SYSTEM AND FRAMEWORK FOR IDENTIFICATION OF INFO MALWARE ON THE ANDROID OPERATING SYSTEM

BY

ROBERT MARSAN

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2013

Urbana, Illinois

Adviser:

Professor Roy H. Campbell

Abstract

The tremendous rise of mobile computing has led to an equally strong rise in mobile malware. We address this issue with several novel concepts, permission fingerprints and user-app agreements. We also present a groundbreaking and substantial new dataset of the state of Android Apps, and provide unique results gathered from it. Finally, we present AndroMEDA, an Android Security Extension which helps address many of the shortcomings to existing techniques of malware identification currently available.

This *seriously* needs to be extended.

To my

Parents, Lisa and Mark

Acknowledgments

I would like to thank my advisor Roy H. Campbell for his mentorship and guidance, as well as Alejandro Gutierrez for putting up with me.

Table of Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

Thesis Statement: Malware on Mobile Operating Systems, especially Android, is better understood when not just analyzing the capabilities of an application, but the expectations the user has as to how it utilizes those capabilities as well.

The rise of smartphones in the last 6 years has been nothing short of meteoric. Since the launch of the Apple iPhone in 2007, over ?1 billion or so? smartphones have been sold, from over ?100 or so?(Author, Year) manufacturers. These new devices marked an unprecedented shift in our relationship with computers, becoming the center point for many personal endeavors, and superseding almost all previous computing devices from cell phones, to cameras, to GPS devices, and to most uses of a desktop pc (Author, Year). Indeed, smartphones continue to become the focal point of almost all personal computing, and consequently the operating systems they run become more important and powerful.

Mobile operating systems (mobile OSs), like the PC operating systems of the 1990s, have a few major players that wield the most influence. The two largest operating systems in the mobile area are Android and iOS. Apples iOS, made exclusively for the Apple iPhone and iPad, currently runs on over 15%(Author, Year) of all smartphones globally. Googles Android, released as an open source OS, has many different hardware manufacturers, Samsung, LG, HTC, Motorola, and many more. It currently runs the majority of smartphones globally, with a 75% marketshare. Some of the less popular, but still significant mobile operating systems are Windows Phone, with 2%, and Blackberry, with 4%.

?Introduce iOS and Android!?

The diversity of hardware that smartphones were designed to replace, along other constraints and features, requires a mobile OS thats designed from the ground up to deal with many different challenges than the typical PC OS. Some of the main design challenges for a mobile OS are:

- Small memory footprint, battery conscious, and other resource constrictions
- Access to a wide variety of personally identifiable information (PII)
- Access a wide array of hardware

In order to effectively enforce rules on battery consumption, low-latency UI, and personally identifiable information, a new security model was created, centered around the concept of the App.

This security model has dramatically changed the nature of mobile software, and in turn, mobile malware. By forcing malware to fit inside of this security sandbox, malware authors must choose to either break out of the box, or work inside of it. This constriction has blurred the definition of mobile malware, and ultimately has profound implications for the user of the mobile device. We will examine these implications, and propose new tools and methods to help understand the nuances of modern malware, as well as provide a framework of tools to detect them.

Chapter 2

Background & Modivation

To understand the nature of modern mobile malware, we first examine the context. The two main models, that of iOS and Android, are compared and contrasted here.

2.1 The “App” and Sandboxing

In the mobile world, “Apps” are isolated and sandboxed programs, generally designed with one singular purpose. They lack dependencies, and generally are not as privileges as system software for performing many tasks. The mechanisms for accessing functionality outside of their sandbox is enforced by a set of policies the system holds, specific to that app. On some platforms, like iOS, only one app may run at any given time, and background computation is virtually non-existent (with some exceptions)[1], along with many other restrictions. On Android and other platforms, many more features are available to apps, but in all cases, the “app” lifecycle is well defined and controlled by the system much more than on a PC OS.

There are various reasons for the tight sandboxing of mobile apps. Power and resource consumption are certainly a factor - mobile OSs generally reserve the right to kill apps if they attempt to allocate too much memory. Controlling access to hardware also helps in this: allowing apps to keep the phone awake could easily drain battery. However, another reason for sandboxing, and arguably more important, is protecting Personally Identifiable Information.

2.1.1 Persionally Identifiable Information

Personally Identifiable Information (PII), as defined by NIST, is “any information about an individual maintained by an agency, including (1) any information that can be used to distinguish or trace an individuals identity... and (2) any other information that is linked or linkable to an individual” [2]. Mobile devices, having replaced cameras, cell phones, GPS devices, and PCs, have an extremely diverse amount of PII, from phone numbers, to contacts, to location history, to bank account numbers and pictures. For many of these datasets, mobile OSs actually organize them into databases with the intention of allowing 3rd parties access to them. Contact lists, SMS, Photographs and location

[1]Minor amounts of computation can be done to compute background audio, and other isolated background tasks.

history are available to apps on virtually every mobile platform in some official way. This is a driving motivation for a greatly improved security model for mobile OSs: controlling 3rd party softwares access to PII.

2.2 The “App Stores” and controlled distribution

The final major difference between mobile OSs and PC OSs is the distribution of code. No mobile OS allows code to be ran outside of the sandbox, and all of them require the user to agree before installing an app. All apps must be signed, and in general, there is 1 main distribution channel for all apps on a mobile OS. This tightly controlled distribution both aids in security, as well as controls the ecosystem around that mobile OS.

The first major distribution platform for mobile apps was Apples App Store[1]. Its model has been repeated by almost all major mobile app distribution platforms. The basic premise is simple: developers sign up to the app store, pay a fee (usually yearly), and submit fully-finished apps. A reviewer runs the app in a monitored sandbox, watches for unusual behavior, checks for stability and clarity, and approves it. Once the app has been approved, its released onto the app store, at which time anyone can download it. The approval process, as well as the high monetary fee, act as a way to ensure only safe and high-quality apps are available for that platform. In these times of platforms, typically no apps may be installed from other sources. On iOS, initially this was the main method of security: if the app passed the inspection, it was acknowledged as safe and virtually unmonitored unless someone noticed something unusual and reported it. However, in recent years, after certain incidents, apps still must request permission from the user to perform certain tasks.

Androids distribution platform takes a different approach, and at its core is also Androids core security model: The Permission System. Android Apps declare when they are packaged what capabilities they will use, and the user reviews them at install time. If the user approves the app, it may use the requested capabilities whenever it wants: little restrictions are placed otherwise. With this barrier in mind, the Google Play Store (formally the Android Market), or GPStore, opts for an alternate model to iOS, where the developer pays a smaller fee, and apps go through no formal approval process. After an apps submission, its immediately released into the wild for users to download and run. The assumption Android uses is that the metadata the GPStore provides: App name, Developer Name, Description, Reviews and Ratings, are enough for the user to determine if the app should be trusted with the permission set its given. In fact, Android even allows the device to accept apps from 3rd party sources, a practice known as sideloading, although its disabled by default. This has spawned a large number of 3rd party app sources, all of which rely on the Permission system for user protection.

This core concept behind Android security - Permissions as a static contract of capabilities, is the classic example of a rule-based security model. ¡expand on this¿. However, in this paper we propose an alternate means of conceptualizing security, which focuses on the interrelationship between the users perception of the app, and the apps actual behavior. We call this the user-app agreement, and will elaborate on it more shortly.

The tighter security model of Mobile OSs has a notable effect on mobile malware. With tight control in sandboxing, and app distribution, the usual viruses, trojans, and other exploits are more difficult to employ. The main vectors are either OS-level exploits, sneaking past the app review process, or through sideloading of apps. When looking at the two main mobile OSs, a stark contrast is shown. Over ¡201 or so¿ known OS-level exploits exist for iOS, where only ¡5 or so¿ have been publicly released for Android. On the contrary, no side-loading is possible for iOS, and there have been very few instances of malware sneaking past Apples App Store review process, although it has happened ¡http://www.wired.com/gadgetlab/2012/07/first-ios-malware-found/¿. Androids malware situation is very much a product of the sideloading and lack of review process found in GPStore. Of all the mobile malware found for iOS and Android, over ¡some percent¿ used no system-exploits at all, ¡some percent¿ on the main app distribution platform.

Malware, as defined by the US-DHS, is Short for malicious software. Programming (code, scripts, active content, and other software) designed to disrupt or deny operation, gather information that leads to loss of privacy or exploitation, gain unauthorized access to system resources, and other abusive behavior ¡$http://ics-cert.us-cert.gov/pdf/undirected_{a}ttack0905.pdf$¿. On mobile devices, the dominant goal of malware is to gather information that leads to loss of privacy, found in over ¡some large percent¿ of known mobile malware.

This trend, of malware that possesses no system exploits, but gathers information that leads to loss of privacy, is one that Androids Permission-based security model is ill-equipped to handle. Androids permission system relies on the user to determine at install-time if a list of capabilities should be entrusted with the given app. The user is not given a say in how or when the capabilities may be used, nor the ability to reject specific capabilities. At the same time, the mechanisms that keep mobile OSs safe are forcing malware writers to use more subtle techniques, often times without exploits. This all works against the user.

In this paper, we attempt to address this key issue through various means. We first introduce several novel concepts for analyzing apps and malware on Android. We then analyze the state of Android apps with the most comprehensive android app database available. Finally, we propose several novel improvements to the Android security architecture, called ¡PROJECT¿, aimed at building off of our conceptual work.

Chapter 3

Conclusion and Future Work

References

[1] Apple App Store. `http://itunes.apple.com/`.

[2] E. McCallister. *Guide to protecting the confidentiality of personally identifiable information*. DIANE Publishing, 2010.