Introduction

Application Fundamentals
Application architecture, measurement of a VM
- Memory
- Network
- Performance
Android Architecture
- Application
- Activity
- Service
- Content provider
- Broadcast receiver
- Handler
- Thread
- Context
- Lifecycle
- Android Studio
Embedded Android Profiler to suit our needs

Scope
- Code and analysis accuracy
- Dynamic analysis
- Static analysis
- Android source code
- Android SDK
- Memory usage
- Network usage
- Performance metrics
- Control flow analysis

Android Architecture
- Application
- Activity
- Service
- Content provider
- Broadcast receiver
- Handler
- Thread
- Context
- Lifecycle
- Android Studio
Embedded Android Profiler to suit our needs

Implementation

TraceDroid Analysis Platform
Automated analysis
- Static analysis
- Dynamic analysis
- Control flow analysis
- Memory analysis
- Network analysis
- Performance analysis

TraceDroid Analysis Platform
Inspection tool
- Quickly analyze >100K lines of trace output
- Load trace output into Python objects
- Interactive shell
- Call graphs for control flow analysis

Contributions
- Extend the analysis to other Android apps
- Extend the analysis to other Android versions
- Extend the analysis to other Android devices
- Extend the analysis to other Android platforms

ZiiMo: Zeus in the Mobile
- Collaborates with PC-based Zeus
- Steals mobile TAN codes

Conclusions

Future work
- Extend the analysis to other Android apps
- Extend the analysis to other Android versions
- Extend the analysis to other Android devices
- Extend the analysis to other Android platforms

Evaluation

Demo

Submit your .apk for automated analysis
- .arcs output containing:
  - method traces
  - network dump
  - call graph
Contact me if you would like to analyze a batch

No source or inspect tool available yet
TraceDroid: A Fast and Complete Android Method Tracer

Hack in the Box 2013, Kuala Lumpur

Victor van der Veen

About me

Currently
- 2.04 m
- Security Consultant at ITQ
- ITQ CTF team member

Past
- (partial) Implementation of a trustworthy voting machine
- Worked on Andrubis with the iSecLab team in Vienna
About me

Currently
• 2.04 m
• Security Consultant at ITQ
• ITQ CTF team member

Past
• Memory Errors: The Past, The Present and the Future (RAID 2012)
• (partial) Implementation of a trustworthy voting machine
• Worked on Andrubis with the iSecLab team in Vienna
Mobile Malware

March 2012: 38,689 samples
March 2013: 276,259 samples

+614%

Android: 92%

How do we automate analysis?
Contributions

**TraceDroid**

Android OS with comprehensive method tracing capabilities:
- Parameter resolution
- Return values
- Object representation

**TraceDroid Analysis Platform**

Framework for automated dynamic analysis
Detect suspicious activity
Ease post analysis
Implementation

TraceDroid Analysis Platform

Automated analysis

- Quickly analyze >100K lines of trace output
- Load trace output into Python objects
  - Interactive shell
  - Call graphs for control flow analysis

Contributions

- TraceDroid: Framework for automated dynamic analysis
  - Detect malicious activity
  - Case study available

ZitMo: Zeus in the Mobile

- Collaborates with PC-based Zeus
- Steals mobile TAN codes

Conclusion

TraceDroid

- Plan and implement real-world needs
- May be used as a debugging/hotfix development

Evaluation

Demo

ZitMo: Zeus in the Mobile

http://tracedroid.few.vu.nl/

Submit your apk for automated analysis
.tar.gz output containing:
  - method traces
  - network dump
  - call graphs

Contact me if you would like to analyze a batch

No source or inspect tool available yet
Scope

Limit method tracing to Java code

- Interesting features only accessible via Java
- Existing tools for tracing native code
  
  `strace`
  
  `ltrace`

Use dynamic analysis

- Evade obfuscation
- Existing tools for static analysis
  
  `AndroGuard`
  
  `Dex2Jar`
Application Fundamentals

Apps are written in Java, executed by a VM

Building blocks:
- **Activity** Single screen with a UI
- **Service** Background components
- **Receiver** Listener for specific announcements
  - e.g., *boot completed, sms received*

Distributed as signed jar files (.apk)
Android Architecture

- **App**
- **Applications**
- **Application Framework**
- **Native libraries**
- **Core Libraries**
- **Dalvik VM**
- **Linux Kernel**
Related Work

**Droidbox**
Injests trace methods into bytecode
- Only a small subset of API calls
- Break signature

**Droidbox**
Adds tracing code to core libraries
- Only a small subset of API calls
- Only for Android 2.1

**DroidScope**
Uses VMI to reconstruct instructions
- Bound to an emulator
- Not open source at the time

**TaintDroid**
Uses taint tracking to detect privacy leaks
- No API / method tracing

**Android profiler**
Method tracer for developers
- No object resolution
- Limited start/stop control
- Bloated
**Android Architecture**

- **App**
- **Applications**
- **Application Framework**
- **Native libraries**
- **Core Libraries**
- **Dalvik VM**
- **Linux Kernel**

**Related Work**

- **Droidbox**
  - Injects trace methods into bytecode
  - Only a small subset of API calls
  - Break signature

- **DroidScope**
  - Uses VMI to reconstruct instructions
  - Bound to an emulator
  - Not open source at the time

- **Droidbox**
  - Adds tracing code to core libraries
  - Only a small subset of API calls
  - Only for Android 2.1

- **TaintDroid**
  - Uses taint tracking to detect privacy leaks
  - No API / method tracing

**Android Profiler**

- Trace app internally
- Trace app to application framework
- Trace app to core libraries
- Trace application framework to core libraries
- Trace core libraries internally

Extend Android Profiler to suit our needs
TraceDroid

Extending Android's Profiler implementation

Hook on method invocations

- Fetch parameters from stack frame
- Lookup and invoke .toString() for Objects
- Convert signatures and descriptors
  
  \[
  \text{foo}(\text{java.lang.String;}Z[])V
  \]
  
  \[
  \text{void foo(java.lang.String, boolean, long[][])}
  \]

Hook on method returns

- Parse return value
- Get thrown exceptions
TraceDroid Analysis Platform

Automated analysis

Static Analysis
List Activities and Services

Stimulation
Simulate events to start receivers
- Reboot
- Incoming SMS

Enumerate Activities and Services

Monkey Exerciser
- Stress test GUIs

Post-Processing
Extract features
- Search traces for suspicious activity
- Preliminary results for malware detection: ~99.96%

Code Coverage Computation
- Map statically found methods against trace output

TraceDroid
Extending Android's Profiler implementation

Hook on method invocations
- Fetch parameters from stack frames
- Lookup and invoke 'toString()' for Objects
- Convert signatures and descriptors

Hook on method returns
- Return return value
- Get thrown exceptions

Log output

Capture network traffic
Stimulation

*Simulate events to start receivers*
  - Reboot
  - Incoming SMS

*Enumerate Activities and Services*

*Monkey Exerciser*
  - Stress test GUIs
Post-Processing

Extract features

- Search traces for suspicious activity
- Preliminary results for malware detection: ~93-96%

Code Coverage Computation

- Map statically found methods against trace output
TraceDroid Analysis Platform

**Automated analysis**

Static Analysis
- List Activities and Services

Stimulation
- Simulate events to start receivers
  - Reboot
  - Incoming SMS

Enumerate Activities and Services

Monkey Exerciser
- Stress test UIs

TraceDroid
- Extending Android's Profiler implementation
  - Hook on method invocations
    - Fetch parameters from stack frame
    - Lookup and invoke 'unboxing' for.Objects
    - Convert signatures and descriptors
      - `foo(int,int,java/lang/String)`
      - `void/before(java/lang/String)`
  - Hook on method returns
    - Return return value
    - Get thrown exceptions

Post-Processing
- Extract features
  - Search traces for suspicious activity
  - Preliminary results for malware detection: ~99.96%

Code Coverage Computation
- Map statically found methods against trace output

Capture network traffic

Log output
TraceDroid Analysis Platform

*Inspection tool*

Quickly analyze >100K lines of trace output

Load trace output into Python objects

- Interactive shell
- Call graphs for control flow analysis
TraceDroid is fast

Benchmark: browse to 8 cached webpages
Visit each page 10 times before computing average load time

Speedup of 1.45 compared to original profiler
## Simulation Effectiveness

Compare automated analysis against manual input (180 seconds)

<table>
<thead>
<tr>
<th></th>
<th>Manual</th>
<th>TraceDroid</th>
</tr>
</thead>
<tbody>
<tr>
<td>17x benign</td>
<td>38.49%</td>
<td>−2.45%</td>
</tr>
<tr>
<td>18x malicious</td>
<td>27.61%</td>
<td>+3.79%</td>
</tr>
</tbody>
</table>

TraceDroid's coverage is about as good as manual analysis. Likely of higher quality due to receiver stimulation.

### Analysis of ~500 samples

<table>
<thead>
<tr>
<th></th>
<th>Code Coverage</th>
</tr>
</thead>
<tbody>
<tr>
<td>250x benign</td>
<td>35.02%</td>
</tr>
<tr>
<td>242x malicious</td>
<td>31.10%</td>
</tr>
</tbody>
</table>

~33%
Simulation Effectiveness

Code coverage of 33% is fairly low
  • (third-party) Libraries
  • Unreachable code
  • Complex applications

Simulation effects vary per app
Simulation effects vary per app

Monkeys suck at gaming
Demo

ZitMo: Zeus in the Mobile

- Collaborates with PC-based Zeus
- Steals mobile TAN codes
Future work

**TraceDroid**

Unpack arrays
- Obfuscation could pass parameters in Object[]

Run on real hardware
- Evade emulator detection

**TraceDroid Analysis Platform**

Compute code coverage per package

Improve stimulation
- Symbolic execution?
Conclusions

**TraceDroid**

- Fast and comprehensive Android method tracer
- May be used as a debugging tool by developers

**TraceDroid Analysis Platform**

- Automated analysis of unknown applications
- Quickly identify suspicious applications
- Interactive environment to ease post-analysis
Submit your .apk for automated analysis .tar.gz output containing:

- method traces
- network dump
- call graph

Contact me if you would like to analyze a batch

No source or inspect tool available yet
Implementation

TraceDroid Analysis Platform
Automated analysis

Quickly analyze >100K lines of trace output
Load trace output into Python objects
- Interactive shell
- Call graphs for control flow analysis

TraceDroid Analysis Platform
Inspection tool

Conclusions

Evaluation

http://tracedroid.few.vu.nl/
Submit your .apk for automated analysis
.tar.gz output containing:
- Android dex
- network dump
- call graph

Contact me if you would like to analyze a batch

No source or inspect tool available yet

Demo

ZitMo: Zeus in the Mobile
- Collaborates with PC-based Zeus
- Steals mobile TAN codes

Future work