CS 261: 1/25 Scribe Notes
Emma Lurie

Context for papers:

- Taint tracking is historically used to keep track of which inputs have the potential to be malicious (e.g. SQL injection) … a threat to integrity
- In the papers for today, we are instead interested in whether the program is sending the data to the attacker… a threat to confidentiality.

TaintDroid: what are barriers or challenges to deployment?

- Users have to run it
- Google has to do it. It may not be reasonable for users to use it (bc you have to have root access).
- Economics – ads: Google wants flourishing app marketplace and is a major ad network.
    - Maybe Google would want to reign in the most egregious violators

TaintDroid: Why didn't the authors suggest API changes?

    - Evaluation section gets trickier if you have to reimplement apps to test your system. Would have to test with developers.
- Malicious actors could go around TaintDroid
    - Implicit flow
    - Colluding apps (using local network)
    - Binary files
- There is a distinction to be made between purely malicious apps and apps that are negligent/accidental leakage.

PiOS: What's the problem with unique device identifiers being sent?

- What is a legitimate use of device ID?
    - "State" for device (e.g. uninstall and reinstall).
    - Fraud prevention (1 account per device)
- Apple has since locked down access to unique device identifiers bc of abuses (e.g. COPPA, can be identifying info esp. with other data)

PiOS: TaintDroid does dynamic analysis and PiOS is doing static analysis… seems similar. Both are trying to figure out "does this secret leave the network?" What are the tradeoffs between dynamic and static analysis?

- Need access to source code for static analysis. Trickier to do on iOS… PiOS had to do lots of heavy lifting to get the binary.
- Objective-C made it hard to hook into… In Java, String is a class… Can easily replace (overload) String library (with a track taint version). Because Java is interpreted, taint tracking for strings is easy.
- Java source and compiled code is very similar. iOS—compiled binary access only.
- Static analysis is more pro-active… up-front review process from Apple rather than dynamic analysis (which looks at leakage once app is deployed)

CS 261: 1/25 Scribe Notes
Emma Lurie

- Research has moved to view systems like PiOS and TaintDroid as pre-deployment review.

PiOS: Why mobile?

- Desktop has different trust model. Mobile has assumption that you haven't given all data to app developers. There has been work in desktop—malware detection; outgoing network data (look for location, device id); two copies of the app but gave different locations, where output differed—data leakage.

PiOS: The focus in both papers is on detection. What about remediation or mitigations?

- Notice: Green lights
- More granular control/consent: which photos will you allow the app to access?
- Ban bad actors from app store
- Frameworks like Frida

Student-led Discussion section:

In the use case where Apple wants to do some large scale privacy leak analysis (e.g. App Store review process), what modifications would have to be made?

- . TaintDroid requires some expert knowledge and manual interpretation of output. TaintDroid doesn't work on third party native library (today most apps use)
- PiOS—arrays and aggregate data isn't fully tracked. Use of heuristics…

Note about additional paper: AppsPlayground paper … same authors as TaintDroid. Wanted to make PiOS like system (increase scale) … still Android

Beyond privacy leaks, what also do you want to detect?

- Resource management (internal, data, external)
- User spam (quality)
- Android API use

Scaling of system like TaintDroid… how could you make it be automatic?

- Random exploration, random fuzz testing) … what are some challenges?
  - Can't cover everything (can't test every input value)
  - Scaling to lots of apps. In TaintDroid, humans are driving the app, but automating that is challenging
- Constrain code paths with static analysis tool
- To handle contextual input issues (e.g. logins) … have manual rules (same username and pw)

Overview of Static Analysis @ 30,000 feet:

- Dynamic analysis at compile time (rather than run time) …
- Example of Static Analysis: Let's imagine that all variables are Booleans.

CS 261: 1/25 Scribe Notes
Emma Lurie

```
x = getinput()
y= getinput()
if (y)
      Z= x
else:
      z = false
```

The problem is that we don't know what x and y are! So we treat it as nondeterministic finite automaton… where we look at all possibilities to find the possible values of output (see "comments" below). For conditionals, take the union. For assignment, copy input. Because of halting problem, not exactly correct, but deterministic answer will always be correct.

```
x = getinput()
// x -> {T, F}
y= getinput()
// x → {T,F}, y → {T,F}
if (y)
      // x -- > {T,F} , y== {T}
      Z= x
      // x – {T,F}, y -- > {F}, Z -- > {T,F}
else:
      z = false
      // x → {T,F}, y → {F}, z – >{F}
// x -- > {T, F}, y – {T,F}, z → {T,F}
```

What is the runtime of this?

- Linear in the length of the code.
- But what happens with a loop?
    - This is a potential problem (infinite), but once there is convergence of the set (fixed point convergence), you know you are done.
    - If just using Booleans, can only have max 2 iterations.
    - At the end of the day… polynomial with loops
- This is handy, bc taint status is Boolean!

Integers instead of Booleans:

- Set of predicates, {x <0, x==0, x>0}… but can now grow unbounded
- Need fancy techniques… if more than one value… conservative over-approximation→ Z (all integers)

CS 261: 1/25 Scribe Notes
Emma Lurie

- Pointers, reframe as Boolean question

Next time: more of usable security perspective