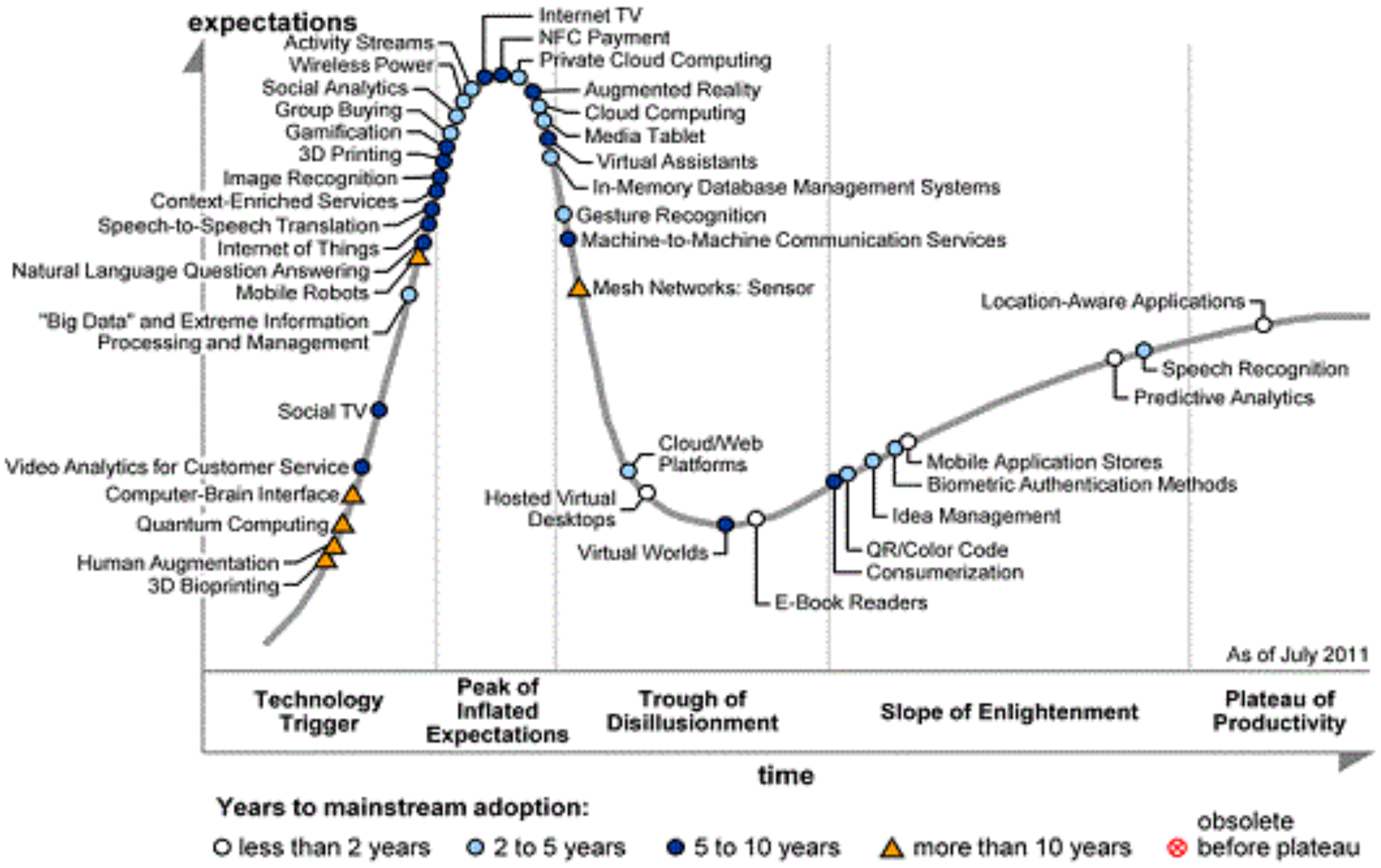
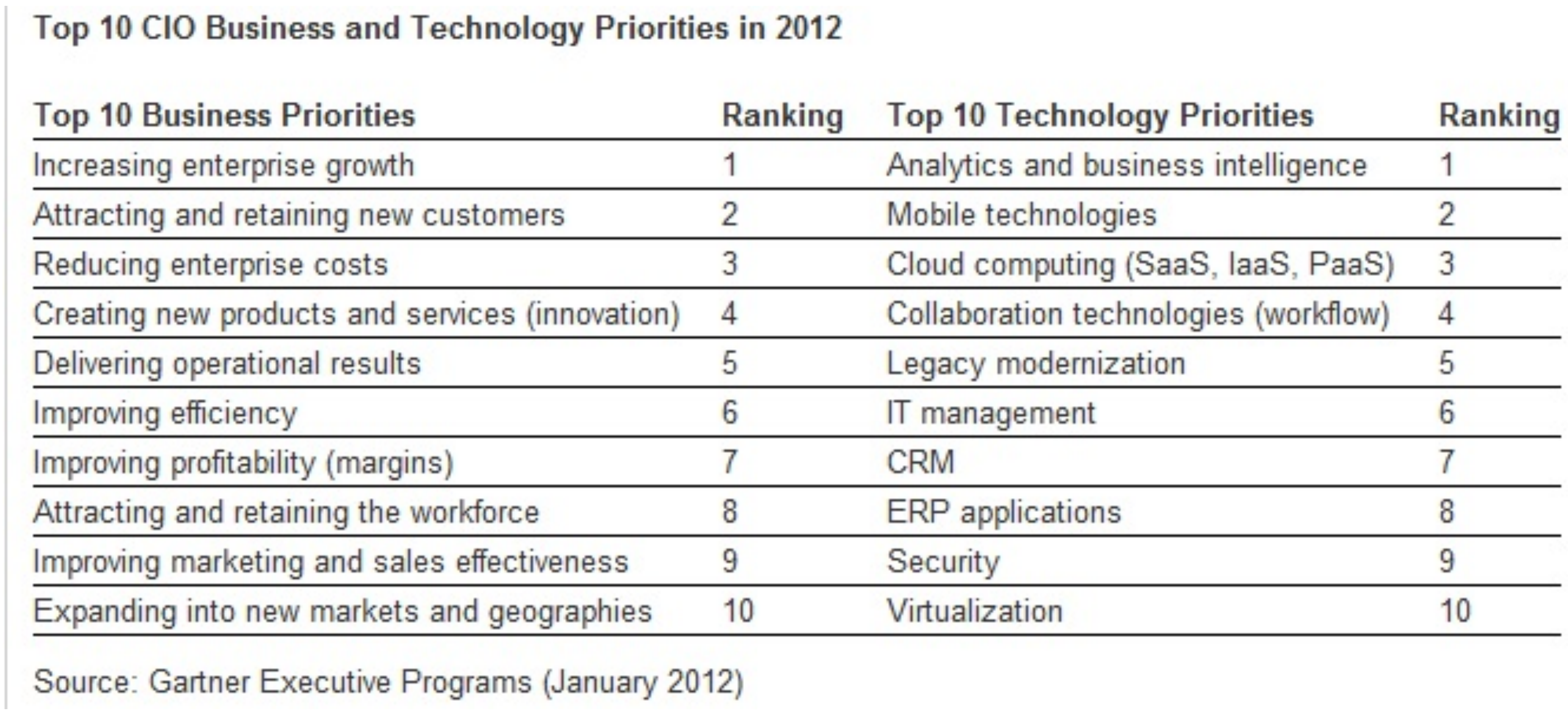
Scalable Software Testing for Android: Challenges & Opportunities

Angelos Stavrou & Jeff Voas
George Mason University & NIST

Maturity of Technologies (source Gartner)



CIO Business Priorities

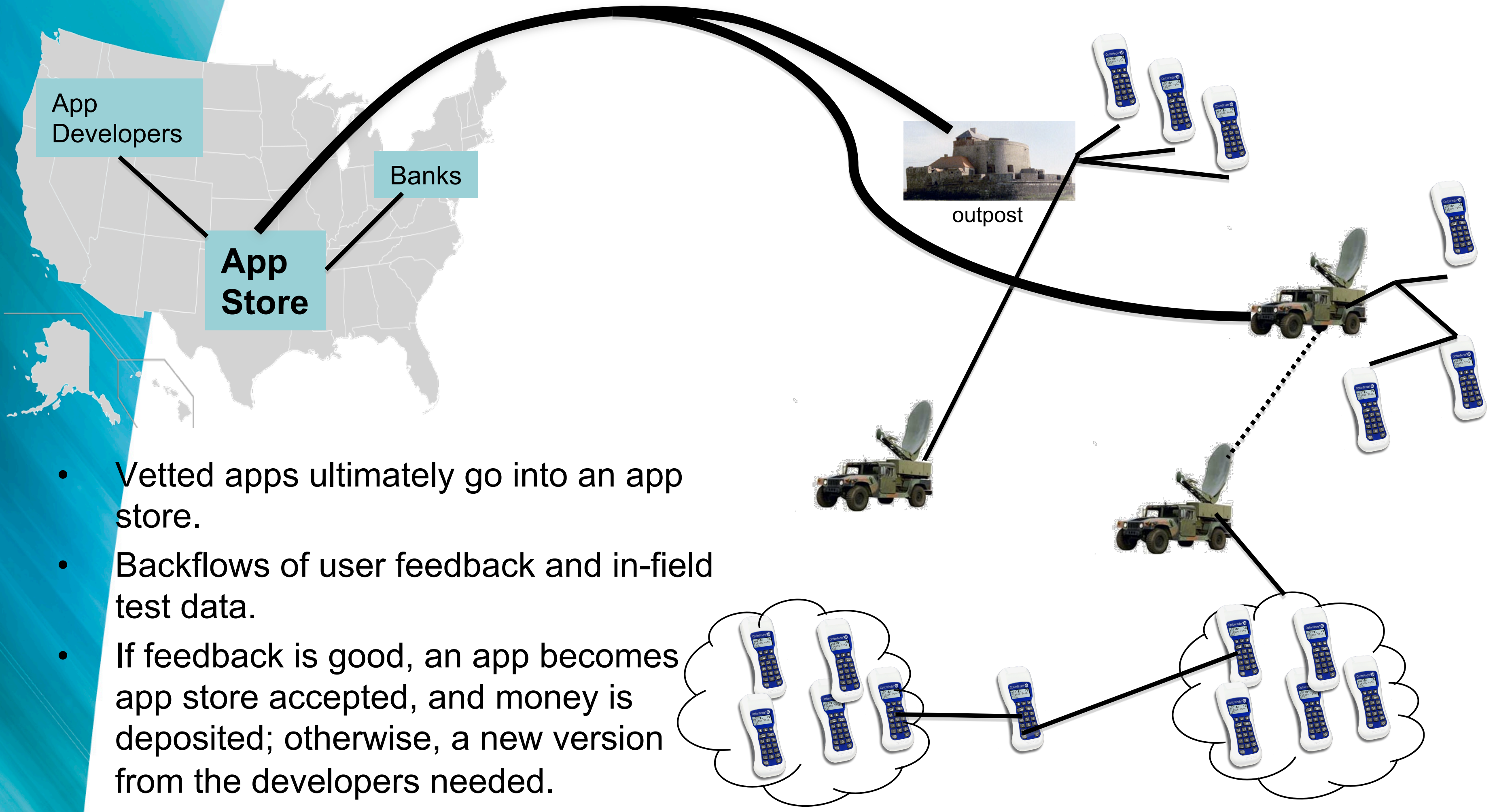


Top 10 CIO Business and Technology Priorities in 2012

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)

High-Level Project Overview



- Vetted apps ultimately go into an app store.
- Backflows of user feedback and in-field test data.
- If feedback is good, an app becomes app store accepted, and money is deposited; otherwise, a new version from the developers needed.

Why Do I Care?

Commercial Mobile Devices have access to a wide-range of functionality and ship with complex code-base:

- Fully Functional Linux system
- Proprietary device drivers with NO access to code
- Permissive policy model
- Capability to perform a wide range of operations
  - 3 (three) different types for location tracking
  - Many more through meta-data (geo-tagging)

BUT, I am secure: I am using Anti-Virus!!! Right?

Current Mobile Anti-Virus

Commercial AV vendors are not ready for mobile:

- Drain battery quickly
- Similar Results with their Desktop Counterparts
- There are no guaranteed for success in detection
  - Cannot Identify non-preclassified threats
  - CarrierIQ is an example, a “benign” and “legitimate” application
  - Some of them “call-back” home and require constant updates

But is it that bad?

The real picture: **Malicious Apps exist...**

Analyzed ~267,000 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

The real picture: **Malicious Apps exist...**

A multifaceted problem:

- 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

What about existing Analysis Tools?

- Commercial application testing tools cover regular, non-Android specific Bugs:
  - No Security Analysis of the Code Functionality
  - No Power Analysis of the Application components and code
  - No Profiling of the resource consumption of individual applications
  - Cannot Regulate/Deny the access and use of phone subsystems (Camera, Microphone, GPS..)
- Existing tools **do not cover Program Functionality**
  - We reveal the application capabilities and access

Application Testing Framework

Application Static Analysis does not cover
Program Functionality

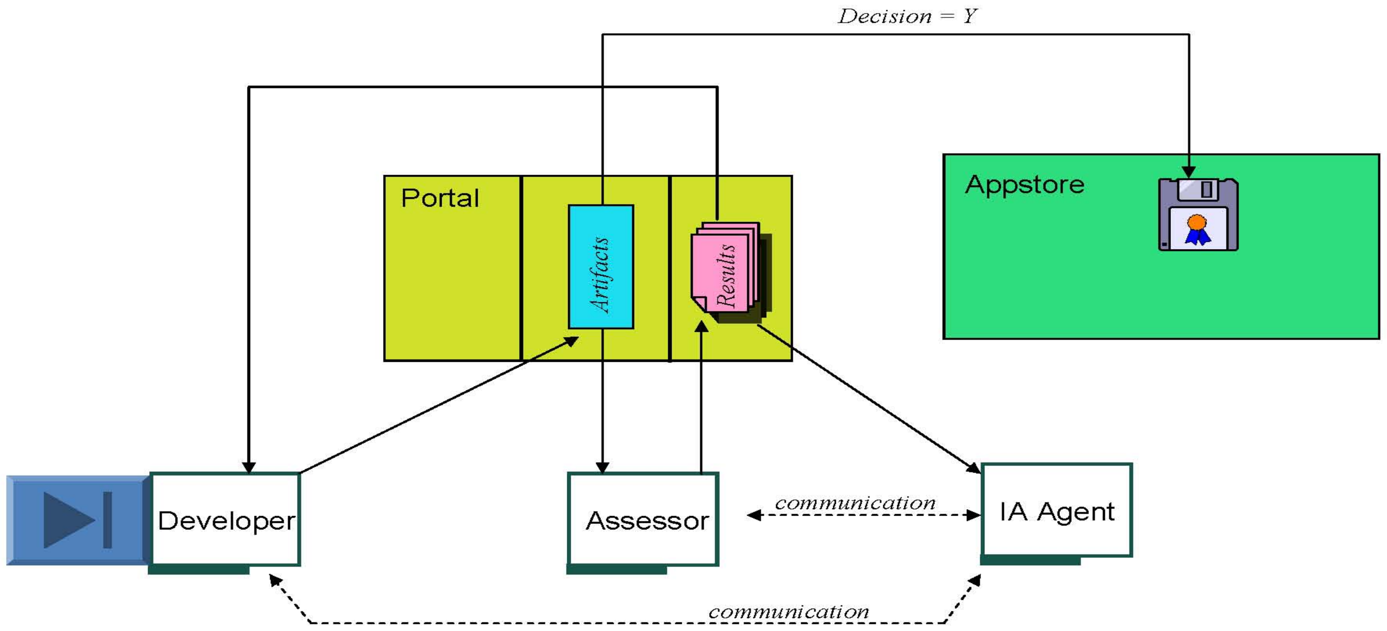
Fortify, Coverity, and other application testing tools cover regular, non-Android **specific Bugs:**

- No Security Analysis of the Code Functionality
- No Power Analysis of the Application components and code
- No **Profiling** of the resource consumption of individual applications
- **Cannot Regulate/Deny** the access and use of phone subsystems (Camera, Microphone, GPS..)

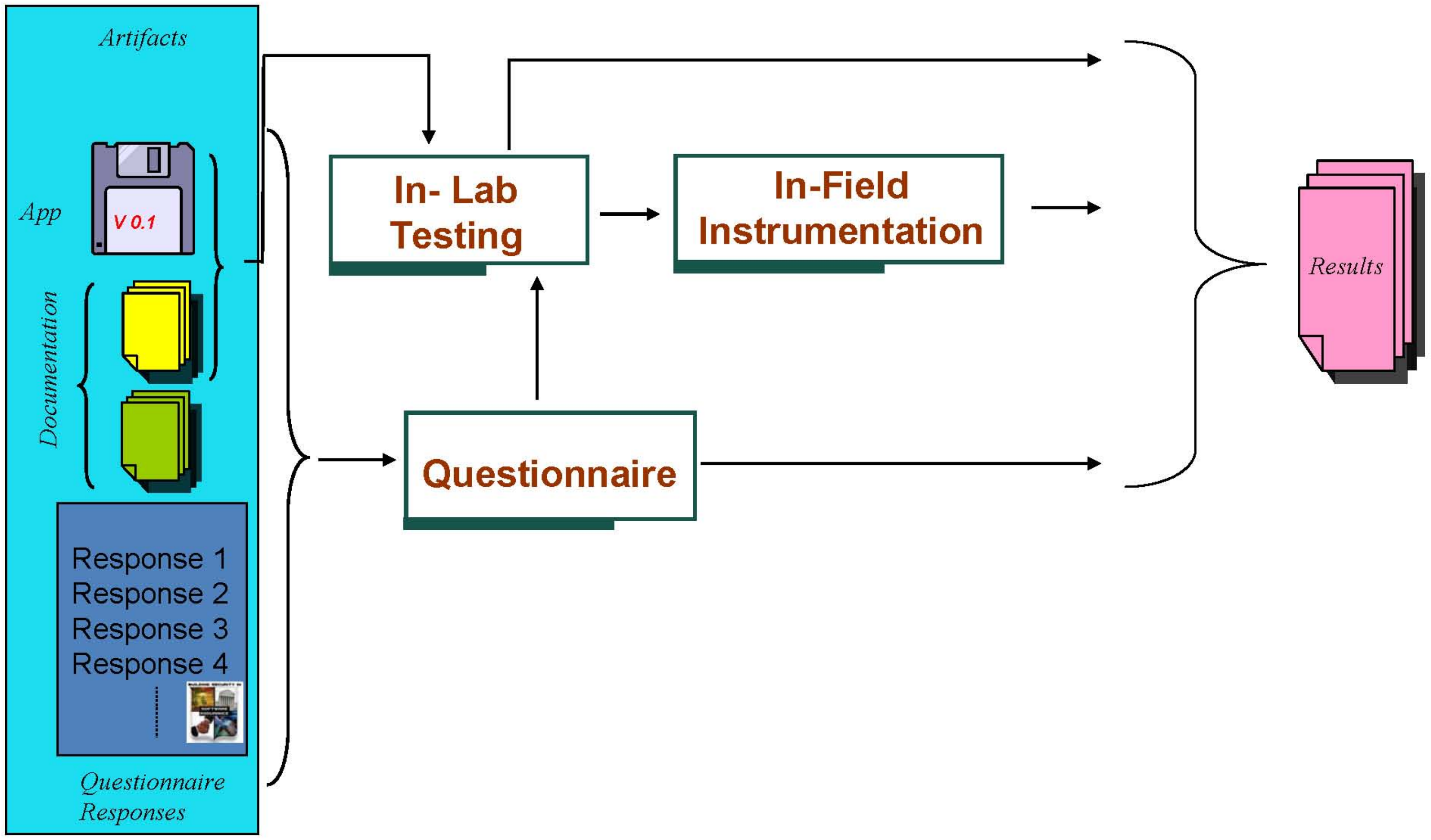
App Vetting & Control

- App Signing – Prevent unauthorized App Execution
  - Approved Apps are signed by the program designated approval authority
  - Only program signed Apps can be installed on the device
    - Customizations made to Android package framework
- App Analysis & Testing
  - All Apps are analyzed for malware and potential vulnerabilities
    - AV Scans
    - Vulnerability Scans (Fortify)
  - Expose hidden & unwanted functionality
    - Hidden in Native Libraries
    - Dynamic or obfuscated code
  - Permissions manifest reconciliation against code

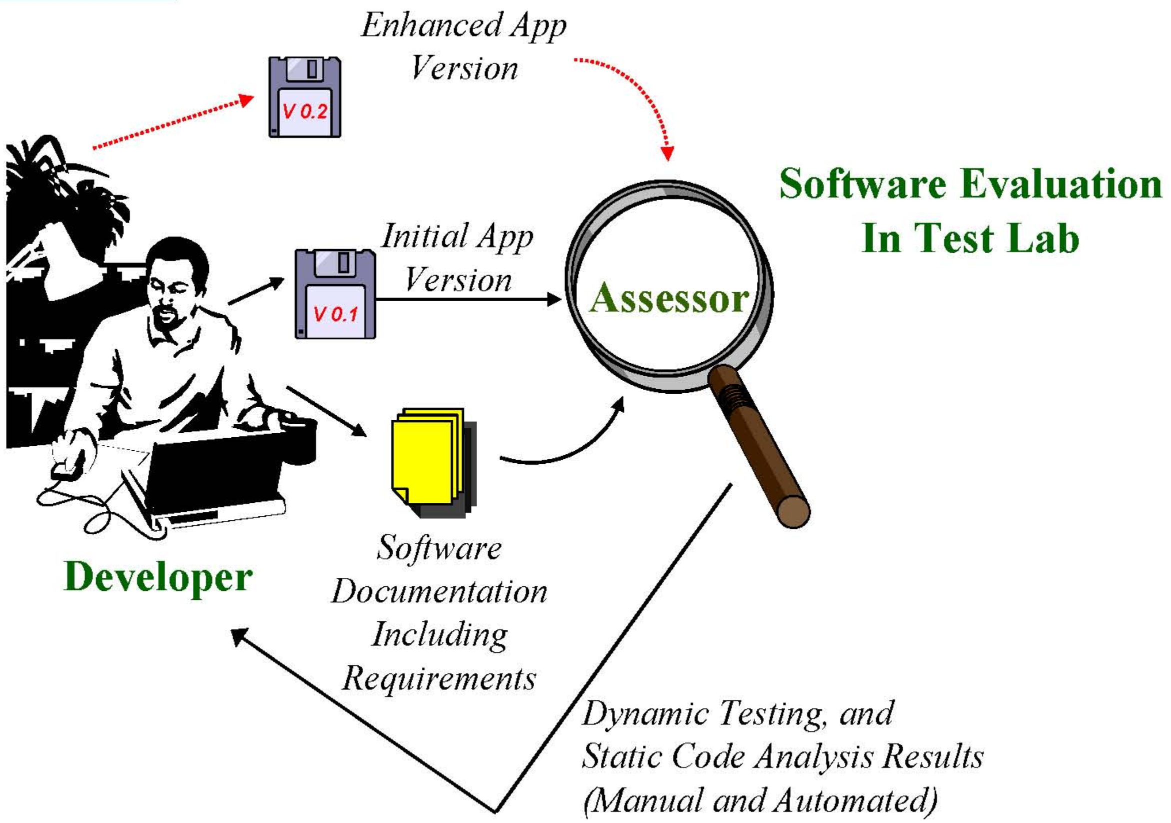
Application Vetting: Big Picture



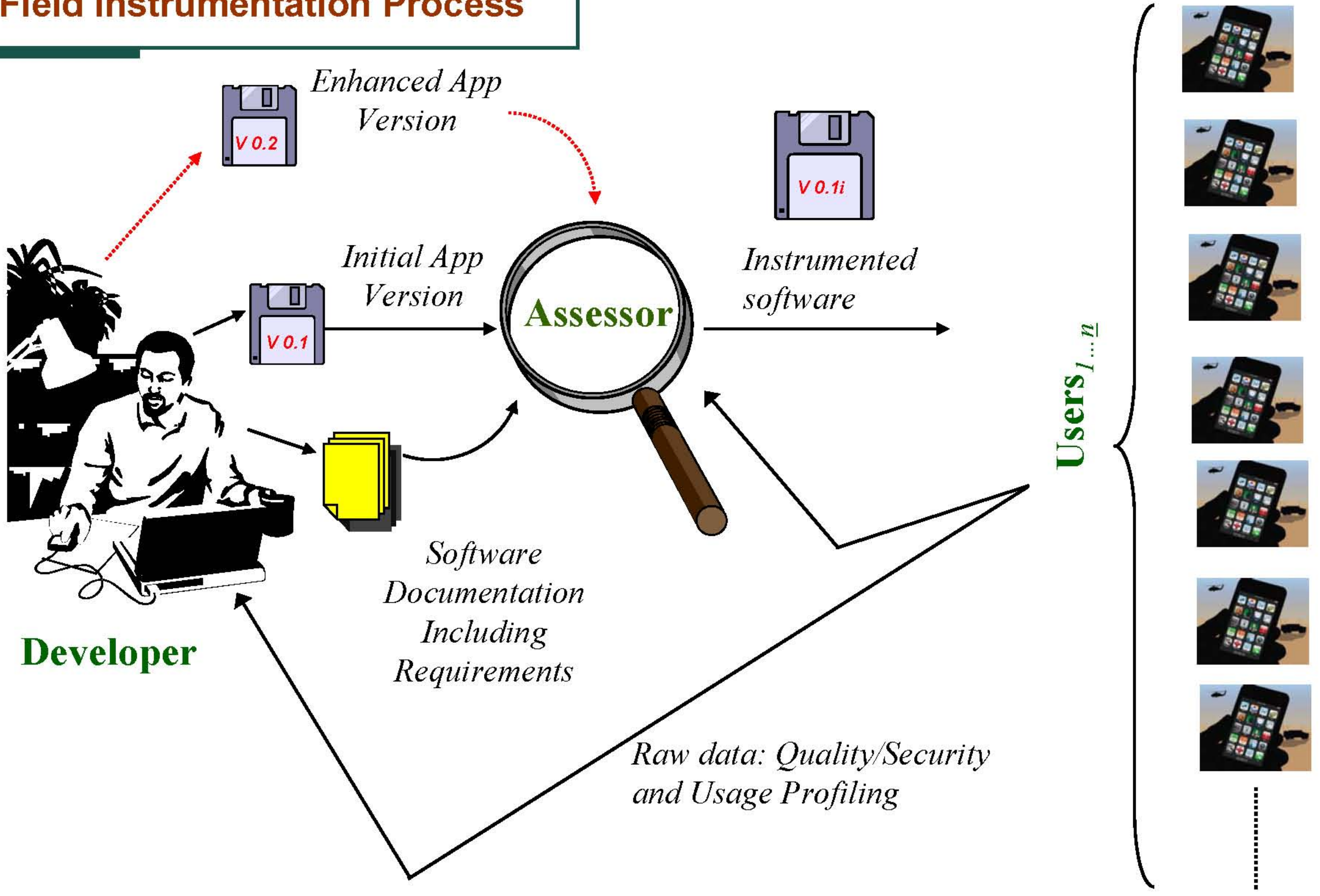
Progression of Testing



In-Lab Testing Process


Enhanced App Version
V 0.2
Software Evaluation In Test Lab
Initial App Version
V 0.1
Assessor
Developer
Software Documentation Including Requirements
Dynamic Testing, and Static Code Analysis Results (Manual and Automated)

In-Field Instrumentation Process



Android Application Control

- Application Signing – Prevent unauthorized App Execution
  - Approved Apps are signed by the program designated approval authority
  - Only program signed Apps can be installed on the device
    - Customizations made to Android package framework
- Application Stress Testing
  - Measure Power Consumption
  - Identify Input Errors / Find UI bugs

Application Analysis Framework

- Android Specific Analysis includes analysis of the Application Security Manifest
  - Tailored to the Android Permission Model
- Verify if the requested permissions are warranted by the submitted code
  - Remove excessive permissions & enforce a tighter security model
- Regulate access to critical/restricted resources
  - 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

Application Policy Enforcement

Solution: Per Application Policy Enforcement

Provide Dalvik mechanisms to

- Enforce application Access & Capabilities
  - Tailored to specific Location or Time
  - Tailored to specific Mission
- Application can still be installed but deprived access to resources and data selectively

Policy Enforcement paired with Device Security can significantly reduce the risk of **Data Exfiltraction**

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

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

Power Metering Framework

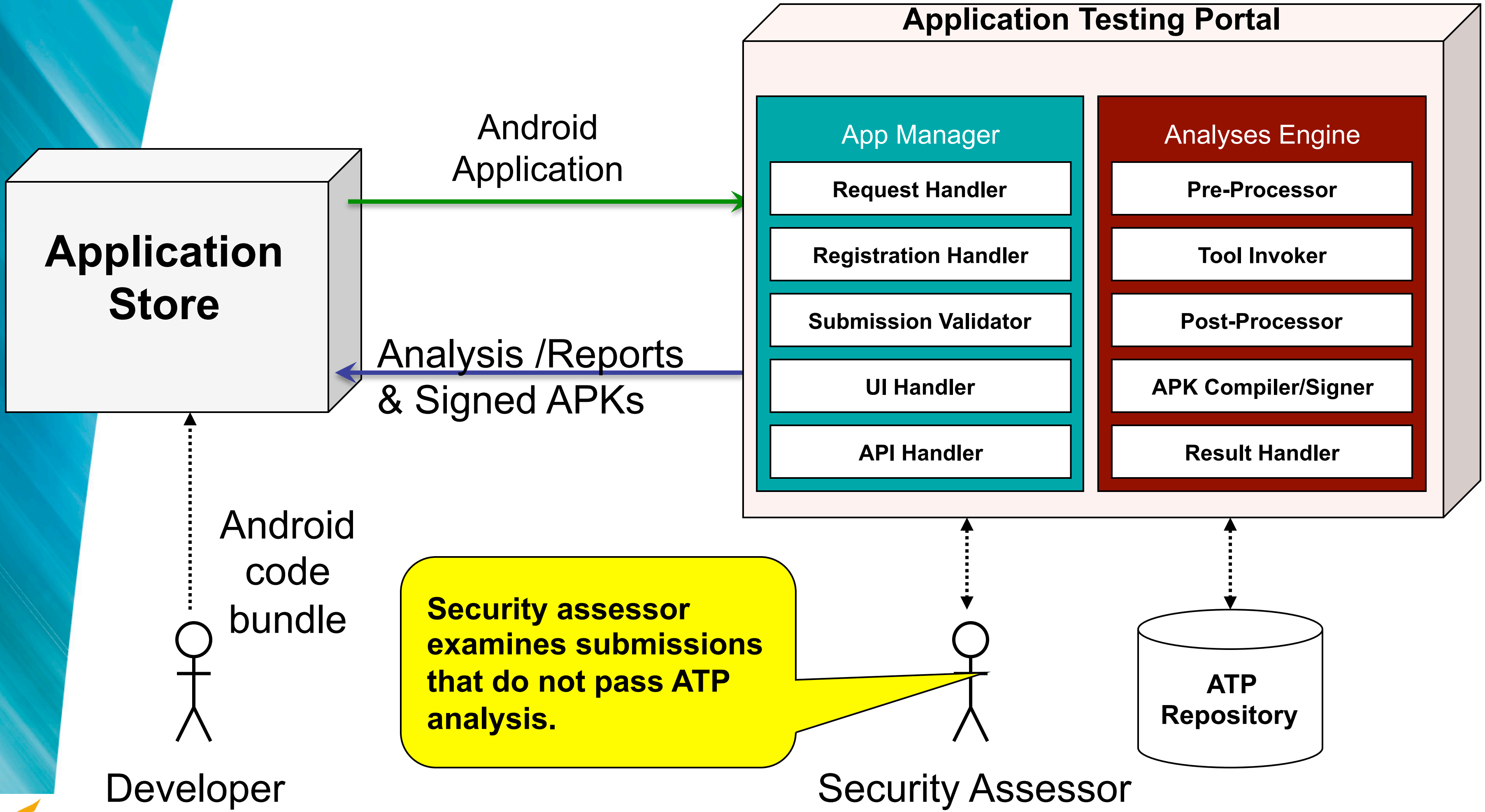
- Design & Implement an accurate model for accounting and policing energy consumption

- 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

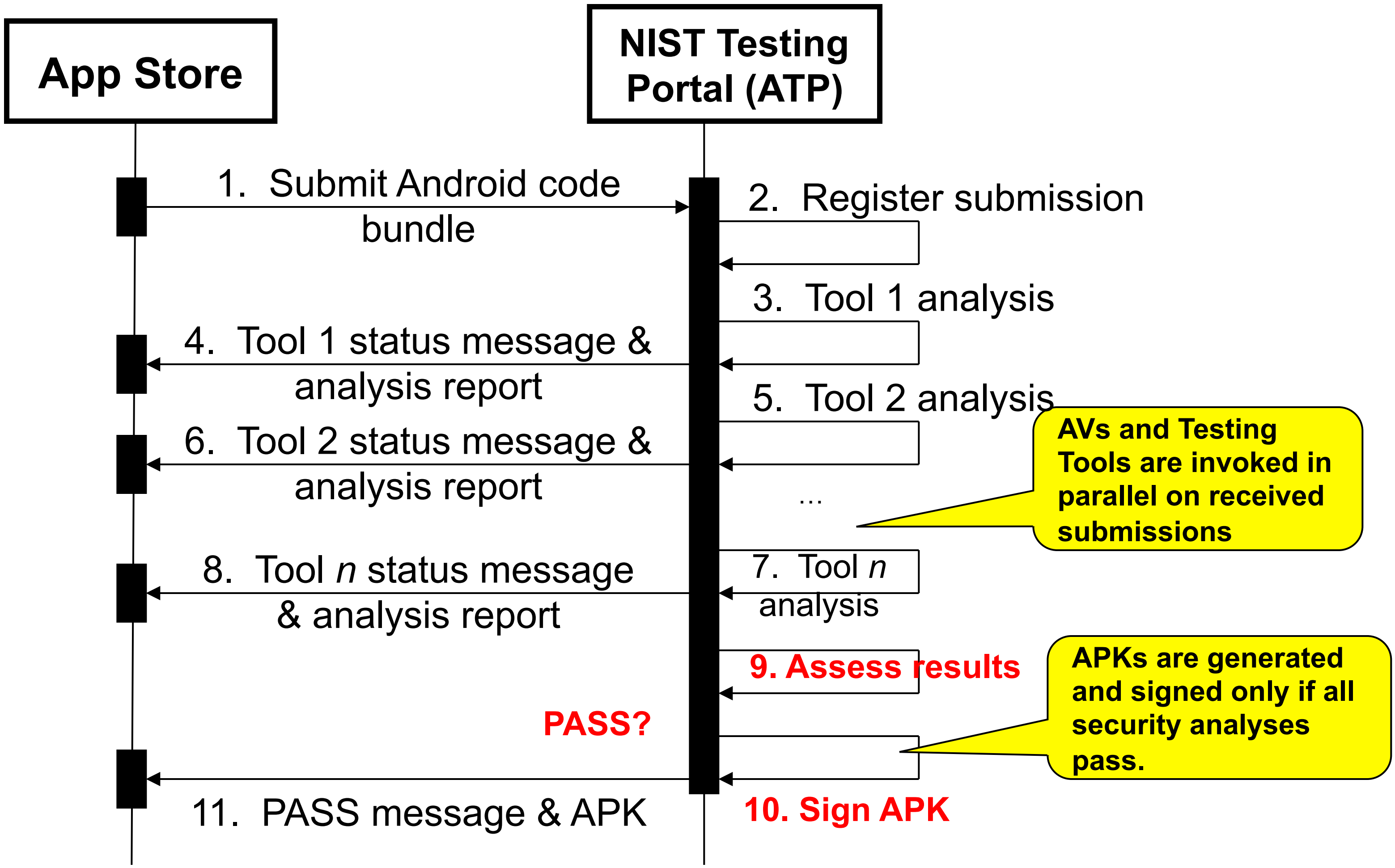
ATP Architecture

ATP analyzes Android code bundles and returns messages, analysis reports, and signed APKs



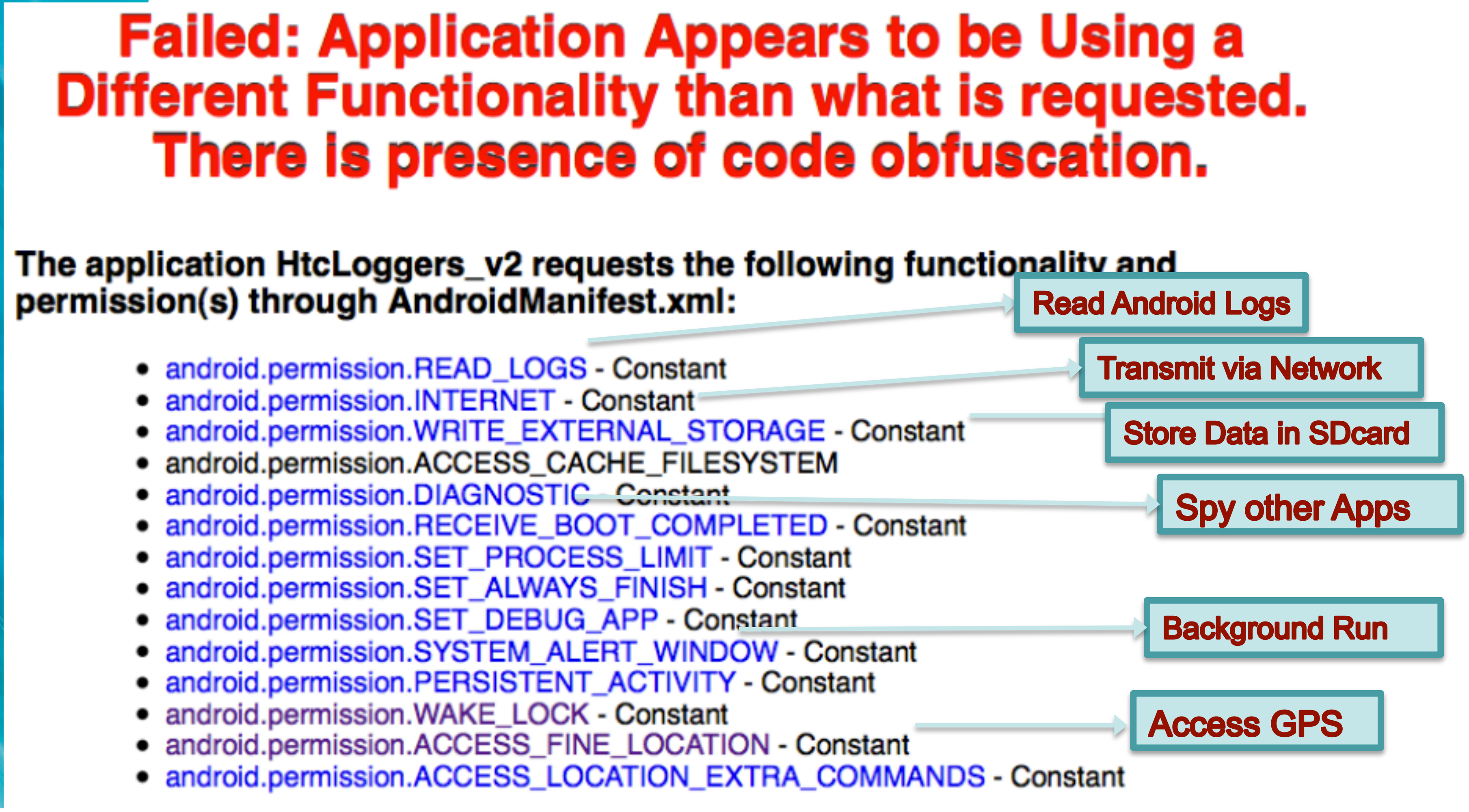
Mobilize-ATP Workflow (PASS Use-Case)

ATP applies Testing to Analyze Android code bundles



App Store
NIST Testing Portal (ATP)
1. Submit Android code bundle
2. Register submission
3. Tool 1 analysis
4. Tool 1 status message & analysis report
5. Tool 2 analysis
6. Tool 2 status message & analysis report
...
AVs and Testing Tools are invoked in parallel on received submissions
7. Tool n analysis
8. Tool n status message & analysis report
9. Assess results
PASS?
APKs are generated and signed only if all security analyses pass.
11. PASS message & APK
10. Sign APK

Analysis of HTC Logger (CarrierIQ)

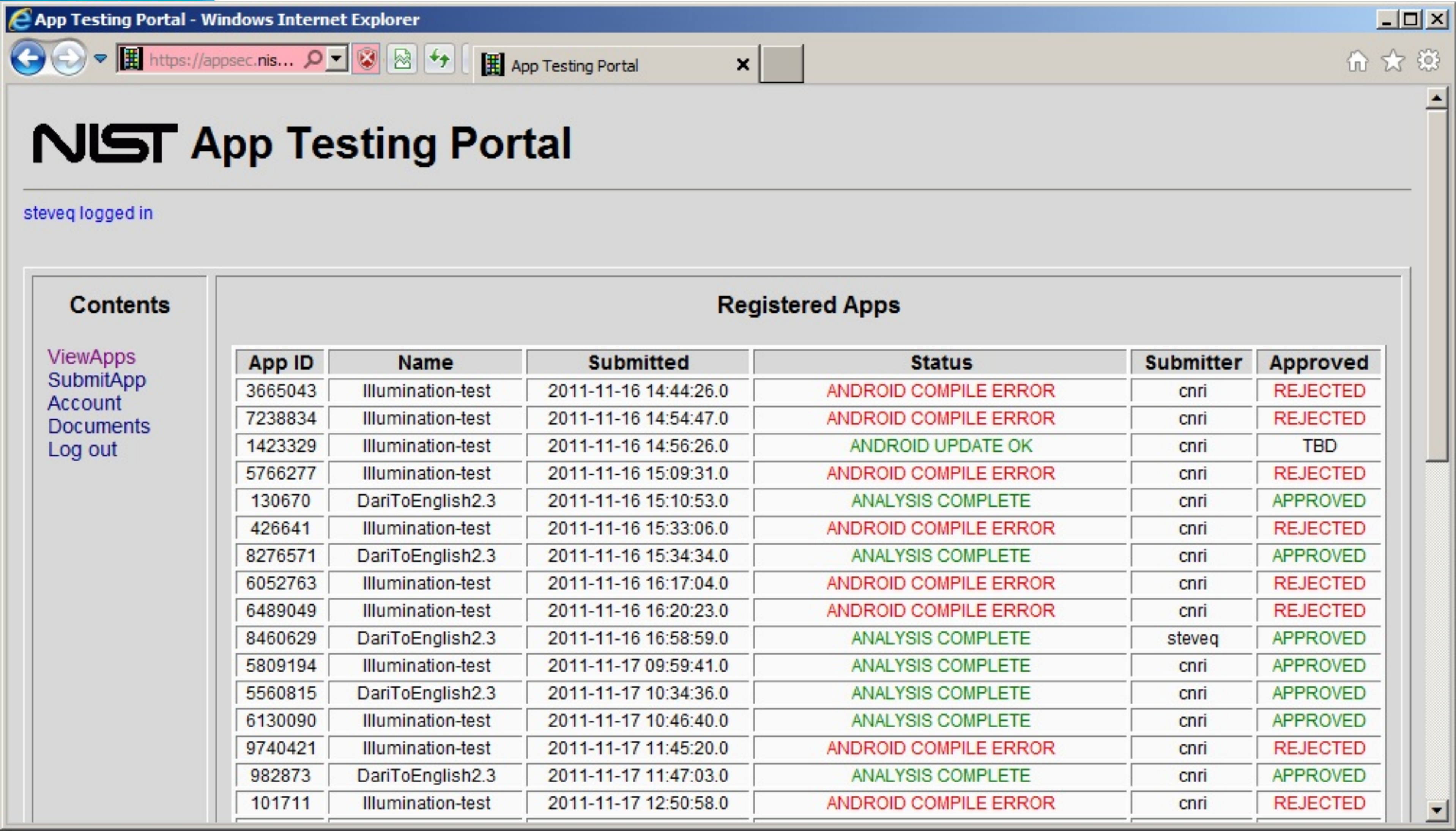


Failed: Application Appears to be Using a Different Functionality than what is requested. There is presence of code obfuscation.

The application HtcLoggers_v2 requests the following functionality and permission(s) through AndroidManifest.xml:

- android.permission.READ_LOGS - Constant
- android.permission.INTERNET - Constant
- android.permission.WRITE_EXTERNAL_STORAGE - Constant
- android.permission.ACCESS_CACHE_FILESYSTEM
- android.permission.DIAGNOSTIC - Constant
- android.permission.RECEIVE_BOOT_COMPLETED - Constant
- android.permission.SET_PROCESS_LIMIT - Constant
- android.permission.SET_ALWAYS_FINISH - Constant
- android.permission.SET_DEBUG_APP - Constant
- android.permission.SYSTEM_ALERT_WINDOW - Constant
- android.permission.PERSISTENT_ACTIVITY - Constant
- android.permission.WAKE_LOCK - Constant
- android.permission.ACCESS_FINE_LOCATION - Constant
- android.permission.ACCESS_LOCATION_EXTRA_COMMANDS - Constant

ATP Monitor



NIST App Testing Portal

steveq logged in

Contents

- ViewApps
- SubmitApp
- Account
- Documents
- Log out

Registered Apps

App ID	Name	Submitted	Status	Submitter	Approved
3665043	Illumination-test	2011-11-16 14:44:26.0	ANDROID COMPILE ERROR	cnri	REJECTED
7238834	Illumination-test	2011-11-16 14:54:47.0	ANDROID COMPILE ERROR	cnri	REJECTED
1423329	Illumination-test	2011-11-16 14:56:26.0	ANDROID UPDATE OK	cnri	TBD
5766277	Illumination-test	2011-11-16 15:09:31.0	ANDROID COMPILE ERROR	cnri	REJECTED
130670	DariToEnglish2.3	2011-11-16 15:10:53.0	ANALYSIS COMPLETE	cnri	APPROVED
426641	Illumination-test	2011-11-16 15:33:06.0	ANDROID COMPILE ERROR	cnri	REJECTED
8276571	DariToEnglish2.3	2011-11-16 15:34:34.0	ANALYSIS COMPLETE	cnri	APPROVED
6052763	Illumination-test	2011-11-16 16:17:04.0	ANDROID COMPILE ERROR	cnri	REJECTED
6489049	Illumination-test	2011-11-16 16:20:23.0	ANDROID COMPILE ERROR	cnri	REJECTED
8460629	DariToEnglish2.3	2011-11-16 16:58:59.0	ANALYSIS COMPLETE	steveq	APPROVED
5809194	Illumination-test	2011-11-17 09:59:41.0	ANALYSIS COMPLETE	cnri	APPROVED
5560815	DariToEnglish2.3	2011-11-17 10:34:36.0	ANALYSIS COMPLETE	cnri	APPROVED
6130090	Illumination-test	2011-11-17 10:46:40.0	ANALYSIS COMPLETE	cnri	APPROVED
9740421	Illumination-test	2011-11-17 11:45:20.0	ANDROID COMPILE ERROR	cnri	REJECTED
982873	DariToEnglish2.3	2011-11-17 11:47:03.0	ANALYSIS COMPLETE	cnri	APPROVED
101711	Illumination-test	2011-11-17 12:50:58.0	ANDROID COMPILE ERROR	cnri	REJECTED

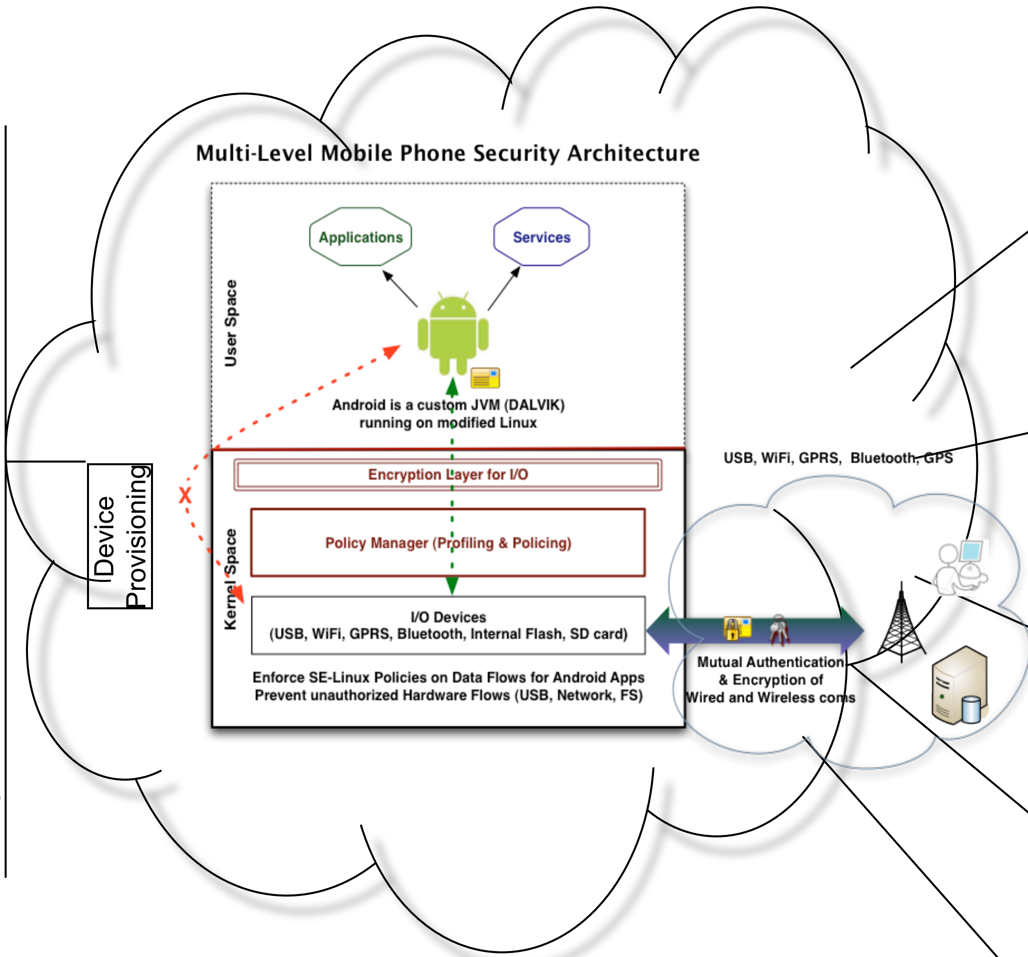
Defense in-Depth: Multiple Levels of Security

- ❖ Application Vetting & Testing
- ❖ Device Lock-down and Encryption of ALL Data and Communications
- ❖ Enforcement of Security Policies in the Android Framework
- ❖ Second-level Defenses placed in the Android Linux Kernel
  - ❖ Prevent Attacks that bypass Android Security Framework
  - ❖ Android has Inherited some (if not all) of the Linux Vulnerabilities
  - ❖ **Java Native Interface to Linux Libraries a potential Avenue for Exploitation**

Risks in Mobile Security Supply Chain

Devices

MDM/Middleware Providers



Device Provisioning
Multi-Level Mobile Phone Security Architecture
User Space
Applications
Services
Android is a custom JVM (DALVIK) running on modified Linux
X
Kernel Space
Encryption Layer for I/O
Policy Manager (Profiling & Policing)
I/O Devices (USB, WiFi, GPRS, Bluetooth, Internal Flash, SD card)
Enforce SE-Linux Policies on Data Flows for Android Apps
Prevent unauthorized Hardware Flows (USB, Network, FS)
Mutual Authentication & Encryption of Wired and Wireless coms
USB, WiFi, GPRS, Bluetooth, GPS

Enterprise Security

Enterprise Security

Enterprise Security

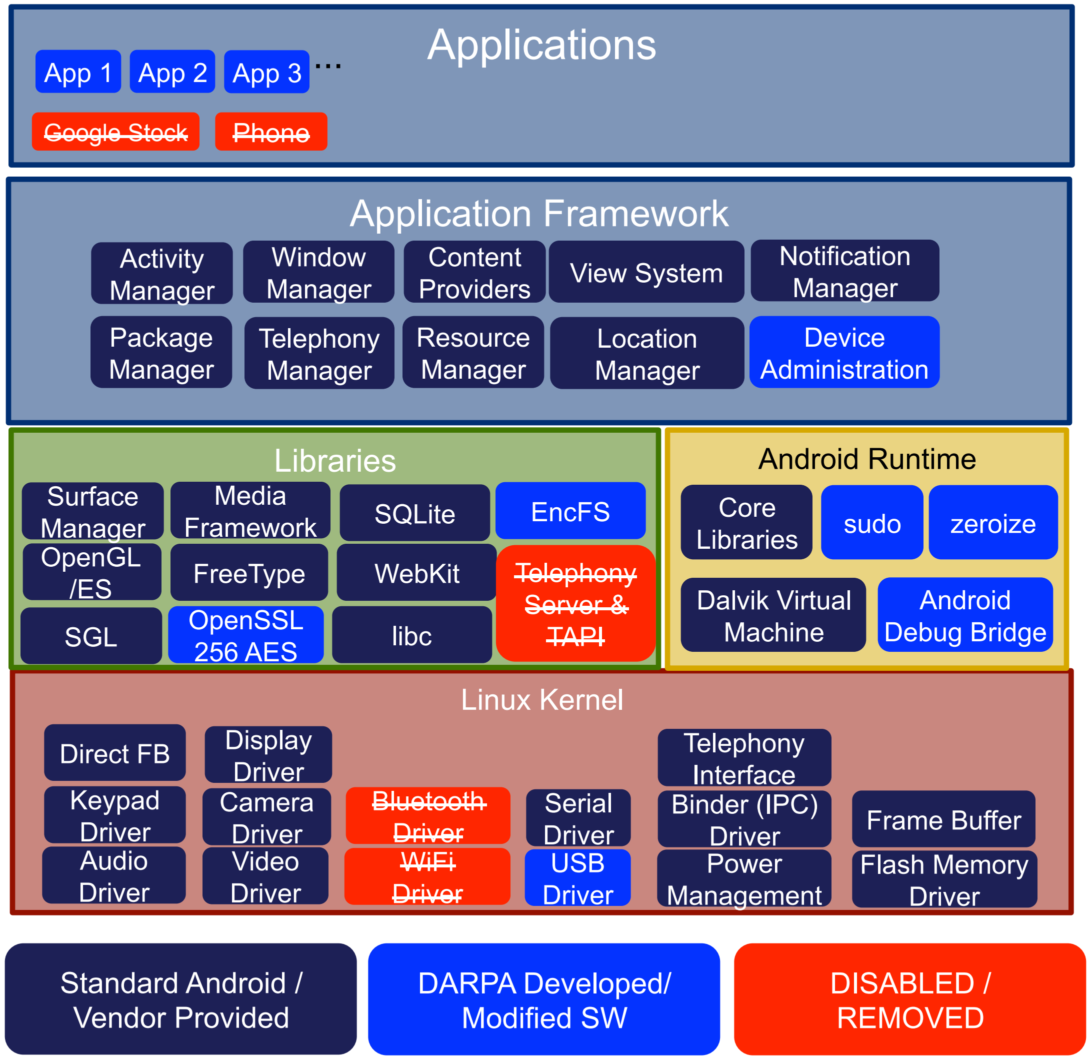
Enterprise Security

Enterprise Security

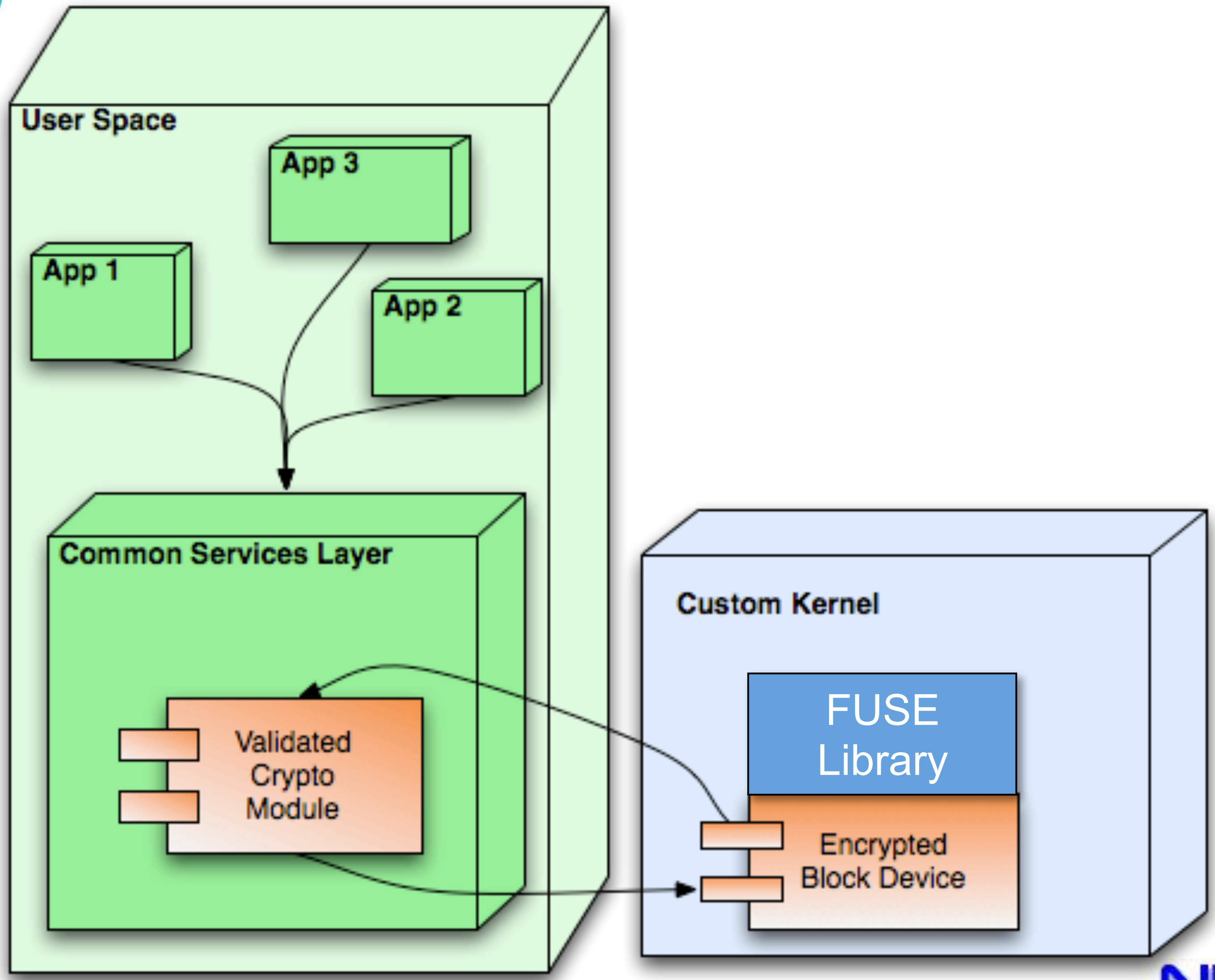
Secure Verify Test Deploy

Hardened Android Platform

- **Custom modified Android OS and Linux Kernel**
  - Additions, deletions, and modifications
  - Preference towards Open Source Solutions
- **Security Stack**
  - Data At Rest Encryption
  - Data In Transit Protections
  - Authentication
  - App Vetting and Control
  - Device Integrity Checks



Encrypted File System



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

Assuring the Secure Operation of Smart Devices has a wide-range of requirements

- Application Testing
  - Static & Dynamic
  - In-Field Instrumentation
  - Power Behavior Metering & Policing

- Physical Device Security
  - Lock-Down of the Device I/O (USB, WiFi, etc.)
  - Encryption of Data both on the Phone & Network
  - Securing Provisioning Process

Thank you!

Questions ?