WILLIAM & MARY
CHARTERED 1693

CSCI 445: Mobile Application Security

Lecture 23 (previously 15)

Prof. Adwait Nadkarni

Running scripts from *home*

- apktool instructions:

`Move both files (apktool.jar & apktool) to /usr/local/bin (root needed)`

- *No-root alternative*:
  - Create a *bin* inside *home*
  - Help the OS find *bin*
  - Export the path inside your *.bashrc* file, so that it is exported during every session.

```
$ mkdir ~/bin
$ export PATH=$PATH:~/bin

$ vi ~/.bashrc
<paste the export command
inside the bashrc, at the
end>.
```

  - Place apktool and other binaries inside this *~/bin*
  - Check if apktool is visible to the OS: `$ which apktool`

How do we study apps?

- *Generally, two ways to do this:*
- *Static analysis tells you want can potentially happen.*
  - *Getting source code: ded, dex2jar, androguard*
  - *Extend existing analysis tools (e.g., Fortify)*
  - *Frameworks: Flowdroid, Amandroid, DroidSafe*
- *Dynamic analysis tells you what actually happens given a specific runtime environment*
  - *TaintDroid, DroidScope*
  - *Derivative environments: Droidbox, andrubis, MarvinSafe*
- *Note: dynamic analysis is hard to automate*

Intro to Dynamic Analysis

Dynamic Analysis

- Execute the program, observe the behavior
- Various abstractions and granularities to monitor: instructions, system calls, processes, API calls, etc.
- Generally, you *monitor* certain *protected operations*
  - E.g., call to sensitive API, network connection
- Additionally, sometimes you *enforce*
  - *Prevent a call, or change returned data*

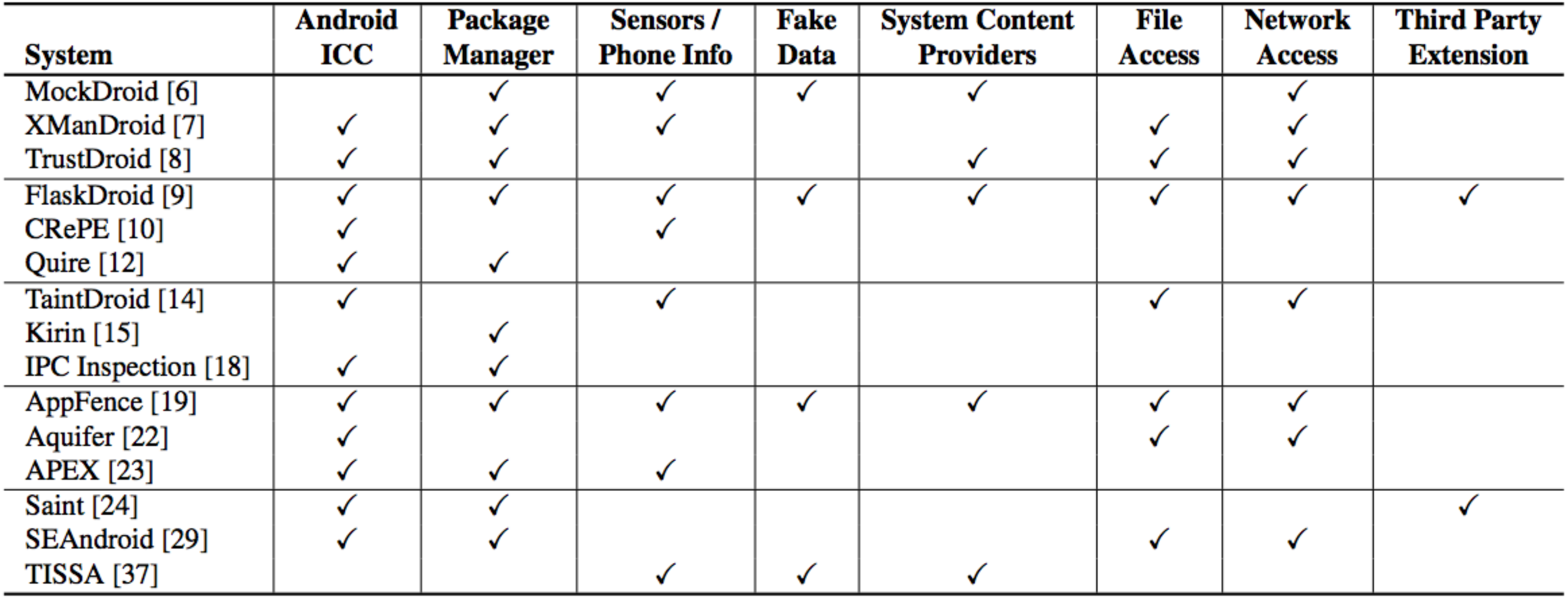
Offline vs Online Analysis

- **Online** Analysis:
  - In a real, production environment, i.e., on the user's phone
  - Factors to consider: Performance, impact of compromise
- **Offline** Analysis:
  - In a test environment (e.g., test device, emulator)
  - Factors to consider: Evasive malware, app exploration

Hooks - 1

- *General approach*: Hook into the relevant protected operation, and monitor programs' execution of it → based on security goal

Table 1: Classification of authorization hook semantics required by Android security enhancements



System	Android ICC	Package Manager	Sensors / Phone Info	Fake Data	System Content Providers	File Access	Network Access	Third Party Extension
MockDroid [6]		✓	✓	✓	✓		✓	
XManDroid [7]	✓	✓	✓			✓	✓	
TrustDroid [8]	✓	✓			✓	✓	✓	
FlaskDroid [9]	✓	✓	✓	✓	✓	✓	✓	✓
CRePE [10]	✓		✓					
Quire [12]	✓	✓						
TaintDroid [14]	✓		✓			✓	✓	
Kirin [15]		✓						
IPC Inspection [18]	✓	✓						
AppFence [19]	✓	✓	✓	✓	✓	✓	✓	
Aquifer [22]	✓					✓	✓	
APEX [23]	✓	✓	✓					
Saint [24]	✓	✓						✓
SEAndroid [29]	✓	✓				✓	✓	
TISSA [37]			✓	✓	✓			

Hooks – II

- *What* does it mean to hook?: Intercept protected operation.
    - Log execution of protected ops, OR get callbacks when they happen
- *Where (relative to the operation)?*
    - Right before, or right after the operation (e.g., for auditing)
- *How* would you accomplish this?
    - Modify the OS
    - Modify the app (i.e., place an *inline reference monitor (IRM)*)

What property do we want from our ***mechanism****?*

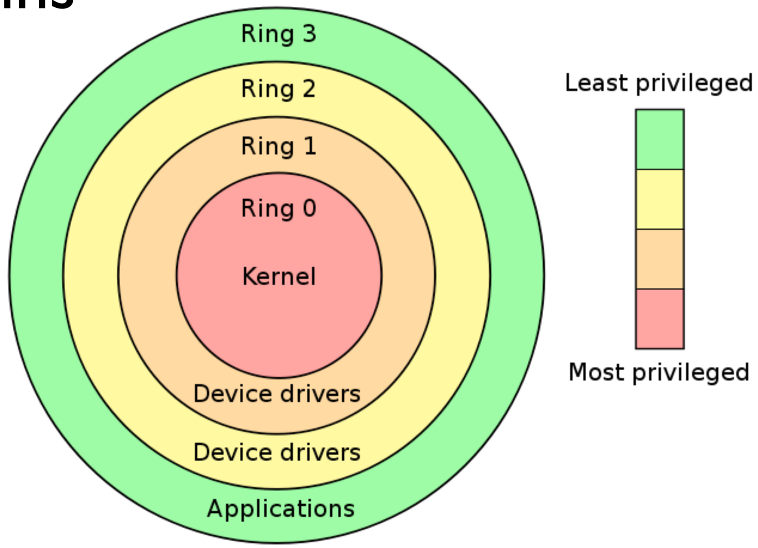
Recall: Reference monitor

- What *three* properties should a reference monitor possess?
    - Complete mediation
    - Tamperproof
    - Easy to verify

- *How* would you accomplish this?
    - Modify the OS
    - Modify the app (i.e., place an *inline reference monitor (IRM)*)

Background: Protection Rings

- Successively less-privileged "domains"
- Modern CPUs support 4 rings
  - Use 2 mainly: Kernel and user
- Intel x86 rings
  - Ring 0 has kernel
  - Ring 3 has application code
- Kernel: Can access physical memory
- Application process: Can only access its own virtual memory space (i.e., not even memory space of other processes)



Where to hook? -1

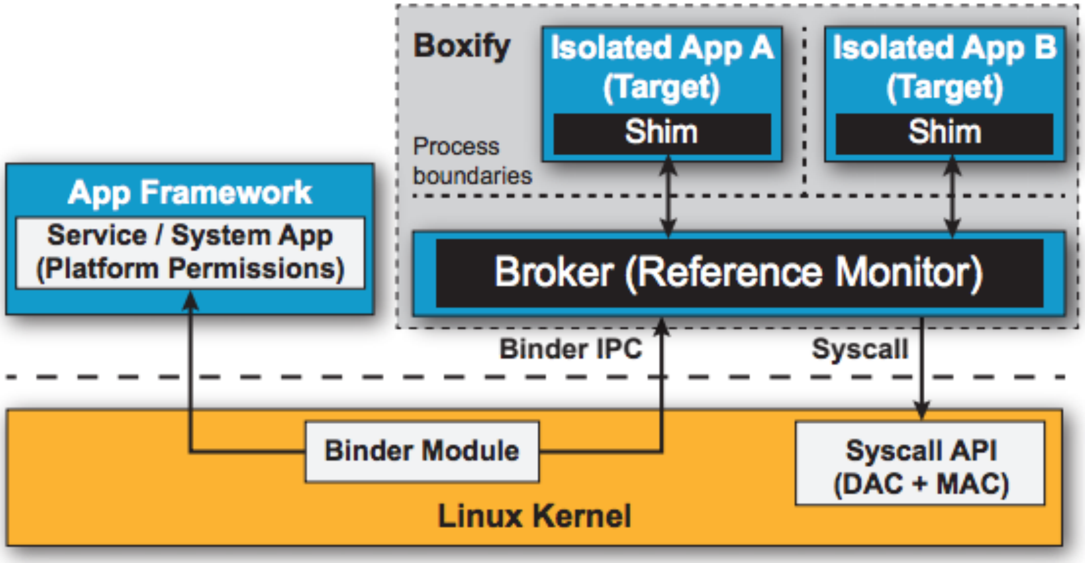
- Goal: Monitoring/Analyze an *untrusted application*
- **Option A:** Hook into the OS (e.g., Android Security Modules (ASM) Framework)
- Complete mediation, and tamper-proof?
    - Yes! The kernel can intercept all system calls
        - Processes can't access kernel memory (as long as the kernel or trusted services are not compromised)
- Is *online* analysis feasible? (i.e., during real-time use)
    - If you can get people to use the modified OS
- Is *offline* analysis feasible:
    - Yes! But may not capture all behavior

Where to Hook? - II

- Goal: Monitoring/Analyze an *untrusted application*
- **Option B:** Inline reference monitor (IRM) (e.g., Aurasium)
  - *Rewrite the APK* to place a check/callback whenever every protected operation is called
- Complete mediation, and tamper-proof?
  - The reference monitor and the program are loaded into the *same process memory* space. So what?
    - App can circumvent/tamper with monitor code!
- Is *online analysis feasible?*
  - Depends. Breaks app update cycle, but the user does not have to use custom firmware.

Where to Hook? - III

- *Boxify:* Provides the security of an OS-based reference monitor, without modifying the OS.
- Uses OS support to enforce a secure IRM
  - The "isolated process" abstraction available in Android



https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-backes.pdf

Where to Hook? - III

- *Boxify:* Provides the security of an OS-based reference monitor, without modifying the OS.
- Uses OS support to enforce a secure IRM
  - The "isolated process" abstraction available in Android
- Rewrites the app, starts it in an isolated process, and another process as a reference monitor
  - OS hooks allow the reference monitor process to get callbacks for protected events executed by the isolated process.
- However, practicality challenges (e.g., signed app updates) still remain

https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-backes.pdf

Challenges for Dynamic Analysis

1. Performance/resource Overhead
2. Granularity/Precision of Analysis
3. Evasive Malware
    a. Malware that circumvents the monitor (discussed previously)
    b. Malware that adapts behavior
4. Application Exploration (coverage)
    1. Higher FNs, but lower FPs (gross generalization), relative to static

Evasive Malware

- Case 1: Offline Analysis, on an emulator
  - How would malware avoid detection?
    - Detect emulator (e.g., arch, OS build)
      - Don’t execute malicious payload!
- Case 2: Offline Analysis, on a real test device
  - How would malware avoid detection?
    - Look for signs of *real* use (e.g., storage, contacts, calendar)
      - Only then execute payload

Application Exploration

- Two ways to do this: manual and automatic
- **Option A:** Manual
  - Use human intuition to guide the exploration of the app
  - Advantages?:
    - Explore likely scenarios
  - Disadvantages?:
    - Costly (time and effort)
    - Coverage may be subjective

Application Exploration

- Two ways to do this: manual and automatic
- **Option B:** Automatic/ semi-automatic (e.g., Monkey (simplest), CrashScope, SMVHunter)
  - Automate app exploration, guided by some heuristics
  - Advantages?:
    - Low manual efforts
  - Disadvantages?:
    - Covered behavior may be unrealistic and/or insufficient
- We are getting better at this (e.g., CrashScope exercises UI in a deterministic fashion), but still a research challenge
- Other practical challenges: Getting past user accounts, paid apps/services

Granularity of Analysis - I

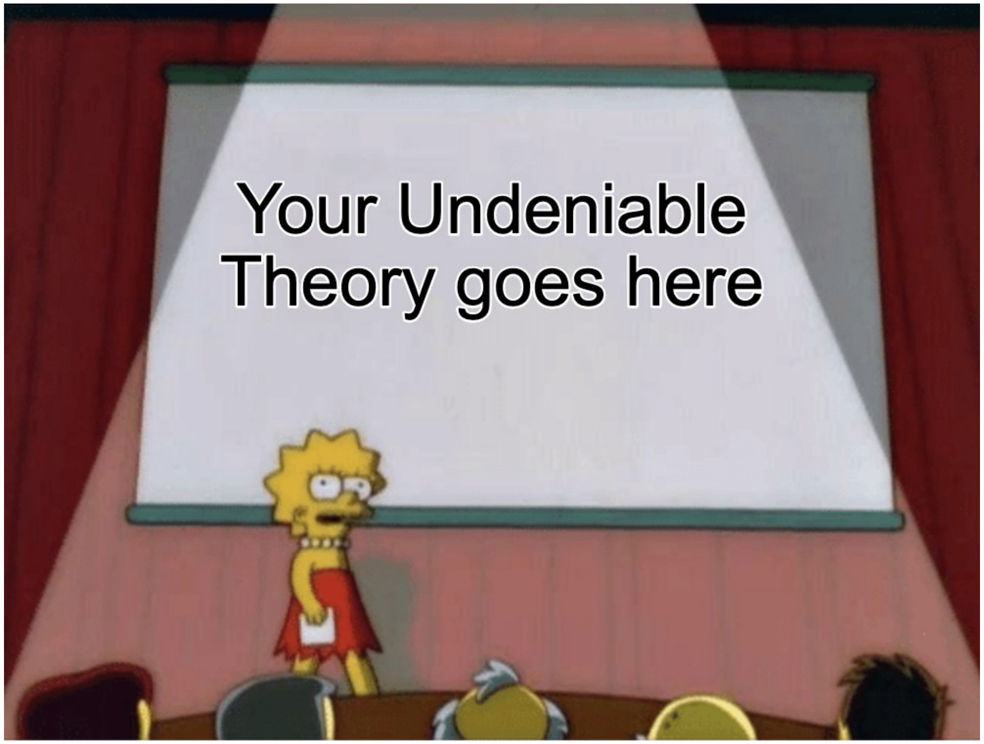
- The *precision* of the analysis depends on the granularity
  - i.e., high precision means low FPR
- Example 1: Detecting information stealing behavior
  - **Analysis 1:** Raises alarm when IMEI is accessed
  - **Analysis 2:** Notes when IMEI is accessed, *keeps track of where it flows*, and raises alarm when it (or copies) is exported to the network
  - Which is more precise?
    - Analysis 2, as it is relatively *fine grained*

Granularity of Analysis - II

- The *precision* of the analysis depends on the granularity
    - i.e., high precision means low FPR
- Example 2: Detecting information stealing behavior (IMEI)
    - **Analysis 2:** Tracks information flows among *processes*
    - **Analysis 3:** Tracks information flows among *program variables*
    - Which is more *precise*?
        - Analysis 3, as it is relatively *fine grained*
    - Which is likely to be more *sound*?
        - Analysis 2, as the OS has complete mediation over process interactions

Project Presentations

- Next Tuesday
- These are "status" presentations of *10 minute duration*
  - RQs
  - Analysis you are doing
  - Findings (optional)
  - Anticipated Results and Findings
- 1 – 5 bonus credits
- *Let me know by EoD today if you want to present.*



The End