Software Security Goes Mobile

Jacob West
CTO, Fortify Products
HP Enterprise Security

July 12, 2012
Motivation

Redefining the phone and the computer

Money: Beyond ringtones and 99¢ games
Landscape

What is mobile?

What matters?

Who cares?
Mobile Threats

Seven ways to hang yourself with Google Android
Parting Thoughts

Questions you can ask to begin improving your mobile security today
Motivation
Smartphones > Feature Phones

Source: Morgan Stanley Research
Smartphones > PCs

Source: Morgan Stanley Research
Page Views on the Rise

Source: Morgan Stanley Research
Smartphones Serve As Pocket PCs and Extend Desktop Experience

81% browsed the internet
77% used a search engine
68% used an app
48% watch videos

Smartphone Activities Within Past Week (Excluding Calls)

Source: The Mobile Movement Study, Google, Ipsos OTX MediaCT, Apr 2011
Base: Smartphone Users (501)

Q. Aside from making or receiving calls, which of the following activities, if any, have you done on your smartphone in the past week?
Mobile is an Emerging Point of Purchase

Have Purchased on Smartphone

- 45%
- 29%
- 28%
- 28%
- 17%

Source: Google/MMA, Global Perspectives: The Smartphone User & Mobile Marketer, June 2011
Q: Have you ever purchased a product or service over the Internet on your smartphone?
Mobile Opportunities

Please select the most important benefit that your organization ultimately expects to gain from current or future mobile solutions deployments (whether or not you are currently receiving those benefits)

N = 600, Source: IDC’s Mobile Enterprise Software Survey (2011)
Mobile Purchasers

% of purchasers

- Entertainment
- Electronics
- Clothing
- Wireless Service
- Travel
- Retail and Tech
- Beauty
- Jewelry

$300/year per user

Source: Google The Mobile Movement Study
Why Mobile Users Don’t Buy

- Security is #2 reason to avoid purchases

Source: Google/MMA, Global Perspectives: The Smartphone User & Mobile Marketer, June 2011
Base: Smartphone Users Who Have Not Made a Purchase on Device (US: 4444; UK: 1559; FR: 1853; DE: 1442; JP: 554)
Q. Why have you not made a purchase using your smartphone?
Mobile Landscape
What is Mobile?
Familiar Model

Device

Browser

Server
Same Ol’ Server
Client-Side Persistence

- Local data persistence
- Similar to HTML 5
- Invisible to users and always available
Mobile OS

- Benefit of hindsight
- Security features
  - Read-only stack
  - Data encryption
  - Permissions
- Confusing
  - Wait, permissions?
Can’t We All Get Along?

- Formal communication
  - Inter-application
  - Intra-application
  - With the OS

- A new trust boundary
What Matters?

Old
- Handling sensitive user and app data
- Environment and configuration
- Standbys like XSS and SQL injection

New
- Local storage (e.g. SD card)
- Communication (SMS, MMS, GPS)
- Security features (Privileges, crypto)
Mobile Threats
# Google Android Vulnerabilities

<table>
<thead>
<tr>
<th></th>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Intent Hijacking</td>
</tr>
<tr>
<td>2</td>
<td>Intent Spoofing</td>
</tr>
<tr>
<td>3</td>
<td>Sticky Broadcast Tampering</td>
</tr>
<tr>
<td>4</td>
<td>Insecure Storage</td>
</tr>
<tr>
<td>5</td>
<td>Insecure Network Communication</td>
</tr>
<tr>
<td>6</td>
<td>SQL Injection</td>
</tr>
<tr>
<td>7</td>
<td>Promiscuous Privileges</td>
</tr>
</tbody>
</table>
Google Android Vulnerabilities

**Intent Hijacking**

**Description:** Malicious app intercepts an intent bound for another app to compromise data or alter behavior

**Cause:** Implicit intents (do not require strong permissions to receive)

**Fix:** Explicit intents and receiver permissions
Google Android Vulnerabilities

Intent Hijacking | Intent Spoofing | Sticky Broadcast Tampering | Insecure Storage | Insecure Network Communication | SQL Injection | Promiscuous Privileges

**IMDb App**

- **Showtime Search**
  - Implicit Intent Action: `willUpdateShowtimes`

- **Results UI**

  Handles Actions: `willUpdateShowtimes`, `showtimesNoLocationError`
Google Android Vulnerabilities

- Intent Hijacking
- Intent Spoofing
- Sticky Broadcast Tampering
- Insecure Storage
- Insecure Network Communication
- SQL Injection
- Promiscuous Privileges
Google Android Vulnerabilities

IMDb App
Showtime Search
Implicit Intent Action: willUpdateShowtimes

Results UI

Handles Actions: willUpdateShowtimes, showtimesNoLocationError

Eavesdropping App
Malicious Receiver

Handles Actions: willUpdateShowtimes, showtimesNoLocationError
Description: Malicious app spoofs a legitimate intent to inject data or alter behavior

Cause: Public components (necessary to receive implicit intents)

Fix: Explicit intents and receiver permissions
    Sensitive operations in private components
Google Android Vulnerabilities

Intent Hijacking  Intent Spoofing  Sticky Broadcast Tampering  Insecure Storage  Insecure Network Communication  SQL Injection  Promiscuous Privileges

Spoofing App

Action: showtimesNoLocationError

Malicious Component

IMDb App

Showtime Search  Results UI

Handles Actions: willUpdateShowtimes, showtimesNoLocationError
Google Android Vulnerabilities

- Intent Hijacking
- Intent Spoofing
- Sticky Broadcast Tampering
- Insecure Storage
- Insecure Network Communication
- SQL Injection
- Promiscuous Privileges

- Showtimes

Date & Location

Thursday, June 23

Current Location

Please specify a location
No showtimes were found for the selected date and location.
# Google Android Vulnerabilities

<table>
<thead>
<tr>
<th>Intent Hijacking</th>
<th>Intent Spoofing</th>
<th>Sticky Broadcast Tampering</th>
<th>Insecure Storage</th>
<th>Insecure Network Communication</th>
<th>SQL Injection</th>
<th>Promiscuous Privileges</th>
</tr>
</thead>
</table>

**Description:** Persistent intents can be accessed and removed by malicious apps

**Cause:** BROADCAST_STICKY allows to full access to any sticky broadcasts

**Fix:** Explicit, non-sticky broadcasts and receiver permissions
Google Android Vulnerabilities

Sticky Broadcasts (intents)

SB1

SB2

SB3

Malicious App
Requests
BROADCAST_STIC
KY Permission

Victim App
Receiver
(expects SB2)
**Google Android Vulnerabilities**

**Description:** Local storage visible to attackers and can compromise sensitive data

**Cause:** Local files are world-readable and persist

**Fix:**
- Use SQLite or internal storage for private data
- Encrypt the data (keep keys off the SD)
Google Android Vulnerabilities

- Kindle app saves e-books (.mbp and .prc) in a folder on the SD card
  - Depending on DRM, accessible to other apps
  - Saves covers of books (privacy violation)
  - Folder is retained after uninstall of app
Google Android Vulnerabilities

Description: Unencrypted channels can be intercepted by attackers sniffing network

Cause: Non-HTTPS WebView connections

Fix: Send sensitive data only over encrypted channels
Google Android Vulnerabilities

- Intent Hijacking
- Intent Spoofing
- Sticky Broadcast Tampering
- Insecure Storage
- Insecure Network Communication
- SQL Injection
- Promiscuous Privileges

Twitter: Tweets are sent in the clear

Google Android Vulnerabilities

Facebook: Despite ‘fully encrypted’ option on the Web, mobile app sends in the clear

![Image of Facebook logo and TCP stream content]

- Intent Hijacking
- Intent Spoofing
- Sticky Broadcast Tampering
- Insecure Storage
- Insecure Network Communication
- SQL Injection
- Promiscuous Privileges
Google Android Vulnerabilities

Description: Malicious users alter or view (query string injection) database records

Cause: Untrusted data used to construct a SQL query or clause

Fix: Parameterized queries
c = invoicesDB.query(
    Uri.parse(invoices),
    columns,
    "productCategory = " +
        productCategory + "" and
        customerId = "" + customerId + ",
    null, null, null,
    "" + sortColumn + ",
    null
);
Google Android Vulnerabilities

productCategory = Fax Machines
customerID = 12345678
sortColumn = price

Select * from invoices
where productCategory = 'Fax Machines'
and customerID = '12345678'
order by 'price'
Google Android Vulnerabilities

Intent Hijacking  Intent Spoofing  Sticky Broadcast Tampering  Insecure Storage  Insecure Network Communication  SQL Injection  Promiscuous Privileges

\[
\text{productCategory} = \text{Fax Machines}' \; \text{or} \; \text{productCategory} = "\\n\text{customerID} = 12345678 \\
\text{sortColumn} = " \text{order by} \; \text{'price}'
\]

\[
\text{select} \; \ast \; \text{from invoices} \\
\text{where} \\
\text{productCategory} = '\text{Fax Machines}' \; \text{or} \; \text{productCategory} = " \\
\text{and customerID} = '12345678' \; \text{order by} \; " \\
\text{order by 'price'}
\]
Google Android Vulnerabilities

- Intent Hijacking
- Intent Spoofing
- Sticky Broadcast Tampering
- Insecure Storage
- Insecure Network Communication
- SQL Injection
- Promiscuous Privileges

```
c = invoicesDB.query(
    Uri.parse(invoices),
    columns,
    "productCategory = ? and customerID = ?",
    {productCategory, customerID},
    null,
    null,
    null,
    "sortColumn = ?",
    sortColumn
);
```
## Google Android Vulnerabilities

<table>
<thead>
<tr>
<th>Vulnerability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intent Hijacking</td>
</tr>
<tr>
<td>Intent Spoofing</td>
</tr>
<tr>
<td>Sticky Broadcast Tampering</td>
</tr>
<tr>
<td>Insecure Storage</td>
</tr>
<tr>
<td>Insecure Network Communication</td>
</tr>
<tr>
<td>SQL Injection</td>
</tr>
<tr>
<td>Promiscuous Privileges</td>
</tr>
</tbody>
</table>

**Description:** Extra permissions permit privilege escalation and desensitizes users

**Cause:** Deputies, Artifacts from testing, Confusion (inaccurate/incomplete resources)

**Fix:** Identify unnecessary permissions
Google Android Vulnerabilities

User App

Does NOT need CAMERA permission

Wants Picture

Implicit Intent Action: IMAGE_CAPTURE

Camera App

Needs CAMERA permission

Takes Picture

Handles Action: IMAGE_CAPTURE

Intent Hijacking | Intent Spoofing | Sticky Broadcast Tampering | Insecure Storage | Insecure Network Communication | SQL Injection | Promiscuous Privileges
Google Android Vulnerabilities

- Third hit on Google search

It broadcasts whenever you connect or disconnect from Wifi, in other words, Wifi State.

You can do it using the following intent-filters:
- android.net.wifi.WIFI_STATE_CHANGED
- action android:name="android.net.wifi.STATE_CHANGE"
- android.net.wifi.supplicant.CONNECTION_CHANGE

Which needs the following permission:
- uses-permission android:name="android.permission.ACCESS_WIFI_STATE"

Not true for android.net.wifi.STATE_CHANGE

http://stackoverflow.com/questions/2676044/broadcast-intent-when-network-state-has-changend
## Empirical Results: DEFCON ‘11

<table>
<thead>
<tr>
<th>Vulnerability Type</th>
<th>% of Apps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Intent Hijacking</td>
<td>50%</td>
</tr>
<tr>
<td>2. Intent Spoofing</td>
<td>40%</td>
</tr>
<tr>
<td>3. Sticky Broadcast Tampering</td>
<td>6%</td>
</tr>
<tr>
<td>4. Insecure Storage</td>
<td>28%</td>
</tr>
<tr>
<td>5. Insecure Communication</td>
<td>N/A</td>
</tr>
<tr>
<td>6. SQL Injection</td>
<td>17%</td>
</tr>
<tr>
<td>7. Promiscuous Privileges</td>
<td>31%</td>
</tr>
</tbody>
</table>
Bonus: iGoat

- iGoat 1.0 documents 5 vulnerabilities
  - We find 15+

- iGoat 1.2 documents 7 vulnerabilities
  - We find 20+
Parting Thoughts
What Questions to Ask?

- What do your apps do and for whom?
- What platform(s) do your apps support and how?
- Who develops your apps and where?
- Is there an existing SDL for other development?
- Do you rely on platform providers or app distributors for any security assurance?
- Are mobile apps prompting back-end changes?
- Are your apps appropriately permissioned?