Mobile Security

Dominic Duggan
Stevens Institute of Technology
Based on materials by
Erika Chin, William Enck, Patrick McDaniels, Greg Morrisett

1

Cellphone OS Security vs OS Security

- Connected to critical infrastructure
  - Phone botnets
- Connected to people
  - Physical attacks
- Multiple Stakeholders
  - Provider, mfg, enterprise, 3rd-party app developer, end user, etc.
- Specific usage scenarios
  - Always with you
  - Only want to carry one (for business and personal)

2

Cellphone Stakeholders

- Mobile Network Operators (MNO)
  - Concerns: network abuse (e.g. tethering, SIM network locking)
- OS/Handset Manufacturers
  - Concerns: DRM for apps, bootloading, firmware tampering

3

Cellphone Stakeholders

- Enterprises
  - Consumerization of enterprise devices (Blackberry vs iPhone)
- App Developers
  - DRM, advertising to monetize app
- End Users
  - Privacy, identity theft, m-commerce, …

Sponsored Links

Adobe DRM Solutions
Read How Companies Protect IP While Collaborating Across Supply Chains.
Adobe.com/manufacturing

Drm
Remove DRM protection instantly from any music file. Free trial.
www.UltimateDRMRemoval.com

4

Controlling Security Policy

- Who controls security policy?
  - Providers: keep users from unlocking phone
  - Users: providers as a privacy risk
  - Third-party apps want DRM protection
  - Some users want to circumvent DRM
- Tension: stakeholders can be adversaries

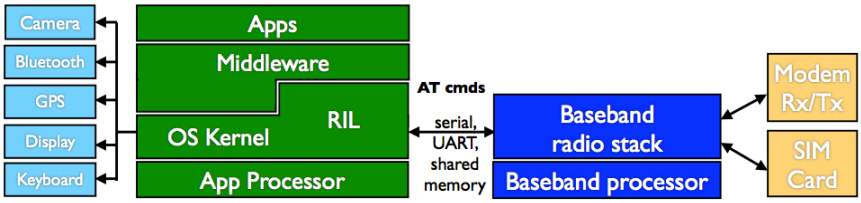
5

PHONE ARCHITECTURE

6

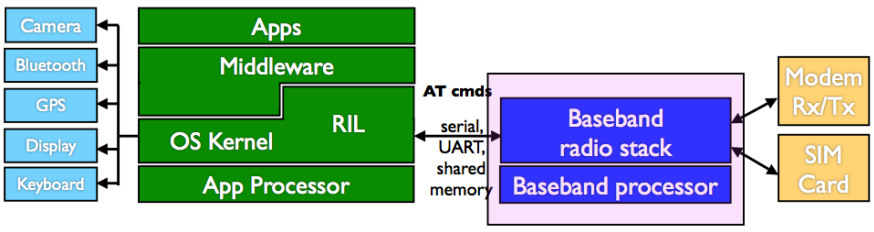
Phone Architecture

- Application processor
- Baseband processor
- May bundle hardware features
  - e.g. GPS, Bluetooth



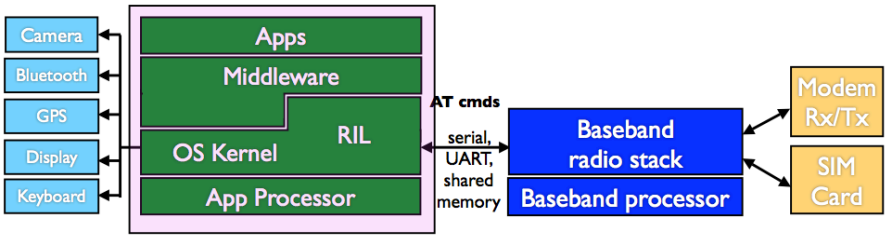
Baseband OS

- Responsible for controlling the radio
  - "radio firmware"
- Network registration, authentication, mobility
- Often, in-call audio bypasses app OS



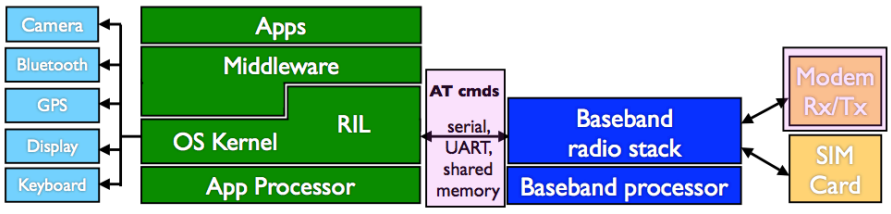
Application OS

- Traditionally dialer, PIM, etc., but now much more elaborate
  - iOS, Android, etc.
  - Middleware of some sort (e.g., Android Linux IPC, .NET, etc)



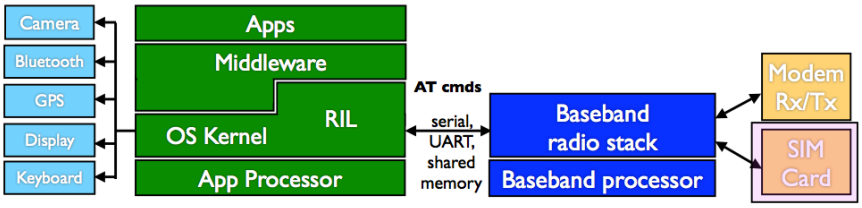
OS Communication

- Indirect access from app OS to cellular radio
- Radio Interface Layer (RIL) communicates with baseband
  - Modem-like interface (AT Commands - 3GPP TS 27.007)
  - Universal Asynchronous Recv/Xmit(UART) or shared mem
  - Userspace and kernelspace components



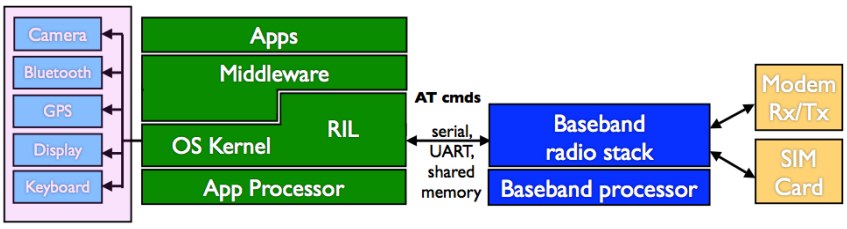
SIM Card

- Stores IMSI, authentication tokens, address book
  - Crypto operations performed on-chip
- Provider identifier ... used to "lock" a phone
  - Black market for SIM network unlocking phones
- Application OS accesses SIM using AT commands



Hardware Features

- Cameras and Bluetooth
- Other standard hardware features (part of SoC, e.g, MSM, OMAP):
  - GPS (A-GPS), accelerometer, compass
  - Video processing chips



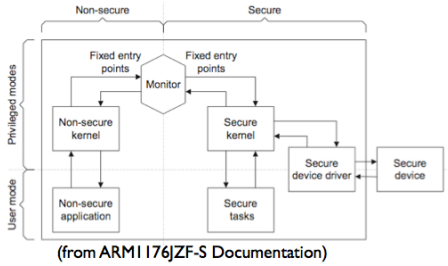
ARM Architecture

- Common in embedded systems
  - Low transister count
- Jazelle DBX (direct bytecode execution)
- Processes: *user* or *privileged*
- Domains: grouping of memory regions
  - Domains either a "client" or "manager"
  - fast switching w/out TLB flush
  - Access permissions specified at the domain level
  - Bits in page table (and TLB)
  - Execute Never (XN) bit restricts page execution
- Additional protection provided by TrustZone ...

13

ARM TrustZone

- Two virtual CPUs that execute on one physical
  - "Normal world" and "secure world".
  - TrustZone is typically only a 5% area overhead
- Virtual CPUs also allow fast and efficient (speed, power) data transfers



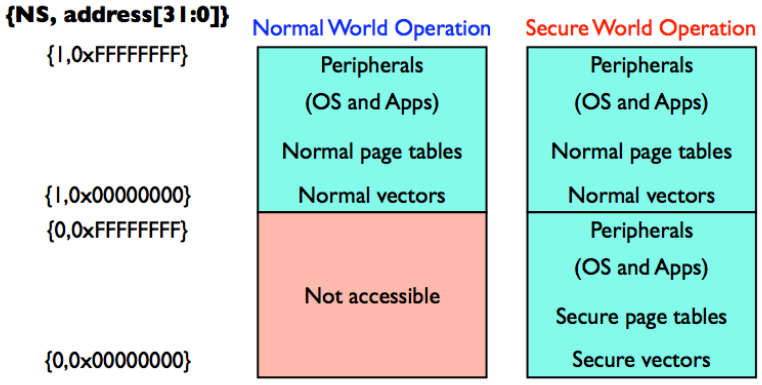
(from ARM1176JZF-S Documentation)

Why TrustZone?

- Protect access to keys (SIM lock functionality) or DRM (decode music)
- Arbitrary "secure services" defined with client stubs (using TrustZone API) in normal OS
- Also protects security sensitive hardware (e.g., secure storage)
  - SoC: harder to place a reader on data lines
  - Security state is propagated on the SoC bus

15

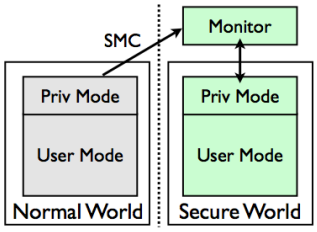
Memory Address Spaces



16

Mode Switching

- Three basic "modes": normal, secure, and monitor
- Normal OS initiates a change into Secure OS via an exception model
  - E.g., "secure monitor call" (SMC)
  - Puts the CPU in monitor mode, restores registers, branch to secure OS
- Each mode has own vector table
  - The monitor is itself a small OS



17

TrustZone Software

- Provides secure service and client APIs.
  - Incl. cryptography and secure storage facilities
- Client API: secure channel driver
  - Similar to DMA and controls secure world scheduling
- Software API allows new services to be created
  - e.g., for DRM or SIM-locking
  - Native services in SSDI
  - Java services in STIP (Small Terminal Interoperability Platform API), which includes a byte-code verifier

18

JAILBREAKING

19

What is Jailbreak?

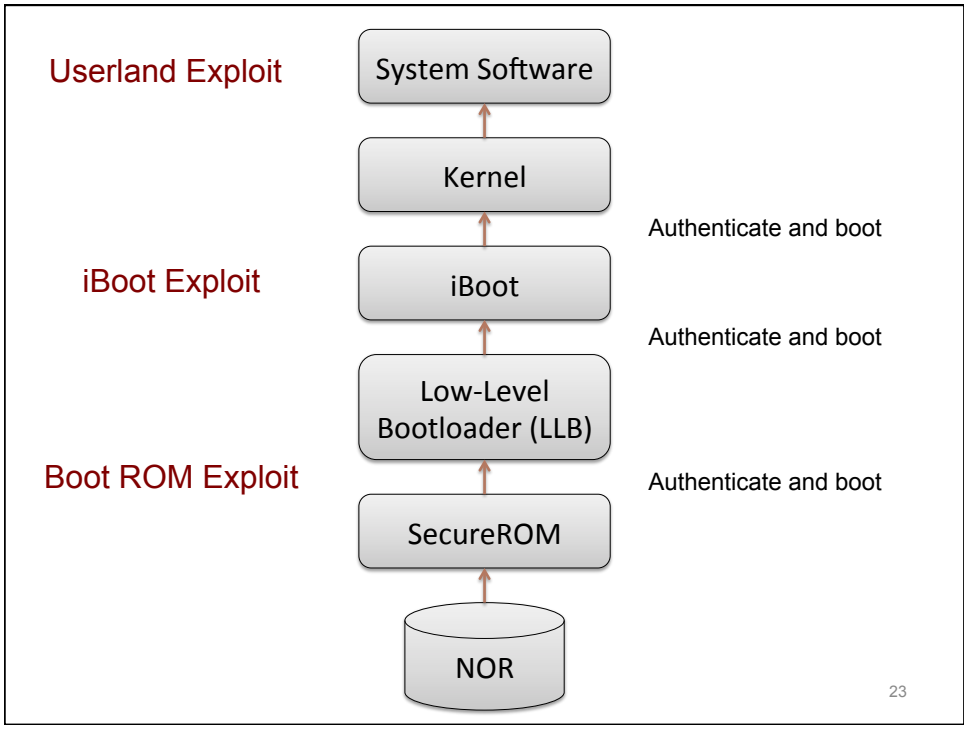
- iOS: Unix-like operating system
- Apple: walled garden
- All iPhone/iPad apps installed via iTunes
- Jailbreak: get read/write access to file system
  - Install apps using e.g. dpkg

21

Why to Jailbreak?

- Download additional extensions and themes
  - E.g. via Cydia
- Install non-Apple operating systems such as Linux
- Freely downgrade to certain version of firmware
- Cydia creator Jay Freeman estimates that over 10% of all iPhones are jailbroken

22



Tethered and Untethered Jailbreaks

- *Tethered:*
  - Fail signature check
  - Device hangs in DFU (recovery) mode
  - Start from "tethered" computer
- *Semi-tethered*
  - Can start device without computer
  - Boot from "tethered" computer to run add-ons
- *Untethered*
  - Device stays jailbroken after full reboot
  - iBoot bypassed normally

24

Userland Exploits

- *Userland:* code that is not running in kernel space but in user space e.g. Safari
- Tethered
- Example: kernel debugger in iOS (!)
  - Connect via serial lines
  - Return-oriented programming (ROP)
- Drawbacks :
  - File system access only
  - Easily fixed

25

iBoot Exploits

- Render low level control over iOS, iBoot
- Found in the iDevice's third bootloader (iBoot)
- Break the code signing mechanism of iBoot
  - Install and run arbitrary programs
  - User can enable device to accept custom firmware
  - Semi-tethered: If iBoot modified, LLB puts device into recovery mode

26

Boot ROM Exploits

- The lowest level control to the iOS
  - Completely bypass code signing
- iDevice's 1st bootloader: SecureROM
- Example: 0x24000 Segment Overflow
  - SecureROM did not check size of loaded LLB image
  - Change return address for SHA1
  - Disable signature checks and then load patched NOR firmware
- Cannot be patched easily by Apple
  - Need new chip

27

28

iOS Firmware Authentication

- SHSH Blob:
  - Signature HaSH
  - 128-byte RSA signature for firmware
- Exclusive Chip ID, or ECID
  - 16-digit hexadecimal number used to uniquely identify Apple iDevices
  - *Stop jailbreaking of future firmwares*

29

iOS Firmware Authentication

- Challenge: Firmware hash and ECID combined as a challenge key
- Response: (from Apple) is the SHSH Blob itself,
  - Digital signature required to validate the firmware

30

iOS Firmware Authentication

31

iOS Firmware Authentication

- During the restore process, users see "Verifying restore with Apple..."
- iTunes signs software and enforces protocol for iOS upgrade or restore
- iTunes contacts Apple servers
  - Generate signature, SHSH, just for your device
  - Only authorize latest firmware version
  - No downgrade available

32

Bypassing Firmware Authentication

33

Bypassing Firmware Authentication

- Static challenge key (ECID+SHSH)
- Replay attack to trick iTunes into validating an old firmware
  - Man-in-the-middle signature server
- Point iTunes to fake server by attaching a pair, like "126.192. 208.100 gs.apple.com" to local hosts
- Fake server responds with "on file" blob rather than active challenge/response
- *Key: Save your SHSH blobs!*
- Apple added *tickets* to prevent replay

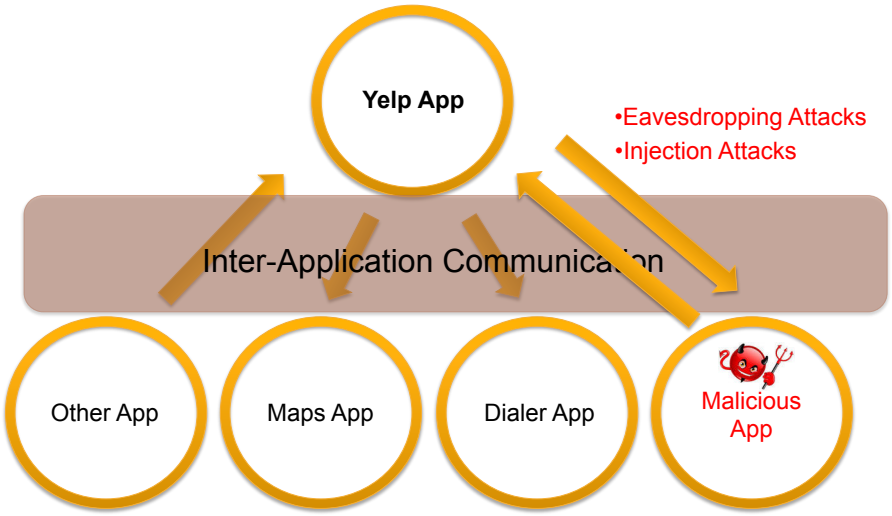
Legal Issues

- July 26, 2010, The U.S. Copyright Office has approved exemptions to the DMCA to legalize jailbreaking
- Apple can patch but cannot sue
- Unclear whether it is legal to traffic in the tools used to make jailbreaking easy
- Illegally shared paid App Store applications caused some strife within the jailbreaking community

INTER-APP EXPLOITS IN ANDROID

36

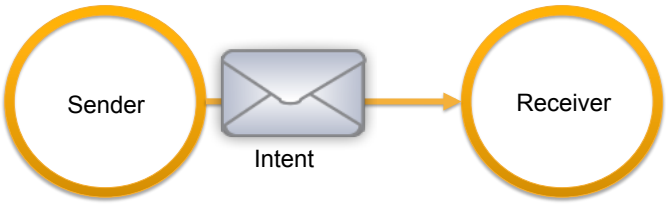
Inter-Application Communication



37

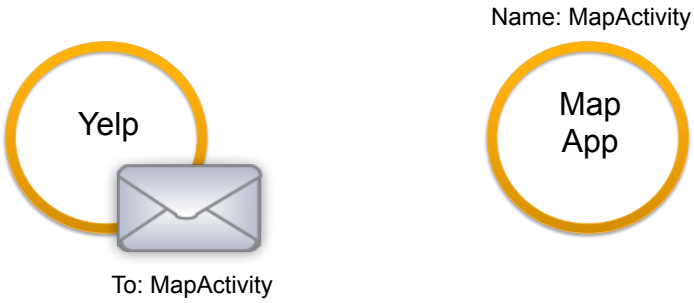
Intents

- Intents = Android IPC
- Intents sent between components



38

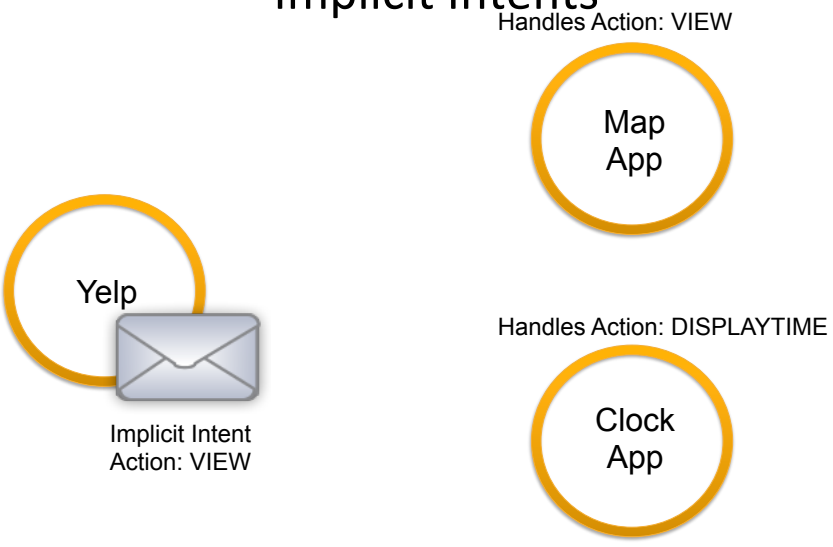
Explicit Intents



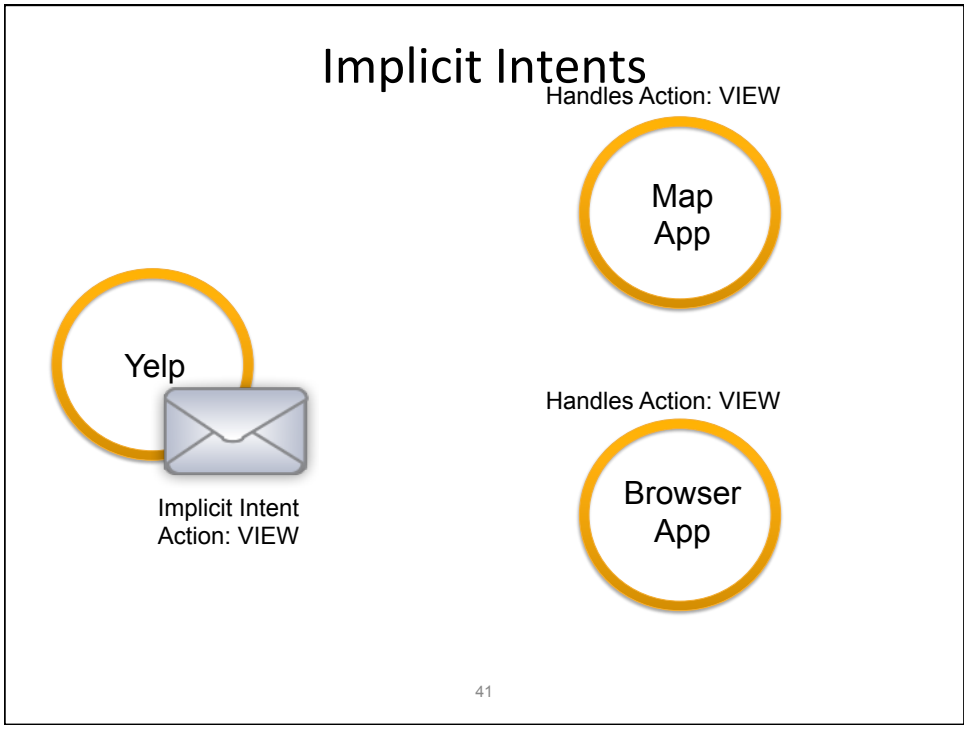
Only the specified destination receives this message

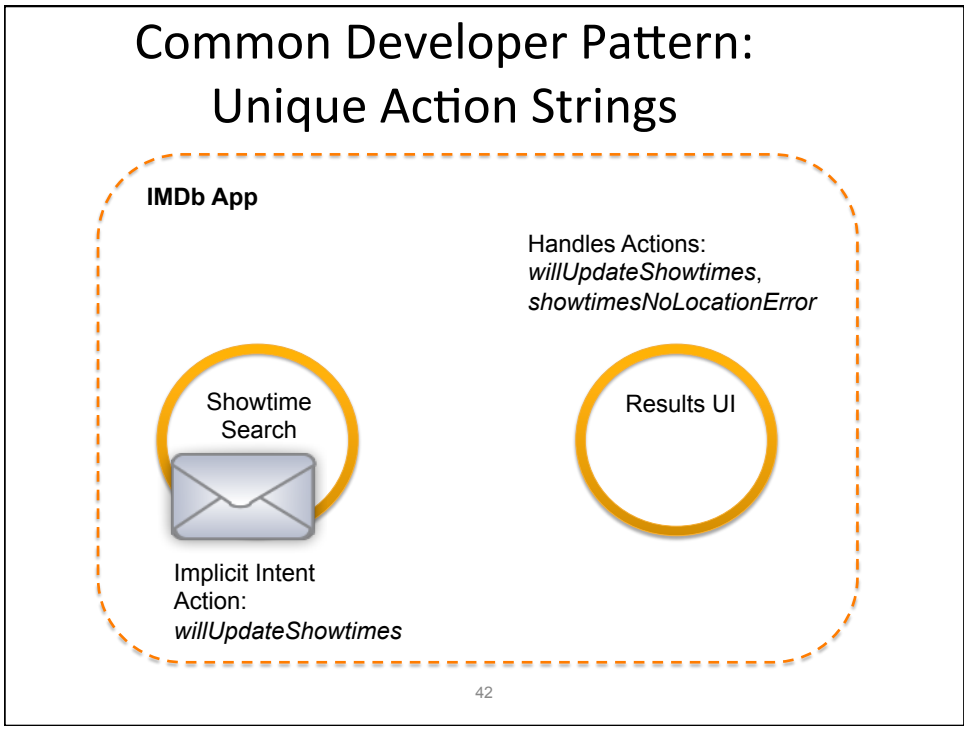
39

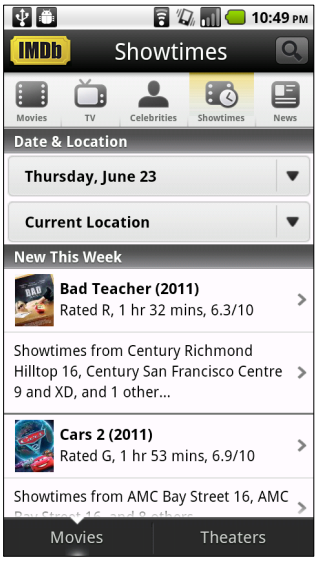
Implicit Intents



40

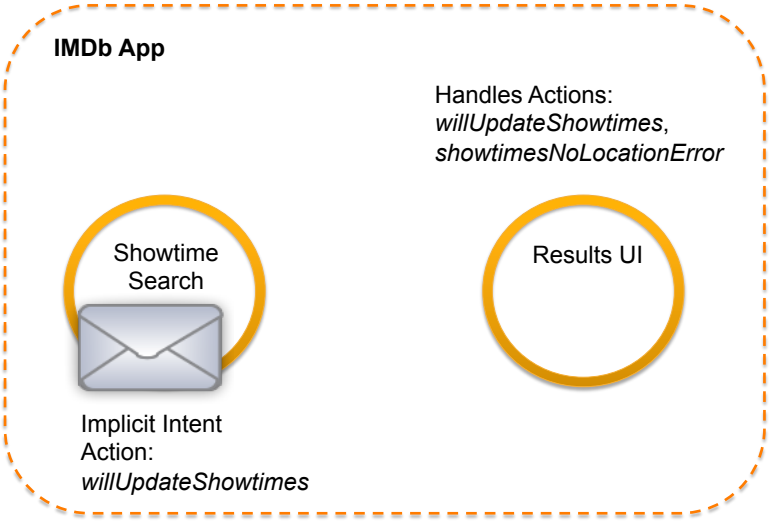

Implicit Intents
Handles Action: VIEW
Map App
Yelp
Implicit Intent
Action: VIEW
Handles Action: VIEW
Browser App
41


Common Developer Pattern:
Unique Action Strings
IMDb App
Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError
Showtime Search
Results UI
Implicit Intent
Action:
willUpdateShowtimes
42



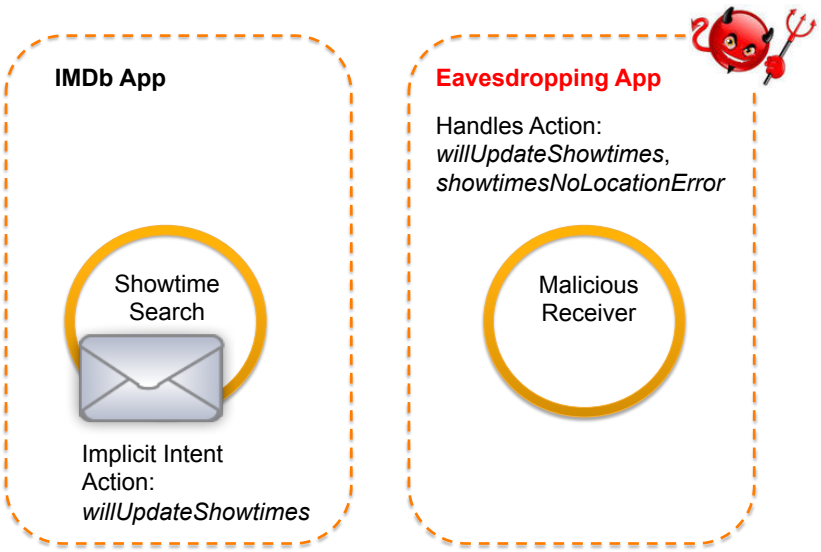
43

Common Developer Pattern: Unique Action Strings



44

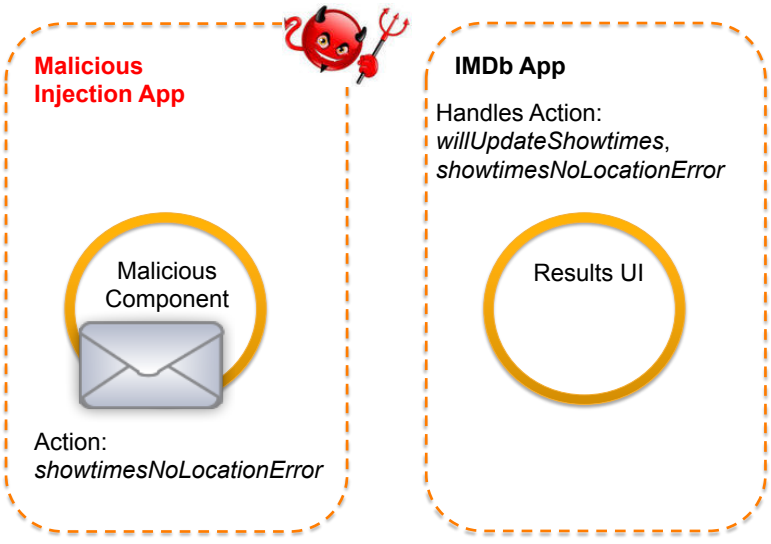
ATTACK #1: Eavesdropping



Sending Implicit Intents makes communication public

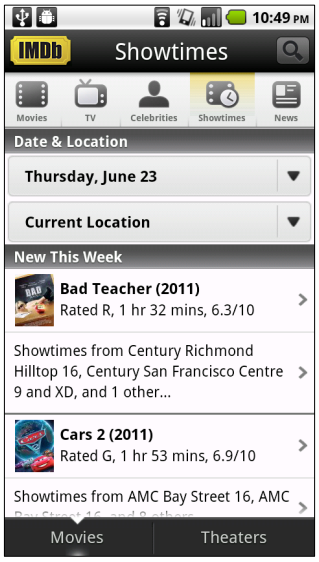
45

ATTACK #2: Intent Spoofing

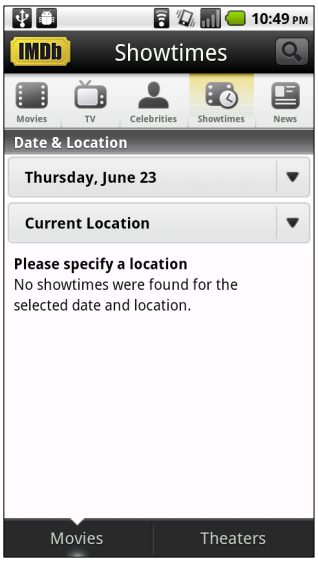


Receiving Implicit Intents makes the component public

46



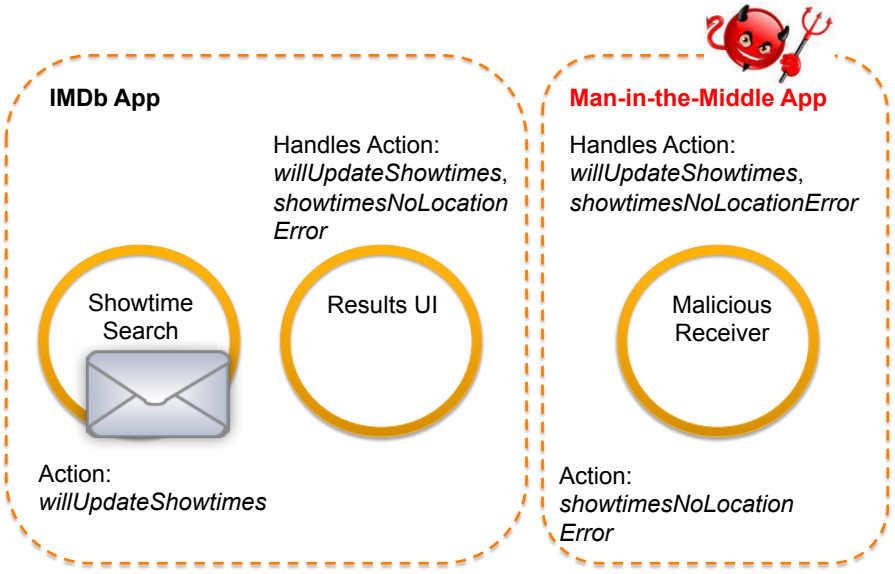
Typical case



Attack case

47

ATTACK #3: Man in the Middle



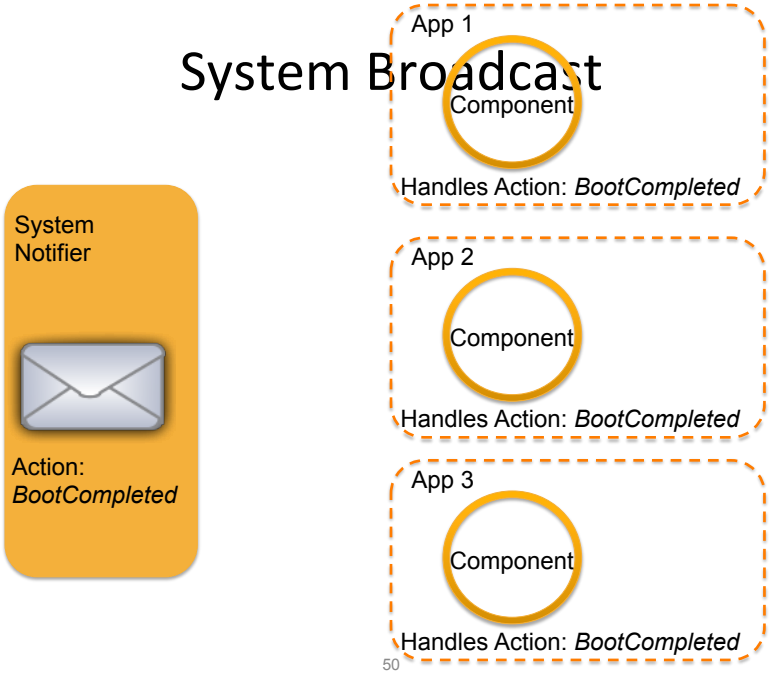
48

ATTACK #4: System Intent Spoofing

- System Broadcast
  - Event notifications sent by the system
  - Some can only be sent by the system
- Receivers become accessible to all applications when listening for system broadcast

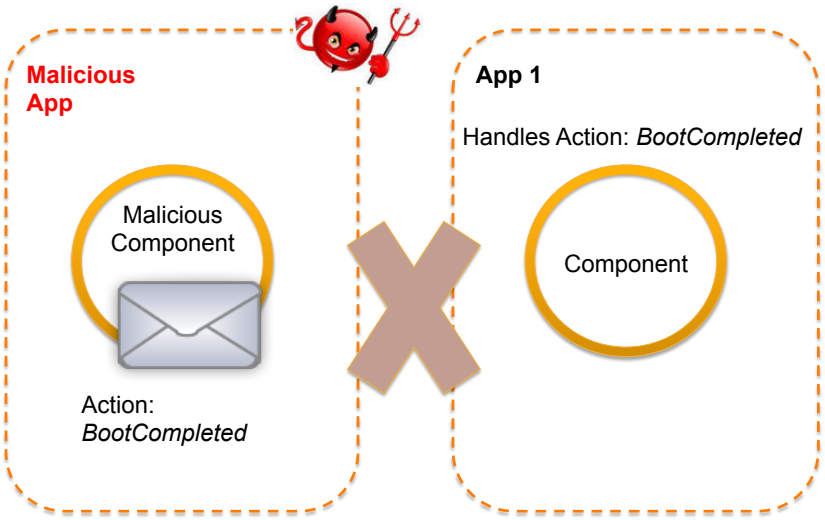
49

System Broadcast



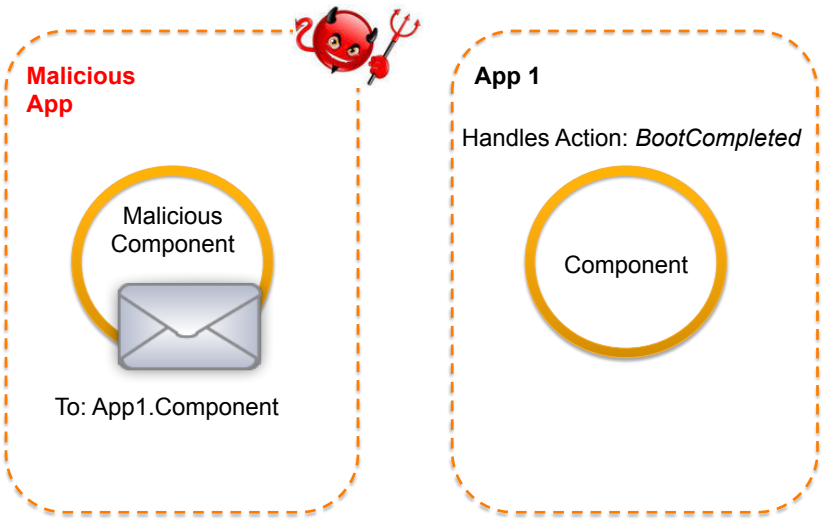
50

System Intent Spoofing: Failed Attack



51

System Intent Spoofing: Successful Attack



52

Explicit inter-app intent

- Explicit intent with class:

```
Intent intent = new Intent(context, MyClass.class);
```

- Implicit intent with intent filters:

```
Intent intent = new Intent();
intent.setAction("org.example.app.MyAction");
```

- Explicit intent with component name:

```
Intent intent = new Intent();
ComponentName componentName =
   new ComponentName("org.example.app", "MyAction");
intent.setComponent(componentName);
```

53

Real World Example: ICE App

- ICE App: Allows doctors access to medical information on phones
- Contains a component that listens for the *BootCompleted* system broadcast
- On receipt of the Intent, exits the application and locks the screen

54

Real World Example: ICE

55

Example Analysis

- **60%** of applications examined have at least 1 exploitable IPC vulnerability

Type	# of Warnings	# of Apps
Severe Vulnerability	34	12
Bad Practice	16	6
Spurious Warning	6	6

56

Recommendations

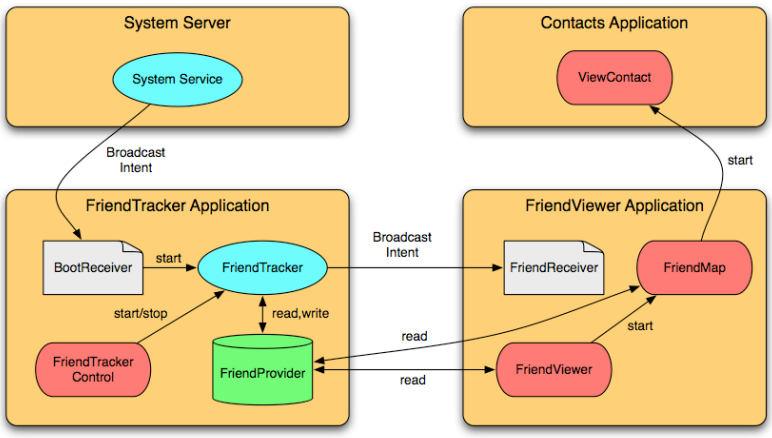
- Treat inter- and intra-application communication as different cases
- Prevent public internal communication
  - 21% of severe vulnerabilities
  - 63% of bugs due to bad practice
  - android:exported="false"
  - LocalBroadcastManager (v4 support library)
- Verify system broadcasts
  - 6% of severe vulnerabilities
  - 13% of bugs due to bad practice

57

ANDROID SECURITY

58

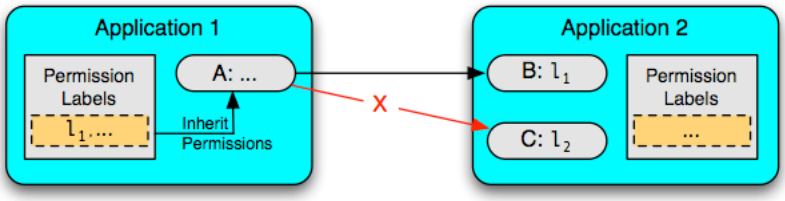
Example App Architecture



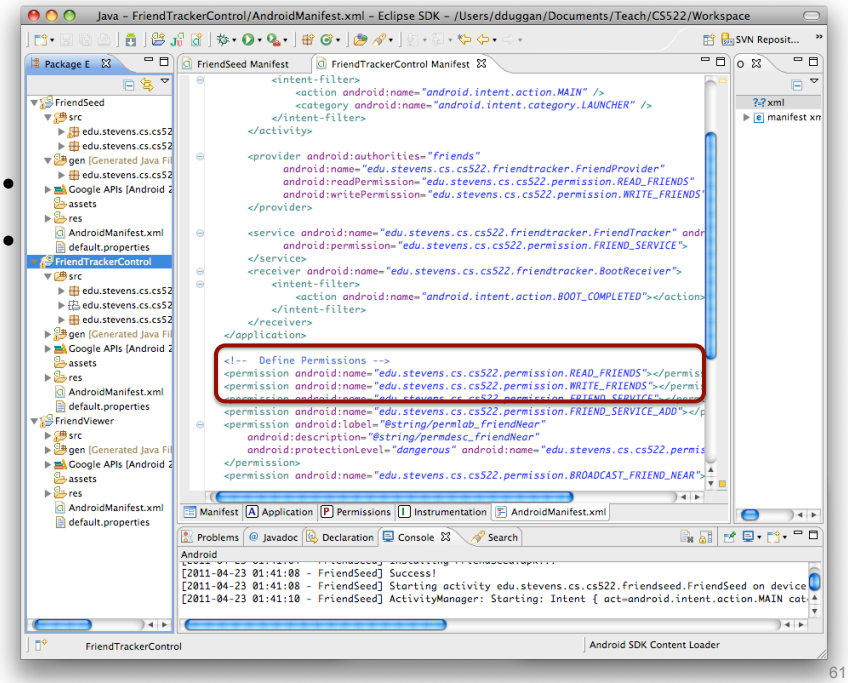
59

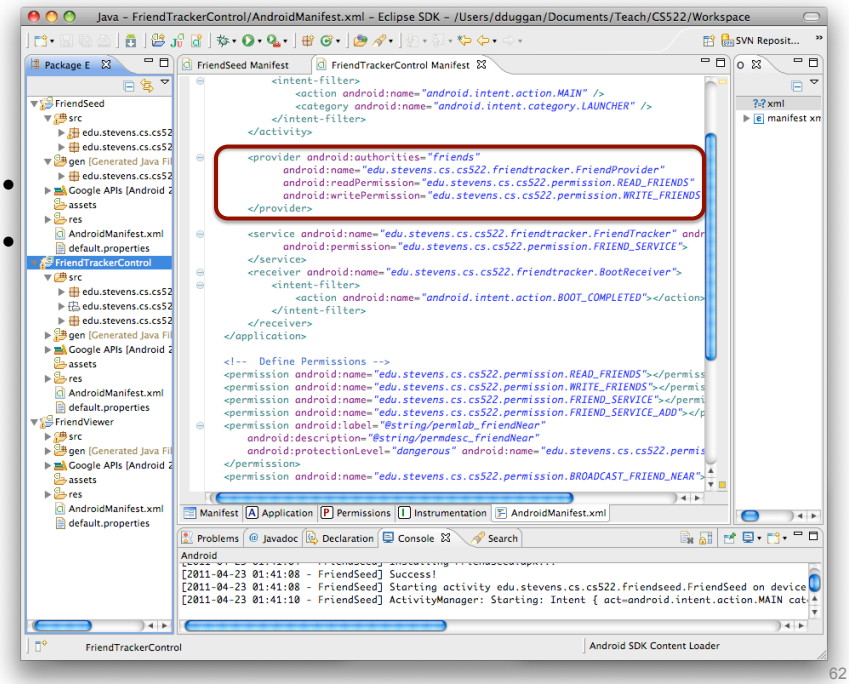
Android Security Model

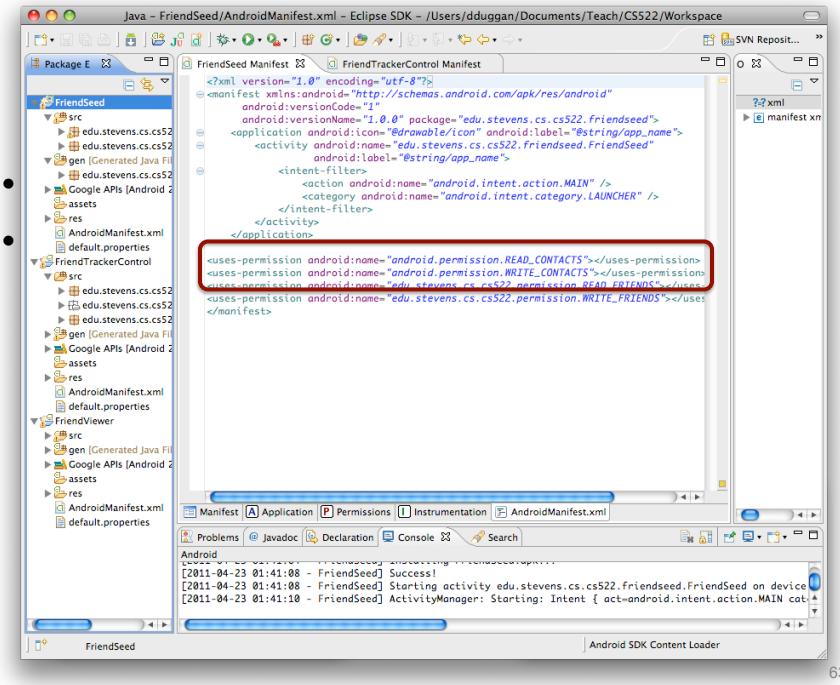
- Focus on Inter Component Communication (ICC)
- Manifest file defines an access control policy
  - Each component assigned access permission label
  - Each application requests a list of permission labels



60


Java – FriendTrackerControl/AndroidManifest.xml – Eclipse SDK – /Users/dduggan/Documents/Teach/CS522/Workspace
FriendSeed Manifest
FriendTrackerControl Manifest
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<provider android:authorities="friends"
android:name="edu.stevens.cs.cs522.friendtracker.FriendProvider"
android:readPermission="edu.stevens.cs.cs522.permission.READ_FRIENDS"
android:writePermission="edu.stevens.cs.cs522.permission.WRITE_FRIENDS"
</provider>
<service android:name="edu.stevens.cs.cs522.friendtracker.FriendTracker" andr
android:permission="edu.stevens.cs.cs522.permission.FRIEND_SERVICE">
</service>
<receiver android:name="edu.stevens.cs.cs522.friendtracker.BootReceiver">
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED"></action>
</intent-filter>
</receiver>
</application>
<!-- Define Permissions -->
<permission android:name="edu.stevens.cs.cs522.permission.READ_FRIENDS"></permis
<permission android:name="edu.stevens.cs.cs522.permission.WRITE_FRIENDS"></permi
<permission android:name="edu.stevens.cs.cs522.permission.FRIEND_SERVICE_ADD"></p
<permission android:label="@string/permlab_friendNear"
android:description="@string/permdesc_friendNear"
android:protectionLevel="dangerous" android:name="edu.stevens.cs.cs522.permis
</permission>
<permission android:name="edu.stevens.cs.cs522.permission.BROADCAST_FRIEND_NEAR">
Manifest
Application
Permissions
Instrumentation
AndroidManifest.xml
Problems
Javadoc
Declaration
Console
Search
Android
[2011-04-23 01:41:08 - FriendSeed] Success!
[2011-04-23 01:41:08 - FriendSeed] Starting activity edu.stevens.cs.cs522.friendseed.FriendSeed on device
[2011-04-23 01:41:10 - FriendSeed] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat
FriendTrackerControl
Android SDK Content Loader
61


Java – FriendTrackerControl/AndroidManifest.xml – Eclipse SDK – /Users/dduggan/Documents/Teach/CS522/Workspace
FriendSeed Manifest
FriendTrackerControl Manifest
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<provider android:authorities="friends"
android:name="edu.stevens.cs.cs522.friendtracker.FriendProvider"
android:readPermission="edu.stevens.cs.cs522.permission.READ_FRIENDS"
android:writePermission="edu.stevens.cs.cs522.permission.WRITE_FRIENDS"
</provider>
<service android:name="edu.stevens.cs.cs522.friendtracker.FriendTracker" andr
android:permission="edu.stevens.cs.cs522.permission.FRIEND_SERVICE">
</service>
<receiver android:name="edu.stevens.cs.cs522.friendtracker.BootReceiver">
<intent-filter>
<action android:name="android.intent.action.BOOT_COMPLETED"></action>
</intent-filter>
</receiver>
</application>
<!-- Define Permissions -->
<permission android:name="edu.stevens.cs.cs522.permission.READ_FRIENDS"></permis
<permission android:name="edu.stevens.cs.cs522.permission.WRITE_FRIENDS"></permi
<permission android:name="edu.stevens.cs.cs522.permission.FRIEND_SERVICE"></permi
<permission android:name="edu.stevens.cs.cs522.permission.FRIEND_SERVICE_ADD"></p
<permission android:label="@string/permlab_friendNear"
android:description="@string/permdesc_friendNear"
android:protectionLevel="dangerous" android:name="edu.stevens.cs.cs522.permis
</permission>
<permission android:name="edu.stevens.cs.cs522.permission.BROADCAST_FRIEND_NEAR">
Manifest
Application
Permissions
Instrumentation
AndroidManifest.xml
Problems
Javadoc
Declaration
Console
Search
Android
[2011-04-23 01:41:08 - FriendSeed] Success!
[2011-04-23 01:41:08 - FriendSeed] Starting activity edu.stevens.cs.cs522.friendseed.FriendSeed on device
[2011-04-23 01:41:10 - FriendSeed] ActivityManager: Starting: Intent { act=android.intent.action.MAIN cat
FriendTrackerControl
Android SDK Content Loader
62



63

Public and Private Components

- Components can be public or private.
  - Default is dependent on "intent-filter" rules
  - Manifest: "exported" attribute
- Why: Protect internal components
  - Especially useful if a "sub-Activity" returns a result
  - e.g., FriendMap Activity

```
<activity android:name="FriendMap" android:exported="false"></activity>
```

- Best Practice: Always set the "exported" attribute

64

Implicitly Open Components

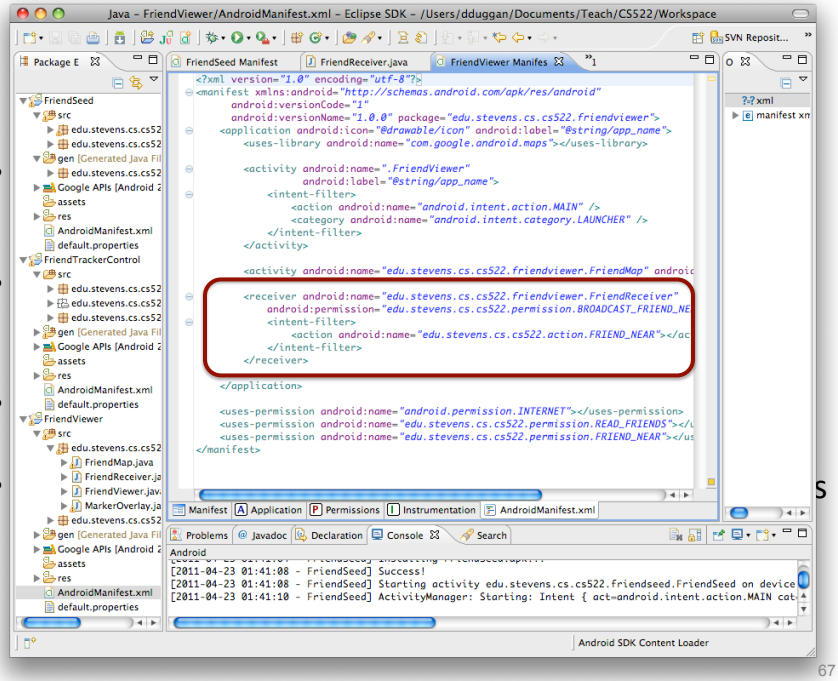
- Without permission checks, any component in any application can access a **public** component
- Why: "Global" access
  - e.g., the main Activity for an Application
- Implication: Unprivileged applications have access
  - e.g., FriendReceiver (spoof notification)
- Best Practice: Components without access permissions should be exceptional cases
- Consider splitting components

65

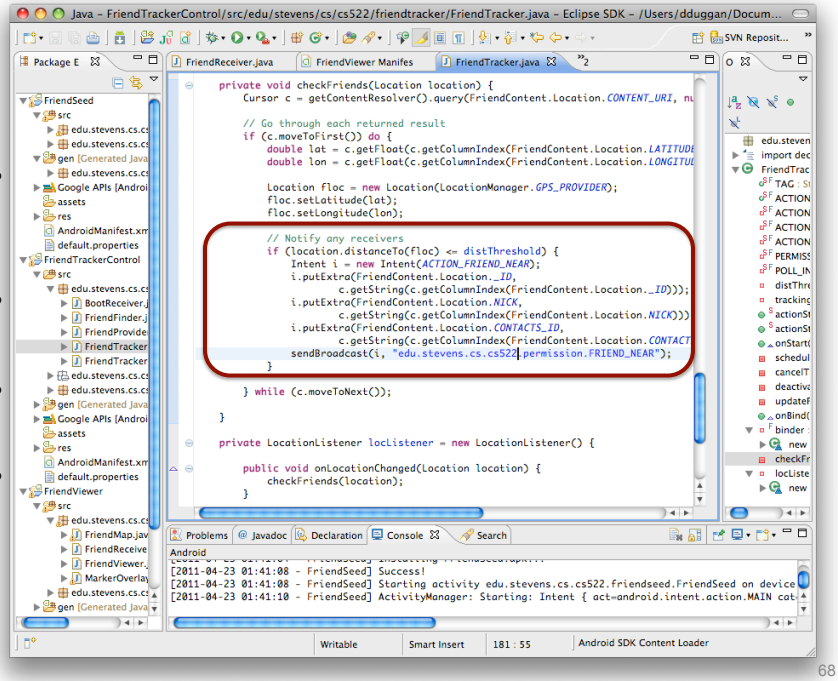
Intent Broadcast Permissions

- Code broadcasting an Intent can set an access permission
- Why: Define what applications can read broadcasts
  - e.g., FRIEND_NEAR message
- Best Practice: Always specify an access permission on Intent broadcasts
  - unless explicit destination
  - LocalBroadcastManager (v4 support library)

66


Java - FriendViewer/AndroidManifest.xml - Eclipse SDK - /Users/dduggan/Documents/Teach/CS522/Workspace
FriendSeed Manifest
FriendReceiver.java
FriendViewer Manifes
Manifest
Application
Permissions
Instrumentation
AndroidManifest.xml
Problems
Javadoc
Declaration
Console
Search
Android SDK Content Loader

67

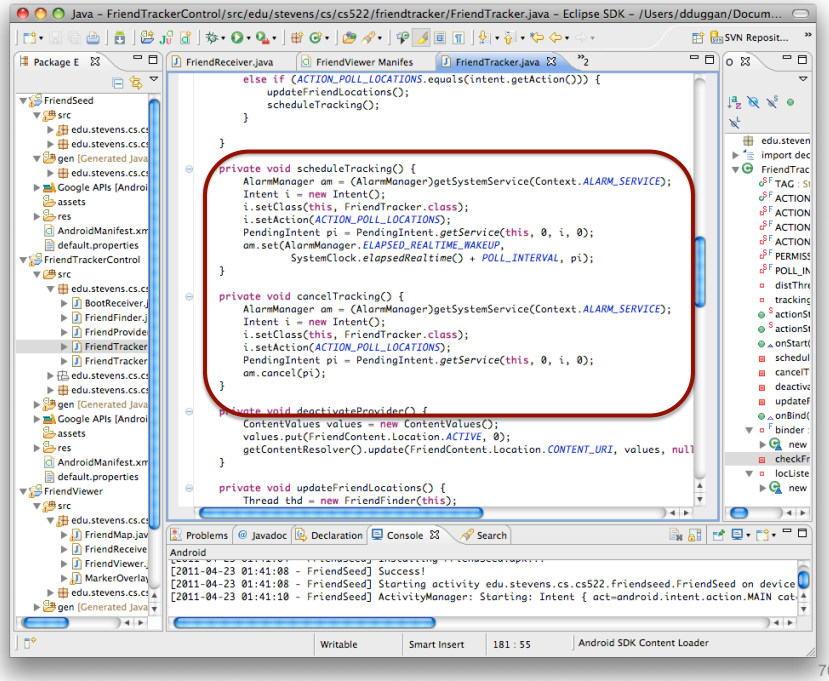

Java - FriendTrackerControl/src/edu/stevens/cs/cs522/friendtracker/FriendTracker.java - Eclipse SDK - /Users/dduggan/Docum...
FriendReceiver.java
FriendViewer Manifes
FriendTracker.java
Problems
Javadoc
Declaration
Console
Search
Writable
Smart Insert
181 : 55
Android SDK Content Loader

68

Pending Intents

- Allow another application to "finish" an operation
  - Execution occurs **in the originating application's "process" space**
- Why: Allows external applications to send to private components
  - E.g., Alarm, Location, Notification
  - E.g., Timer in FriendTracker Service (polling)
- Best Practice: Only use Pending Intents as "delayed callbacks"
  - Delegate to private Broadcast Receivers/Activities
  - Always fully specify the Intent destination

69

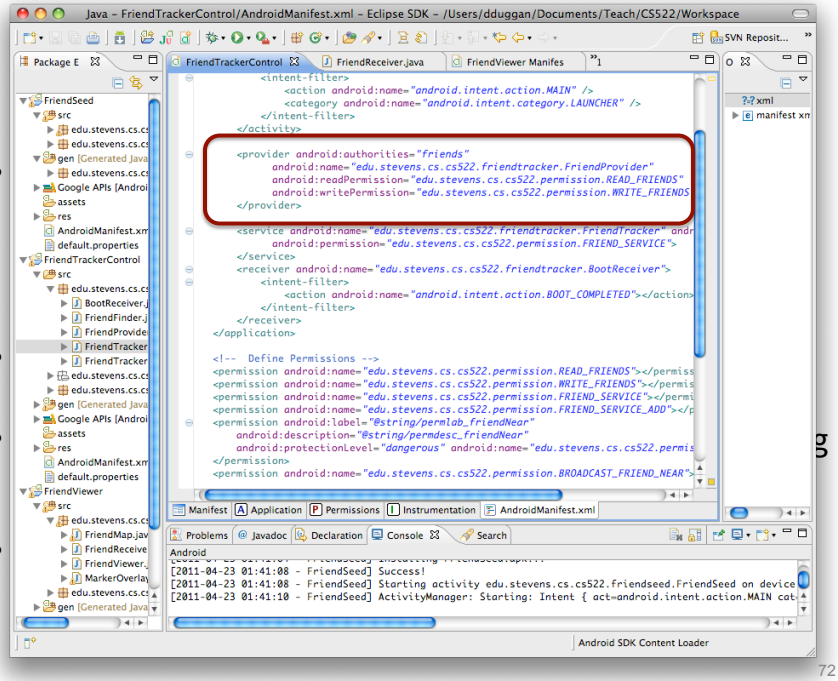


70

Content Provider Permissions

- Content Providers have two additional security features
  - Separate “read” and “write” access permission labels
- Content sharing need not be all or nothing
  - **URI permissions** allow record level delegation
  - must be allowed by Provider
- Best Practice: Always define separate read and write permissions.
  - Allow URI permissions when necessary

71

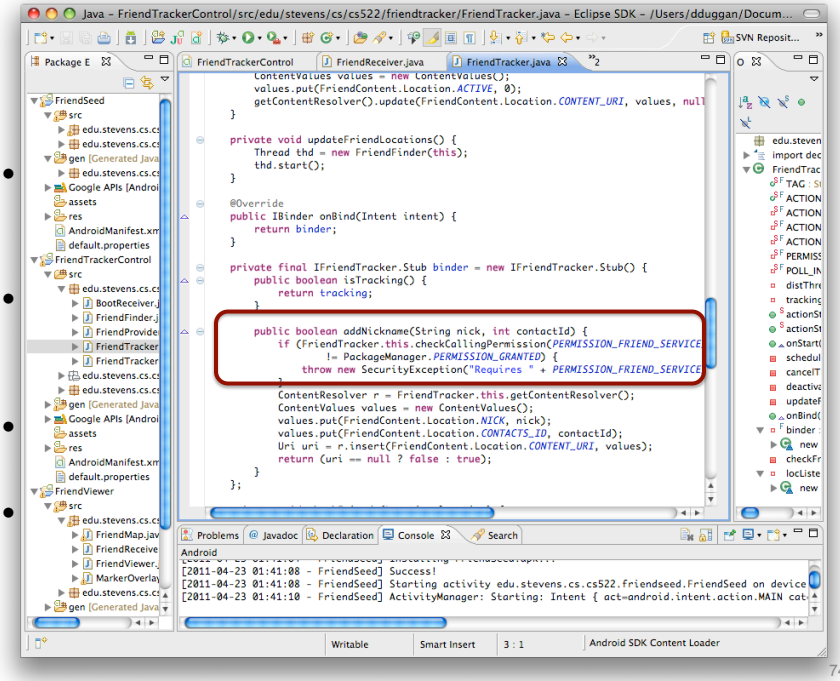


72

Service Hooks

- Exception: A component (e.g., Service) may arbitrarily invoke checkPermission()
- Why: Allows Services to differentiate access to specific methods
  - e.g., .addNickname() method of IFriendTracker
- Best Practice: Use checkPermission() to mediate "administrative" operations
- Alternatively, create separate Services

73



74

Protected APIs

- Exception: Permission labels to mediate access to certain resource APIs
  - e.g., `android.permission.INTERNET` for network connections

```
<uses-permission android:name="android.permission.INTERNET"></uses-permission>
```

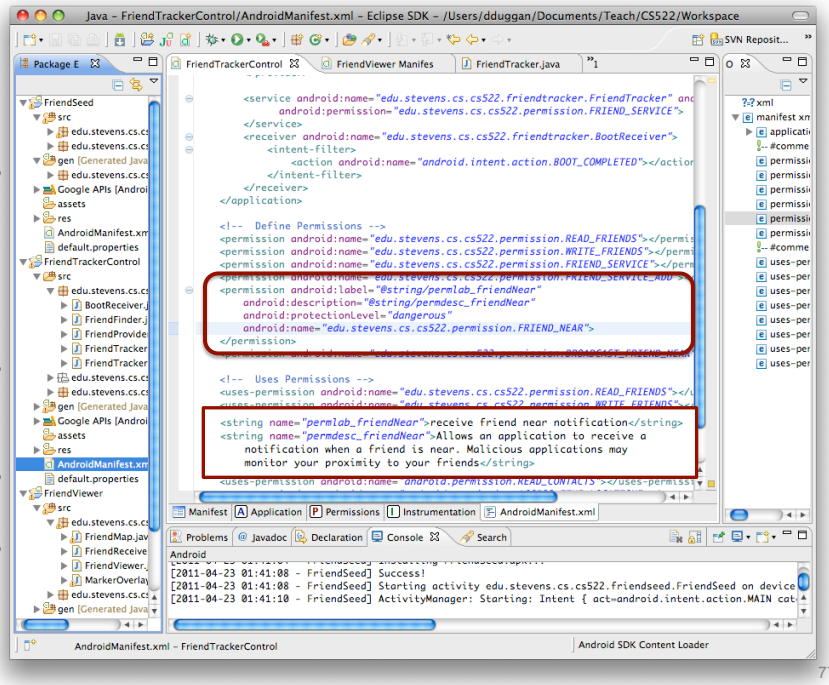
- Implication: Allows the system or a user to assess how "dangerous" an application may be
- Best Practices: Judiciously request permissions for protected APIs

75

Permission Protection Levels

- Permissions can be:
  - normal - always granted
  - dangerous - requires user approval
  - signature - matching signature key
  - signature or system - also system apps
- Why: Malicious apps may request harmful permissions
  - e.g., privacy implications of receiving FRIEND_NEAR
- Implication: Users may not understand implications
- Best Practice:
  - Use signature permissions for application "suites"
  - Use dangerous permissions otherwise
  - Include informative descriptions

76



77

Lessons in Defining Policy

- Relatively straightforward model with policy defined in the manifest file … but many exceptions
- Some thought is needed to avoid …
  - "Spoofing" Intent messages (FriendReceiver)
  - Privacy leaks (e.g., FRIEND_NEAR broadcast)
- The policy expands into the code
  - Broadcast permissions, `checkPermission()`, etc
- Keeping malicious applications from acquiring permissions is tricky

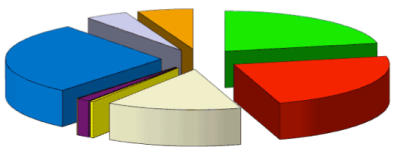
78

MOBILE MALWARE

79

Categorizing Mobile Malware

- Based on Target (e.g., base station, HLR) (Guo et al., HotNets'04)
- Based on Goal (e.g., DoS, identity theft) (Dagon et al., '04)
- Based on Infection Vector al., MobiSys'07)
  - Cellular Network
  - Bluetooth
  - Internet
  - USB/ActiveSync/Docking (crossover viruses)
  - Peripherals (e.g., SDCard)



80

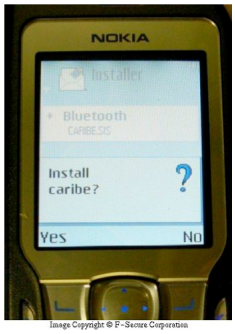
Early "Mobile" Malware

- 2000
  - Spanish Timofonica: sent annoying SMSs to random numbers via specific SMS gateway (PC/email based virus)
  - Phage: Palm OS virus that spread through IR and docking station. Originally benign (just spread), later versions destroyed apps files.
- 2001
  - PalmOS.Liberty.A: manual installation and execution resulting in deleted application files and databases.

81

Cabir (2004-2005)

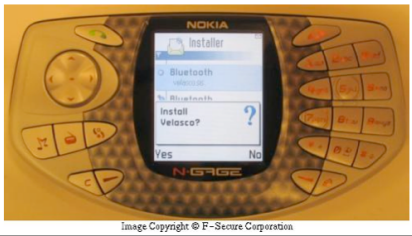
- The original Carbir.A virus spread via bluetooth for Symbian S60
  - Arrives as "caribe.sis", needs confirmation
  - Once installed, starts looking for new victims
  - Benign other than replication side-effects
- Mabir.A variant includes MMS replication
  - Replies to any SMS or MMS with an MMS message containing info.sis file



82

Lasco (2005)

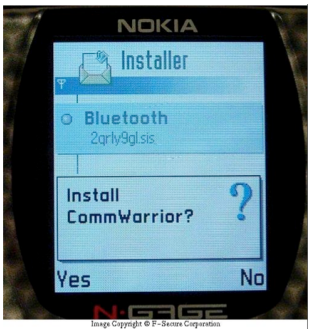
- Lasco.A is another Bluetooth/Symbian S60 virus (velasco.sis)
  - Once installed, looks for new victims
  - Based on same source code as Cabir.H
- Also replicates itself by infecting other SIS files
  - Infected files not automatically propagated



83

Commwarrior (2005)

- The Commwarrior worm infects Symbian S60 2nd edition
  - Spreads via both Bluetooth and MMS (variety of text) using the device's address book (thought to be 1st MMS)
  - Uses random file names
- Commwarrior.A is one of many variants
  - Bluetooth replication between 08:00 and 23:59
  - MMS replication between 00:00 and 06:59
  - Other variants (e.g., Commwarrior.B) does not use system clock for replication techniques



84

Skulls (2005)

- Skulls originated as a SIS Trojan (theme manager) for Nokia 7610 ("Extended theme.sis")
  - Replaces all application icons with "skull and cross bones"
  - Icons can't be used to start applications
- Symbian OS: files in the C: drive override similar filenames on the Z: drive (ROM)
  - Replaces build in applications with non-functional variants
  - Later variants try to disable AntiVirus applications (e.g., F-Secure)

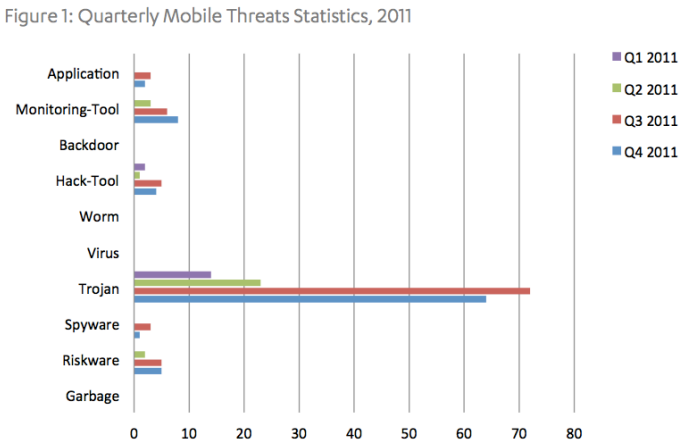
85

Other Trojans

- Drever (2005) is a Trojan pretending to be an update for Symbian OS
  - It's variants attempt disable various AntiVirus products (e.g., Kaspersky, Simworks, F-Secure) by overwriting their startup files.
- Locknut (2005) also pretends to be a Symbian OS update
  - Crashes a critical system service that prevents applications from launching (i.e., it is "locked").
  - Variations include versions of Cabir, but Locknut keeps Cabir from running too.

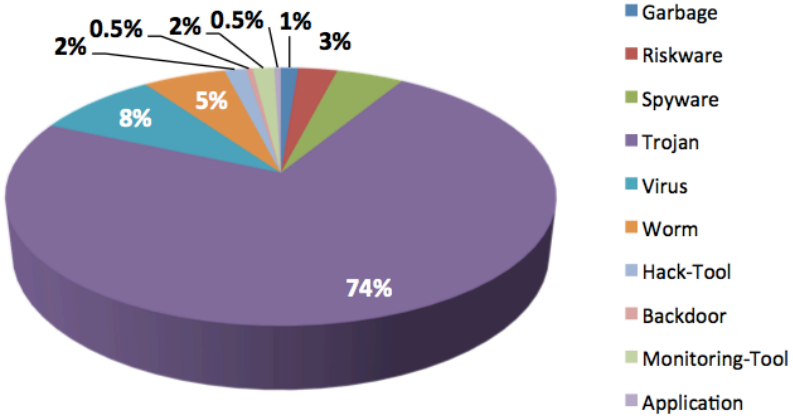
86

Malware by Type: 2011



87

Malware by Type: 2004-2011



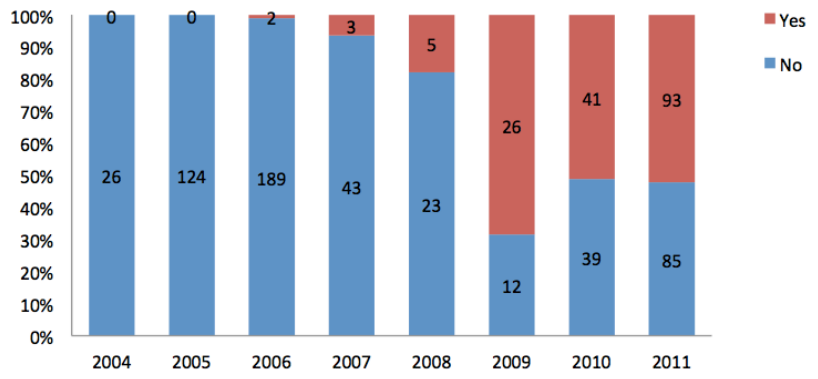
88

Malware by Type: 2004-2011

	Year								
Type	2004	2005	2006	2007	2008	2009	2010	2011	Total
Garbage			8						8
Riskware			1		1	8	1	10	21
Spyware			5	15	6		2	4	32
Trojan	11	105	160	23	13	24	47	136	519
Virus	14	19	17	6					56
Worm				2	8	6	22		38
Hack-Tool							4	9	13
Backdoor							3		3
Monitoring-Tool							1	14	15
Application								5	5
Total	25	124	191	46	28	38	80	178	710

89

Threats Motivated by Profit



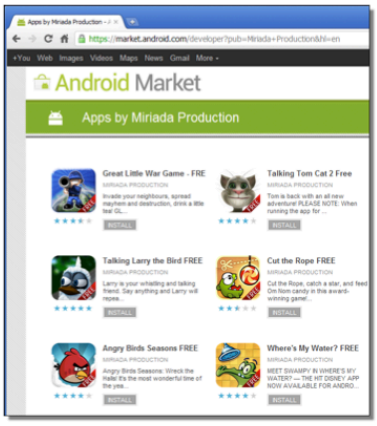
90

Android Malware

- EuropaSMS.A
  - Premium rate SMS-sending malware that targets European countries.
  - Two batches of application that come from different developers
    - Logastrod
    - Miriada Production
  - Developers managed to publish their malicious apps on the official Android Market

91

Android Malware



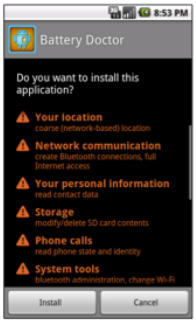
Destination SMS number	ISO country code	Country name
1121	AM	Armenia
1171	TJ	Tajikistan
1645	LT	Lithuania
17013	EE	Estonia
1874	LV	Latvia
4157	KG	Kyrgyzstan
4545	IL	Israel
7540	UA	Ukraine
7781	BY	Belarus
7781	RU	Russian Federation
7790	KZ	Kazakhstan
79067	GB	United Kingdom
8014	GE	Georgia
80888	DE	Germany
81185	FR	France
9014	AZ	Azerbaijan
90901599	CZ	Czech Republic
92525	PL	Poland

Applications by Miriada Productions were available on the Android Market for a short while

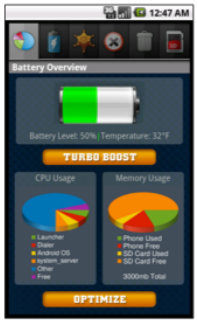
92

Android Malware

- FakeBattScar.A
  - Fake battery doctor
  - Secretly connects to remote website
  - Sends:
    - Device model
    - Locaton
    - Package name
    - IMEI
    - OS version
    - Browser version
    - Network operator



Requested permissions



Fake visual on the battery usage

93

Android Malware

- Fake Netflix login page
- Forwards credentials and uninstalls itself

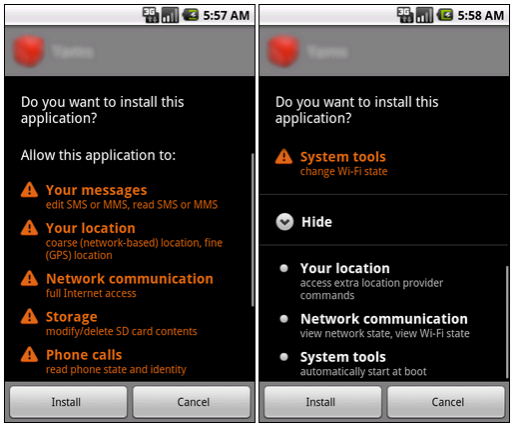
Fake login page

Error message that appears before the Trojan uninstall itself

94

Android Malware

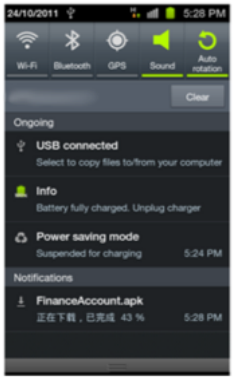
- DroidKungFu.C
  - Attempts to root the phone (exploits encrypted in the package)
  - Sends info (including phone memory, SD card memory) to remote server

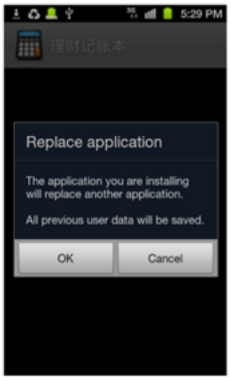


95

Android Malware

- DroidKungFu.E
  - Update attack

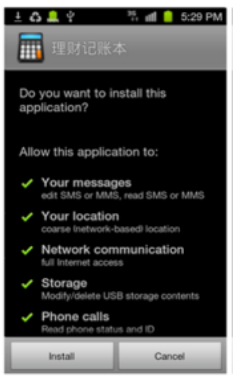


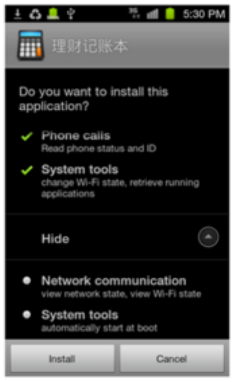


96

Android Malware

- DroidKungFu.E
  - Update attack





97