Analyzing ARM Native Code for Tracking Information Flow

Woo-Yeon Lee, Seo-Yoon Choi, Tae-Hun Kim, Byung-Gon Chun,
Cloud and Mobile System (CMS) Laboratory, Seoul National University (SNU)

1. Introduction

- Third-party “apps” may leak users’ privacy-sensitive data or manifest malicious behavior.
- Why ARM native code?
  - More and more apps use ARM native code.
    - Android: 49% of the apps are packaged with third-party native library
    - Tizen: Native apps are written as ARM native code.
  - Lots of studies about information flow tracking, but not in ARM-instruction level.

2. ARM Architecture

- Advanced RISC architecture
  - 32bit-fixed instruction length
  - PC as a general register
  - Single execution cycle
  - Conditional execution
- Extension
  - Thumb / Thumb-2 mode (16bit)

Some of these features are challenging to handle.

3. System Architecture

![System Architecture Diagram]

4. Dynamic Taint Tracking

- Taint Tracking is a technique used to track information dependencies from an origin.
- Three Factors
  - Taint Source
  - Taint Propagation
  - Taint Sink

- Update Taint map during execution

5. Dynamic Binary Instrumentation (DBI)

- What is DBI?
  - DBI manipulates executing binary at runtime and controls process’s behavior.
  - Main challenge is an Application Transparency, keeping application behaving same as before instrumentation.

- Taint Tracking with DBI,
  - Low overhead enables real-time tracking
- Implementation
  - Code Injector inserts initial code into application to load analysis modules into application's memory space
  - Taint Map is implemented with Shadow Memory, which is consisted of shadow bytes mapping to bytes in main memory.

---

CMS Labs Seoul National University