MADAM: A Multi Level Anomaly Detector for Android Malware

Fabio Martinelli
National Research Council of Italy
(CNR)

Joint work with Andrea Saracino, Daniele Sgandurra et al.
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
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:

- 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
Global Smartphone Market Share By Platform

- **Android**
- **Apple iOS**
- **Microsoft**
- **BlackBerry**
- **Other**

Source: IDC, Strategy Analytics
Security

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

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

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

• Not enough yet?
  – Android is **Open Source**
  – Loose control on official market
  – 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
Android Security State of the Art (2)

• 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 implementation 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**

  – 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
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.
Static Analysis (2)

Trust Value

Multi-Criteria Decision System

Permission Analysis

Developer Reputation

Marketplace

User Feedback

Rating

User Reputation

Consiglio Nazionale delle Ricerche - Pisa
Istituto di Informatica e Telematica
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.

Where is my 7-Day Battery?
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

Levels
- Package-Level
  - App/Market

Features
- App Metadata
  - Permissions Rating
  - Market Developer Downloads
- Critical API
  - Admin Apps Install Apps Process List
- User Activity
  - User Present On Call Screen On
- SMS
  - SMS Num SMS Susp
- Sys Calls
  - open read ... rcvmag

MADAM Architecture
- App Risk Assessment
  - App Evaluator
  - App Suspicious List
  - Heuristics
  - Signature-based Detector

Per App Monitor
- User Activity Monitor
- Message Monitor
- Actions Logger
- SysCall Monitor

Global Monitor

Prevention

Classifier

Notify and Remove Malicious App

App Evaluation

User Interface
MADAM Workflow

App Evaluator → App Suspicious List → Prevention

Heuristics → Prevention

Per App Monitoring:
- User Activity Monitor
- Message Monitor
- SysCall Monitor

Global Monitoring:
- Actions Logger → Classifier → Alert Module
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*.
Policies (2)

• 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)

<table>
<thead>
<tr>
<th>open</th>
<th>ioctl</th>
<th>brk</th>
<th>read</th>
<th>write</th>
<th>exit</th>
<th>close</th>
<th>sendto</th>
<th>sendmsg</th>
<th>recvfrom</th>
<th>recvmsg</th>
<th>Idleness</th>
<th>SMS Num</th>
<th>SMS Susp</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>19</td>
<td>18</td>
<td>1</td>
<td>4</td>
<td>0</td>
<td>7</td>
<td>16</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>147</td>
<td>652</td>
<td>192</td>
<td>711</td>
<td>4</td>
<td>282</td>
<td>229</td>
<td>7</td>
<td>15</td>
<td>7</td>
<td>13</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

• 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: 2800
  – Number of families: 125

• Global Accuracy: 96,1%

• 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.

• Detection Results compared with VirusTotal.
  – Comparable accuracy (96,4% vs 96,1%)
  – Almost complementary results
    • Possible merging with Virus Total for higher accuracy
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 $1 \times 10^{-5}$).
Requirements

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

• Increasing the number of policies, their extraction methods and evaluation strategies.
• Merging the approach with other static analysis tools like VirusTotal.
• Porting the MADAM approach on Windows and iOS platforms.
• Using collaborative approaches for intrusion detection
• Using privacy aware techniques for IDS
Thank You

fabio.martinelli@iit.cnr.it