Peeking into Your App without Actually Seeing It: UI State Inference and Novel Android Attacks

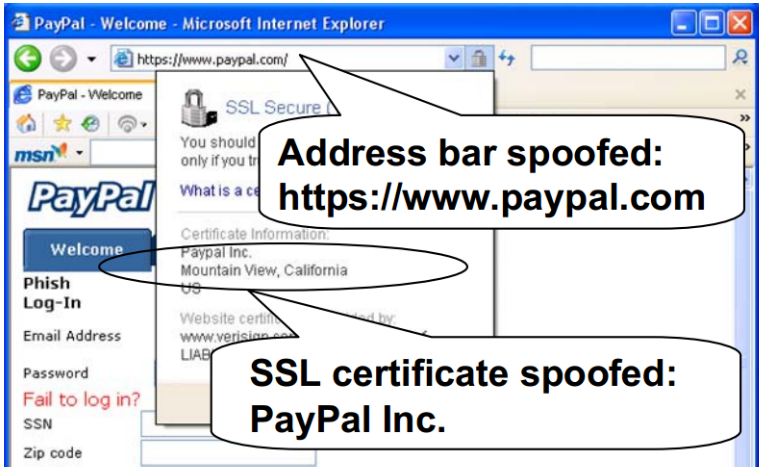
Qi Alfred Chen, Zhiyun Qian†, Z. Morley Mao

University of Michigan, †University of California - Riverside

RobustNet Research Group
University of Michigan

Importance of GUI Security

- GUI content confidentiality and integrity are critical for end-to-end security
  - UI Spoofing in desktop/browsers[1]
  - Screenshot capture on Android without privilege[2]

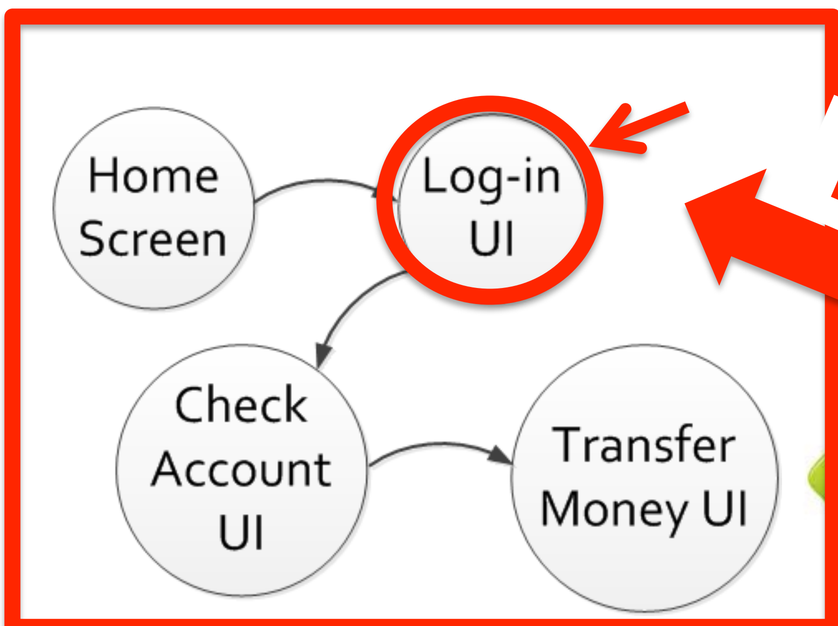


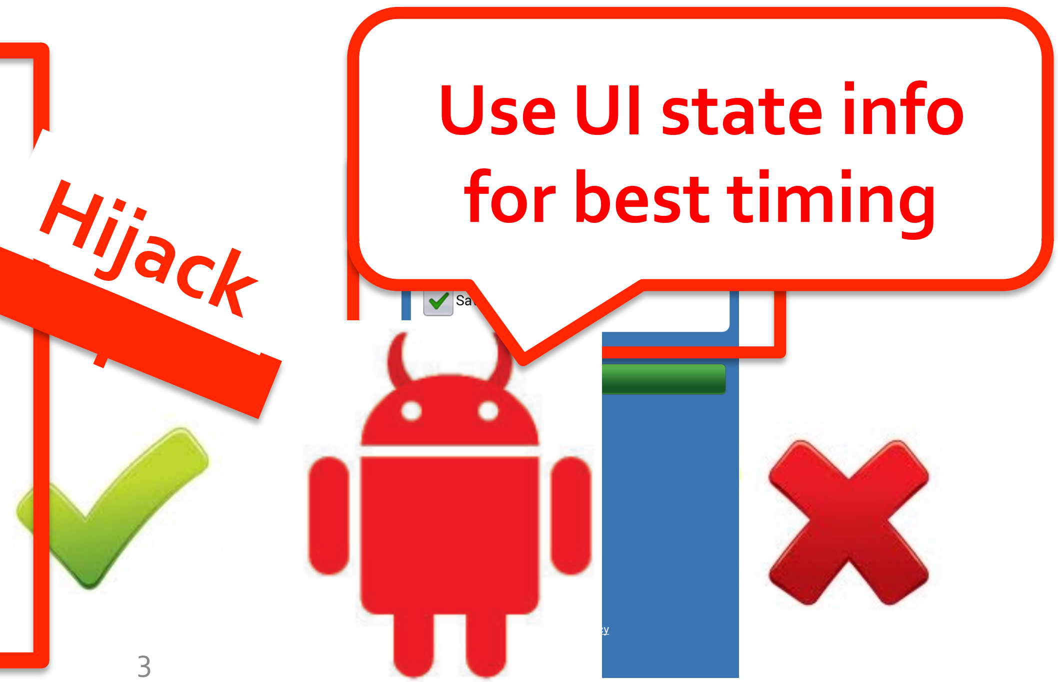
[1]Chen, Oakland'07

[2]ScreenMilker, NDSS'14

Another Form of GUI Confidentiality Breach

- A weaker form
  - UI state an app is in (*e.g., login state*) ***without knowing the exact pixels of the screen***





Serious security implications!

Enabled Attack: UI State Hijacking

- Hijack sen private input

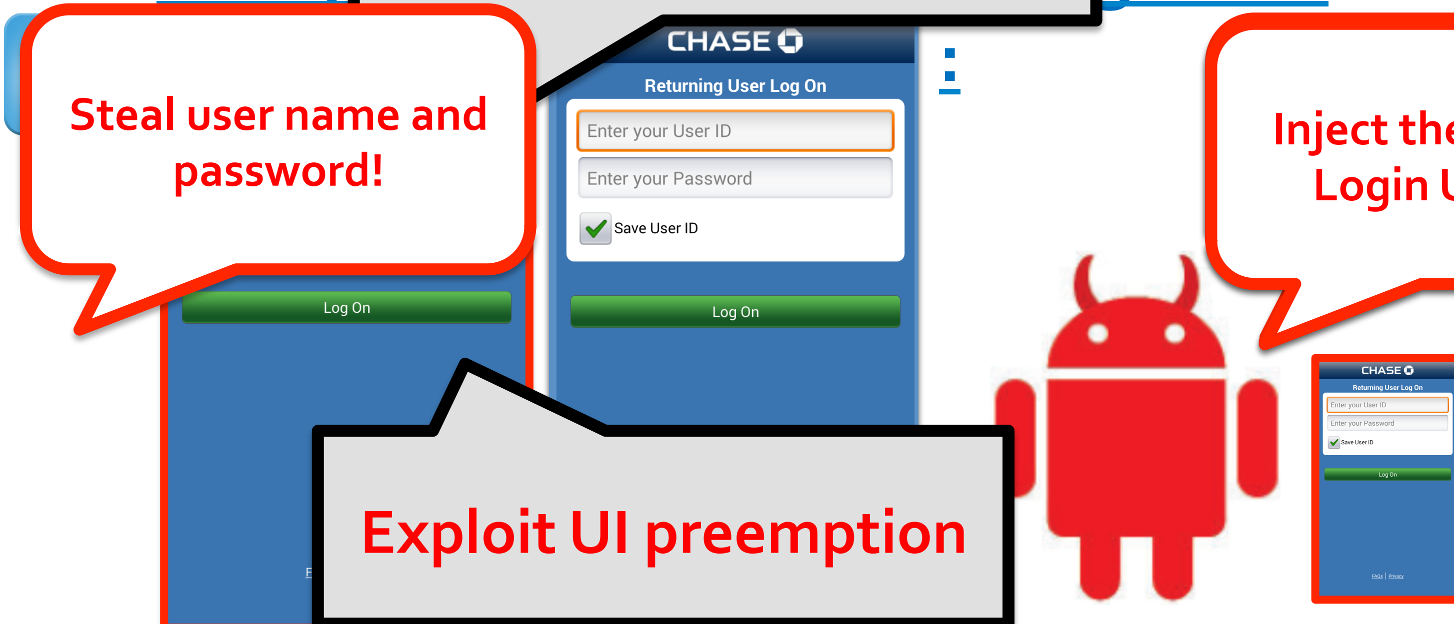
No glitches as we **disable the animation**

+ precise attack timing

Foreg ground

Steal user name and password!

Inject the phishing Login UI state!



Exploit UI preemption

UI State Hijacking Attack Demo

- Video demo: UI state hijacking attack steals your **password** in H&R Block app

Other Enabled Attacks

- An enabled attack: camera peeking
  - Steal **<u>sensitive pictures</u>** taken in Android apps

  - Breaks GUI confidentiality!
- Monitor and analyze user behavior
  - Breaks GUI confidentiality!
- Enhance existing attacks in both stealthiness and effectiveness

UI State Leakage is Dangerous

- Lead to both GUI **integrity** and **confidentiality** breaches
- UI state information **is not protected well**
  - An **unprivileged application** can track another app's UI states in real time

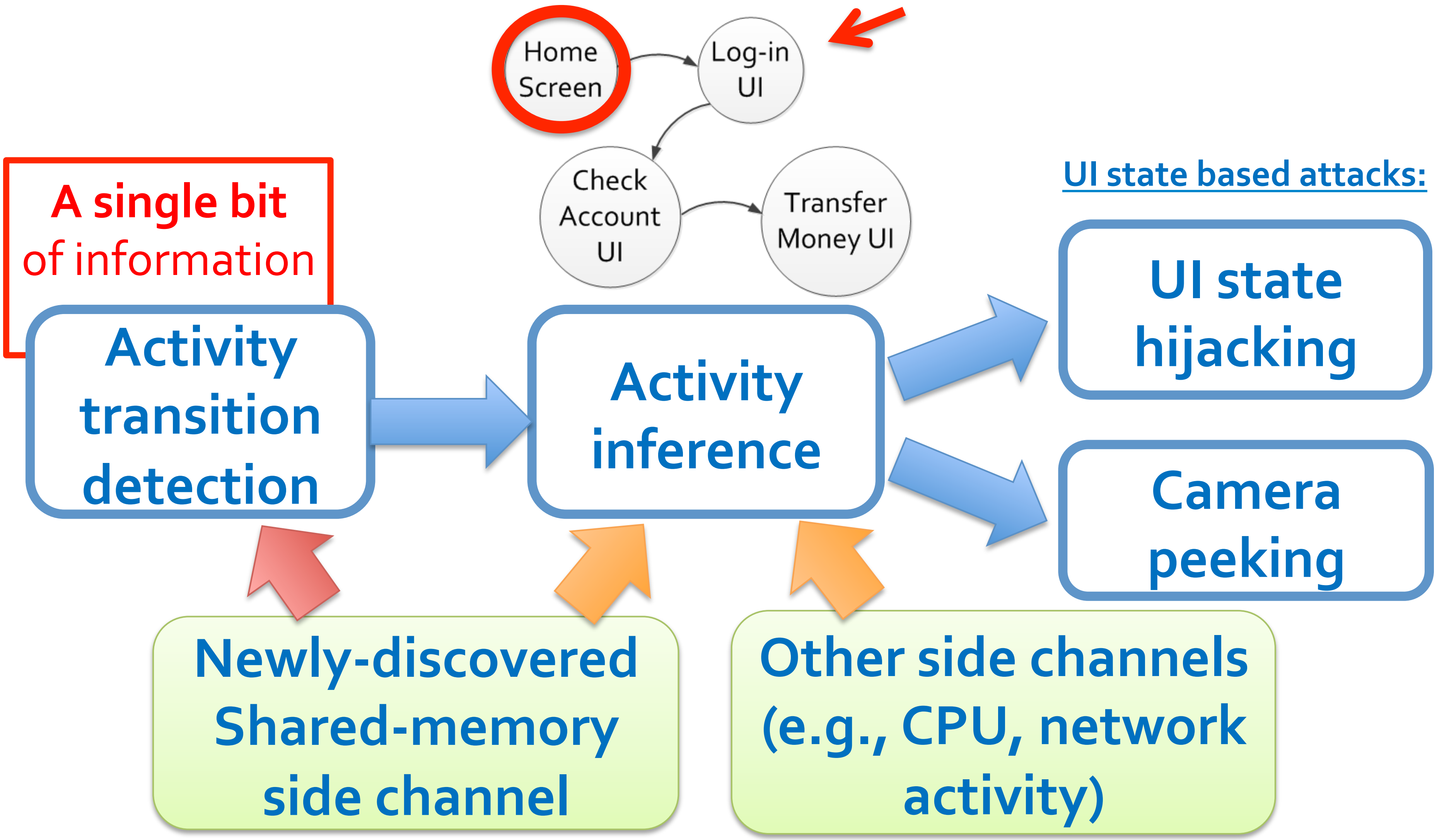
UI State Inference Attack

- **UI state**: a mostly consistent UI **at window level** for certain functionality (e.g., log-in)
  - On Android: **Activity (full-screen window)**
- Also called **Activity inference attack**
  - An unprivileged app can infer the foreground Activity in real time
  - Requires **no permission**

Underlying Causes

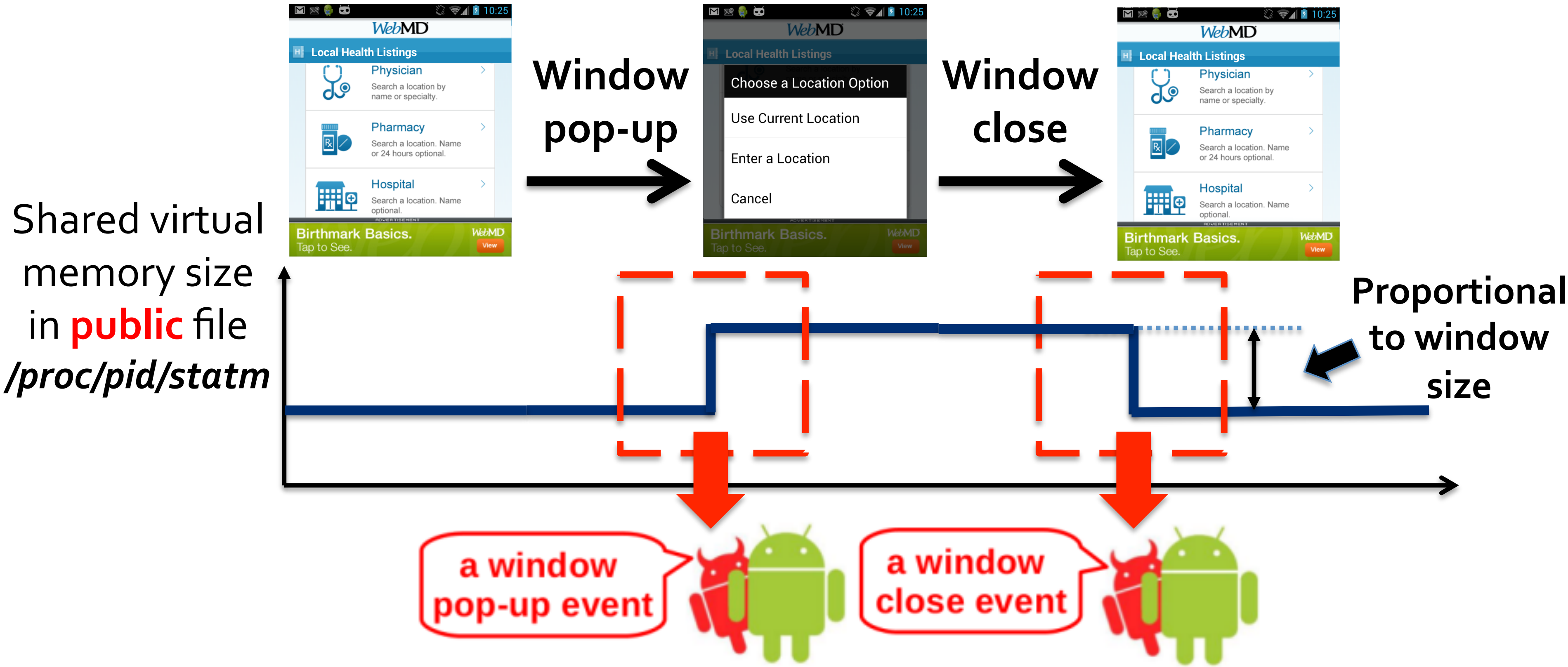
- **Android GUI framework design** leaks **UI state changes** through ***a publicly-accessible side channel***
  - A newly-discovered shared-memory side channel
  - Affects nearly **all popular OSes**

Attack General Steps



Shared-Memory Side Channel

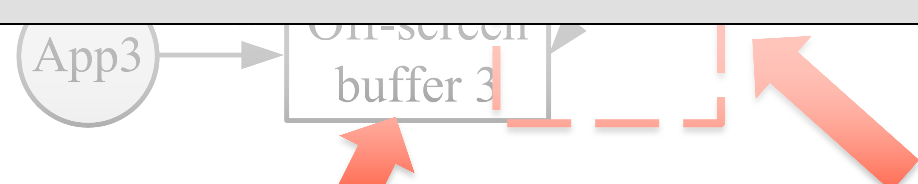
- **Finding**: shared virtual memory size changes are correlated with Android window events



Shared-Memory Side Channel

- Root cause for this correlation

Confirmed that **shared memory is used in GUI design for many OSes,** including



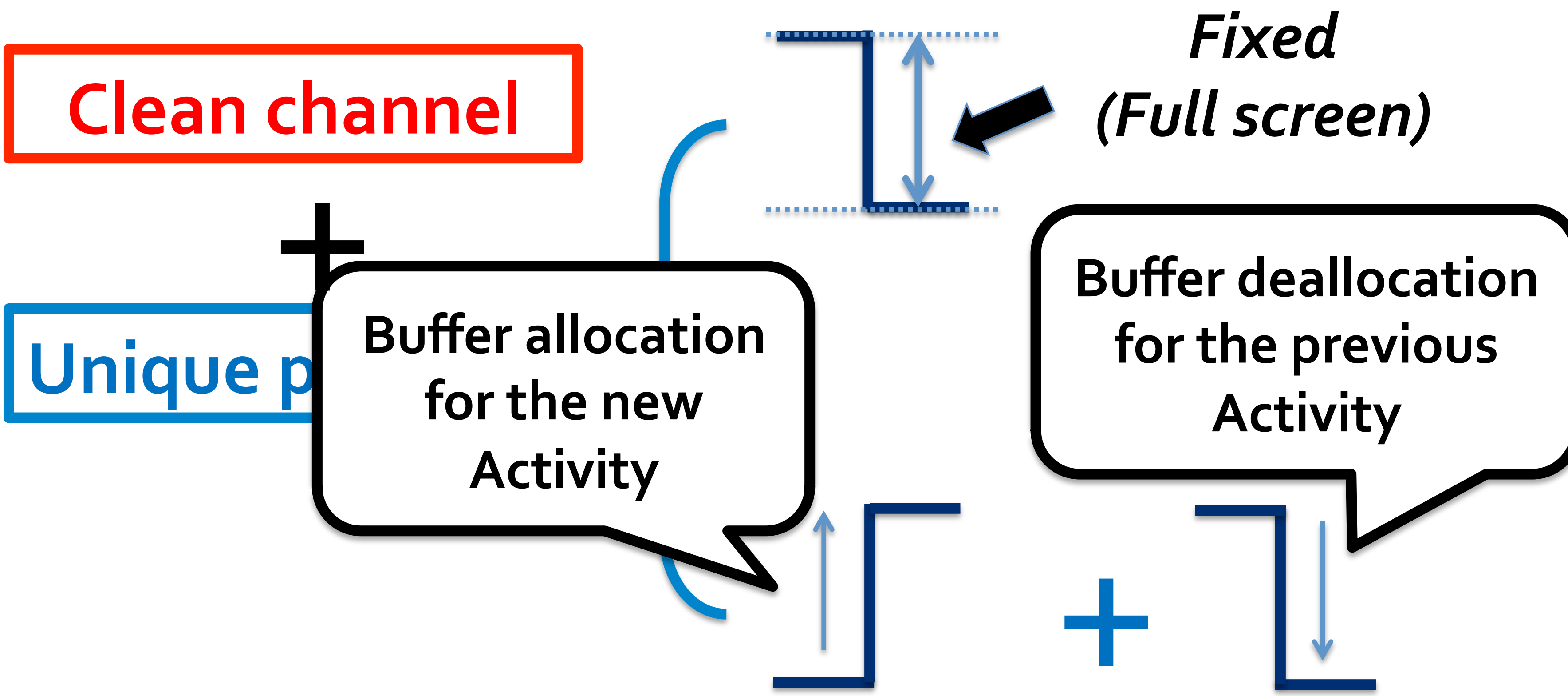
The changed size is the off-screen buffer size

The root cause is here

For better UI drawing performance, Android uses **shared memory** as IPC

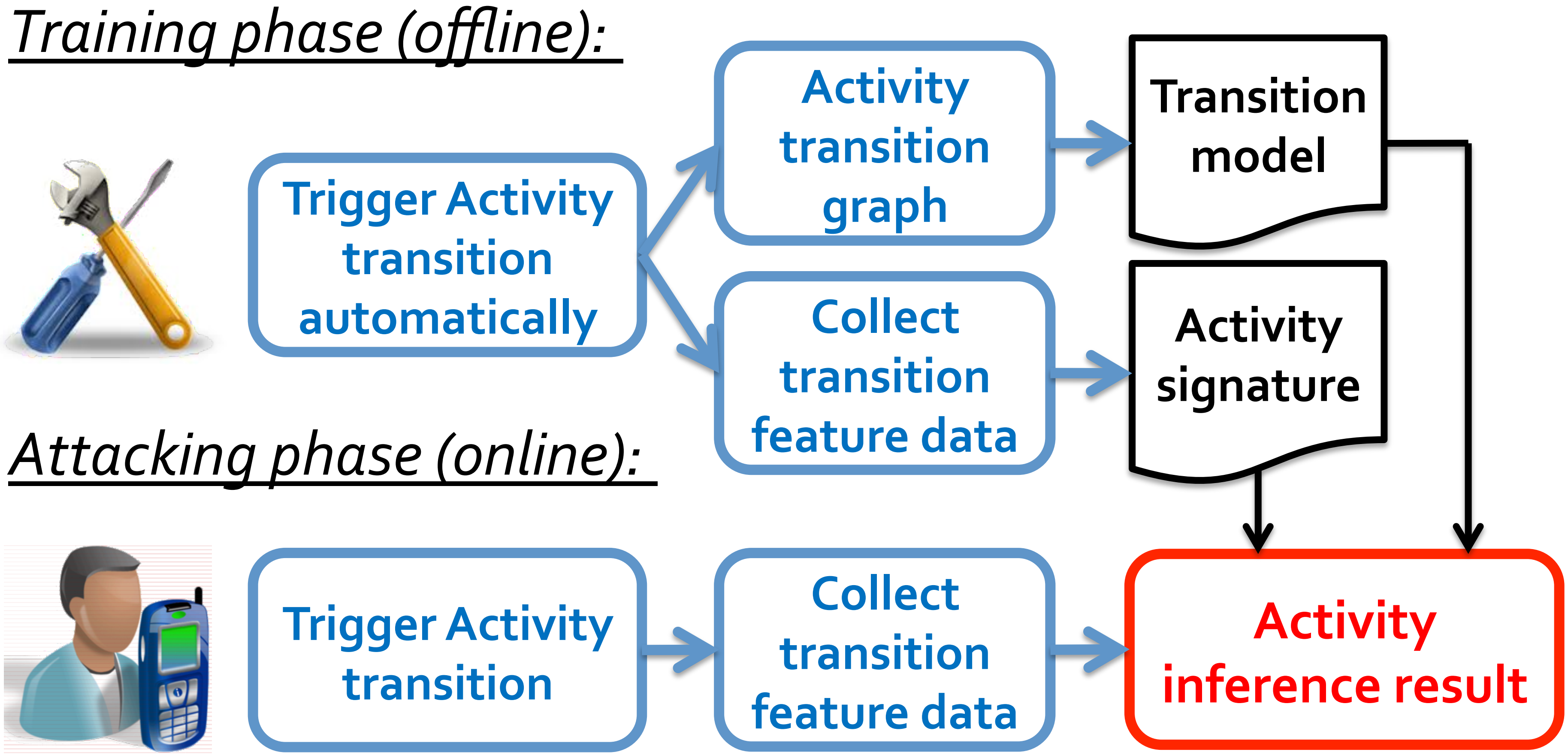
Activity Transition Detection

- Detect shared-memory size change pattern
  - **<u>Nice properties</u>**:



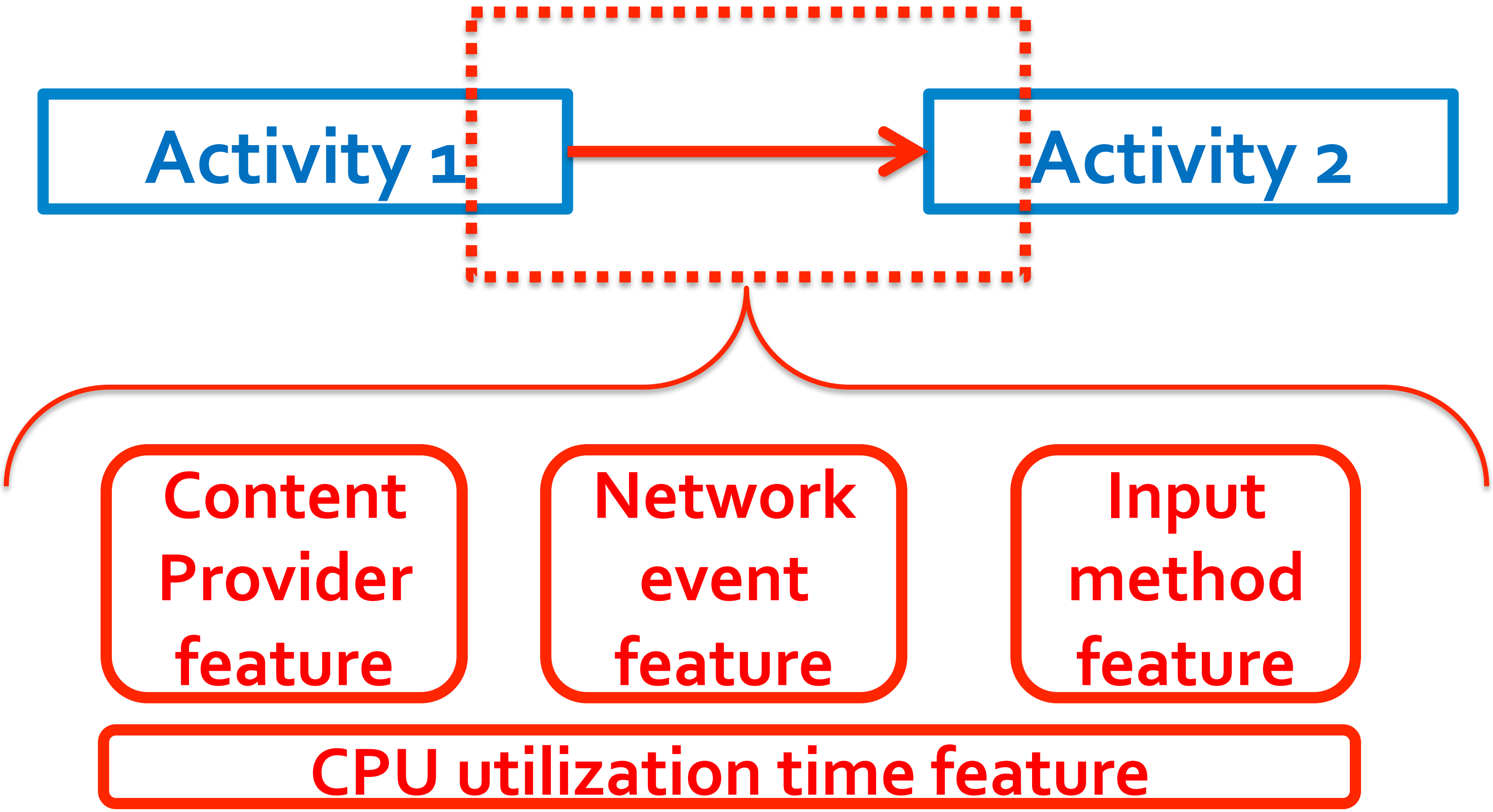
Activity Inference

- **Activity signature + Activity transition graph**



Activity Signature Design

- Consists of various features



Remaining Steps of Activity Inference

- **Create an Activity transition model**
  - Hidden Markov Model (HMM)
- **Inference results**
  - A list of Activities in decreasing order of their probabilities

Evaluation Methodology

- **Implementation**: ~ 2300 lines of C++ code compiled with Android NDK
- **Data collection**: using automated Activity transition tool on Samsung Galaxy S3 devices with Android 4.2
- **Experimented on 7 popular Android apps:**

Evaluation Results

- **Activity transition detection**, for all apps
  - Detection accuracy **≥ 96.5%**
  - FP and FN rates both **≤ 4%**
- **Activity inference accuracy**
  - **80–90%** for 6 out of 7 popular apps
    - **Important features**: CPU, network, transition model
- **Inference computation & delay**
  - Inference computation time: **≤ 10 ms**
  - Delay (Activity transition → inference result): **≤ 1.3 sec**
    - Improved to **≤ 500 ms** for faster and more seamless Activity hijacking
- **Overhead**
  - Increase power usage by **2.2–6.0%**

Defense Discussion

- **Eliminate the side channel**
  - **Proc file system access control**
    - Android already limits some, but more is needed
  - **Window buffer reuse**
    - Pre-allocate double the buffers and reuse them
    - More memory consumption (<u>several MBytes per buffer</u>)
- **Mitigate those follow-up attacks**
  - For example, for UI state hijacking
    - Build trusted paths between user and app
- Defense is non-trivial, more effort is required

Summary

Demonstrated serious security implications for a new form of GUI confidentiality breach

- Formulated a general UI state inference attack
  - Infer UI state in real time
- Discovered a new side channel for UI state inference
  - Potentially affecting all popular GUI systems
- Designed and implemented it on Android, and further built several new attacks (e.g., UI state hijacking)
- Attack video demos at our website

http://tinyurl.com/UIStateInference

- Questions?

http://tinyurl.com/UIStateInference