Android Security Extensions

- SSL Usability
- Taint tracking
- Extensible Security
- Inlined Reference Monitoring

Overview Selected Security Extensions and Tools

Malware Detection
- **Kirin** [Enck et al., ACM CCS 2009]
- **Apex** [Naumann et al., AsiaCCS 2010]
- **Paranoid** [Portokalidis et al., ACSAC 2010]
- **Airmid** [Nadji et al., ACSAC 2011]
- **DroidScope** [Yan et al., USENIX Sec. 2012]

In-App Ad Library Malware
- **AdRisk** [Grace et al., WiSec 2012]
- **AdDroid** [Pearce et al., AsiaCCS 2012]
- **AdSplit** [Dietz et al., USENIX Sec. 2012]

Privilege Escalation (Application-Level)
- **Confused Deputy**
  - **IPC Inspection** [Felt et al., USENIX Sec. 2012]
  - **QUIRE** [Dietz et al., USENIX Sec. 2012]
  - **XManDroid** [Bugiel et al., NDSS 2012]
  - **SORBET** [Fragkaki et al., TR 2012]

Privilege Escalation (Multi-Level)
- **FlaskDroid** [Bugiel et al., USENIX Sec. 2013]
- **SE Android** [Smalley and Craig, NDSS 2013]
- **ASM** [Heuser et al., USENIX Sec. 2014]
- **ASF** [Backes et al., ACSAC 2014]
Overview Selected Security Extensions and Tools

Detecting and Preventing Private Data Leakage

- **TaintDroid** [Enck et al., USENIX OSDI 2010]

TISSA [Zhou et al., TRUST 2011]

AppFence [Hornyack et al., ACM CCS 2011]

Application Hardening and Context-Based Policies

**System extension**

- **SAINT** [Ongtang et al., ACSAC 2009]
- **CRePE** [Conti et al., ISC 2010]

**Inlined Reference Monitor**

- **Aurasium** [Xu et al., USENIX Sec. 2012]
- **AppGuard** [Backes et al., DPM 2013]
- **Dr Android and Mr. Hide** [Jeon et al., SPSM 2012]

DRM Policies and Domain Isolation

- **Porscha** [Ongtang et al., ACSAC 2010]
- **MOSES** [Russello et al., SACMAT 2012]
- **TrustDroid** [Bugiel et al., ACM SPSM 2011]
- **DroidRanger** [Zhou et al., NDSS 2012]
- **DroidMOSS** [Zhou et al., CODASPY 2012]

Security Aspects of App Stores

- **Meteor** [Barrera et al., IEEE MoST 2012]
Android Security Extensions

SSL Usability [33,35]
Rethinking SSL Development in an Appified World

Sascha Fahl
Marian Harbach
Henning Perl
Markus Kötter
Matthew Smith
Manual App Testing Results

- cherry-picked 100 Apps
- 21 Apps trust all certificates
- 20 Apps accept all hostnames

What we found:
Talking to Developers

- Finding broken SSL in Apps is good…
  …knowing what the root causes are is even better

- We contacted 80 developers of broken apps
  - informed them
  - offered further assistance
  - asked them for an interview

- 15 developers agreed
Common: Blaming Developers

“It’s all the developers’ fault!”
It's Time to Rethink how we code SSL

Rethinking SSL Development in an Appified World
Developers’ Wishlist

- Self-Signed Certificates – Development
- Self-Signed Certificates – Production
- Less SSL Coding
- Certificate Pinning / Trusted Roots
- Easy-to-use Warning Message
Getting out of the Dilemma

Current Situation:
- Developers have the freedom to customize certificate validation
- Developers mostly are not security experts
- Developers find the current situation too inflexible

Future Situation:
- Protect the user!
- Make the common use cases easy
- Adapt certificate handling to the developers’ needs

Solution: Improve usability of certificate handling for developers!
Patching Android OS

Rethinking SSL Development in an Appified World

Force hostname verification

Force certificate validation; Configurable by the users

android.net.ssl

TrustManagerClient (in app)

uses replaced by

android.net.ssl

TrustManagerService (in system)

configures

Pluggable Certificate Validation: (CA-based validation, CT, AKI, TACK, etc.)

Turn on/off SSLPinning, Accept all certificates on developer devices

Warn the user if connection is insecure

org.apache.http.conn.ssl

SSLSocketFactory

start

uses

removed

javax.net.ssl

TrustManager

warn if SSL validation fails

User options

Developer options

decisions

Human Computer Interface

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.

Certificate Validation

Certificate Pinning

CA Validation

Development Mode

Warning message: May prevent careless users from turning off certificate validation for specific apps installed on their devices.
Self-Signed Certificates – Development

enable developer options

disable SSL validation for this app only
This is easy!

```xml
<uses-ssl>
  <pins host="seuressl.com">
    <pin type="ca" comment="Verisign Root CA">
      8F:57:5A:C8:5B:09:63:B0:24:2B:90...
    </pin>
    <pin type="cert" comment="Self-Signed">
      18:DA:D1:9E:26:7D:E8:BB:4A:21:58...
    </pin>
  </pins>
</uses-ssl>
```
Certificate Pinning / Trusted Issuers (2)

This is easier!

```java
URL url = new URL("https://www.dcsec.uni-hannover.de");
HttpsURLConnection conn = (HttpsURLConnection) url.openConnection();
conn.setReadTimeout(1000000000000L); // milliseconds
conn.setConnectTimeout(1500000000000L); // milliseconds
conn.setRequestMethod("GET");
conn.setDoInput(true);
```

Team

Compare With

Replace With

Properties

Import certificate for SSL Pinning
Conclusion

✔ Eve and Mallory no longer love Android
✔ Backwards compatible – no broken apps, except
  ❌ apps that implemented pinning (19 in 13500 tested Android apps)
  ✔ updating them to the new pinning system is very easy
✔ New features for Android
   ✔ Easy to use self-signed certs for development
   ✔ Easy to use pinning / custom CAs
   ✔ Central and easy to use warning messages
   ✔ Central place to plug in new validation strategies – such as CT, TACK, etc
✔ Contacted developers –
   ✔ got positive feedback

Download the Code and have a go: http://android-ssl.org
Android Security Extensions

Taint tracking [36,37]
TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones

OSDI’10

William Enck, Peter Gilbert, Byung-Gon Chun, Landon P. Cox, Jaeyeon Jung, Patrick McDaniel, and Anmol N. Sheth
Monitoring Smartphone Behavior

• There are tens of thousands of smartphone apps that provide both fun and valuable utility.

• *General challenge*: balance fun and utility with privacy

• Step 1: “look inside” of applications to watch how they use privacy sensitive data
  
  ‣ location
  ‣ phone identifiers
  ‣ microphone
  ‣ camera
  ‣ address book
Challenges

• **Goal**: Monitor app behavior to determine when privacy sensitive information leaves the phone

• **Challenges** ...
  
  ‣ Smartphones are resource constrained
  
  ‣ Third-party applications are entrusted with several types of privacy sensitive information
  
  ‣ Context-based privacy information is dynamic and can be difficult to identify even when sent in the clear
  
  ‣ Applications can share information
Dynamic Taint Analysis

• Dynamic taint analysis is a technique that tracks information dependencies from an origin

• Conceptual idea:
  ‣ Taint source
  ‣ Taint propagation
  ‣ Taint sink

```
c = taint_source()
...
a = b + c
...
network_send(a)
```

• Limitations: performance and granularity is a trade-off
• **TaintDroid** is a system-wide integration of taint tracking into the Android platform

  ‣ Variable tracking throughout Dalvik VM environment
  ‣ Patches state after native method invocation
  ‣ Extends tracking between applications and to storage

• **TaintDroid is a firmware modification, not an app**
VM Variable-level Tracking

• We modified the Dalvik VM interpreter to **store** and **propagate** taint tags (a taint bit-vector) on variables.

• **Local variables and args**: taint tags stored adjacent to variables on the internal execution stack.
  ‣ 64-bit variables span 32-bit storage

• **Class fields**: similar to locals, but inside static and instance field heap objects

• **Arrays**: one taint tag per array to minimize overhead
Native Methods

• Applications execute *native methods* through the Java Native Interface (JNI)

• TaintDroid uses a combination of heuristics and *method profiles* to patch VM tracking state
  ‣ Applications are restricted to only invoking native methods in system-provided libraries
IPC and File Propagation

- TaintDroid uses *message level* tracking for IPC
  - Applications marshall and unmarshall individual data items

- Persistent storage tracked at the *file level*
  - Single taint tag stored in the file system XATTR
Performance

CaffeineMark 3.0 benchmark (higher is better)

- Memory overhead: 4.4%
- IPC overhead: 27%
- Macro-benchmark:
  - App load: 3% (2ms)
  - Address book: (< 20 ms)
    - 5.5% create, 18% read
  - Phone call: 10% (10ms)
  - Take picture: 29% (0.5s)

CaffeineMark score roughly corresponds to the number of Java instructions per second.
Taint Adaptors

• Taint sources and sinks must be carefully integrated into the existing architectural framework.

• Depends on information properties
  ▸ Low-bandwidth sensors: location, accelerometer
  ▸ High-bandwidth sensors: microphone, camera
  ▸ Information databases: address book, SMS storage
  ▸ Device identifiers: IMEI, IMSI*, ICC-ID, Ph. #
  ▸ Network taint sink
Application Study

- Selected 30 applications with bias on popularity and access to *Internet*, *location*, *microphone*, and *camera*

<table>
<thead>
<tr>
<th>applications</th>
<th>#</th>
<th>permissions</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Weather Channel, Cetos, Solitarie, Movies, Babble, Manga Browser</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Bump, Wertago, Antivirus, ABC --- Animals, Traffic Jam, Hearts, Blackjack, Horoscope, 3001 Wisdom Quotes Lite, Yellow Pages, Datelefonbuch, Astrid, BBC News Live Stream, Ringtones</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Layer, Knocking, Coupons, Trapster, Spongebot Slide, ProBasketBall</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>MySpace, Barcode Scanner, ixMAT</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Evernote</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

- Of 105 flagged connections, only 37 clearly legitimate
Findings - Location

• 15 of the 30 applications shared physical location with an ad server (admob.com, ad.qwapi.com, ads.mobclix.com, data.flurry.com)

• Most traffic was plaintext (e.g., AdMob HTTP GET):

```plaintext
...&s=a14a4a93f1e4c68&...&t=062A1CB1D476DE85B717D9195A6722A9&d%5Bcoord%5D=47.66122789000006%2C-122.31589477&...```

• In no case was sharing obvious to user or in EULA
  ‣ In some cases, periodic and occurred without app use
Findings - Phone Identifiers

• 7 applications sent device (IMEI) and 2 apps sent phone info (Ph. #, IMSI*, ICC-ID) to a remote server without informing the user.
  ‣ One app’s EULA indicated the IMEI was sent
  ‣ Another app sent the hash of the IMEI

• Frequency was app-specific, e.g., one app sent phone information every time the phone booted.

• Appeared to be sent to app developers ...

“There have been cases in the past on other mobile platforms where well-intentioned developers are simply over-zealous in their data gathering, without having malicious intent.” -- Lookout
Limitations

• **Approach limitations:**
  ‣ TaintDroid only tracks data flows (i.e., explicit flows).

• **Taint source limitations:**
  ‣ IMSI contains country (MCC) and network (MNC) codes
  ‣ File databases must be all one type
Summary

- TaintDroid provides efficient, system-wide, dynamic taint tracking and analysis for Android.
- We found 20 of the 30 studied applications to share information in a way that was not expected.
- Source code will be available soon: appanalysis.org
- Future investigations:
  - Provide direct feedback to users
  - Potential for realtime enforcement
  - Integration with expert rating systems
Demo

• Demo available at http://appanalysis.org/demo/

TaintDroid running on Nexus One

* video produced by Peter Gilbert (gilbert@cs.duke.edu)
* special thanks to Gabriel Maganis (maganis@cs.ucdavis.edu) for TaintDroid UI
These Aren’t the Droids You’re Looking For

Retrofitting Android to Protect Data from Imperious Applications

Peter Hornyack, Seungyeop Han, Jaeyeon Jung, Stuart Schechter, David Wetherall
What is “sensitive data”?

We identified 12 types of privacy-sensitive data on Android:

<table>
<thead>
<tr>
<th>data types</th>
</tr>
</thead>
<tbody>
<tr>
<td>device id</td>
</tr>
<tr>
<td>location</td>
</tr>
<tr>
<td>phone number</td>
</tr>
<tr>
<td>contacts</td>
</tr>
<tr>
<td>camera</td>
</tr>
<tr>
<td>accounts</td>
</tr>
<tr>
<td>logs</td>
</tr>
<tr>
<td>microphone</td>
</tr>
<tr>
<td>SMS messages</td>
</tr>
<tr>
<td>history &amp; bookmarks</td>
</tr>
<tr>
<td>calendar</td>
</tr>
<tr>
<td>subscribed feeds</td>
</tr>
</tbody>
</table>
How can we tell what apps are doing?

* **TaintDroid**: dynamic taint tracking for Android applications [Enck10]

```
loc = getLocation();     //taint tag applied
...
loc_copy = loc;          //taint propagated
...
network_send(loc_copy);  //checked for taint
```

* Apps can’t transform, obfuscate or encrypt data to remove taint
* We enhanced TaintDroid: added tracking for all 12 data types

Gives us runtime detection of sensitive data transmission for *unmodified apps*

(example taken from William Enck OSDI’10)
© 2011 Peter Hornyack
How can we defend against these apps?

- Threat: applications may misappropriate users’ sensitive data
- We have a tool, TaintDroid, that can monitor unmodified applications at runtime
- Can we do something simple to unmodified applications to defend against this threat?

Our system: AppFence
AppFence uses two privacy controls

- Two complementary privacy controls:
  - Shadowing: app doesn’t get sensitive data at all
  - Blocking: app gets sensitive data, but can’t send it out

data shadowing

exfiltration blocking
How data shadowing works

Without data shadowing:

Unmodified Application

Phone #?

Android

(206) 555-4321

analytics.com

(206) 555-4321
How data shadowing works

With data shadowing:

Unmodified Application

Phone #?

Android

analytics.com

(206) 555-4321
(123) 456-7890

Shadow data
Three kinds of shadow data

* Blank data
  * e.g. contacts: {S. Han, 206-555-4321} ➞ {}  

* Fake data
  * e.g. location: {47.653,-122.306} ➞ {41.887,-87.619}  

* Constructed data
  * e.g. device ID = hash(app name, true device ID)
    * Consistent for each application, but different across applications
How exfiltration blocking works

Without exfiltration blocking:

Unmodified Application

Phone #?

Android

© 2011 Peter Hornyack
How exfiltration blocking works

With exfiltration blocking:

Unmodified Application

Phone #?
(206) 555-4321

Android

analytics.com
How exfiltration blocking works

With exfiltration blocking:

Unmodified Application

Phone #?
(206) 555-4321

Airplane mode: no network available

analytics.com

Android
## Side effects shown by 50 apps

<table>
<thead>
<tr>
<th></th>
<th>Data shadowing</th>
<th>Exfiltration blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>None</strong></td>
<td>28 (56%)</td>
<td>16 (32%)</td>
</tr>
<tr>
<td><strong>Ads absent</strong></td>
<td>0 (0%)</td>
<td>11 (22%)</td>
</tr>
<tr>
<td><strong>Less functional</strong></td>
<td>14 (28%)</td>
<td>10 (20%)</td>
</tr>
<tr>
<td><strong>Broken</strong></td>
<td>8 (16%)</td>
<td>13 (26%)</td>
</tr>
</tbody>
</table>

* Slightly more than half of the apps ran with limited or no side effects
Side effects shown by 50 apps

<table>
<thead>
<tr>
<th></th>
<th>Data shadowing</th>
<th>Exfiltration blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>28 (56%)</td>
<td>16 (32%)</td>
</tr>
<tr>
<td>Ads absent</td>
<td>0 (0%)</td>
<td>11 (22%)</td>
</tr>
<tr>
<td>Less functional</td>
<td>14 (28%)</td>
<td>10 (20%)</td>
</tr>
<tr>
<td>Broken</td>
<td>8 (16%)</td>
<td>13 (26%)</td>
</tr>
</tbody>
</table>

- Slightly more than half of the apps ran with limited or no side effects
- Data shadowing was less disruptive than exfiltration blocking
## Side effects shown by 50 apps

<table>
<thead>
<tr>
<th></th>
<th>Data shadowing</th>
<th>Exfiltration blocking</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>28 (56%)</td>
<td>16 (32%)</td>
</tr>
<tr>
<td>Ads absent</td>
<td>0 (0%)</td>
<td>11 (22%)</td>
</tr>
<tr>
<td>Less functional</td>
<td>14 (28%)</td>
<td>10 (20%)</td>
</tr>
<tr>
<td>Broken</td>
<td>8 (16%)</td>
<td>13 (26%)</td>
</tr>
</tbody>
</table>

* Remember, we applied a *single* privacy control (one or the other) to *all* applications
## Side effects shown by 50 apps

<table>
<thead>
<tr>
<th></th>
<th>Data shadowing</th>
<th>Exfiltration blocking</th>
<th>Choose least-disruptive</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>28 (56%)</td>
<td>16 (32%)</td>
<td>30 (60%)</td>
</tr>
<tr>
<td>Ads absent</td>
<td>0 (0%)</td>
<td>11 (22%)</td>
<td>3 (6%)</td>
</tr>
<tr>
<td>Less functional</td>
<td>14 (28%)</td>
<td>10 (20%)</td>
<td>11 (22%)</td>
</tr>
<tr>
<td>Broken</td>
<td>8 (16%)</td>
<td>13 (26%)</td>
<td>6 (12%)</td>
</tr>
</tbody>
</table>

* Choose the control that caused least-severe side effects for each app: 33 apps (66%) had no side effects or ads absent
* We used profiling to choose; determining in advance is challenging

© 2011 Peter Hornyack
So 34% of applications didn’t work?

These apps had four kinds of functionality that *directly conflict* with our configuration (sensitive data should never leave the device):

- Location broadcast (location)
- Geographic search (location)
- Find friends (contacts)
- Cross-application gaming profiles (device ID)
What does this mean for AppFence?

- Some applications force the user to choose between functionality and privacy
  - Protecting sensitive data will always cause side effects for these applications

- Remaining apps: AppFence can prevent misappropriation without side effects
  - Choosing the least-disruptive privacy control in advance is still an open problem
  - Each control was less disruptive for certain sensitive data types
Conclusion

* AppFence breaks the power of the installation ultimatum

* We revealed side effects by *never* allowing sensitive data to leave the device

* Some apps: user must choose between functionality and privacy

* Majority of apps: two privacy controls can prevent misappropriation without side effects
Android Security Extensions

Extensible Security [39,40]

ASM: A Programmable Interface for Extending Android Security

Stephan Heuser, Ahmad-Reza Sadeghi
Intel Collaborative Research Institute for Secure Computing at TU Darmstadt, Germany

Adwait Nadkarni, William Enck
NC State University, USA
Android Security Framework:
Extensible multi-layered access control on Android

M. Backes, S. Bugiel, S. Gerling, and P. von Styp-Rekowsky

To appear at 30th Annual Computer Security Applications Conference (ACSAC’14)
Android Security Extensions (selected)

Security extensions focus on specific use cases and/or security and privacy models

- **Privacy**
  - TaintDroid, AppFence, MockDroid

- **IPC Provenance**
  - QUIRE, IPC Inspection

- **Fine-Grained Permissions**
  - APEX, CRePE

- **Permission Constraints**
  - Kirin

- **Context-based Apps**
  - CRePE, ConXSense

- **App Communication**
  - Saint, XManDroid, TrustDroid, Aquifer

- **Mock Data**
  - MockDroid, TISSA, AppFence

- **Type Enforcement**
  - SEAndroid, FlaskDroid
Android Security Extensions

Access control (hooks) are embedded in sensitive components

System Apps

3rd Party Apps

3rd Party App (Inlined Reference Monitor)

Access Control

System Content Providers (e.g. contacts)

Framework Libraries

Activity Manager Service

Package Manager Service

Access Control

Android OS

Linux Kernel

ASM - Android Security Modules
Research Question

Is it possible to provide a *programmable* and *generic* security architecture on top of which many of these solutions can be instantiated?
# Observations

Diverse Goals, but use similar security hooks and mechanisms

<table>
<thead>
<tr>
<th>System</th>
<th>Android ICC</th>
<th>Package Manager</th>
<th>Sensors / Phone Info</th>
<th>Fake Data</th>
<th>System Content Providers</th>
<th>File Access</th>
<th>Network Access</th>
<th>3rd Party Hooks</th>
</tr>
</thead>
<tbody>
<tr>
<td>MockDroid</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XManDroid</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TrustDroid</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>FlaskDroid</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>CRePE</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Quire</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>TaintDroid</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Kirin</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPC Inspection</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>AppFence</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Aquifer</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>APEX</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Saint</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>SEAndroid</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>TISSA</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
High-level Idea of ASM

- A modular access control architecture supporting multiple stakeholders
- Deploy Android Security Modules (ASMs) as apps
ASM Framework

1. Register Callback Service
2. Query Hooks

System Providers (e.g. contacts)

Reference Monitor

System Services (e.g. ActivityManager)

ASM Bridge

Applications

Android OS

Linux Kernel

ASM User
ASM Provider
ASM Enterprise
3rd Party App
WhatsApp

ASM LSM
SELinux LSM

ASM - Android Security Modules
Hook Invocation

Applications

ASM User
ASM Provider
ASM Enterprise
3rd Party App (e.g., WhatsApp)

Android OS

SSL
System ContentProviders (e.g., contacts)
System Services (e.g., ActivityManager)

ASMA Bridge

Linux Kernel

SELinux LSM
ASM LSM
Support for 3rd-Party Hooks

- **ASM User**
- **ASM Provider**
- **ASM Enterprise**
- **ASM aware 3rd Party App**

- **System ContentProviders** *(e.g. contacts)*

- **ASM Bridge**

- **SELinux LSM**
- **ASM LSM**

Applications

Android OS

Linux Kernel

ASM - Android Security Modules
ConXSense
Context Aware Access Control

- Goal: Context-aware access control
- Context-aware access control enforcing policies by user context profiling
- Includes access control on sensors (e.g., GPS and camera), sensitive information (e.g., contacts) and apps

- ASM based implementation:

ConXSense ASM

- User Interface
- Context Profiler
- ASM Callback Service

ActivityManager Service
LocationManager Service
System ContentProviders
CameraService

ConXSense [ASIACCS 2014]
ASF API AND USE-CASES

- Security API
  - 136 functions for enforcement hooks in all major system services and apps
  - Hooks support *edit automata* policies (i.e., modification of return values instead of only deny/allow decision)

- Use-cases: Porting previous research prototypes as modules on ASF
  - FlaskDroid/SE Android (type enforcement at middleware and kernel level)
  - TrustDroid (domain isolation private vs work)
  - CRePE (Context-aware access control)
  - AppOps and IntentFirewall (dormant Android features by Google)
  - AppGuard (IRM)
  - XManDroid (Chinese wall policies)
  - Saint (Developer policies)
  - Data shadowing

- In general minimal overhead of porting those use-cases to our system
Android Security Extensions

Inlined Reference Monitoring [38]

**Motivation**

**Existing permission system**

- **To restrictive:**
  - Either accept permissions, or do not install the app.

**Understand an apps behavior**

**Enforce a desired level of privacy**
How to enforce such dynamic permissions?
**Problem Description**

- Ideally performed at OS / Middleware layer
  - Requires firmware modification!
**Problem Description**

- Ideally performed at OS / Middleware layer
  - Requires firmware modification!
- Android isolates app processes: “all apps are created equal”
  - Monitor not privileged enough!

![Diagram showing Operating System, Untr. App, and Monitor App]
**Problem Description**

- Ideally performed at OS / Middleware layer
  - Requires firmware modification!
- Android isolates app processes: “all apps are created equal”
  - Monitor not privileged enough!
- **Solution**: Combine monitor and app into “self-monitoring” app

---

Diagram:

1. Operating System
2. Monitor
3. Untrusted App
APPGUARD – CONCEPTUAL OVERVIEW

Implemented as stand-alone app:
→ easily deployable
**Inline Reference Monitoring**

- **Dynamic Access Control**
  - Prevent apps from accessing certain system resources
  - Revocation and re-granting of permissions

- **Fine-granular Security Policies**
  - Comprehensible for user
  - Expressive for developer

- **“Graceful degradation”**
  - Apps should not crash after access to restricted resource

- **No change to the OS**
  - Deployment as regular Android app (no root)
INLINE REFERENCE MONITORING

- **Goal**: Mediate security-relevant operations
  - Monitor program behavior at critical points
  - Instrument program to redirect control flow to the monitor
  - Take action based on security policy
    - Terminate program
    - Suppress operation

- Security-relevant operations
  - Function calls: Java Core API, Android API
  - Control flow redirection either at **caller-site** or **callee-site**

- Typically by bytecode modification
**Caller- vs. Callee-Site Rewriting**

```java
String s;
URL u;

s = "http://attacker.com/";
u = new URL(s);
u.openConnection();
```

**Untrusted App**

Application.main()

**System Library**

URL.openConnection()

... return connection;

Untrusted App

Application.main()

String s;
URL u;

s = “http://attacker.com/“;
u = new URL(s);
u.openConnection();
...

System Library

URL.openConnection()

Monitor

Monitor.checkConnection(url)

if (!connectionAllowed(url)) {
  System.exit();
}

Monitor.checkConnection(this);
...

return connection;
String s;
URL u;

s = "http://attacker.com/";
u = new URL(s);

Monitor.openConnection(u);

Monitor.openConnection(url)

if (connectionAllowed(url)) {
    return url.openConnection();
} else {
    System.exit();
}

... return connection;
CALLER- VS. CALLEE-SIDE REWRITING

**Caller-side**

- ✗ Many places to instrument
- ✗ Dynamically loaded code
- ☐ Reflection

- ✔ Possible in practice for end-users

**Callee-side**

- ✔ Few places to instrument
- ✔ Dynamically loaded code
- ✔ Reflection

- ✗ Impossible in practice for end-users
**AppGuard: Rewriter**

- **Rewriter**
  - Works directly on Dalvik executable (DEX) bytecode
  - Generates runtime monitor from policies and merges it into the target app
  - Identifies invocations of security-relevant methods within the target app’s bytecode
  - Rewrites target app to call into the monitor right before every invocation of a security-relevant method (caller-site rewriting)
  - Additional try-catch block allows monitor to suppress the security-relevant method call and return a mock value
**AppGuard: Rewriter**

**Original code**

```
TelephonyManager tm = getTelephonyManager();
String deviceId = tm.getDeviceId();

URL url = new URL(loc);
try {
    url.openConnection();
} catch (IOException) {
    // handle IOException
}
```

**After rewriting**

```
TelephonyManager tm = getTelephonyManager();
String deviceId;
try {
    Monitor.checkConnection(url);
    deviceId = tm.getDeviceId();
} catch (MonitorException e) {
    deviceId = e.mockValue();
}
```

```
URL url = new URL(loc);
try {
    url.openConnection();
} catch (IOException) {
    // handle IOException
}
```

```
URL url = new URL(loc);
try {
    Monitor.checkConnection(url);
    url.openConnection();
} catch (IOException) {
    // handle IOException
}
```

```java
} catch (MonitorException e) {
    deviceId = e.mockValue();
}
```

```java
} // no return value, ignore
```
APP GUARD: MANAGEMENT

- UI for rewriting apps on the phone
AppGuard: Management

- UI for rewriting apps on the phone

AppGuard will secure this application by inserting additional security checks into its code.

This requires three steps:

- Scan and secure application
- Uninstall existing application
- Install secured application

Android will prompt you to confirm the uninstallation and installation step.

Securing the application will result in all your settings and data of the application being lost!

Cancel    |    OK
**AppGuard: Management**

- UI for rewriting apps on the phone
- Policy configuration per app
  - Passed to target app via world-readable config file
  - Fine-grained configuration supported
APPGuard: Management

- UI for rewriting apps on the phone
- Policy configuration per app
  - Passed to target app via world-readable config file
  - Fine-grained configuration supported
**APP GUARD: MANAGEMENT**

- UI for rewriting apps on the phone
- Policy configuration per app
  - Passed to target app via world-readable config file
  - Fine-grained configuration supported
- Log of security-relevant events
  - Pushed via IPC from inlined monitor
**Case Studies**

- **Wetter.com**
  - Provides weather information & forecast
  - Displays advertisements
  - **Situation**
    - Retrieves weather data from [wetter.com](http://wetter.com)
    - Requests **INTERNET** permission for full Internet access
  - **Solution**
    - Selectively allow access to [wetter.com](http://wetter.com) servers only
    - No more advertisements displayed
**Case Studies**

**Twitter**

- **Mobile client for popular micro-blogging service**

- **Situation**
  - Automatically transfers contact data to Twitter servers without user’s knowledge or consent
  - Part of Twitter’s „find friends“ feature

- **Solution**
  - Block access to user’s contact data
  - Friends can still be added manually
Endomondo Sports Tracker

- Tracks your outdoor sport activities (running, cycling, etc...)
- Creates personal sports profile

**Situation**
-Leaks authentication token via **HTTP**

**Solution**
-Intercept **HTTP** connections and redirect to encrypted **HTTPS**
CASE STUDIES

(Evil) Tea Timer

- Simple timer app
- Requires INTERNET permission only

Situation
- Uploads user’s personal photos to public photo sharing site
- No permission required to access photos storage

Solution
- Block access to photo storage
APPGUARD: DISCUSSION

- Practical solution to a pressing security problem
  - Negligible runtime overhead (< 6%)
  - Reasonable rewriting time (5-60 seconds)
  - Deployed & widely adopted (~1 million downloads over 8 months)
- General purpose lightweight runtime instrumentation
  - Only minimal static rewriting (caller-site) necessary
- Limitations
  - Native code
  - Re-implementation
  - Stealth
CONCLUSION

- Mobile security a very active research area
  - Feature-rich smartphones and “appification” have induced security research on various (new) aspects
- Android’s open-source nature has made Android very attractive to security researchers
  - Many lessons learned for concurrent and future mobiles Oses
ACKNOWLEDGEMENTS

- Luca Davi (CASED/TU Darmstadt/ICRI-SC @ TU Darmstadt) for his overview slides
- Sascha Fahl (Uni Hannover) for sharing his CCS’12 and CCS’13 slides on SSL vulnerabilities
- Stephan Heuser (CASED/TU Darmstadt) and William Enck (NCSU) for publishing their slides on ASM and TaintDroid, respectively