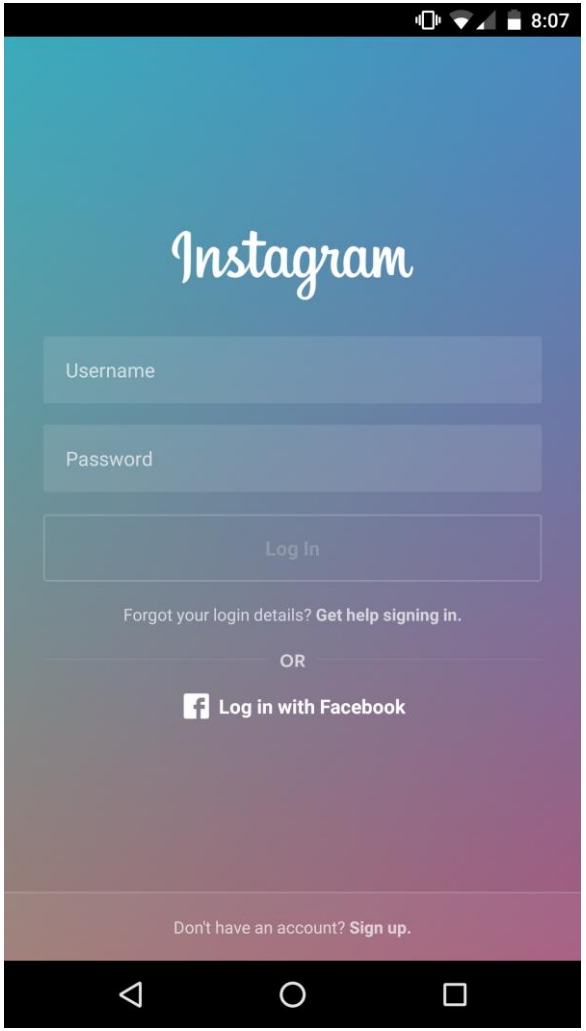
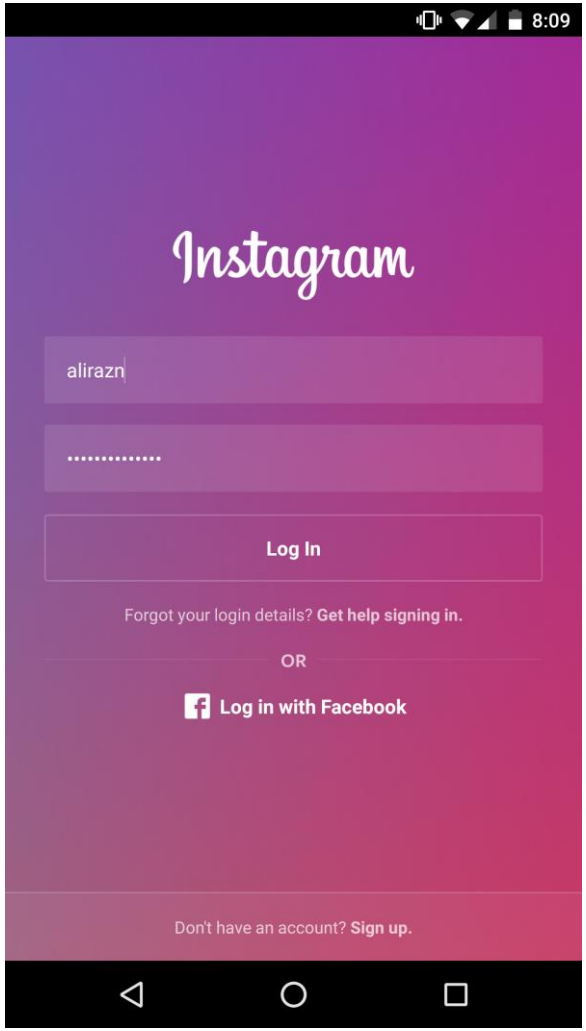
Dynamic Taint-Tracking on Modern Mobile Platforms

Ali Razeen

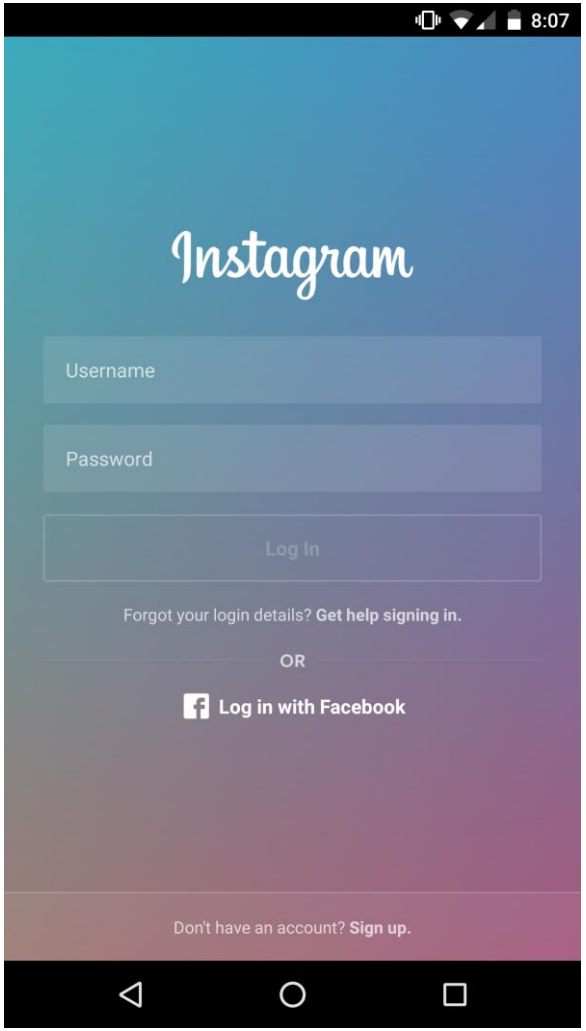
Duke University

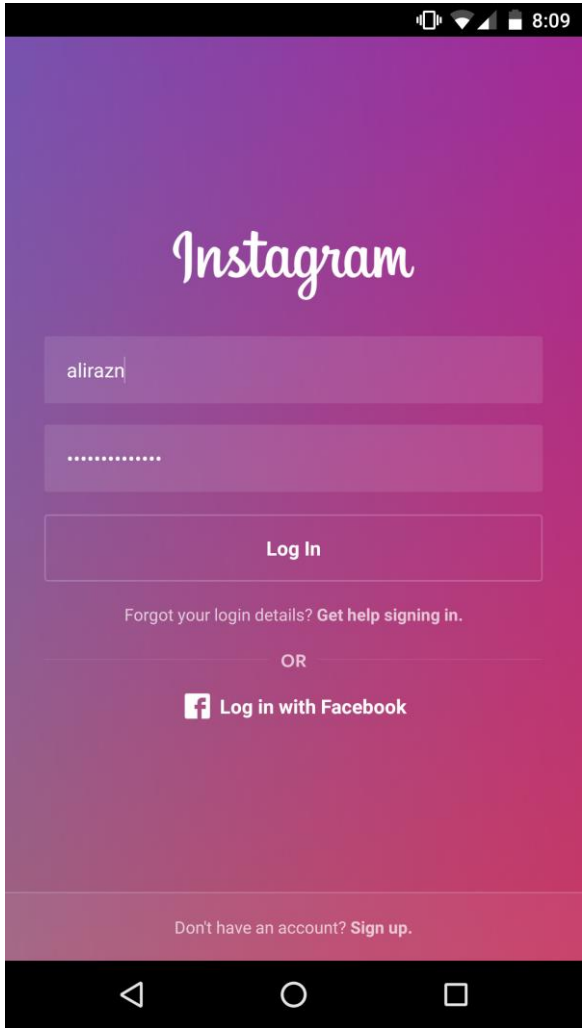
A Typical App Today



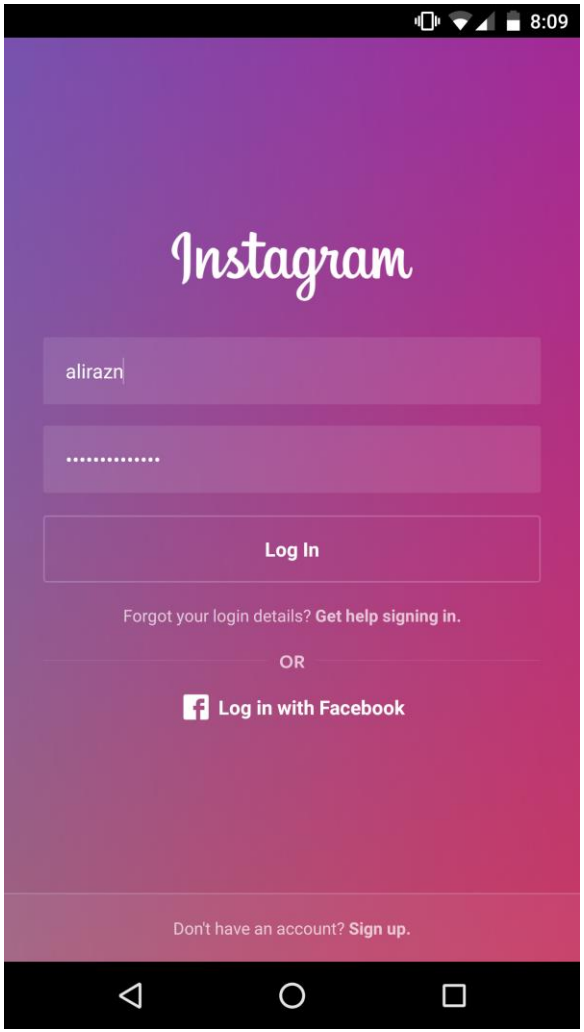


A Typical App Today



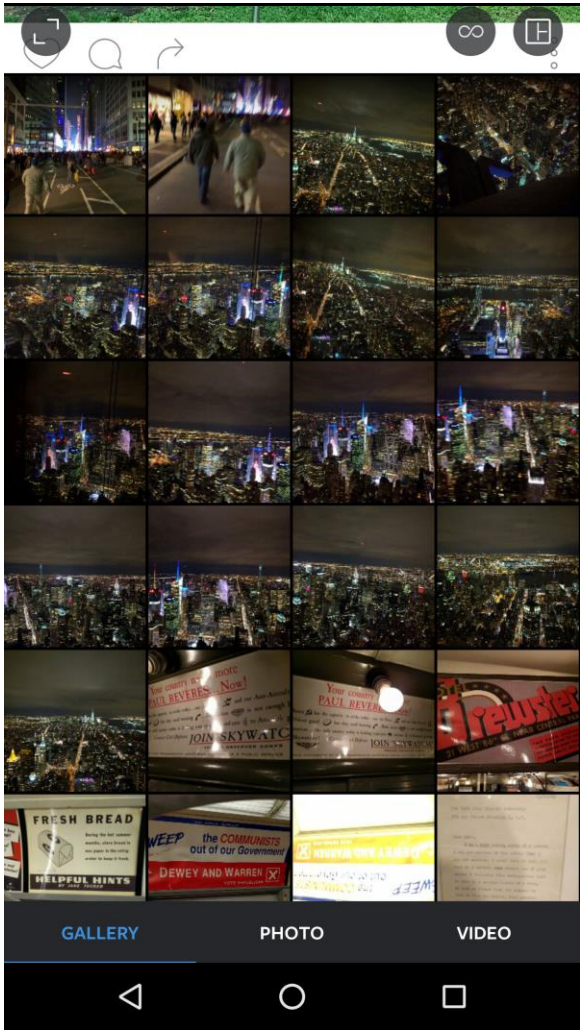


“Is my password safe?”



- Saved to a text file?
- Sent in clear text over the network?
- Sent to a suspicious server?

“… and what about my other data?”



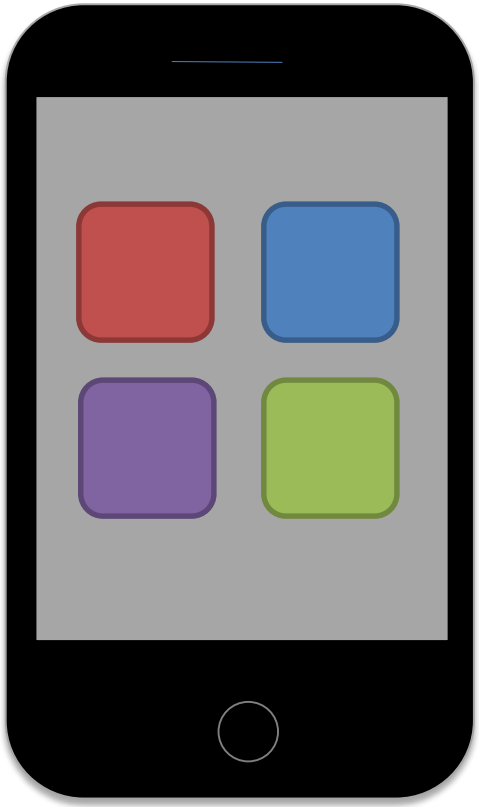
- Are copies of my data created?
- Are other apps accessing the copies?
- Is something “bad” happening?

How do we track how apps use our data?

Dynamic Taint Tracking

Track tainted data as it is used.

Taint Propagation



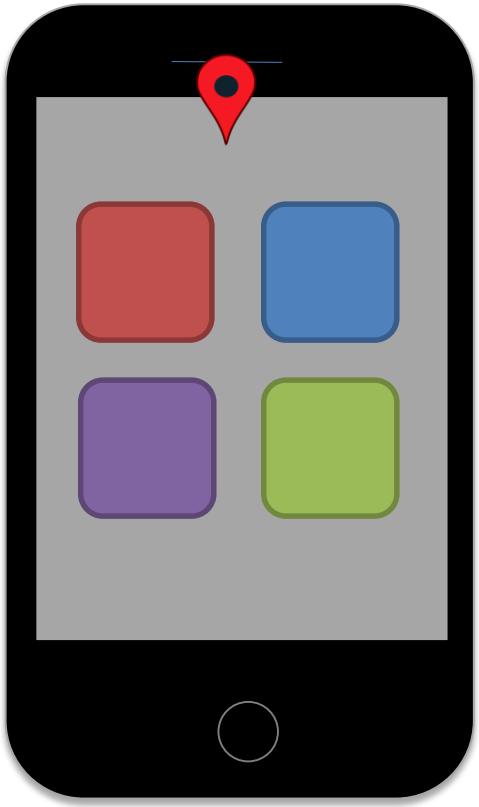
Taint Sources

What data should we track?

How may tainted data leave the system?

Taint Sinks

Dynamic Taint Tracking



```
01: double lng, lat = GetLocation();

02: double newLng = filter(lng);
03: double newLat = filter(lat);

04: sendToServer(newLng, newLat)
```

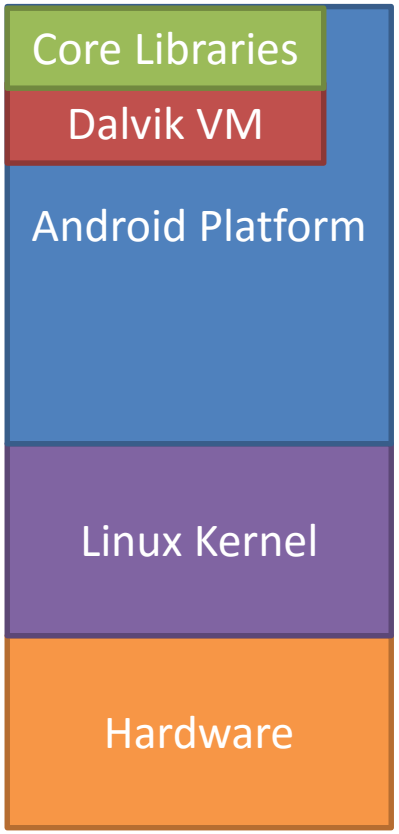
Uses of Dynamic Taint Tracking

- Attest to the authenticity of sensor data.
  YouProve (Gilbert'11)
- Minimize exposure of sensitive data.
  Clean OS (Tang '12)
- Manage app data in logical units.
  Pebbles (Spahn '14)
- Track how apps use passwords.
  SpanDex (Cox '14)
- Improve energy efficiency of apps.
  MobileHub (Shen '15)

Not just for detecting and defeating malware! **TaintCheck** (Newsome '05), **Panorama** (Yin '07)

TaintDroid [Enck '10]

- Implemented within the Dalvik VM.
  - (Dalvik is a managed runtime like the Java VM.)
- Reasonably good performance.
  - Overhead of about 14%.
  - Running apps on a VM is slow to begin with.
  - Taint tracking logic does not add significant overhead.



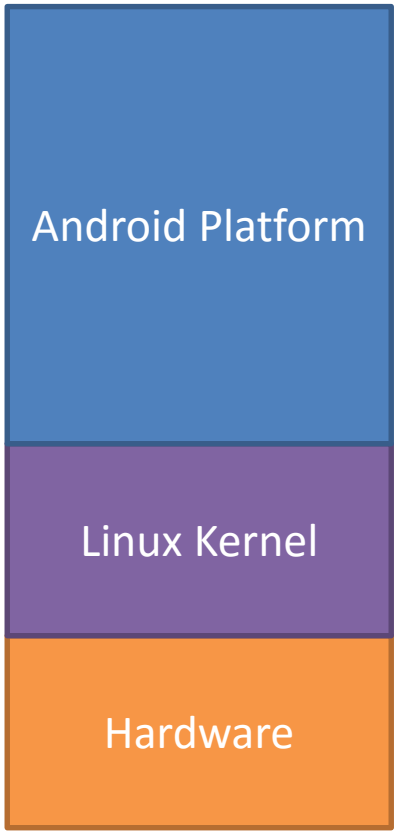
The Trend to Native Code

- The new Android Runtime (ART):
  - Compiles apps from bytecode to native.
- Developer-supplied native code:
  - Native libraries used for performance reasons
    - (E.g.: To perform computer vision.)

TaintDroid will not work anymore.

- Native code taint tracking is expensive!
  - 10x to 30x slowdown.

TaintCheck (Newsome '05), **Dytan** (Clause '07)



Selective Taint Tracking

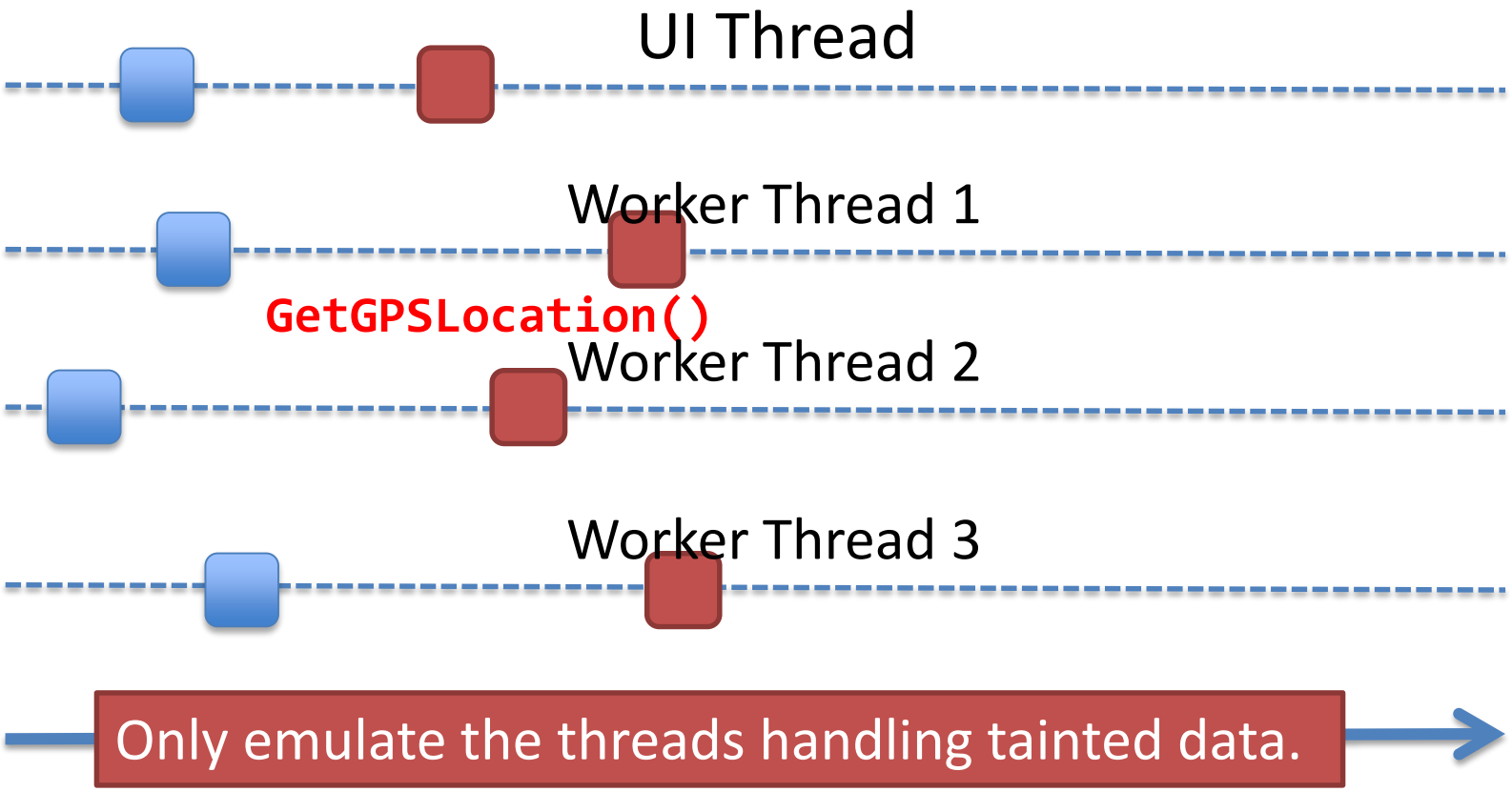
- Perform tracking only when passwords are used.
  - 1. Disassemble an app's instructions.
  - 2. Perform taint propagation.
  - 3. Execute the app instruction.
  - 4. Proceed until app stops using tainted data.
- Otherwise, apps run normally.

Overhead is amortized over app's lifetime!

- Implemented using page protections.

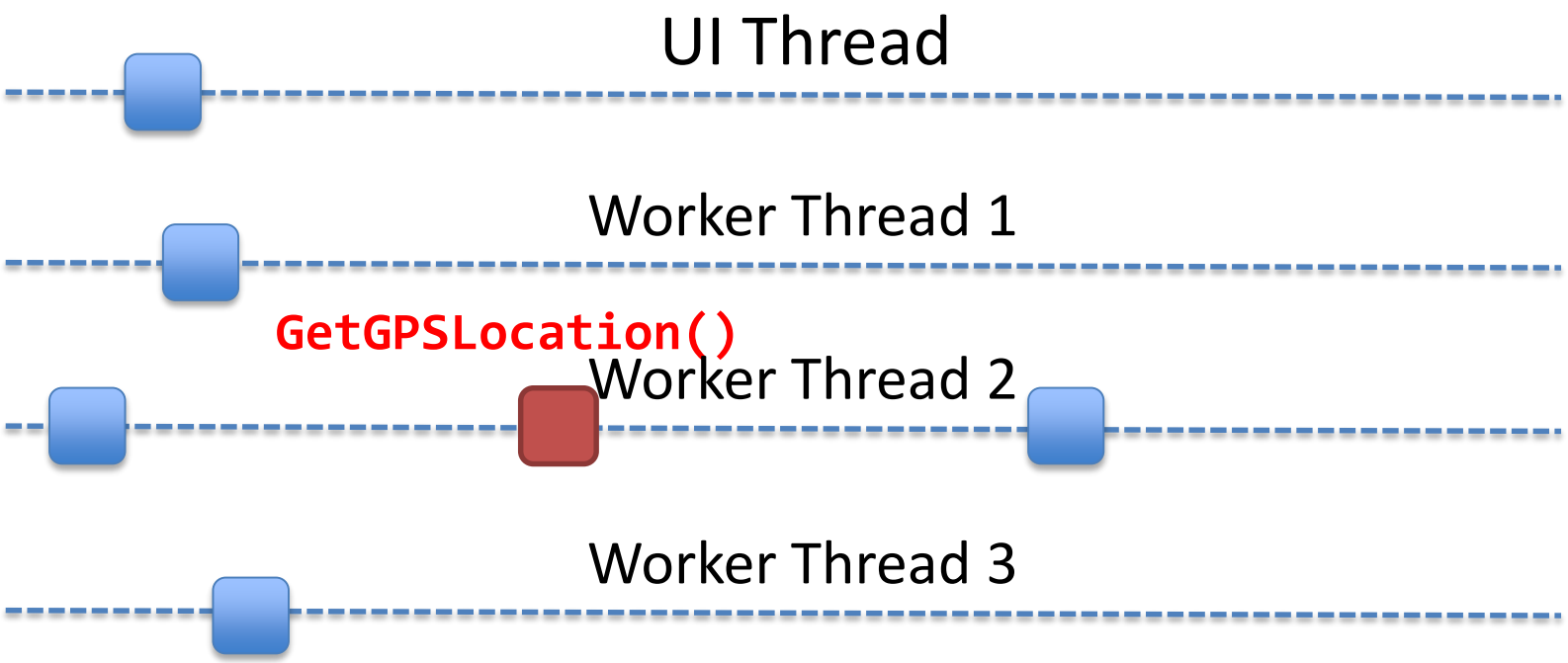
  Demand Emulation (Ho '06)

Apps are multi-threaded



Thread-local Emulation

Only the threads handling tainted data are emulated.



How can we interpose on memory accesses on a thread-level basis?

Implementation Techniques

- Software Techniques:
  - Virtual page table tricks. (Appel and Li '91)
- Hardware Techniques:
  - ARM Memory Domains.
  - Virtualization Features. **Dune** (Belay '12)

Look out for my HotMobile 2016 paper and come chat with me!