Mobile device security or MADAM: A Multi Level Anomaly Detector for Android Malware

Fabio Martinelli
National Research Council of Italy (CNR)

Joint work with Andrea Saracino, Daniele Sgandurra, Gianluca Dini, Francesco Mer caldo ...
Outline

• National Research Council of Italy in a nutshell
• Security for mobile devices (android)
• Madam framework
• Future work
CNR in a nutshell

• The Italian National Research Council is the main public research organization in Italy
  – CNR has near 9000 employees split in:
    • 100 research Institutes
  – The main Italian organization as capability to attract EU project funding

• My Institute of informatics and Telematics (IIT-CNR)
  • Location: Pisa, Tuscany, Italy.
  • Has 4 research groups:
    – Security, networking, Algorithms, Web technologies
  • IIT-CNR manages the ccTLD “.it” and it is part of EURid consortium that manages “.eu”

• Fabio Martinelli is the coordinator of all the cyber security activities at CNR

• Security Group of IIT-CNR:
  – 6 researchers
  – 4 Post-docs
  – 3 PhD students
  – 1 Administrative
  – 4 software engineers
  – 3 associate researchers from University
EU projects/ Research Areas

- ID and TRUST
- Usage Control
- Secure communication
- Data Protection
- Formal methods for modelling, analysis, and synthesis

- Bionets
- S3MS
- Consequence
- Sensoria
- Aniketos
- SESAMO
- COCO-Cloud
- CONTRAIL
- GRIDTrust
- Secure!
Current Main Activities

• Developing an promoting the **European Cyber Security Strategic Research Agenda** produced by the European Commission promoted Public Private Platform for Network and Information Security (NIS)
  – I am the coordinator of the WG3 on secure ICT research and innovation
    • More than 200 researchers from all the main research/academic/governmental institutions
  – Current Agenda is available at the ENISA URL:
  – Agenda is taken as basis the cPPP on cyber security

• Coordination of the **European Research and Training Network in Cyber Security (NeCS)**
  – More than 12 partners
  – The objectives if to create an active community of PhD/young post docs students interested
  – Research and training opportunities
  – Fellowships in several European countries (including CNR in Italy) and travel available for young students
Security

• Android is the target of 99% of security attacks on mobile devices.

• Apps are practically the main vector to bring security attacks on Android.

• Yearly malware increase: exponential
Malware Increase on Android
Why Android

• Not enough yet?
  – Android is **Open Source**
  – Availability of unofficial market
Android Markets

• Installing applications from unknown sources.
• Free versions of apps which have a cost on the official market.
• Limited-to-no control on the applications.
• Repackaged apps
  – Trojanized apps
Android Markets (2)

• Dangerous and malicious applications have been found **even** on the official market (Google Play).
  – Loose controls (**Bouncer**) not effective against zero day attacks.
  – Policy of forced removal of malicious apps from victim’s devices.
Android Security State of the Art

• Producer Side:
  – Native Security Mechanisms:
    • App Isolation
    • Permission System (access control)
    • Blocking unknown sources by default
    • Online detection of malicious apps at install time (online antivirus).
  – Pro: Native, no overhead.
  – Cons: Easy to deceive
• Commercial Side:
  – Anti-Virus code base – signature based.
    • Pretty much as standard computer AVs.
    • Also same brands -> Mobile edition
  – Pro:
    • Ease of use and no false positives
  – Cons:
    • Uneffective against new threats (zero day)
Android Security State of the Art (3)

• Research Side:
  – Static analysis framework
    • Decompiles and analyzes security relevant features of app code.
    • Pro: Can be run offline and almost accurate.
    • Cons: Attack specific and could miss run time misbehaviors
  – Information flow analysis
    • Detection of privacy leakage and app vulnerability
    • Example: Taintdroid
    • Pro: Effective in finding exploitable vulnerabilities.
    • Cons: Mainly concern only the subset of privacy related attacks
Android Security State of the Art (4)

• Still more research:
  – Security policies enforcement
    • Code instrumentation-based (Example: App Guard).
    • **Pro:** Fine grained control.
    • **Cons:** Require modification of device OS.
  – Behavior based Intrusion Detection System:
    • Monitor and classify behaviors as genuine or malicious at runtime.
    • **Pro:** Can detect zero days.
    • **Cons:** Can raise False Positives
Detecting Malicious Behaviors

- Works at runtime.
- Code independency:
  - Not tricked by obfuscation
  - Not tricked by polymorphic malware
  - Not tricked by malware which download malicious code at runtime.
Malicious Behaviors

• Steal privacy sensitive data
  – Contacts
  – Text messages
• Steal user’s money
  – Send text message
  – Register to premium services
  – Try to intercept bank transactions
• Show undesired advertisements (spam)
• Take control of the mobile device
• …
Malware: Some Numbers

• Almost 1 M malicious apps in the wild.
• More than 200 different malware families.
  – Family: Different applications with the same malicious code.

• Finding: Several implementations for the same misbehavior
Malware Classes

• **Malware Class**: Different applications with different malicious code, performing however the same (or very similar) misbehavior.

• 7 Malware classes identified... out of 150 analyzed families.
Malware Classes (2)

• **SMS Trojan**: Send SMS messages without user authorization.
• **Rootkit**: Attempt to take super user privileges.
• **Botnet**: Open a backdoor and wait for commands from a C&C server.
• **Spyware**: Steal sensitive information related to user privacy.
Malware Classes (3)

• **Installer**: Try to download and install additional malicious applications, without the user authorization.

• **Ransomware**: Attempt to take control of the device, blocking it till a fee is not paid by the user.

• **Trojan**: The few families (5/125) with custom misbehaviors not falling in anyone of the former categories.
MADAM

• Multi-Level Anomaly Detector for Android Malware

– It combines several approaches:
  • Anomaly Based Intrusion Detection and Prevention System.
  • Host based.
  • White list.
  • Zero day attacks.
Multi-Level for Higher Detection

• MADAM monitors 5 sets of features.
• Each set as standalone or in cooperation with others is used to spot a specific misbehavior class.
Global Analysis

• Monitor device at different levels:
  – System Calls
    • 13 SysCalls relevant
  – API Calls
    • Outgoing SMS
    • Active processes
    • Package installation
  – User Activity
    • User Present / Not Present
Per App Analysis

- Issued System Calls
- Sent Text Messages
  - Recipient
  - Message text
  - Frequency
- Number of processes per package
- Static Information
  - Required permissions
  - Market of provenance
  - Developer reputation
  - Rating and user feedbacks
  - Code analysis (n-grams)
Static Analysis

• Performed at *deploy time*, before app can be executed.
• Controls app installed from any sources (not deceived by Installer malware).
• Analysis of app metadata.
  – Does not require to decompile binaries.
  – Low performance overhead.
• Analysis of n-grams (code analysis)
Static Analysis (2)

- N-grams analysis: analysis of frequencies of opCode n-grams (sequences of actions) to be used as features for classifiers.
- Classifier trained with known malicious frequencies.
- Application for static recognition of malicious apps.
Static Analysis (2)

Trust Value

Multi-Criteria Decision System

Permission Analysis

Developer Reputation

Marketplace

User Feedback

Rating

User Reputation
Static Analysis (3)

• Permission analysis:
  – Extracted from *Manifest* file of APKs *(AndroidManifest.xml)*
  – Threat score assigned to each permission on three parameters:
    • Privacy Threat
    • Financial Threat
    • System Threat
Privacy Threat

• Permissions that allow an application to:
  – Read Contacts
  – Read text messages
  – Access user’s accounts and passwords
  – Read IMEI and location
Financial Threat

• Permissions that allow an application to:
  – Perform phone calls.
  – Send SMS messages.
  – Use the internet connection.
  – Modify connection settings.
System Threat

• Permissions that allow an application to:
  – Install/Uninstall applications on the phone.
  – Enable/Disable connection interfaces (Wi-Fi, Bluetooth, ...).
  – Switch on/off the smartphone screen.
Static Analysis (4)

• Based on the Analytical Hierachy Process (AHP)
  – Weighted sum of scores assigned to the 5 parameters

• Simultaneously analyzes all the parameters and returns a decision:
  – Trusted
  – Untrusted
Madam Architecture
MADAM Workflow

App Evaluator → App Suspicious List → Prevention

Heuristics → Prevention

User Activity Monitor → Actions Logger
Message Monitor → Actions Logger
SysCall Monitor → Actions Logger

Actions Logger → Classifier → Alert Module

Per App Monitoring

Global Monitoring
Policies

• Potentially malicious action evaluated against custom security policies.

• Security Policies can be:
  – Manually selected (security policies)
  – Inferred from classifiers (conditions on system calls).
  – Based on specific behavioral probabilistic patterns expressed through probabilistic automata or logic formula.
Examples:

- More than 5k reads when user non active -> misbehavior.
- SMS sent to number not in contacts -> misbehavior
- App behavior deviates from expected one -> misbehavior
- App behavior does not match policy specification -> misbehavior
- Probabilistic graph from execution logs to describe expected behavior.
- Markov Chain representation.
- Runtime behavior reconstruction and matching.
Prevention

• If an action violates a policy, it is blocked.
• User is notified of the violation if performed by a suspicious-listed activity.
• Active policies can be set by the user at any time.
Global Monitor

• Classification done through a K-NN classifier with \(k=1\) (1-NN).

• Based on numerical features
  – Issued SysCalls
  – Sent Messages
  – Seconds of user activity

• Good behavior and Bad behaviors form different clusters.
Global Monitor (2)

- Comparison between 2 behaviors (vectors)
  - User Idle (top) VS User Active (bottom)

- Classification performed through vectors similarity

\[ \text{Similarity}(x, y) = -\sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \]
Detection Result (Statistics)

- Training Set: 30000 behavior vectors.
- Malicious Vectors: 800
  - Real malware + Artificially generated (SMOTE)
- TPR = 100%
- FPR = 0.01%
Malware Detection Results

• Three tested datasets of malicious apps:
  – Total number of tested apps: 2784
  – Number of families: 123

• Global Accuracy: 99.7%

• 100% accuracy against, SMS Trojan, Installer, Ransomware, Rootkit and general trojan.

• Able to detect the Android.Poder trojan, still undetected by most AV.
Discussion

• Malware perform malicious action demanding OS or other components to effectively do the misbehavior.
  – Difficult to find anomalies in syscall issued by apps.
  – Easy to find globally.
  – Static and dynamic approaches are complementary.
• Detection Results compared with VirusTotal.
  – Better accuracy (99,7% (MADAM) vs 98% (VT))
Detailed Results

<table>
<thead>
<tr>
<th>Malware Type</th>
<th>Families Samples</th>
<th>Static Fam Sam</th>
<th>Dynamic Fam Sam</th>
<th>MADAM</th>
<th>VirusTotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Botnet</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Installer</td>
<td>6</td>
<td>3</td>
<td>6</td>
<td>406</td>
<td>400</td>
</tr>
<tr>
<td>Ransomware</td>
<td>6</td>
<td>2</td>
<td>3</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Rootkit</td>
<td>13</td>
<td>10</td>
<td>13</td>
<td>543</td>
<td>541</td>
</tr>
<tr>
<td>SMS Trojan</td>
<td>41</td>
<td>34</td>
<td>41</td>
<td>1309</td>
<td>1290</td>
</tr>
<tr>
<td>Spyware</td>
<td>38</td>
<td>38</td>
<td>21</td>
<td>231</td>
<td>220</td>
</tr>
<tr>
<td>Trojan</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>Hybrid</td>
<td>14</td>
<td>10</td>
<td>14</td>
<td>243</td>
<td>243</td>
</tr>
<tr>
<td>Composition</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>123</td>
<td>103</td>
<td>102</td>
<td>2784</td>
<td>2753</td>
</tr>
</tbody>
</table>

**Accuracy**

|                  | **99.7%** | **98%** |

[Image of the table]
Performance

• Testbed:
  – LG Nexus 4

• Overhead (Quadrant tool):
  – Global 1,4%
  – CPU: 0,9%
  – Memory: 9,4%
  – Video 0%
  – Battery: 3%
False Positive Analysis

• On a set of 9804 genuine apps the 0.2% has been considered suspicious by the static analysis module.
• At runtime:
  – Results extracted as average of one week of experiments on three devices with different users.
  – the average amount of FP per day is of 1 ($FPR = 1*10^{-5}$).
Requirements

• Non custom operative device.
• Necessary to have the device rooted (jailbreak).
  – Activate the kernel module.
  – Intercept events and stopping them.
References


Future Works

• Increasing the number of policies, their extraction methods and evaluation strategies.
• Using collaborative approaches for intrusion detection
• Using privacy aware techniques for IDS
Thank You

fabio.martinelli@iit.cnr.it
Related Work

- **Copperdroid**: performs malware stimulation to discover hidden behavior.
  - Considers actions at Java and JNI level.
  - VM and System call – based.
  - Runs offline (not on device).
- **TaintDroid**: analyzes information flows to detect and stops privacy leakage.
  - Allows the definition of security policies for data protection.
  - Requires OS modifications.
  - Attack Specific
Related Work (2)

• **Alterdroid**: Framework for analysis of Android application to detect faults in static resources.
  – Effective in detecting repackaged apps.
  – Attack specific.

• **Aurasium**: Framework for enforcement of app specific security policies.
  – Does not consider global features and can be evaded.
Related Work (3)

• **MOSSDroid**: static analysis framework for Android malware.
  – Signature based detection.
  – Evaded by zero day threats.

• **Drebin**: framework for static analysis and classification of Android malware.
  – Analyzed a large amount of samples.
  – Offline analysis.
  – Drebin DB analyzed by MADAM.
Probabilistic Contract Based Security

• Verifying if app behavior matches security policies.
• Probabilistic security policies:
  – Greater flexibility
  – Smaller fall-out (FPR)
• Generation of probabilistic contract from app execution (sandbox).
• Learning user probabilistic behavior.