Detecting Mobile Malicious Webpages in Real Time

Syeda Sumaira Mohammadi 1, Ghazala Afreen 2, Syeda Humaira Mohammadi 3
Mohammed Liaqat Ali Khan 4
1BE Student, 2BE Student, 3BE Student 4Asst. Professor
CSE Dept, ISL ENGINEERING COLLEGE, Hyd.

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 kAYO1, 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.

Objective of the Project

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.

2. LITERATURE SURVEY

VulnerableMe: Measuring systemic weaknesses in mobile browser security.

Porting browsers to mobile platforms may lead to new vulnerabilities whose solutions require careful balancing between usability and security and might not always be equivalent to those in desktop browsers. In this paper, we perform the first large-scale security comparison between mobile and desktop browsers. We focus our efforts on display security given the inherent screen limitations of mobile phones. We evaluate display elements in ten mobile, three tablet and five desktop browsers. We identify two new classes of vulnerabilities specific to mobile browsers and demonstrate their risk by launching real-world attacks including display ballooning, login CSRF and clickjacking. Additionally, we implement a new phishing attack that exploits a default
policy in mobile browsers. These previously unknown vulnerabilities have been confirmed by browser vendors. Our observations, inputs from browser vendors and the pervasive nature of the discovered vulnerabilities illustrate that new implementation errors leading to serious attacks are introduced when browser software is ported from the desktop to mobile environment. We conclude that usability considerations are crucial while designing mobile solutions and display security in mobile browsers is not comparable to that in desktop browsers.

Measuring SSL indicators on mobile browsers: Extended life, or end of the road?

Mobile browsers are increasingly being relied upon to perform security sensitive operations. Like their desktop counterparts, these applications can enable SSL/TLS to provide strong security guarantees for communications over the web. However, the drastic reduction in screen size and the accompanying reorganization of screen real estate significantly changes the use and consistency of the security indicators and certificate information that alert users of site identity and the presence of strong cryptographic algorithms. In this paper, we perform the first measurement of the state of critical security indicators in mobile browsers. We evaluate ten mobile and two tablet browsers, representing over 90% of the market share, using the recommended guidelines for web user interface to convey security set forth by the World Wide Web Consortium (W3C). While desktop browsers follow the majority of guidelines, our analysis shows that mobile browsers fall significantly short. We also observe notable inconsistencies across mobile browsers when such mechanisms actually are implemented. Finally, we use this evidence to argue that the combination of reduced screen space and an independent selection of security indicators not only make it difficult for experts to determine the security standing of mobile browsers, but actually make mobile browsing more dangerous for average users as they provide a false sense of security.

Building a dynamic reputation system for DNS.

The Domain Name System (DNS) is an essential protocol used by both legitimate Internet applications and cyber attacks. For example, botnets rely on DNS to support agile command and control infrastructures. An effective way to disrupt these attacks is to place malicious domains on a "blocklist" (or "blacklist") or to add a filtering rule in a firewall or network intrusion detection system. To evade such security countermeasures, attackers have used DNS agility, e.g., by using new domains daily to evade static blacklists and firewalls. In this paper we propose Notos, a dynamic reputation system for DNS. The premise of this system is that malicious, agile use of DNS has unique characteristics and can be distinguished from legitimate, professionally provisioned DNS services. Notos uses passive DNS query data and analyzes the network and zone features of domains. It builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a
new domain indicative of whether the domain is malicious or legitimate. We have evaluated Notos in a large ISP's network with DNS traffic from 1.4 million users. Our results show that Notos can identify malicious domains with high accuracy (true positive rate of 96.8%) and low false positive rate (0.38%), and can identify these domains weeks or even months before they appear in public blacklists.

MAST: Triage for Market scale Mobile Malware Analysis.

Malware is a pressing concern for mobile application market operators. While current mitigation techniques are keeping pace with the relatively infrequent presence of malicious code, the rapidly increasing rate of application development makes manual and resource-intensive automated analysis costly at market-scale. To address this resource imbalance, we present the Mobile Application Security Triage (MAST) architecture, a tool that helps to direct scarce malware analysis resources towards the applications with the greatest potential to exhibit malicious behavior. MAST analyzes attributes extracted from just the application package using Multiple Correspondence Analysis (MCA), a statistical method that measures the correlation between multiple categorical (i.e., qualitative) data. We train MAST using over 15,000 applications from Google Play and a dataset of 732 known-malicious applications. We then use MAST to perform triage on three third-party markets of different size and malware composition---36,710 applications in total. Our experiments show that MAST is both effective and performant. Using MAST ordered ranking, malware-analysis tools can find 95% of malware at the cost of analyzing 13% of the non-malicious applications on average across multiple markets, and MAST triage processes markets in less than a quarter of the time required to perform signature detection. More importantly, we show that successful triage can dramatically reduce the costs of removing malicious applications from markets.

Evilseed: A guided approach to finding malicious web pages.

Malicious web pages that use drive-by download attacks or social engineering techniques to install unwanted software on a user's computer have become the main avenue for the propagation of malicious code. To search for malicious web pages, the first step is typically to use a crawler to collect URLs that are live on the Internet. Then, fast prefiltering techniques are employed to reduce the amount of pages that need to be examined by more precise, but slower, analysis tools (such as honey clients). While effective, these techniques require a substantial amount of resources. A key reason is that the crawler encounters many pages on the web that are benign, that is, the "toxicity" of the stream of URLs being analyzed is low. In this paper, we present EVILSEED, an approach to search the web more efficiently for pages that are likely malicious. EVILSEED starts from an initial seed of known, malicious web pages. Using this seed, our system automatically generates search engines queries to identify other malicious pages that are similar or related to the ones in the initial seed. By doing so, EVILSEED leverages the crawling
infrastructure of search engines to retrieve URLs that are much more likely to be malicious than a random page on the web. In other words EVILSEED increases the "toxicity" of the input URL stream. Also, we envision that the features that EVILSEED presents could be directly applied by search engines in their prefilters. We have implemented our approach, and we evaluated it on a large-scale dataset. The results show that EVILSEED is able to identify malicious web pages more efficiently when compared to crawler-based approaches.

3. ANALYSIS

Introduction

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system the software development process.

Existing System

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. 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.

Disadvantages of Existing System:

1. Previous techniques also fail to consider mobile specific webpage elements such as calls to mobile APIs.

Proposed System

In this paper, we present kAYO, 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.

Advantages of Proposed System:

1. kAYO’s performance matches or exceeds that of existing static techniques used in the desktop space.
2. kAYO also detects a number of malicious mobile webpages not precisely detected by existing techniques such as VirusTotal and Google Safe Browsing.

3.1. PROCESS MODEL USED WITH JUSTIFICATION

Stages in SDLC:

- Requirement Gathering
- Analysis
- Designing
- Coding
- Testing
- Maintenance

Requirements Gathering stage:

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports.

A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and textual description.

These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are not included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term requirements traceability.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

- Feasibility study is all about identification of problems in a project.
- No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
- Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator.

**Analysis Stage:**

The planning stage establishes a bird’s eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.

The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype
Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.

When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.

The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.
**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

**Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.

The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.
Maintenance:

Outer rectangle represents maintenance of a project. Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category. For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

4. SCREEN SHOTS

To execute project double click on ‘run.bat’ file to get below screen

Click on ‘Upload Mobile Webpages’ to upload entire dataset folder. This folder contains 12 web pages

After dataset upload will get below screen
Now click on “Process Pages For Kayo Features’ to read dataset and prepare processing features.

In above screen displaying process web page no, name and web page size.

Now click on ‘Javascript Features Set’ to check for no of javascript tags used in web pages by web designer.

Now click on ‘HTML features set’ to get no of redirection pages used by web designer in webpage.
Now click on ‘URL features set’ to get no of sub domains used by web designer in web page.

<table>
<thead>
<tr>
<th>Page Name</th>
<th>Subdomain Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.html</td>
<td>10</td>
</tr>
<tr>
<td>10.html</td>
<td>2</td>
</tr>
<tr>
<td>11.html</td>
<td>1</td>
</tr>
<tr>
<td>7.html</td>
<td>7</td>
</tr>
<tr>
<td>1.html</td>
<td>0</td>
</tr>
<tr>
<td>4.html</td>
<td>0</td>
</tr>
<tr>
<td>5.html</td>
<td>6</td>
</tr>
<tr>
<td>6.html</td>
<td>8</td>
</tr>
<tr>
<td>7.html</td>
<td>7</td>
</tr>
<tr>
<td>8.html</td>
<td>9</td>
</tr>
<tr>
<td>9.html</td>
<td>1</td>
</tr>
</tbody>
</table>

Now click on ‘Mobile features set’ to get no of time mobile api (mobile api such as sms, smsto, mms etc) used by web designer inside web page.

<table>
<thead>
<tr>
<th>Page Name</th>
<th>SMS Count</th>
<th>MMS Count</th>
<th>Telephone Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11.html</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5.html</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6.html</td>
<td>4</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>7.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>technical.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>wikiworldmedia.html</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Now click on ‘View Detection Classification’ button to predict weather web page is malicious or not by analyzing above features count. If count > threshold then that page consider as malicious otherwise consider as benign.

<table>
<thead>
<tr>
<th>Page Name</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>3.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>4.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>6.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>8.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>11.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>9.html</td>
<td>Malicious Page</td>
</tr>
<tr>
<td>10.html</td>
<td>Benign Page</td>
</tr>
<tr>
<td>7.html</td>
<td>Benign Page</td>
</tr>
<tr>
<td>technical.html</td>
<td>Benign Page</td>
</tr>
<tr>
<td>wikiworldmedia.html</td>
<td>Benign Page</td>
</tr>
</tbody>
</table>

Now click on ‘Detection Chart’ button to display graph for no of malicious and benign pages.
5. 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.

6. BIBLIOGRAPHY

browsing/, 2012.