Lightweight Information Tracking for Mobile Phones

William Enck and Patrick McDaniel (Penn State)
Jaeyeon Jung, Byung-gon Chun, and Anmol Sheth (Intel Labs)

Mobile Phone OS Security

Smartphones running downloaded third party applications have become common in the past few years. Users are already downloading many applications without regard for security. As new OS architectures are designed, we must learn from PC security lessons while adapting technologies for this new environment.

Primarily, our research efforts focus on:

- How to protect the phone from applications
- How to protect applications from each other
- How to protect the user's privacy

Android as a Security Research Platform:

- Open source platform
- Unique and collaborative application architecture
- Lots of third-party applications (9,000+ and growing)
- Acceptance by network providers world-wide
- T-Mobile G1 in top-5 best selling smartphones (Q1'09, NPD Group)

Android Security

Android includes beneficial security features:

- Applications execute as unique user identities
- Separate system and data partitions
- Applications request functionality permissions

However, our previous research shows that Android is deficient in expressing desirable security policies.

- Dangerous permission combinations are frequently required for desirable application functionality (e.g., access both Internet and location state)
- Frequently, policy has insufficient context to make decisions (context is inside applications)

Previous Work:

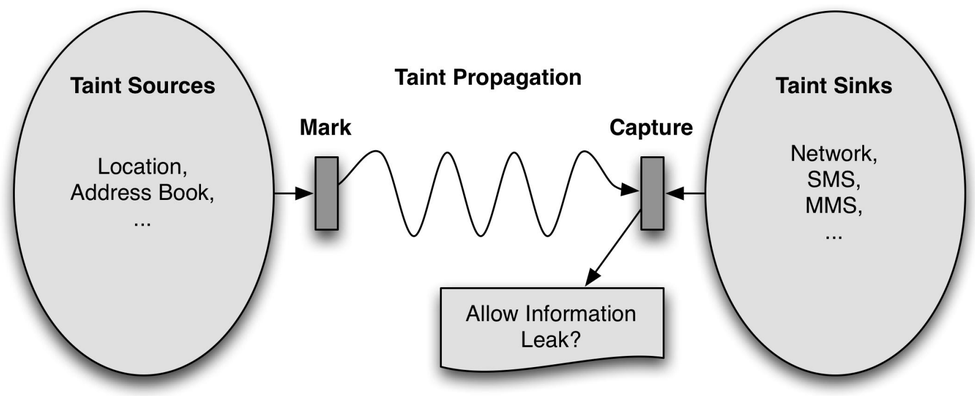
- William Enck, Machigar Ongtang, and Patrick McDaniel. On Lightweight Mobile Phone Application Certification. *Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS)*, November 2009.

Taint Tracking on Android

Holistic enforcement of information flow provides context regarding runtime execution decisions.

Desirable mobile phone security policies are achievable with taint tracking

- *Auto-tag* sensitive information at interfaces (e.g., location provider, address book, microphone, etc)
- Capture tainted information at the network boundary



Challenge: *performance* vs. *accuracy*

TaintDroid

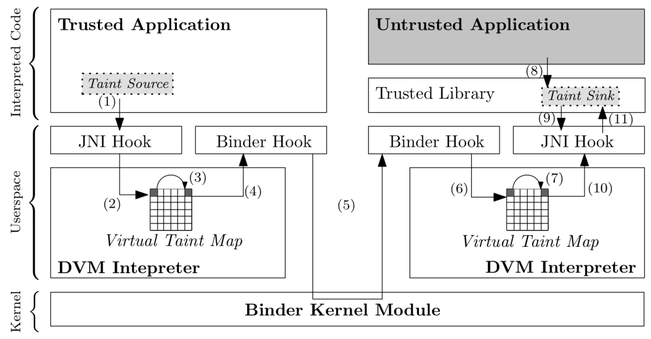
The system-wide taint tracking mechanism must:

- … run on the phone
- … track multiple taint sources

We leverage Android's architectural features to design a practical system: Java-like VM, specialized message passing IPC.

Approach: Instrument Android's virtual machine interpreter

- The virtual machine provides external processing context
- More efficient by tracking variables *not* memory addresses
- IPC taint tracking on message granularity



Trade-offs: requires careful handling of native code and no implicit flow tracking, but provides lightweight, fine-grained taint tracking

Performance

We have a modified firmware running on an HTC Dream

Many applications are primarily user interfaces with minimal processing: *overhead is difficult to perceive*!

CaffeineMark shows worst case performance overhead:

- Portable vs. "fast" interpreter: 24% loss
- Removing DEX optimization: 7% loss
- TaintDroid firmware: 59% loss, however …

… *only 28% is due to our changes*!

Providing Policy Context

TaintDroid provides only a mechanism to enhance Android's security. With this tool, we will explore a range of security policies, beginning with user privacy and then extending to traditional information flow policies involving environmental context such as geographic location and "processing modes."

Supported by NSF and Intel Corp.