DeepDroid: Dynamically Enforcing Enterprise Policy on Android Devices

Xueqiang Wang\textsuperscript{1}, Kun Sun\textsuperscript{2}, Yuewu Wang\textsuperscript{1}, Jiwu Jing\textsuperscript{1}

\textsuperscript{1}Institute of Information Engineering, CAS
\textsuperscript{2}College of William and Mary

Mon, Feb. 9\textsuperscript{th}, 2015
Outline

- Introduction
- Related Work
- DeepDroid
- Evaluation
- Discussion
- Conclusion
Introduction

- Mobile devices are widely used for work purposes.
  - “51% of end users rely on smartphones to perform daily business activities.” ——Cisco
  - “Android hit 84% smartphone share in Q3 2014” ——IDC
Outline

- Introduction
- Related Work
- DeepDroid
- Evaluation
- Discussion
- Conclusion
Related Work

- Evolutionary support from Google
  - **Android Permission**
    - Coarse-grained
    - All-or-nothing
    - Lack of run-time configuration
  
  - **Device Administration APIs**
    - Only provide device-level control on password policy, camera, device wipe, etc.
    - Very limited interfaces (43 in KitKat VS 500+ in BlackBerry)
Related Work

- Evolutionary support from Google
  - Introduction of **SEAndroid**
    - Brings flexible MAC to Android
    - Middleware MAC has not been included, even in Android 5.0
    - Unavailable on legacy phones (58.7%<version 4.4)
  - Incorporation of **Knox APIs**
    - A large step towards “Android for Enterprise”
    - Introduces Knox features into AOSP except hardware-based ones
    - Unavailable on legacy phones (98.4%<version 5.0)
Related Work

- Possible solutions
  - **Device OEMs’ API**, e.g., SAFE, HTC, 3LM, LG.

- **Other solutions based on source code modification**
  - Extending permission, e.g., Compac[CODASPY’14].
  - Introducing MAC, e.g., FlaskDroid[USENIX Security’13], SEAndroid[NDSS’13].
  - Dynamic taint tracking, e.g., TaintDroid[OSDI’10].
  - Data shadowing, e.g., AppFence[CCS’11]

- **Portability** issue caused by tremendous source code modification.
Related Work

Possible solutions

- **Rewriting Android apps**
  - Dalvik bytecode rewriting, e.g., *I-ARM-Droid*[MoST’12]
  - Low-level libc interposition, e.g., *Aurasium*[USENIX Security’12]
  - On-the-phone instrumentation, e.g., *AppGuard*[TACAS’13]

- Require no modification to smartphone’s firmware and require no root access
- Lack of **isolation** between app and monitoring code.
Basic Idea - Middleware

- The system_server
  - centralized controller for middleware permissions

- The client-server architecture
  - system services, content providers, etc.

- Binder IPC
  - RPC to services/content providers
  - Intent
  - Broadcast
  - Messengers
  - ashmem
  - ...

- Dynamic Memory Instrumentation

configure middleware permissions

configure middleware behaviors
Basic Idea-Linux

- The zygote
  - centralized controller for Linux groups (a.k.a. Linux permissions)

- App works based on Linux system calls.

- Tracing System Calls

Activity Manager

Create process that can:
- read/write sdcard
- access network
- use camera
- read contacts

The zygote

The zygote is a centralized controller for Linux groups (a.k.a. Linux permissions). It allows apps to perform certain operations such as reading and writing to the sdcard, accessing the network, using the camera, and reading contacts. The apps work based on Linux system calls configured with permissions and behaviors.
DeepDroid-Middleware Permission

- *system_server* opens a few interfaces for middleware permission check.

- **Key:** Java method interposition
DeepDroid-Middleware Permission

Diagram showing the process of byte code execution, with classes.dex, byte code, interpreter, monitoring code, and libx.so.
DeepDroid-Middleware Behavior

- Transactions between apps and system services
  - ioctl(binderFd, BINDER_WRITE_READ, &bwr)
  - By tampering Global Offset Table (GOT) of libbinder.so

Transactions between apps and system services:

- app
- access to services
- Intent
- Broadcast
- Messenger
- ashmem
  
- upper layers

- system_server
- upper layers
  - libc.so
  - libbinder.so

- android.process.acore
- upper layers
  - libc.so
  - libbinder.so

- …

- supervise behaviors

Binder driver
DeepDroid-Middleware Behavior

- Synchronous invocation
  - E.g., getLastKnownLocation(), getDeviceId()
DeepDroid-Middleware Behavior

- Asynchronous invocation
  - One-way callbacks, e.g., onLocationChanged()

**system process**

- get a remote handle
- data callback

BC_TRANSACTION

- interfaces defined in aidl or in .java

counterpart recognition
1) servicemanager
2) IBinder instances

Primitives, IBinder, FD, Parcelable
DeepDroid-Linux Permission

- Configure Linux permissions (e.g., groups)

**Diagram:**
- system_server
- monitoring
- zygote
- app process

1: process creation request
2: recognize app
3: reset groups & track until setuid

---

--runtime-init
--setuid=10028
--setgid=10028
--setgroups=1015, 3003, 1006, 1007
android.app.ActivityThread
DeepDroid-Linux Behavior

- Configuration on Linux permissions is irreversible.
  - Tracking system calls of Application

```
syscall
ptrace
enforce
```

```
App Process
Monitoring Code
```

```
ptrace
syscall
```

enforce
DeepDroid-Properties

- Fine-grained access control
  - Both permission and behavior level

- Portable
  - Based on stable system architecture, e.g., system services, permission mechanism, binder.

- Dynamic instrumentation
  - Reduce the work on system customization
Outline

- Introduction
- Related Work
- DeepDroid
- **Evaluation**
- Discussion
- Conclusion
## Evaluated Resources

<table>
<thead>
<tr>
<th>Resource</th>
<th>Permission</th>
<th>Group</th>
<th>Permission Enforcement</th>
<th>Behavior Enforcement</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMEI</td>
<td>READ_PHONE_STATE</td>
<td>package</td>
<td>com.android.phone</td>
<td></td>
</tr>
<tr>
<td>Phone #</td>
<td>READ_PHONE_STATE</td>
<td>package</td>
<td></td>
<td></td>
</tr>
<tr>
<td>location</td>
<td>ACCESS_FINE_LOCATION</td>
<td>package</td>
<td>system_server</td>
<td></td>
</tr>
<tr>
<td>contacts</td>
<td>READ_CONTACTS</td>
<td>package</td>
<td>android.process.acore</td>
<td></td>
</tr>
<tr>
<td>camera</td>
<td>CAMERA</td>
<td>camera</td>
<td>mediaserver</td>
<td></td>
</tr>
<tr>
<td>account</td>
<td>GET_ACCOUNTS</td>
<td>package</td>
<td>system_server</td>
<td></td>
</tr>
<tr>
<td>logs</td>
<td>READ_LOGS</td>
<td>log</td>
<td>Process Creation</td>
<td>app process</td>
</tr>
<tr>
<td>network</td>
<td>INTERNET</td>
<td>inet</td>
<td>package/Process Creation</td>
<td></td>
</tr>
<tr>
<td>SMS</td>
<td>SEND_SMS</td>
<td>package</td>
<td>com.android.phone</td>
<td></td>
</tr>
</tbody>
</table>
## Evaluated Devices

<table>
<thead>
<tr>
<th>Device</th>
<th>Android OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus S (Samsung)</td>
<td>2.3.6</td>
</tr>
<tr>
<td>Sony LT29i</td>
<td>4.1.2</td>
</tr>
<tr>
<td></td>
<td>4.2.2</td>
</tr>
<tr>
<td>Galaxy Nexus (Samsung)</td>
<td>4.0</td>
</tr>
<tr>
<td>Samsung Galaxy Note II</td>
<td>4.1</td>
</tr>
<tr>
<td>Samsung Galaxy Note 3</td>
<td>4.3</td>
</tr>
<tr>
<td>Nexus 5 (LG)</td>
<td>4.4</td>
</tr>
<tr>
<td>Meizu MX II</td>
<td>Flyme 3.2</td>
</tr>
<tr>
<td></td>
<td>(4.2.1)</td>
</tr>
<tr>
<td>Huawei Honor 3c</td>
<td>4.2</td>
</tr>
</tbody>
</table>
Performance

Overhead of Sensitive RPC

- **phone_state**
  - Normal Mode (Success)
  - Normal Mode (Fail)

- **contacts**
  - Normal Mode (Success)
  - DeepDroid Mode (Success)
  - Normal Mode (Fail)
  - DeepDroid Mode (Fail)

- **SMS message**
  - Normal Mode (Success)
  - DeepDroid Mode (Success)
  - Normal Mode (Fail)
  - DeepDroid Mode (Fail)
Performance

Zygote Overhead (Time of startService)

- MX II
- LT29i
- Nexus S

- Normal Zygote
- Traced Zygote

ms

0
20
40
60
80
100
120
# Performance

## Quadrant Scores

<table>
<thead>
<tr>
<th></th>
<th>Normal Quadrant</th>
<th>Traced Quadrant</th>
</tr>
</thead>
<tbody>
<tr>
<td>MX II</td>
<td>2508.5</td>
<td>2507.6</td>
</tr>
<tr>
<td>LT29i</td>
<td>4653.8</td>
<td>4553.6</td>
</tr>
<tr>
<td>Nexus S</td>
<td>1750.0</td>
<td>1705.6</td>
</tr>
</tbody>
</table>

## CaffeineMark Scores

<table>
<thead>
<tr>
<th></th>
<th>Normal CaffeineMark</th>
<th>Traced CaffeineMark</th>
</tr>
</thead>
<tbody>
<tr>
<td>MX II</td>
<td>6367.2</td>
<td>6207.5</td>
</tr>
<tr>
<td>LT 29i</td>
<td>14125.5</td>
<td>13998.5</td>
</tr>
<tr>
<td>Nexus S</td>
<td>5982.8</td>
<td>5959.9</td>
</tr>
</tbody>
</table>
Discussion

- **Root access**
  - Required to instrument system components and trace *zygote*.
  - DeepDroid is a *self-contained* app and can be easily inserted as a system component.
  - DeepDroid carries little burden on vendor customization.

- **Compared to other solutions**
  - SEAndroid is enforced on Android 4.4.
  - Knox is fully supported only on some Samsung devices.
  - DeepDroid is based on stable architecture of Android, therefore, it can be easily adopted on *phones from other OEMs and legacy phones*. 
Discussion

- policy misuse
  - We used software-based scheme to protect policies.
  - On future devices, we can adopt some hardware-based schemes (e.g., TrustZone-based integrity checking scheme).
Outline

- Introduction
- Related Work
- DeepDroid
- Evaluation
- Discussion
- Conclusion
Conclusion

- We propose a dynamic security policy enforcement scheme named DeepDroid.

- DeepDroid enables fine-grained control on both permission and apps’ behavior.

- DeepDroid is relatively portable on different devices compared to direct system customization.
Thank You
References

- **Compac**[CODASPY’14]: “Compac: enforce component-level access control in android”
- **FlaskDroid**[USENIX Security’13]: “Flexible and Fine-Grained Mandatory Access Control on Android for Diverse Security and Privacy Policies”
- **SEAndroid**[NDSS’13]: “Security Enhanced (SE) Android: Bringing Flexible MAC to Android”
- **TaintDroid**[OSDI’10]: “TaintDroid: An Information-Flow Tracking System for Realtime Privacy Monitoring on Smartphones”
- **AppFence**[CCS’11]: “These aren't the droids you're looking for: retrofitting android to protect data from imperious applications”
- **I-ARM-Droid**[MoST’12]: “I-ARM-Droid: A Rewriting Framework for In-App Reference Monitors for Android Applications”
- **Aurasium**[USENIX Security’12]: “Aurasium: Practical Policy Enforcement for Android Applications”
- **AppGuard**[TACAS’13]: “AppGuard: Enforcing User Requirements on Android Apps”