Rage Against the Virtual Machine: Hindering Dynamic Analysis of Android Malware

Thanasis Petsas,1 Giannis Voyatzis,1 Elias Athanassopoulos,1 Michalis Polychronakis,2 and Sotiris Ioannidis 1

1 Institute of Computer Science, Foundation for Research and Technology—Hellas, Greece
2 Columbia University, USA

{petsas,jvoyatz,elathan,sotiris}@ics.forth.gr, mikepo@cs.columbia.edu

Motivation

- Android anti-virus products that offer real-time protection to mobile users can be evaded through transformation techniques[2]
- There exist many tools and web services that dynamically analyze Android apps in order to detect zero-day malware and enhance anti-virus capabilities
- Can these dynamic analysis tools also be evaded?
- How can we protect these tools from evasion techniques?

Anti-analysis Techniques

Static Heuristics

- Checking pre-initialized static information
  - Device ID (idH)
  - Current build (buildH)
  - Routing table (netH)

Dynamic Heuristics

- Sensors produce always the same values at equal intervals
  - Accelerometer (accelH)
  - Magnetic field (magnFH)
  - Rotation vector (rotVecH)
  - Proximity (proximH)
  - Gyroscope (gyrosH)

Examples

- IMEI, GSMI, etc
  - By default IMEI=null in Android Emulator
- Fixed Build attributes
  - PRODUCT=google
- Hardware=goldfish
- Android Emulator behind a virtual router
  - address space: 10.0.2/24

Hyervisor Heuristics

- Cases where native code runs differently
  - Identifying QEMU scheduling (BTdetectH)
  - Identifying QEMU caching behavior (xFlowH)

BTdetectH [1]

- QEMU optimization: Virtual PC is updated only after branch
- Device: Various scheduling points
- Emulator: A unique scheduling point
- xFlowH
  - QEMU does not emulate the ARM split cache

BTdetectH Heuristic Effectiveness

Due to optimizations many of the scheduling events that can take place are not exhibited on an emulated environment.

Heuristics APK

Web Server

Dynamic Analysis Service

Evaluation Methodology

Implementation

- Heuristics implementation: Use of Android SDK and NDK
- Android app that reports the effectiveness of the heuristics
- Incorporation of the heuristics in known Android malware samples
- Patch the Dalvik bytecode with the bytecode of the heuristics
- Use of Small/Baksmali and Apktool for disassembling and reassembling

Evasion Results

Countermeasures

- Emulator Modifications
- Realistic Sensor Event Simulation
- Accurate Binary Translation
- Hardware-Assisted Virtualization
- Hybrid Application Execution

References

  http://www.dexlabs.org/blog/btdetect


This work was supported in part by the FP7 projects NECOMA, OPTET, SysSec and by the FP7-PEOPLE-2010-IOF project XHUNTER, funded by the European Commission under Grant Agreements No. 600833, No. 317631, No. 254116 and No. 273765.