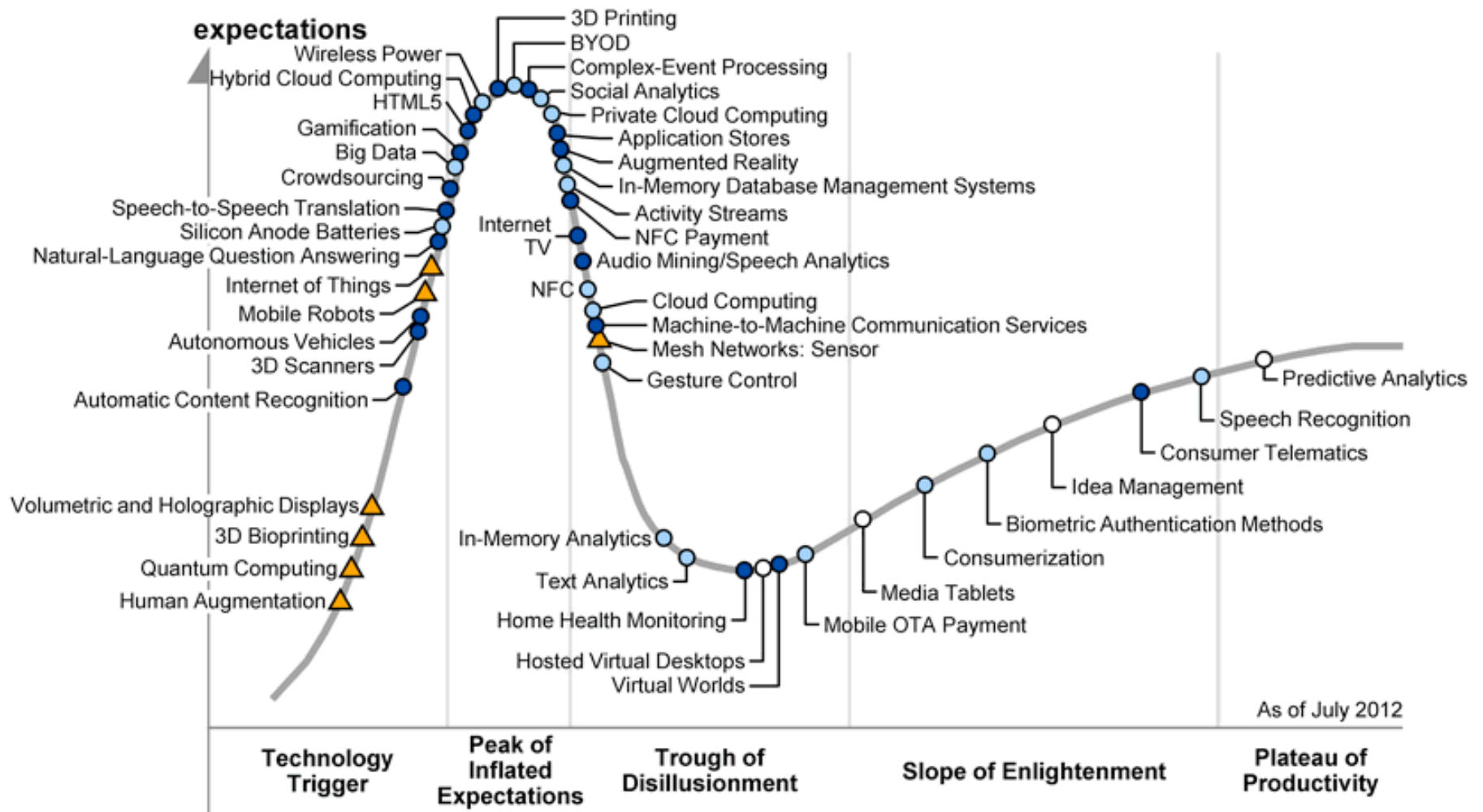


Programming on Android: Best Practices for Security & Reliability

*Angelos Stavrou, Ryan Johnson,
Rahul Murmuria, Mohamed Elsabagh*
George Mason University & Kryptowire

Why do I care? Maturity of Technologies

(source Gartner)



Plateau will be reached in:

- less than 2 years
- 2 to 5 years
- 5 to 10 years
- ▲ more than 10 years
- ⊗ obsolete before plateau

CIO Business Priorities

Top 10 CIO Business and Technology Priorities in 2012

Top 10 Business Priorities	Ranking	Top 10 Technology Priorities	Ranking
Increasing enterprise growth	1	Analytics and business intelligence	1
Attracting and retaining new customers	2	Mobile technologies	2
Reducing enterprise costs	3	Cloud computing (SaaS, IaaS, PaaS)	3
Creating new products and services (innovation)	4	Collaboration technologies (workflow)	4
Delivering operational results	5	Legacy modernization	5
Improving efficiency	6	IT management	6
Improving profitability (margins)	7	CRM	7
Attracting and retaining the workforce	8	ERP applications	8
Improving marketing and sales effectiveness	9	Security	9
Expanding into new markets and geographies	10	Virtualization	10

Source: Gartner Executive Programs (January 2012)

Traditional Device Concerns

Security Requirements

- Confidentiality
- Integrity
- Authenticity
- Availability
- Accountability
- Non Repudiation

Attacks

- Physical Attacks
- Application Attacks
- Telecommunications
- Infrastructure – App Store
- Supply Chain

Threats

- Eavesdropping
- Integrity
- Data Exfiltration
- Denial of Service
- Masquerading

Vulnerabilities

- Hardware
- Software
- OS
- Communication Protocols

Mobile & Smart Devices Risks

Assumptions

- Networked or Stand-alone Apps
- Public or Private Network
- Tethered or Untethered Synchronization
- Standard or Proprietary Protocols
- Ad Hoc Network or Base Station
- Configuration Management
- Classification Level of Data
- Connect back to DoD or IC Networks
- Interoperable with DoD or IC
- Federated or Enterprise Model

Threats

- Capture or loss of device
- Poor configuration management, administrative backdoor, automatic updates
- Eavesdropping wireless communications
- Infection from compromised PC during data synchronization
- Peer smart-phone attack or infection (via Bluetooth or WiFi)
- Attacks on Telecom Network - Base Station
- Malware - viruses, trojans, or worms spread the same way as PCs
- Location tracking
- Proper Device disposal – forensic tools
- DoS – Spam

Risks in Mobile Security Supply Chain

Devices



MDM/Middleware Providers

Enterprise Security



Enterprise Security



Enterprise Security



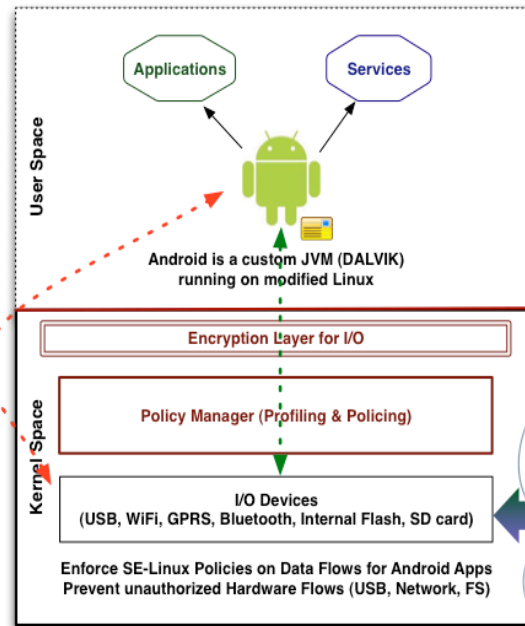
Enterprise Security



Enterprise Security



Multi-Level Mobile Phone Security Architecture



Secure Verify Test Deploy

Android Security & Reliability

- State of the Art: Anti-virus, Code Analysis
- Static Permissions checking
- Functional Static & Dynamic Analysis
- Resource Restriction
- Power Metering

Android Security & Reliability

What we will cover in this Tutorial:

Overview of available Mobile Analysis:

- ❖ Permissions checking
- ❖ Functional Static & Dynamic Analysis
- ❖ Power Metering & Resource Restriction
- ❖ Android vs iOS similarities & differences

Android Security & Reliability

What we will cover in this Tutorial:

Code Analysis Examples & Case Studies

- ❖ User Interface Testing
- ❖ Code Examples & Case Studies
- ❖ Multi-threading, JNI, Best Practices
- ❖ Questions & Discussion

Android Security & Reliability

Overview of Existing
Security & Reliability
Automated Testing Tools

Security & Reliability Challenges

Why Mobile Testing Is Difficult (Gartner Study 2013)

1. Diversity in platforms, OSs and devices

The most popular mobile OS — the Android — is the most fragmented, with five major versions corresponding to nine API sets that had over 1% market share in April 2012.

This complexity is compounded by hundreds of different device designs, screen sizes and form-factor variations, such as tablets and handsets.

Security & Reliability Challenges

Why Mobile Testing Is Difficult (Gartner Study 2013)

2. Automation challenges

Sophisticated user experiences involve touch, gestures, GPS location, audio, sensors (such as accelerometers) and physical actions (such as touching the handset to Near Field Communication [NFC] readers).

Such interactions can't be fully scripted or simulated, and may involve manual testing on real devices.

Security & Reliability Challenges

Why Mobile Testing Is Difficult (Gartner Study 2013)

3. Application complexity and sophistication:

Mobile devices and applications are becoming more sophisticated, using techniques such as context, 3D graphics and gaming. Greater sophistication implies more complex testing

Security & Reliability Challenges

Why Mobile Testing Is Difficult (Gartner Study 2013)

3. New OS versions often break applications:

Developers have no control over when new OS versions will appear, and when or whether users will upgrade. Thus, it's common for new OS releases to break existing native applications.

4. Bug-fix latency: Some app stores have a submission latency of one to two weeks, meaning that bugs cannot be corrected rapidly, making application quality more important.

Security & Reliability Challenges

Mobile Testing Tools and Services

- ❖ Automated Tools for Testing
- ❖ Instrumentation, Monitoring and analytics
- ❖ Best Code Practices based on a Feedback Loop
- ❖ Expand to Multiple Devices through testing on multiple Devices and Emulators
- ❖ Common Failure: UI, POWER, OS Version

Security: Current Mobile Anti-Virus

Commercial AV vendors are not ready for mobile:

- Drain battery quickly
 - Unacceptable in tactical setting
 - Cannot be regulated
- Worse detection capabilities compared to their Desktop Counterparts
- Detection not guaranteed:
 - Cannot Identify non-preclassified threats (see next slide)
 - Some of them “call-back” home and require constant updates

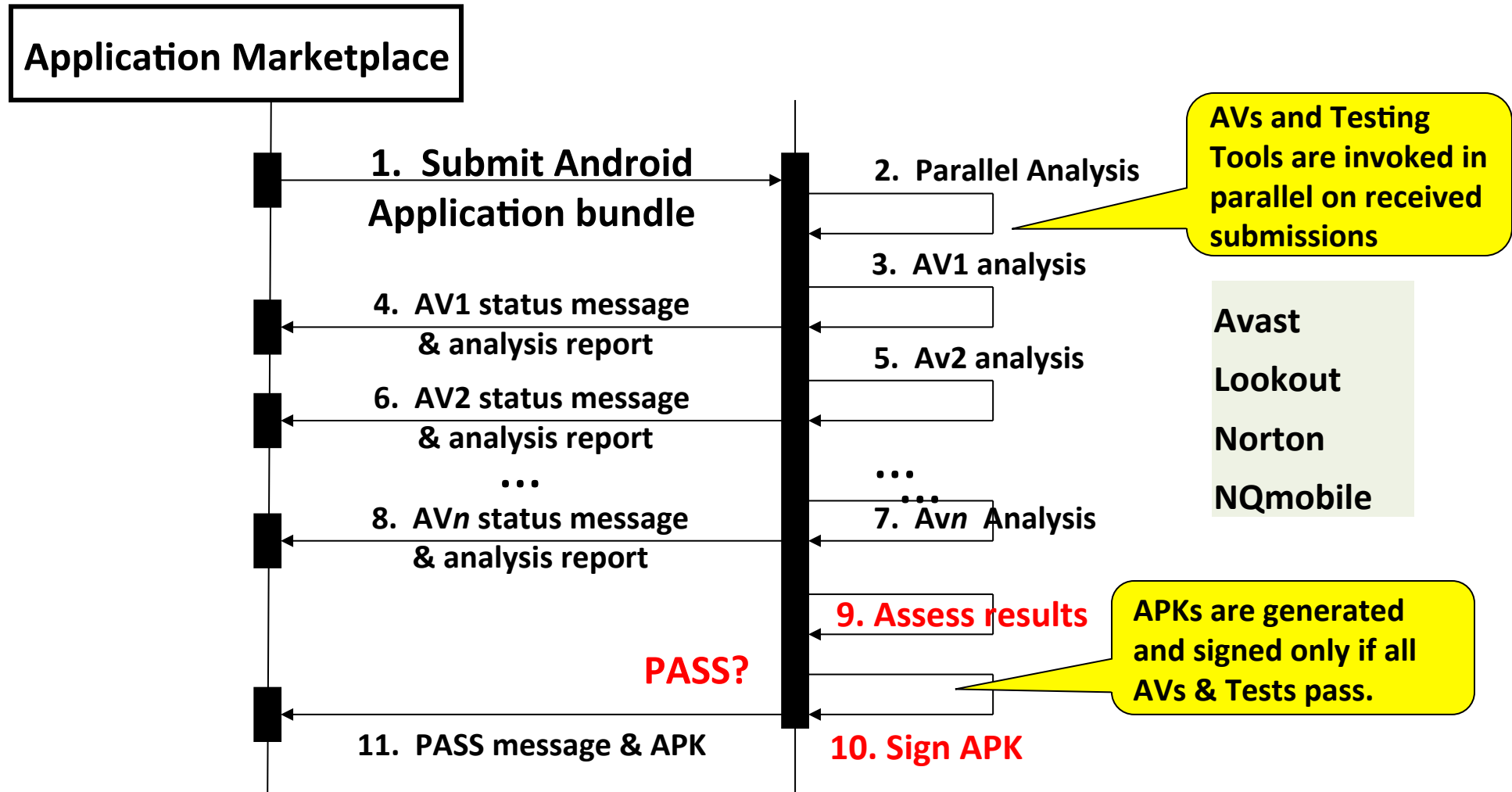
Security: Mobile Anti-Virus Testing

- Obtain a test set of malware from the wild
 - Include both recent (zero-day) and older
 - We used three (3) (28 zero-day, 95+144 >4 weeks) malware sets
- Download popular AVs from the Android Market
- Create a “Cloud” system that allows for large-scale app testing (malicious and benign)
- Measure metrics:
 - Overall Efficiency: % of malware detected
 - Detection Latency: Efficiency versus timeliness of threat

Android Mobile Anti-Virus Testing

<u>AV Name</u>	<u>2-Weeks</u>	<u>>4 Weeks Detection</u>	<u>"Zero" Day</u>
Avast	50.00%	94.38%	14/28
Lookout	57.14%	93.26%	16/28
Norton	53.57%	90.64%	15/28
NQmobile	25.00%	88.76%	7/28
Comodo	N/A	87.64%	0
BitDefender	N/A	87.27%	0
AVG	N/A	86.52%	0
TrustGo	N/A	86.14%	0
Kaspersky	N/A	85.02%	0
Zoner	N/A	81.65%	0
GData	N/A	77.53%	0
DrWeb	N/A	72.66%	0
WebRoot	N/A	22.10%	0
ALYac	N/A	13.48%	0
MobileBot	N/A	0.00%	0

Testing Portal Solution: Pre-Production Scanning



- Fast, Scalable, no burden to device or end-user
- Better Coverage through Multiple AV vendors

Shortcomings of Mobile Anti-Virus

- Do not detect 0-day malware
- Do not detect Polymorphic Android Malware
- Do not detect Embedded malware
 - Media
 - PDF
 - Threats that attack non-mobile devices
- Do not support any behavioral analysis or heuristics
 - Cannot operate correctly without network connection

Android Analysis Tools: Why?

- ❖ Developers maybe well-intended but...
 - ❖ They do not necessarily understand the mission or the security/policy requirements
 - ❖ They make mistakes
 - ❖ They use third-party libraries and code
- ❖ The Android permission model is **neither sound nor complete**
 - ❖ Intentions, Reflection, JNI, Webkit, others...
 - ❖ Android permissions are enforced inside Dalvik not everywhere in the device
- ❖ Zero-day and polymorphic threats remain undetected with current commercial tools

Badly Designed & Malicious Apps exist...

Analyzed >600,000 Applications from the Google Android Market

- **Thousands** with incorrect/permissive manifest
- **Hundreds** with excessive functionality that can be constituted as malicious
- **Hundreds** of Trojans (i.e. take over existing, legitimate applications)
 - Who will download these apps?
 - People who use SEARCH to find apps
 - Virtually everyone...
 - Two infection vectors:
 - Regular Web Search
 - Search inside the Mobile App Market

Badly Designed Apps - Defined

- Mobile Software Developers make mistakes...
 - Collect and/or Store **sensitive (PII)** information without notifying the End- User
 - Transmit PII information to their website or third parties
 - Enable other programs to get access to PII data
- Good Intentions but **undisclosed** to the End-User
 - The application makes use of resources not disclosed to the user: Camera, GPS Location, Microphone, Read of PII (contacts, phone #s, IMEI, etc.)
 - Perform Functionality without explicit End-User permission

Malicious / Rogue Mobile Apps - Defined

- Rogue mobile apps can be best defined as follows:
 - Created by non-authorized individuals or entities
 - Seek to confuse consumer to believe it is published from an authorized source – similar name, use of logo, or similar publisher
 - Similar to other applications but its objectives are to compromise other apps on the device
- Malware mobile apps have different objectives:
 - Similar to desktop malware or viruses – device disabling
 - Data syphon – attempt to steal device data and PII information to third parties
 - Man in the middle – serve as a proxy - behavior to end user is seamless, credentials are taken

Example of Malicious code

```
public class WebViewExample1 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        WebView wv = null;

        String toScreen = "";
        try {

            wv = new WebView(this.getApplicationContext());
            Uri uri = Uri.parse("http://www.gmu.edu");
            Intent intent = new Intent(Intent.ACTION_VIEW, uri);

            startActivity(intent);
            toScreen = "The application just loaded a webpage without the
android.permission.INTERNET permission!";
        }
        catch (Exception e) {
            toScreen = e.toString() + "\n" + e.getCause();
        }
        TextView tv = new TextView(this);
        tv.setText(toScreen);
        setContentView(tv);
    }
}
```

Rogue Mobile Apps – Example

Mobile Banking | Best Free Download Android games, applications market, android phones

http://topandroid.net/mobile-banking-2

SIRA News | SIRA Infinitive Portal (IP) Google Apple AOL Apple - Joint Venture Google Maps YouTube Wikipedia News (388) Popular

US Bank
Valley Credit Union
Wachovia
Washington Mutual
Wells Fargo
West Suburban Banks
Vancity
Servus Credit Union
Meridian Credit Union
First Calgary Credit Union

Do you bank with more than one banking institution? What about your credit cards, are they offered by banks other than the one you have an account with? Well look no further!

For you convenience, we have bundled together an application that gives you access to online banking at every major bank in the United States and Canada!

App Screenshots

Click to enlarge images

User Reviews

No bank info would load. Willing to try again once issue has been resolved
★★★★★ by **Drake** – June 17, 2011

Twitchy , buttons to small for fingers, keeps telling u to reload. POS app
★★★★★ by **Warren** – July 28, 2011

Useless. I got this thinking it was gonna be more than a bank URL storage

Functional Static and Dynamic Analysis: Detect Zero-Day Threats

- **Functional Static Analyzer: Permissions Analysis**
 - Android Specific Analysis includes analysis of the Application Security Manifest
 - Functional Analysis Verify if the requested permissions are warranted by the submitted code
- **Dynamic Code Analyzer: Run-Time/Behavioral Analysis**
 - Identifies Access to Critical Resources & Behavior (Network, Intents, Reflection, File Access, etc.)
 - Provides context for the behavior by resolving Data Structures at Run-time in a Cloud
 - Exercise all available data and control flow paths through an application while keeping state
 - Analysis on binaries includes Java Code, third-party Java and Native libraries
 - Computes Code Coverage (In our experiments ~90% within 5mins 99.99% within 60 minutes depending on LoC)
 - Power Consumption Analysis

Android Application Testing Framework

Android-specific analysis includes analysis of the Application Security Manifest (not supported by third-party vendors)

- **Tailored to the Android Permission Model**
- Verify if the requested permissions **are warranted** by the submitted code
- Curtails excessive permissions and enforces a tighter security model

Future: Modifications to the Android engine to enable dynamic policies

- Control the underlying Dalvik engine to report **absence/depletion of resources** instead of lack of permissions
- Regulate access to critical/restricted resources

Example of Dalvik Application Permission Query & Verification

Static Analysis of ammocore-release-1.5.14-0-g4deo308

Failed: Application Appears to be Using a Different Functionality Than What is Requested

Requested Permissions

- android.permission.READ_LOGS
- android.permission.RAISED_THREAD_PRIORITY
- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE
- android.permission.CHANGE_NETWORK_STATE
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_WIFI_STATE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.CHANGE_WIFI_MULTICAST_STATE
- android.permission.ACCESS_FINE_LOCATION

Missing Permissions

- android.permission.READ_CONTACTS
- android.permission.BROADCAST_STICKY
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.WAKE_LOCK
- android.permission.BROADCAST_PACKAGE_REMOVED


Required Permissions based on Functionality

- android.permission.INTERNET
- android.permission.ACCESS_NETWORK_STATE
- android.permission.READ_PHONE_STATE
- android.permission.ACCESS_WIFI_STATE
- android.permission.CHANGE_WIFI_STATE
- android.permission.WRITE_EXTERNAL_STORAGE
- android.permission.CHANGE_WIFI_MULTICAST_STATE
- android.permission.ACCESS_FINE_LOCATION
- android.permission.READ_CONTACTS
- android.permission.BROADCAST_STICKY
- android.permission.RECEIVE_BOOT_COMPLETED
- android.permission.WAKE_LOCK
- android.permission.BROADCAST_PACKAGE_REMOVED

Not Required Permissions

- android.permission.READ_LOGS
- android.permission.RAISED_THREAD_PRIORITY
- android.permission.CHANGE_NETWORK_STATE

kryptowire Sample of Dynamic Analysis

Version 1.3.0

Analysis of activator.apk

Found instances of:



Show all details

Telephony events:

- `android.telephony.SmsManager.sendMessage(java.lang.String, java.lang.String, java.lang.String, android.app.PendingIntent, android.app.PendingIntent)`

- File name: `ActivatorActivity.smali` line 61
- Parameters:
 - Destination number: 770656
 - Service center number: 0x0
 - Text message body: DEF1773

Line in the Disassembled Code

Phone Number Attempted

Pointer to the SMS Message

Dynamic analysis Provides the context (URLs, Intents, I/O) at Runtime

Android Market Application Meta-data Collection

- Developed & Maintained an Application DB from official US Markets
 - Create a time-line for new and old Applications
 - Query the Market for new Versions of Applications
 - **Provide a mechanism to Efficiently Store Metadata related to each Application**
- Host the Database in a Secure environment
 - Access to the DB is going to be restricted
 - Capability to Store history of Queries
 - Secure Way to exchange information between Analysts

Sample Example of Collected Application Meta-Data

Title: Cashew
App type: APPLICATION
Id number: 3781153483681195048
Category: Productivity
Price:
Price currency:
Creator: Kawet
Creator Id: Kawet
Package name: com.madebykawet.cashew
Rating: 0.0
Ratings count: 0
Screenshots count: 2
Version: 0.9beta
Version code: 2
Serialized size: 1851
Contact email: madebykawet@gmail.com
Contact phone:
Contact website: http://madebykawet.com

Description: *** You need to create an account on <http://cashew.madebykawet.com> to get your login & pass credentials and test your apps! ***

Cashew is an iPhone & Android app creation platform that requires no technical knowledge at all.

It provides an all-in-one solution to create and update (design + content) an app very easily and quickly using a CMS. Changes can even be done once the app is available on the App Store or the Android Market!

This app is the Android test platform: this is where you can see, instantly, the changes you do on the web platform exactly as the final result will be. When you are happy with your app, we will send it on the Android Market. Nothing won't change: you will still be able to use the CMS to update and customize your app. Apps generated are fully natives. Cache is supported (you can access to your app even if you have no Internet connection) and the design of each app can be entirely customized.

The features supported by each app are:

- unlimited views (cells inside other cells)
- unlimited tabs
- slideshow for pictures & images
- advanced support of RSS feeds
- custom videos
- videos through YouTube, Vimeo and Brightcove
- custom geolocation (unlimited number of POI on a map)
- mail & sms in app
- web browser in app
- push notifications (coming soon on Android)
- music streaming/mp3 player (coming soon)
- augmented reality (coming soon)
- ...

Keywords: portfolio, kawet, cashew, madebykawet

Promo text: Cashew is a mobile app creation platform.

Promotional video:

Recent changes:

Install size: 2328927

Permission Id count: 4

List of permissions: android.permission.INTERNET
android.permission.WRITE_EXTERNAL_STORAGE
android.permission.ACCESS_FINE_LOCATION
android.permission.ACCESS_COARSE_LOCATION

Application Testing for iOS

Challenges:

- Application & Device Encryption
- Lack of Source Code (Binaries/IPA)
- Communications might be Encrypted
 - SSL
- Mixed language package
 - Objective C
 - HTML 5
 - Meta Data
 - Resource Files
 - Sqlite3 Database
- Analysis can have many requirements
 - Data integrity
 - Compliance
 - Other (?)

Analysis of mobile-specific Capabilities

What we can offer now (Summary)

- Capability to bypass device and data encryption
- Automation of extraction process from an binary file (both iOS and Android)
- Static Analysis that covers both Code and Data
- Dynamic Analysis that complements Static
- Flexible Reporting based on the requirements from a customer
- Implement & Integrate customer Requirements
 - Customer usually needs to be educated about what can be reported

iOS-specific Testing Capabilities Analysis

Package Analysis

Capability to bypass iOS Device and Data Security

- iOS package encryption can be bypassed
- Requires a jailbroken device
- Process is quick (seconds)

Automation of extraction process from an IPA file

- Full Package is available including Code & Data
- Analyst gains access to Database & Metadata

iOS-specific Testing Capabilities Analysis

Static Binary Analysis

Objective C Code

- Extraction of Classes, Methods, Types
- Code Capabilities (Ads, DB access, UI elements)

HTML 5

- Images, code, styles

Meta-Data

- File Locations
- Database Schemas
- Others

iOS-specific Testing Capabilities Analysis

Dynamic Binary Analysis

System-wide observations

- Instrument the underlying OS and Libc libraries
- Collect information about what is being invoked on both low-level and high level
- System Instrumentation for
 - Network I/O
 - Record ALL Communications (SSL)
 - File I/O
 - Database Transactions
 - Encryption including keys and password
 - Others

iOS-specific Testing Capabilities Analysis

Dynamic Binary Analysis

Objective C Code

- Execute code and observe behavior
- UI Exploration to Code Exploration (Demo)
- Code Instrumentation for Code Injection
 - File I/O
 - Database Transactions
 - Network I/O
 - Record ALL Communications (SSL)

Identify Vulnerable Code

- ❖ Hardcoded Passwords and Sensitive Data
- ❖ Code Flow Problems, recover Key material
- ❖ Other

UI Testing

- Part of dynamic app analysis
- Tests how the app is responding when sequences of inputs are performed
- Tests are performed automatically without actual user input
- Exercises paths automatically and without user Intervention

Why UI testing

- Frequently apps under analysis do not exhibit all their programmed functionality (malicious/benevolent) with no or little input
 - UI needs to be “exercised” in order for “malicious” or other behavior to appear
- Simulates real user behavior and how the app responds.
- Is an automated process that can provide useful insight in the app’s behavior beyond what analysis provides
 - Code updates
 - Third Party Advertisement

Challenges

- Source code might not be available for the apps under analysis
 - No way to instrument the entire UI in order to find out how UI is laid out
- Tested platform is not open
 - Not able to execute the app outside the device
 - There is no clear mechanism to
- What about Libraries? Their source code is usually not available

Methodology

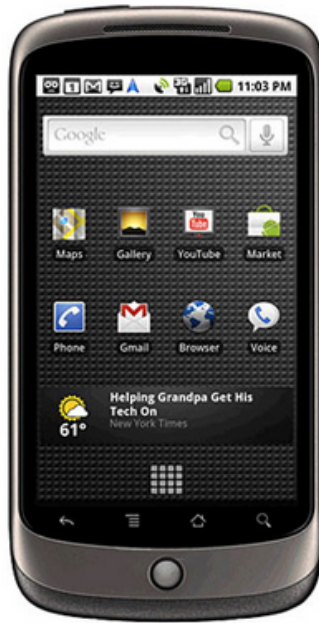
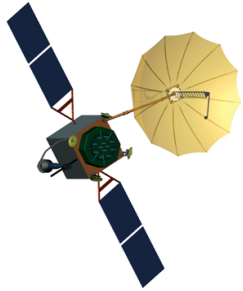
- Prepare iDevice
 - Setup, initial configuration, necessary tool installation
- Establish remote connection to the tested iDevice running
 - Used for being able to keep track what is happening on the device's screen and simulate user interaction
 - OCR is being used to monitor how apps UI changes by the inputs.
- Start the app and exercise UI
 - Perform a series of user inputs
 - Various strategies employed to increase efficiency (not just random input).
 - Record necessary data for analysis
 - Stop the app and repeat the steps many times to increase coverage

Mobile Device Testing Capabilities

What we can achieve now:

- Known Malware Detection
- Encrypted Storage of Sensitive Data
- Encrypted Transport of Sensitive Data
- Cryptographic Operations and Standard APIs
- Known Vulnerabilities and Good Coding Practices
- Necessary and Sufficient Permissions
- Dynamically Loaded Objects
- IPC Using Standard APIs with Safeguards for Components
- App Stability Following Changes and During Unexpected Events

Mobile Devices' Hardware components



- CPU
- Display
- Graphics
- Audio
- Microphone
- Wi-fi
- GPS
- Touchscreen
- Accelerometer
- Compass
- And more...

Why is power profiling important?

- Market study: Up to **75%** of total power consumption spent by applications powering 3rd party advertisements^[*]
- There is incentive for developers and users to use a proper energy accounting infrastructure to make more informed decisions about where to spend remaining device power

[*] <http://www.bbc.co.uk/news/technology-17431109>

The Need for Power Analysis

Power Consumption is important:

- Devices are heavily used (Maps, Comms, GPS)
- Tactical Environments but also first responders
- Power not readily available

Applications can **cause Power Exhaustion**

- Power Exhaustion cannot be detected through mere code analysis
- Different Devices have different power consumption
- Badly designed or Malicious Apps can deplete the battery
- Individual Apps can behave well but can drain power when operating in parallel

There is a need for Power Analysis

Challenges for Power Metering

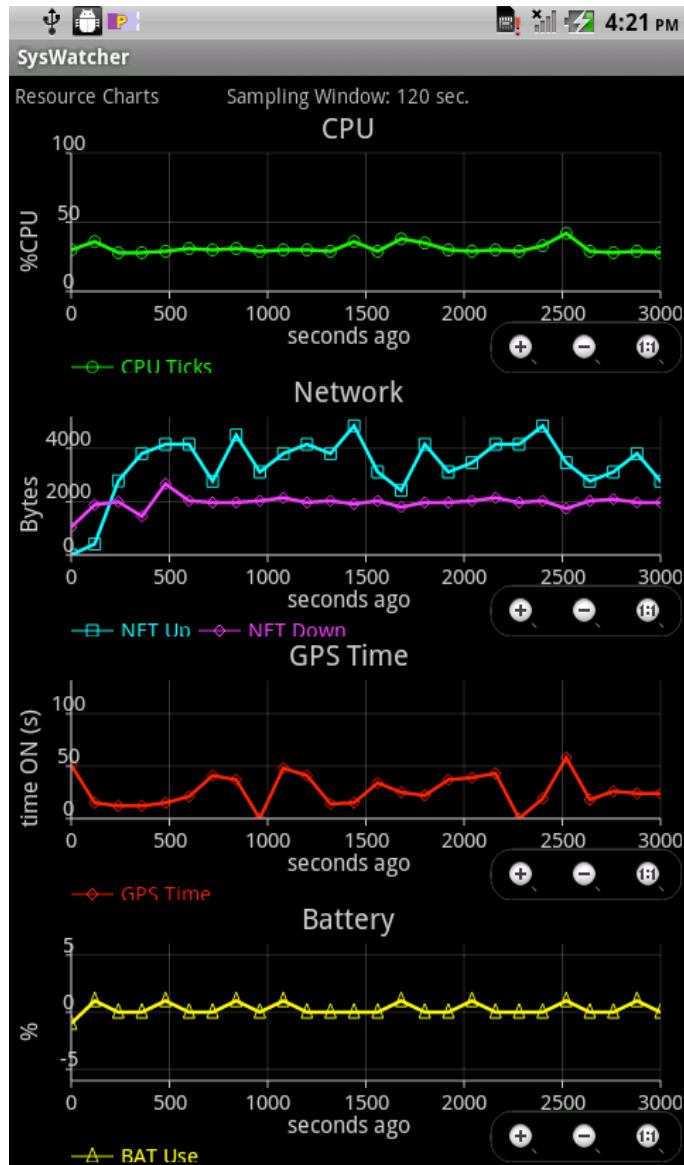
- A process can **evade energy metering**
 - Outsource the “expensive operations” to the Kernel
 - Network operations
 - Storage operations
 - Use Devices that themselves cause power drain
 - Wi-Fi, GPS, Bluetooth
 - **Display**
 - Spawn other sub-processes
- **Changing Energy Consumption**
 - **Over Time**
 - **Per User**
 - **Based on Location**

Power Metering Framework

- Design & Implement an accurate model for **accounting and policing** energy consumption
- Two-pronged approach
 - Meter the **per-process** CPU & Device utilization over time
 - Identify the **relative impact of each device** component on energy consumption
- Design an **Android kernel subsystem** to estimate energy
 - Meter energy consumption for each App/process
 - Use for characterizing application behavior
 - This behavior is **Application dependent**
 - Sometimes the behavior is also **User dependent**

Evaluation

Screenshots from our metering application



The screenshot displays the SysWatcher application settings menu. The interface is dark-themed and includes the following sections and options:

- System Resources:**
 - Show History Plot (CPU, Network and Battery)
 - Show Live Plot (CPU, Network, LCD, GPS)
- System Loggers (CPU, MEM, NET):**
 - Choose Sort By (Sort by PID, Load, Memory, Name, Network)
 - Choose Time Range (Default is boot time)
 - Show Task Manager (List processes sorted as per chosen parameter)
 - Choose Timer Frequency (Set the timer frequency of the logger)
- GPS Profiler Settings:**
 - GPS Logger (Log location to sdcard)
 - Choose Switcher Type (Turn GPS device or Location Listener on/off)
 - Choose Timer Frequency (Set the timer frequency of the logger)

Evaluation (cont)

- We were able to produce accurate real-time estimations
 - Per-process CPU utilization
 - Per-process Network utilization
 - Overall battery usage
 - Localize that information (GPS coordinates list)
- Analyze the individual measurements
 - Generate reference values for per-device energy consumption rate
 - Approximate per-process energy consumption
 - **User and Location Dependent!**

Application Policy Enforcement

Ultimately the Testing assists in POLICY Enforcement

- **Tailored to the Android Permission Model**
- Can allow **Location-Based** Policies
- Curtails excessive permissions and enforces a tighter security model

Modifications on the Android Engine to enable dynamic policies

- Control the underlying Dalvik engine to report **absence/depletion of resources** instead of lack of permissions
- Regulate access to critical/restricted resources

Conclusions

- Security and Reliability demands tools for Mobile (Android & iOS) App Testing
 - Tested on commercial apps (Android Marketplace)
 - Used in Apps Testing Portal for TransApps
- Four Tools for Reliability Analysis & Protection :
 - Cloud-based pre-release AV scanning
 - Functional Static Analysis
 - Dynamic code analysis
 - Power Usage analysis

Thank you!



Questions ?
(More to Come!)