Security Evaluation of App Runtime for Chrome

Meng Xu

Georgia Institute of Technology

meng.xu@gatech.edu

December 1, 2014
Overview

1. ARC Introduction
2. Permission Shift
3. Inter-”Component” Communication
The goal of ARC is to build the **minimum** codebase to run a **single** Android app

Implicitly constrained by multiple sandboxes

Privileges operations handled by Chrome broker process (by the design of NaCl)

Complemented by a re-packaging script
Permission model is the core of Android security

**Problem**: Repackaged app has access to privileged operations even without declaring corresponding permissions.

**Approach**: Instrument the repackaging script to declare correct Chrome permissions given declared Android permissions.

**Demo**
Difficulties

- Extract declarable permission list
  → Write an app to probe PackageManager on ARC dynamically
- Map Android permissions to Chrome permissions
  → Manual process
Permission Shift

Cause of the problem: two ways of enforcing permissions in Android
- Assign GID to the app
  → Broken
- Intercept API/system call to check permission
  → Still works

Demo
Future work

- Enhance the completeness of the permission shift
  - Currently support "dangerous" permissions only
  - Declarable permissions in Android and Chrome are not perfect matches
- Bring back the GID enforcement
Possible communications
- App $\leftrightarrow$ App
- App $\leftrightarrow$ Extension
- App $\leftrightarrow$ Webpage
- App $\leftrightarrow$ System

General conclusion: since ARC is heavily sandboxed, there is no particular advantage gained by attacking the ARC model compared with attacking Chrome or writing an Android malware.
Inter-"Component" Communication

- App $\leftrightarrow$ App
- App $\leftrightarrow$ Extension
- App $\leftrightarrow$ Webpage
- App $\leftrightarrow$ System
App $\leftrightarrow$ App

- System privilege escalation attack does not make sense
- Component hijacking (of another app) is not possible
Inter-"Component" Communication

- App ←→ Extension
  - Chrome extension may cause a DoS on Android app (Demo)
  - Chrome extension may view the cookies generated from Android WebView (Demo)
  - Android app has no way of influencing Chrome extensions
Inter-"Component" Communication

- App ←→ Webpage
  - No interaction
App $\leftrightarrow$ System

- Data stored in both "internal storage" or "external storage" are not safe (Demo)
- Apps may have access to OS filesystem (via browser file chooser), but will not be able to modify them (Demo)
Future work

- Going systematical
  - Apply dynamic taint analysis or static model checking to test the interaction between ARC and extension/webpage/system
- Side-channels / covert-channels
Questions?