 91% true positive rate and 7% false positive rate on the cross-validation set. We used the best parameters obtained from the training and cross-validation steps to test the 10% labeled dataset set aside. Our test set shows 90% accuracy, 8% false positive rate and 89% true positive rate. We believe that this rate is equivalent to that of the desktop-specific schemes, which use dramatically smaller datasets than our own, as accuracy falls significantly when datasets increase in size. We also anticipate that the false positive rate on the test set would be lower than what was found using the labeled samples because kAYO detected a number of malicious mobile URLs in the wild that we hand verified, and were not detected by tools that we used for establishing ground truth of our datatset. More details on examples and in-depth analysis of mobile malicious URLs detected by kAYO in the wild can be found in Section 7.
Comparison with existing static techniques:[8] We have identified and used 11 new mobile-relevant features previously not studied. We note that none of the existing techniques account for mobile specific features considered in kAYO. The non-commercial static analysis technique closest to kAYO is Cantina [59]. It detects phishing webpages in real-time using static features of webpages. We compare kAYO with the methodology,

6. False positive rate is equal to (1 - recall) or (1 - sensitivity).
7. False negative rate is also known as precision or specificity.
8. To the best of our knowledge, kAYO is the first technique that uses static features of webpages to detect malicious *mobile* pages. Therefore, we compare against existing desktop techniques. We also could not secure access to the code or software of these related desktop techniques from the respective authors upon request. Thus, our only option was to base kAYO's comparison on the results discussed in the related research papers of existing techniques.

Technique	Designed for		Tested on	False Pos rate	Evaluation set size
	Environment	Threat			
[20]	Desktop	Drive by Downloads	Drive by download only	9.9	15000
[53]	Desktop	Malicious JavaScript	Drive by download only	13.7	15000
[39]	Desktop	Spam URLs	Drive by download only	14.8	15000
Union of [39], [53] [24], [37]	Desktop	Drive by, malicious JS, spam URLs	Drive by download only	17.1	15000
kAYO	**Mobile**	**Existing mobile web threats**	**Existing mobile web threats**	**8.1**	**34914**

TABLE 4: Comparison of kAYO with five existing static analysis techniques that detect malicious desktop webpages. kAYO provides the lowest false positive rate on an evaluation set twice as large as the one used by other techniques. kAYO also considers mobile web threats, whereas, the other techniques are focused on detecting desktop web threats.

Technique	Time in sec		Considers mobile webpages?
	Feature extraction	Classification	
[20]	3.06	0.24	✗
[53]	0.15	0.034	✗
[39]	3.56	0.020	✗
Union of [24], [37], [39], [53]	N/A	N/A	✗
kAYO	**0.016**	**0.0002**	✓

TABLE 5: Comparison of kAYO with five existing static analysis techniques that detect malicious desktop webpages. kAYO's feature extraction process is 10 times faster than the fastest existing technique [53] and classification time is 100 times faster than the fastest existing technique [39]. kAYO is the *only* technique that considers mobile specific features of webpages.

speed and performance of Cantina given in related research papers [56], [59]. Table 3 summarizes the comparisons. Cantina provides better true positive rate and comparable false positive rate against kAYO. However, there are several drawbacks to Cantina. First, Cantina's depends on the results of Google's search engine. Moreover, Cantina assumes that every webpage not ranked by Google is malicious. We argue that this is a strong assumption and might lead to a high false positive rate. Additionally, this methodology prevents Cantina from analyzing webpages not visited by Google's Safe Browsing kAYO does not depend on any external tools and can detect malicious webpages missed by Google. Second, kAYO's feature extraction process is over two orders of magnitude faster than Cantina. On an average, kAYO takes 0.016 seconds to extract the features of a webpage and Cantina takes 2.82 seconds. We argue that this improvement in the speed of analyzing webpages makes kAYO more usable than Cantina in real-time. Finally, Cantina works only on webpages written in English due to its heuristic features whereas kAYO can work with webpages written in any language.

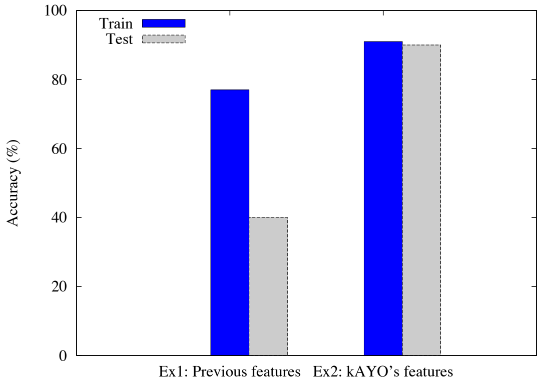


Fig. 4: *Ex1*: Results of a model trained on desktop webpages using desktop features studied in earlier techniques and then tested on mobile webpages. *Ex2*: Results of a model trained on mobile webpages by adding mobile specific features to the feature set and tested on mobile webpages. Ex1 shows that a model trained on desktop pages using features from related work performs poorly when applied to mobile webpages. However, when a model is trained with the same static features and additional mobile specific features exclusively on a mobile datatset, the results of testing on a mobile dataset improve significantly as seen in Ex2.

We also compared kAYO's performance with existing static analysis tools that detect non-phishing attacks. The closest non-commercial tool to kAYO based on the diversity of features and the scale of the evaluation set is Prophiler [20]. Prophiler detects drive-by-downloads on desktop webpages. We compare kAYO's performance with the performance numbers of existing static techniques described by Canali et al. [20]. Canali et al. performed an analysis of 15,000 webpages consisting of about 5,000 known webpages launching drive-by-downloads. The contenders of the comparison were then existing tools detecting malicious JavaScript [24], [37], [53], drive-by-downloads [20] and spam URLs [39]. Table 4 and Table 5 show the comparison of performance of kAYO with each of these techniques. kAYO provides the lowest false positive rate over an evaluation set twice as large as the one used by other techniques as shown in Table 4. Moreover, kAYO's feature extraction process is 10 times faster than the fastest existing technique [53] and classification process is 100 times faster than the fastest existing technique [39]. Finally, all the existing techniques are focused on desktop threats, whereas, kAYO focuses on mobile specific threats. Accordingly, had we been able to run these tools over our dataset, they would have performed more poorly.

Need for mobile specific techniques: Because neither Cantina nor Prophiler were made available to us, we performed an experiment to demonstrate the need for new mobile specific models. Intuitively, due to the disparity in the same static features when measured on mobile and desktop webpages (as discussed in Section 3), and the emergence of new mobile specific features, a model trained on desktop webpages will not generate precise results for mobile webpages. Note that we are not making claims about the *exact* performance of each system against our dataset; rather, we are attempting to demonstrate (in the absence of either being made available and in good faith) that previously published techniques not considering the changes and new features identified in this work perform significantly worse than our own when analyzing malicious mobile webpages.

For this experiment, we created a training dataset of desktop webpages and a test datatset of mobile webpages. We statically crawled Alexa top 10,000 webpages to the two links deep using the desktop Internet Explorer browser version 9.0 for Windows 7. We obtained the desktop malicious webpages by monitoring public blacklists [2], [3], [5] and crawling live URLs two links deep. We verified ground truth of these URLs using Google Safe Browsing and VirusTotal. We randomly shuffled the webpages and chose 10,000 webpages while keeping the proportion of benign and malicious webpages in the dataset equivalent to the mobile dataset described in Section 4.2. We then created a test dataset of 1000 mobile benign and malicious webpages by randomly selecting URLs from the larger dataset described in Section 4.2.

We extracted 33 out of the 44 static features in kAYO from each webpage in the desktop and mobile datasets. We disregarded the 11 new mobile features used in kAYO and instead focused our analysis on the 33 features previously used in similar desktop static techniques. We note that the goal of this experiment is not to extract all desktop relevant features used earlier, but demonstrate that a model trained on features extracted from desktop webpages does not perform well when applied to mobile webpages. We believe that these 33 features accurately represent the static features used in earlier techniques to detect malicious desktop webpages [20], [24], [28], [37], [39], [53], [56], [59].

We used logistic regression with regularization to train a model on the desktop webpage dataset and tested the model on the mobile dataset. Figure 4 shows the results of our experiments. Ex1 shows that using 33 features, we achieved 77% accuracy in training on desktop webpages. However, when the parameters obtained from this model were applied to the mobile dataset, the accuracy reduced significantly to 40%. *The difference between the accuracy of the training and testing dataset is the important comparison metric in this experiment as it demonstrates the inability of previous desktop-only models to accurately characterize mobile webpages.* Ex2 simply shows kAYO's results (discussed in Section 5.2) of training and testing on mobile webpages considering mobile specific features. Both training (91%) and testing (90%) dataset accuracies improve notably. More importantly, the accuracies of the training and testing datasets in Ex2 are comparable unlike those in Ex1. These results confirm our intuition that mobile specific static techniques are necessary - without using the new mobile features, previously proposed techniques perform poorly.

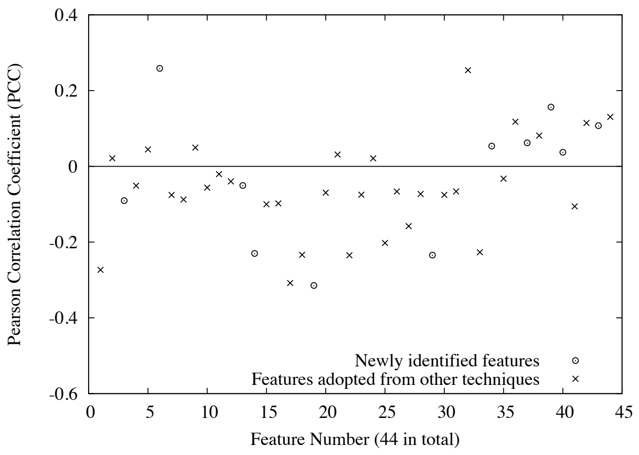


Fig. 5: The Pearson Coefficient Correlation (PCC) of each of the features extracted in kAYO with the label (malicious/benign), found using our evaluation dataset. Each point corresponds to the correlation of a feature with the label. In total there are 44 points corresponding to the 44 features of kAYO, including the newly identified features and the ones adopted from existing techniques. The Y value of each point depicts the predictive power of the corresponding feature i.e. PCC. The greater the absolute value of the PCC of a feature, the better predictive power of the feature. Note that all the PCC values are non-zero implying that every feature in kAYO's feature set is significant and impacts the result of classification.

Proprietary techniques: We note that Google's Safe Browsing tool is proprietary and no information can be retrieved about its performance. Moreover, Google Safe Browsing is not enabled on mobile browsers at present. Therefore, even if Google's tool identifies a mobile webpage as bad, Chrome and Firefox mobile browser users do not benefit from this information unlike the desktop users of these browsers. Even though enabling Safe Browsing on mobile is an engineering effort, we later demonstrate the importance of employing a mobile specific technique such as kAYO.

Significance of kAYO's feature set: It is important to observe that kAYO's feature set has been carefully created to ensure relevance to mobile webpages and negligible extraction time. We experimentally demonstrate the significance of kAYO's features using the Pearson product-moment Correlation Coefficient (*PCC*). PCC is a measure of the linear dependence between two variables giving a value between $+1$ and -1 inclusive [52]. In other words, PCC provides information about the predictive power of a feature over the classification result. The larger the absolute value of the PCC of a feature, the more its predictive power. For example, a feature with PCC -0.6 is a better predictor of whether a webpage is malicious than a feature with PCC 0.21. It is important to note that identifying features with very high PCC values is extremely difficult given the hundreds of different components of webpages and the diversity of threats.

We find the PCC between each feature in kAYO's feature set and the label (benign/malicious), from the test set used for evaluation. Intuitively, if kAYO's features are significant, then the absolute value of the PCC of each feature with the label must be non-zero. Figure 5 shows the plot of the PCC of each of the 44 features of kAYO with the label. The circles show the PCC of the newly identified features of kAYO and the Xs depict the PCC of features adopted from earlier works. As seen in the Figure, all the PCC values are non-zero, implying that every feature in kAYO is significant.

Comparison with existing browser tools: Browser extensions and plugins help protect users from visiting malicious websites. The most prevalent threat on the mobile web at present is phishing. Therefore, we surveyed the most popular anti-phishing Firefox desktop extensions for comparison with kAYO. These 33 extensions were selected by searching for the keyword 'phishing' on the Firefox extension store. Most of the extensions were certificate verifiers, password protectors or file protectors. We did not find any extensions performing content-based static analysis. We disregarded extensions that were built only for one specific website (e.g., FB Phishing Protector and LibertyGuard) or were no longer supported (e.g. Nophish). We then chose the top five extensions (Anti Phishing 1.0, DontPhishMe, Netcraft Toolbar, PhishTank SiteChecker and Phish Tester) based on the number of users for further analysis. We randomly selected a set of 10 known malicious URLs from our dataset and queried each tool with the URLs. PhishTank SiteChecker simply queried PhishTank and returned the result, detecting three of the 10 URLs. Netcraft detected three out of the 10 URLs as well, two of which were also detected by Phish Advisor. Anti Phishing 1.0 detected one URL. Phish Tester and DontPhishMe did not generate any results.

We also tested the freely available trial version of the Lookout safe browsing tool [4]. Lookout is one of the most popular security applications available for mobile devices. This tool protects users of the Android mobile and the Chrome mobile browsers from phishing scams and malicious links on the mobile web. We browsed the same 10 known malicious URLs from both the Android mobile and Chrome mobile browser on a device running the Android 4.0 operating system. *We were presented with alerts for only two out of the 10 URLs by Lookout, while kAYO detected eight out of the 10 webpages.*

Given the paucity of a working extension to detect different threats on mobile webpages, and the unavailability of signature-based tools such as Google Safe Browsing for mobile browsers, we developed a mobile browser extension using kAYO.

6 Browser Extension

Building a browser extension based on kAYO adds value for two reasons. First, the mobile specific design of kAYO enables detection of new threats previously unseen by existing services (e.g., pages including spam phone numbers). Second, building an extension allows immediate use of our technique. We discuss other potential avenues of adopting kAYO in Section 7.3.

We developed a browser extension using kAYO for Firefox Mobile[9], which informs users about the maliciousness of the webpages they intend to visit. Our goal was to build an extension that runs in real-time. Therefore, instead of running the feature extraction process in a mobile browser, we outsourced the processing intensive functions to a backend server. Figure 6 shows the architecture of the extension. User enters the URL he wants to visit in the extension toolbar. The extension then opens a socket and sends the URL and user agent information to kAYO's backend server over HTTPS. The server crawls the mobile URL and extracts static features from the webpage. This feature set is input to kAYO's trained model, which classifies the webpage as malicious or benign. The output is then sent back to the user's browser in real-time. If the URL is benign according to kAYO, the extension renders the intended webpage in the browser automatically. Otherwise, a warning message is shown to the user recommending them not to visit the URL.

Users of the extension will browse both mobile specific and desktop webpages since not all websites offer a mobile specific version. Recall that being a mobile specific technique, kAYO does not perform well on desktop webpages. Consequently, processing all pages of interest through kAYO might output incorrect results for desktop webpages. To address this problem, the backend server first detects whether the intended webpage is mobile specific using the same method explained in Section 4.2. The webpage is processed by kAYO only if it is mobile. The desktop webpages are analyzed using Google Safe Browsing. Note that any other existing technique for detecting desktop malicious webpages can be used instead of Google Safe Browsing.

We performed manual analysis of 100 randomly selected URLs (90 benign and 10 malicious) from our test dataset and measured the performance of kAYO in real-time. On an average, an output was rendered in 829 ms on average from the time the user entered a URL in kAYO's toolbar. We argue that the good performance is due to careful selection of quickly extractable features and lower complexity of mobile webpages as compared to desktop webpages. The maximum delay in result generation was seen in scraping the input webpage from its respective server. Caching already scraped webpages can reduce this delay, as we demonstrated experimentally, by an average of 85%. Figure 7 shows a screen shot of our browser extension at work. We plan to make the extension available publicly post publication.

9. Firefox Mobile is one of the very few mainstream mobile browsers that support browser extensions. Similar extensions can easily be developed on other mobile browsers once supported.

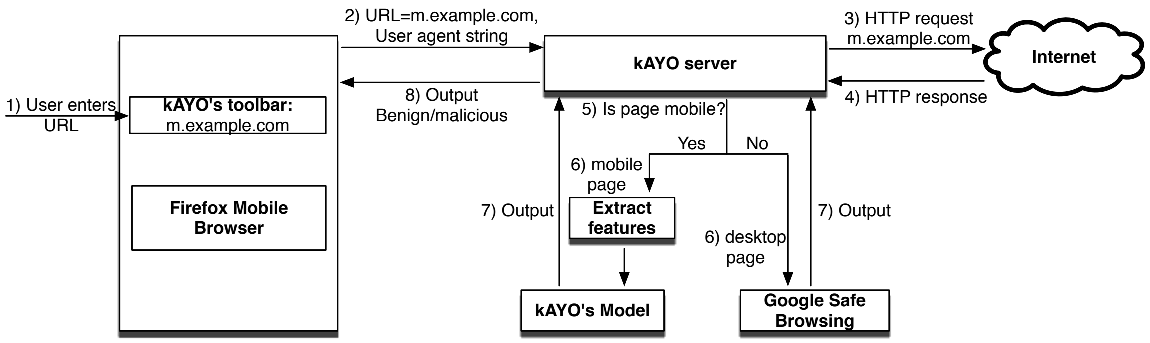
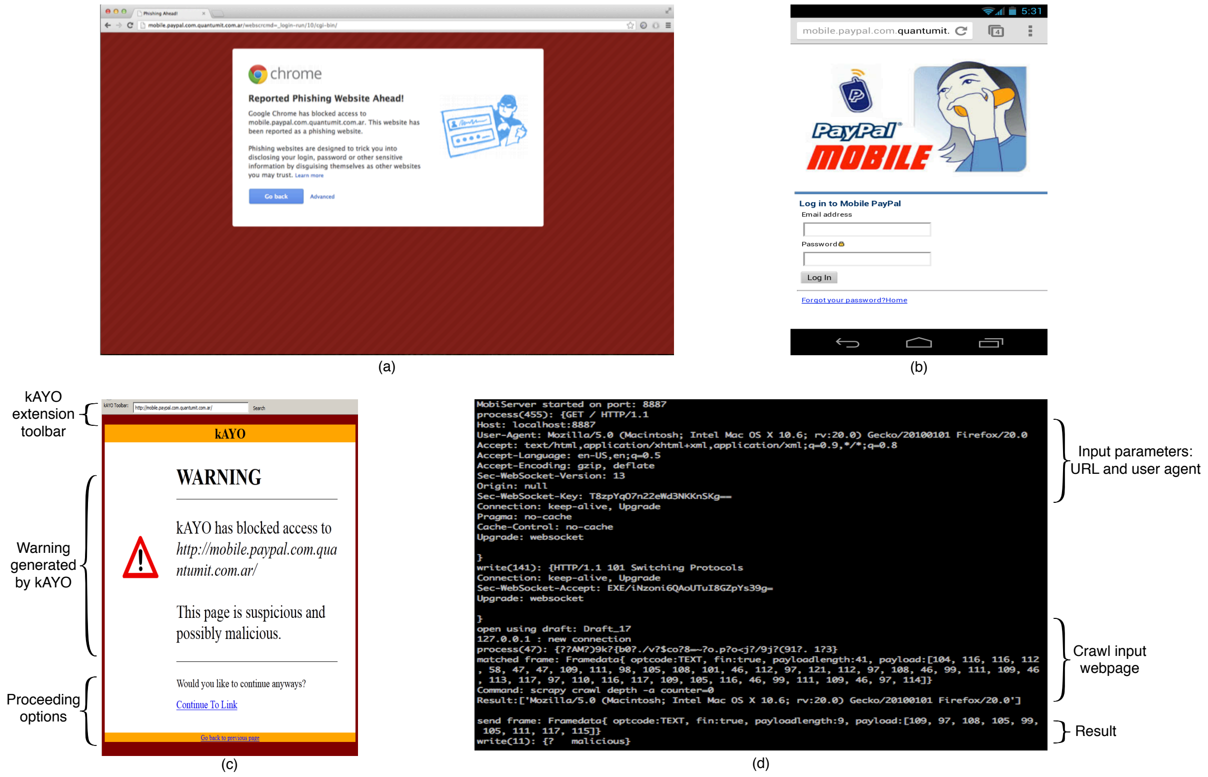


Fig. 6: Architecture of the mobile browser extension based on kAYO. User enters the URL he wants to visit in the extension toolbar and receives a response in real-time from our backend server about the maliciousness of the URL. If the URL is benign according to kAYO, the page of interest is rendered in the browser. Otherwise, the user is shown a warning message to not visit the URL.


(a)
mobile.paypal.com.quantumit.com.ar/webscrcmd=_login-run/10/cgi-bin/
chrome
Go back
Advanced
(b)
mobile.paypal.com.quantumit.
PayPal MOBILE
Log in to Mobile PayPal
Email address
Password
Log In
Forgot your password?Home
(c)
kAYO extension toolbar
Warning generated by kAYO
Proceeding options
kAYO Toolbar: http://mobile.paypal.com.quantumit.com.ar/
Search
kAYO
Continue To Link
Go back to previous page
(d)
Input parameters: URL and user agent
Crawl input webpage
Result

Reported Phishing Website Ahead!

Google Chrome has blocked access to mobile.paypal.com.quantumit.com.ar. This website has been reported as a phishing website.

Phishing websites are designed to trick you into disclosing your login, password or other sensitive information by disguising themselves as other websites you may trust. Learn more

WARNING

kAYO has blocked access to *http://mobile.paypal.com.quantumit.com.ar/*

This page is suspicious and possibly malicious.

Would you like to continue anyways?

```
MobiServer started on port: 8887
process(455): {GET / HTTP/1.1
Host: localhost:8887
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:20.0) Gecko/20100101 Firefox/20.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Sec-WebSocket-Version: 13
Origin: null
Sec-WebSocket-Key: T8zpYqO7n22eWd3NKKnSKg==
Connection: keep-alive, Upgrade
Pragma: no-cache
Cache-Control: no-cache
Upgrade: websocket

}
write(141): {HTTP/1.1 101 Switching Protocols
Connection: keep-alive, Upgrade
Sec-WebSocket-Accept: EXE/iNzoni6QAoUTuI8GZpYs39g=
Upgrade: websocket

}
open using draft: Draft_17
127.0.0.1 : new connection
process(47): {??AM?)9k?{b0?./v?$co?8=~?o.p?o<j?/9j?(91?. 1?3}
matched frame: Framedata{ optcode:TEXT, fin:true, payloadlength:41, payload:[104, 116, 116, 112
, 58, 47, 47, 109, 111, 98, 105, 108, 101, 46, 112, 97, 121, 112, 97, 108, 46, 99, 111, 109, 46
, 113, 117, 97, 110, 116, 117, 109, 105, 116, 46, 99, 111, 109, 46, 97, 114]}
Command: scrapy crawl depth -a counter=0
Result:['Mozilla/5.0 (Macintosh; Intel Mac OS X 10.6; rv:20.0) Gecko/20100101 Firefox/20.0']

send frame: Framedata{ optcode:TEXT, fin:true, payloadlength:9, payload:[109, 97, 108, 105, 99,
 105, 111, 117, 115]}
write(11): {?   malicious}
```

Fig. 7: (a): Chrome desktop browser informing the user of a potentially malicious webpage. The webpage is a known mobile phishing webpage. (b): The same webpage when rendered on the Chrome mobile browser, whose users are the real targets, does not provide any warning. (c): kAYO extension running on the Firefox mobile browser detects the webpage as malicious and warns the user. (d): Screenshot of command line processing at the extension server.

7 Discussion

kAYO detected a number of malicious webpages in the wild that were not found by existing techniques. We investigate these webpages in detail and then describe the limitations and future work of kAYO.

7.1 Investigating False Positives

We used Google Safe Browsing and VirusTotal for establishing ground truth of our dataset for training and evaluating kAYO. However, such dynamic analysis techniques execute webpages on desktop browsers running on virtual machines and miss mobile specific threats. To validate our intuition, we performed manual analysis on webpages that were identified as malicious by kAYO, but were tagged as benign by Google Safe Browsing and VirusTotal. Performing manual in-depth analysis for all webpages classified as malicious by kAYO was not feasible. Therefore, we chose a random subset of 100 URLs from the false positives obtained by running kAYO on the test dataset in Section 5.2.

We verified each of the 100 URLs by visiting them manually from an Android mobile browser version 4.0. We found 10 URLs to be suspicious. These 10 URLs contained survey pages to win iPads or Visa gift cards, uncommon online electronic equipment stores and stores selling health-related products. Most URLs did not have a Google page rank. One particular webpage prompted a user to download a binary file masquerading as a flash update. We downloaded and found the binary file to be malicious by querying VirusTotal. Another webpage had a known bank fraud phone number prefixed with the `tel:` API. Two out of the 10 suspicious URLs were also marked as suspicious by the Lookout safe browsing tool. We have reported these 10 webpages to PhishTank. Five out of the 10 URLs that we submitted have already been validated by PhishTank and marked as malicious. All 10 URLs went offline within one week of submission. This further strengthens our intuition since phishing URLs are usually short lived [31], [44]. We note that PhishTank might not validate some of the URLs we submitted. This is because, PhishTank's validation process is based on crowdsourcing and threats such as known bank fraud numbers on a website might not be detected without the availability of tools such as Pindrop PRS [7].

7.2 Cross-channel threats

We found 173 unique mobile webpages in our dataset (including training and testing) that hosted API prefixed known fraudulent phone numbers and were all tagged as benign by Google Safe Browsing and VirusTotal. These numbers are associated with a number of known financial fraud campaigns against a number of different major US-based institutions) according to our queries to the Pindrop Security PRS [7]. These results show that adversaries have begun to exploit such cross-channels (e.g., create a phishing webpage and include a fraud phone number) to attack mobile users. Moreover, these experiments suggest that *the false positive rate of kAYO might be lower in reality, given that mechanisms fail to classify such pages as malicious*. We intend to conduct a further analysis of such attacks in our future work.

7.3 Limitations and Future Work

The expected concerns of kAYO are similar to those of existing malicious website detection tools using static analysis. Evasion by mimicking the features we consider to be good indicators of a legitimate webpage can be used to defeat kAYO. However, our comprehensive set of features makes it harder to evade kAYO, as seen from our evaluation over a large dataset.

We statically crawled the top million websites of Alexa. Therefore, we did not collect webpages that use JavaScript to detect and redirect to the mobile webpage. We have also missed the mobile webpages represented by ways other than the ones used by the top 1,000 websites. We do not make any claims about gathering all mobile webpages from Alexa top one million. However, given the large set of webpages collected, we believe that our dataset is a representative cross section. Finally, the focus of this work was on mobile webpages designed for phones. We defer the analysis of webpages built for tablets to future work.

kAYO's features reflect current trends in mobile malicious webpages. The potential of bad activity in the mobile web could increase yet further over time. kAYO's feature set and model will need to be updated, according to the new threats faced by the mobile web in the future. However, such updates are necessary in all static techniques that aim to detect new threats.

In-depth dynamic analysis of webpages may provide additional important details. However, because such approaches incur significantly higher costs, this approach conflicts with our design goal of creating a real-time detector. Accordingly, we leave the significant challenge of efficient operation of such tools to future work.

Using signature based blacklist approaches such as Google Safe Browsing might improve the performance of kAYO's browser extension. A blacklist can be synchronized with kAYO's extension server and enforced locally. Although such techniques might reduce the average delay in page rendering, they will also preclude from protection against webpages that change dynamically defeating kAYO's goal of real-time evaluation. We plan to investigate performance enhancing designs that preserve real-time evaluation in future work.

8 Conclusion

Mobile webpages are significantly different than their desktop counterparts in content, functionality and layout. Therefore, existing techniques using static features of desktop webpages to detect malicious behavior do not work well for mobile specific pages. We designed and developed a fast and reliable static analysis technique called kAYO that detects mobile malicious webpages. kAYO makes these detections by measuring 44 mobile relevant features from webpages, out of which 11 are newly identified mobile specific features. kAYO provides 90% accuracy in classification, and detects a number of malicious mobile webpages in the wild that are not detected by existing techniques such as Google Safe Browsing and VirusTotal. Finally, we build a browser extension using kAYO that provides real-time feedback to users. We conclude that kAYO detects new mobile specific threats such as websites hosting known fraud numbers and takes the first step towards identifying new security challenges in the modern mobile web.

Acknowledgments

The authors would like to thank EldoS for access to their driver framework and VirusTotal for providing malware samples. This work was supported in part by the US National Science Foundation under grant numbers CNS-1464087 and CNS-1464088. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

References

[1] Gnu octave: high-level interpreted language. http://www.gnu.org/software/octave/.

[2] hphosts, a community managed hosts file. http://hphosts.gt500.org/hosts.txt.

[3] Joewein.de LLC blacklist. http://www.joewein.net/dl/bl/dom-bl-base.txt.

[4] Lookout. https://play.google.com/store/apps/details?hl=en&id=com.lookout.

[5] Malware Domains List. http://mirror1.malwaredomains.com/files/domains.txt.

[6] Phishtank. http://www.phishtank.com/.

[7] Pindrop phone reputation service. http://pindropsecurity.com/phone-fraud-solutions/phone_reputation_service_prs/.

[8] Scrapy — an open source web scraping framework for python. http://scrapy.org/.

[9] VirusTotal. https://www.virustotal.com/en/.

[10] Google developers: Safe Browsing API. https://developers.google.com/safe-browsing/, 2012.

[11] Alexa, the web information company. http://www.alexa.com/topsites, 2013.

[12] dotmobi. internet made mobile. anywhere, any device. http://dotmobi.com/, 2013.

[13] C. Amrutkar, K. Singh, A. Verma, and P. Traynor. VulnerableMe: Measuring systemic weaknesses in mobile browser security. In *Proceedings of the International Conference on Information Systems Security (ICISS)*, 2012.

[14] C. Amrutkar, P. Traynor, and P. C. van Oorschot. Measuring SSL indicators on mobile browsers: Extended life, or end of the road? In *Proceedings of the Information Security Conference (ISC)*, 2012.

[15] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *Proceedings of the 19th USENIX Conference on Security (SECURITY)*, 2010.

[16] V. A. Balasubramaniyan, A. Poonawalla, M. Ahamad, M. T. Hunter, and P. Traynor. Pindr0p: using single-ended audio features to determine call provenance. In *Proceedings of the 17th ACM Conference on Computer and Communications Security (CCS)*, 2010.

[17] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE : Finding malicious domains using passive DNS analysis. In *Proceedings of the 18th Annual Network and Distributed System Security Symposium (NDSS)*, 2011.

[18] M. Boodaei. Mobile users three times more vulnerable to phishing attacks. http://www.trusteer.com/blog/mobile-users-three-times-more-vulnerable-to-phishing-attacks, 2011.

[19] M. Butkiewicz, Z. Wu, S. Li, P. Murali, V. Hristidis, H. V. Madhyastha, and V. Sekar. Enabling the transition to the mobile web with websieve. In *Proceedings of the 14th Workshop on Mobile Computing Systems and Applications (HotMobile)*, 2013.

[20] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *Proceedings of the 20th International Conference on World Wide Web (WWW)*, 2011.

[21] S. Chakradeo, B. Reaves, P. Traynor, and W. Enck. MAST: Triage for Market-scale Mobile Malware Analysis. In *Proceedings of the ACM Conference on Security and Privacy in Wireless and Mobile Networks (WiSec)*, 2013.

[22] R. Dhamija, J. D. Tygar, and M. Hearst. Why phishing works. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006.

[23] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri. A study of Android application security. In *Proceedings of the 20th USENIX Security Symposium*, 2011.

[24] B. Feinstein and D. Peck. Caffeine monkey: Automated collection, detection and analysis of malicious javascript. In *Proceedings of the Black Hat Security Conference*, 2007.

[25] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner. Android permissions demystified. In *Proceedings of the 18th ACM conference on Computer and communications security*, 2011.

[26] A. P. Felt and D. Wagner. Phishing on mobile devices. In *Web 2.0 Security and Privacy (W2SP)*, 2011.

[27] A. P. Felt, H. J. Wang, A. Moshchuk, S. Hanna, and E. Chin. Permission re-delegation: attacks and defenses. In *Proceedings of the 20th USENIX conference on Security*, 2011.

[28] I. Fette, N. Sadeh, and A. Tomasic. Learning to detect phishing emails. In *Proceedings of the 16th International Conference on World Wide Web (WWW)*, 2007.

[29] S. Gajek, A.-R. Sadeghi, C. Stüble, and M. Winandy. Compartmented security for browsers or how to thwart a phisher with trusted computing. In *Second International Conference on Availability, Reliability and Security (ARES)*, 2007.

[30] S. Garera, N. Provos, M. Chew, and A. D. Rubin. A framework for detection and measurement of phishing attacks. In *Proceedings of the ACM workshop on recurring malcode*, 2007.

[31] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling. Measuring and detecting fast-flux service networks. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2008.

[32] A. Ikinci, T. Holz, and F. Freiling. Monkey-spider: Detecting malicious websites with low-interaction honeyclients. In *Proceedings of Sicherheit, Schutz und Zuverlassigkeit*, 2008.

[33] L. Invernizzi, S. Benvenuti, M. Cova, P. M. Comparetti, C. Kruegel, and G. Vigna. Evilseed: A guided approach to finding malicious web pages. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 2012.

[34] P. Kolari, T. Finin, and A. Joshi. Svms for the blogosphere: Blog identification and splog detection. In *Proceedings of AAAI Spring Symposium on Computational Approaches to Analysing Weblogs*, 2006.

[35] A. Le, A. Markopoulou, and M. Faloutsos. Phishdef: Url names say it all. In *Proceedings of IEEE International Conference on Computer Communications (INFOCOM)*, 2011.

[36] C. Lever, M. Antonakakis, B. Reaves, P. Traynor, and W. Lee. The core of the matter: Analyzing malicious traffic in cellular carriers. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2013.

[37] P. Likarish, E. Jung, and I. Jo. Obfuscated malicious javascript detection using classification techniques. In *Proceedings of Malicious and Unwanted Software (MALWARE)*, 2009.

[38] C. Ludl, S. Mcallister, E. Kirda, and C. Kruegel. On the effectiveness of techniques to detect phishing sites. In *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2007.

[39] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proceedings of the SIGKDD Conference*, 2009.

[40] D. K. McGrath and M. Gupta. Behind phishing: an examination of phisher modi operandi. In *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[41] E. Medvet, E. Kirda, and C. Kruegel. Visual-similarity-based phishing detection. In *Proceedings of International Conference on Security and Privacy in Communication Netowrks (SecureComm)*, 2008.

[42] Y. min Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys: Finding web sites that exploit browser vulnerabilities. In *Proceedings of the Networking and Distributed Systems Security (NDSS)*, 2006.

[43] A. Mohaisen and O. Alrawi. AV-Meter: An Evaluation of Antivirus Scans and Labels. In *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2014.

[44] T. Moore, R. Clayton, and H. Stern. Temporal correlations between spam and phishing websites. In *Proceedings of the 2nd USENIX conference on Large-scale exploits and emergent threats: botnets, spyware, worms, and more (LEET)*, 2009.

[45] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A crawler-based study of spyware on the web. In *Proceedings of Network and Distributed System Security Symposium (NDSS)*, 2006.

[46] J. Nazario. Phoneyc: a virtual client honeypot. In *Proceedings of the 2nd USENIX conference on Large-scale Exploits and Emergent Threats: botnets, spyware, worms, and more (LEET)*, 2009.

[47] J. Nazario and T. Holz. As the net churns: Fast-flux botnet observations. In *Proceedings of 3rd International Conference on Malicious and Unwanted Software (MALWARE)*, 2008.

[48] E. Passerini, R. Paleari, L. Martignoni, and D. Bruschi. Fluxor: Detecting and monitoring fast-flux service networks. In *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2008.

[49] R. Perdisci, I. Corona, D. Dagon, and W. Lee. Detecting malicious flux service networks through passive analysis of recursive DNS traces. In *Proceedings of Annual Computer Security Applications Conference (ACSAC)*, 2009.

[50] G. Podjarny. Mobile web performance optimization. http://www.slideshare.net/blazeio/mobile-web-performance-optimization-tips-and-tricks.

[51] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iframes point to us. In *Proceedings of the 17th USENIX conference on Security (SECURITY)*, 2008.

[52] L. Rodgers and W. A. Nicewander. Thirteen ways to look at the correlation coefficient. *The American Statistician*, 1988.

[53] C. Seifert, I. Welch, and P. Komisarczuk. Identification of malicious web pages with static heuristics. In *Telecommunication Networks and Applications Conference*, 2008.

[54] F. Weimer. Passive DNS replication. 2005.

[55] C. Whittaker, B. Ryner, and M. Nazif. Large-scale automatic classification of phishing pages. In *Proceedings of the Networking and Distributed Systems Security (NDSS)*, 2010.

[56] G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2), Sept. 2011.

[57] C. Yue and H. Wang. Characterizing insecure javascript practices on the web. In *Proceedings of the 18th international conference on World Wide Web (WWW)*, 2009.

[58] B. Zdrnja, N. Brownlee, and D. Wessels. Passive monitoring of DNS anomalies. In *Proceedings of the 4th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2007.

[59] 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 (WWW)*, 2007.