Markets and Malware: Detecting Malicious and Privacy-stealing Apps in Smartphone Marketplaces

Shaun Brandt

CS 591

11/26/2012

Introduction

- Smartphones have quickly managed to become the most dominant portion of the wireless device market
  - And the computing market in general, when combined with tablet devices
- Hundreds of millions of devices are already active, and tens of millions more are activated every year
- 'Apps' are one of the main attractions to smartphone buyers, and they have collectively downloaded billions!
- Apps = executable code = potential for malware

Introduction, continued

- There are many smartphone platforms:
  - Apple: iOS
  - Google: Android
  - Microsoft: Windows Phone
  - Research in Motion: Blackberry OS
  - Others: Symbian, WebOS, MeeGo
- Apple and Google have, by far, the largest marketshare, and the two (especially the latter) are the focus of my paper.

Why target smartphones?

- Contact information
  - Phone numbers, e-mail addresses, even physical addresses in some cases
- Access to services that can make attackers money!
  - Phone calls to international numbers
  - SMS messages to premium numbers and short codes
  - The user won't even notice until they get their bill
- Device serial numbers (IMEIs)

Existing security infrastructure

(NOTE: my paper covers the topic much more extensively)

- Two different models
- Apple's model for iOS devices:
  - Walled garden (their marketplace or none)
  - Only licensed developers can submit apps
  - Apps are checked before publication
  - Apps are digitally signed
  - Apps' ability to read and write files are tightly controlled

Existing security infrastructure

- Google's model for Android devices:
  - Open market (many third-party marketplaces)
  - Apps are sanity checked, but not exhaustively checked for malicious behavior
  - Only apps that declare intent to use dangerous features can call the functions that provide them
  - Account segregation (different user) for each app
  - Java VM / sandbox
  - The user has primary responsibility – a list of required dangerous permissions is shown at install time, and the user must accept **all** of them

Active research

- Detecting malware and preventing privacy leaks are both areas of active research
- Most research concentrates on Android (since it's open source)
- Research is concentrated in a few areas:
  - Detection as part of the app submission process
  - Detection at install time
  - Detection at run time
  - Enhancement of existing security infrastructure

Detection at app submission time

- AppInspector
  - From the paper 'Vision: Automated Security Validation of Mobile Apps at App Markets'
  - Developed by Peter Gilbert et al.
  - Extension of previous work done on a tool called TaintDroid
  - Uses a technique called dynamic taint analysis to track the flow of private information through an app, from the 'source' (address book, GPS location), to a 'sink' (the external network)

AppInspector

- Dynamic taint analysis
  - When sensitive data is accessed, that data has a tag attached to it
  - As the data is moved along through the code, the tag persists
  - If the tagged data leaves the device through a sink, then this is considered a potential private data leak
  - The code is inspected symbolically and through actual execution (in a virtual machine)

AppInspector, continued

- AppInspector is an off-line version of the TaintDroid tool, designed to be run by the vendor as a step of the app submission process
  - TaintDroid was made to run in real-time, tracking private data flows on the device itself

PiOS

- From the paper 'PiOS: Detecting Privacy Leaks in iOS Applications' by Manuel Egele et al.
- PiOS is a tool that behaves in a similar way to AppInspector, but for iOS devices
  - Analysis is only done statically
  - Data is traced from source to sink
- However, iOS apps are usually written in Objective C, which makes tracking data difficult...

PiOS, continued

- Analysis requires that the tool can create a control flow graph (CFG) to trace through execution, but:
  - Objective C uses a message-passing interface instead of direct function calls or structures like vtables
  - All messages are dispatched through a single function!
  - Applications from the official marketplace are encrypted and digitally signed
- Even with these problems, the team was able to develop PiOS

PiOS, continued

- Encrypted apps were grabbed at execution time with a jailbroken phone and debugger (while they were in a decrypted state)
- Apps from Cydia (a third-party marketplace) were also used
- The disassembled apps and binary header information were used to infer class structure and create the CFG
- Static analysis could then be done, linking sources to sinks

PiOS, continued

- Tested on 1400 apps – most were found to respect private data
- Even most apps from the Cydia marketplace (which Apple has no control over) were well-behaved!

Other papers

- Malicious app detection at install time
  - Kirin: an infrastructure to describe 'dangerous' combinations of permissions on Android phones, and block installation
    - Example: an app that wants to check phone state, record audio and connect to the Internet may be a phone call recording/monitoring app
    - The paper describes a security policy, and through Kirin, provides a method of enforcing it

Other papers, continued

- Offloading malware detection to the cloud (yes, the cloud...)
- Paranoid Android
  - The idea: log device behavior, transmit over the network, and replay all actions on a VM clone of the device living on a server somewhere
  - Only monitors activities that cause non-determinism, to save space
  - Analysis can then be done using methods that are too slow to do in real time

Other papers, continued

- You may be asking 'but isn't transmitting all of the phone's activity to a remote server a breach of confidentiality?'
  - The target market is corporate / military environments
  - 'Confidentiality' and 'integrity' are more important to the company or agency that owns the device (and their data), not the user

Other papers, continued

- Enhancing Android's existing permissions infrastructure
  - TISSA – a tool developed by Yajin Zhou et al.
  - Adds new finer-grained permissions
  - More importantly, allows individual permissions to be granted or denied at runtime (as opposed to the all-or-nothing, install-time only option that Android currently offers)
  - Can be configured to return bogus data in place of real data

Other papers, continued

- Detecting modified apps in third-party marketplaces
  - Popular apps are frequently repackaged and put on unofficial marketplaces
  - They may contain malware, or modifications to provide revenue to the person who did the repackaging
- DroidMOSS is a tool that uses fuzzy hashing to fingerprint apps. Apps that are 'mostly' the same can be detected in this manner

Other papers, continued

- Tested on 6 third-party marketplaces: between 5 and 13 percent of apps were repackaged
  - Some redirected ad affiliate credentials to give revenue to the repackager
  - Some add ads to apps that previously didn't have them
  - A few added malware packages (mainly to send messages to premium SMS numbers)