IEEE S&P 2016

No Pardon for the Interruption: New Inference Attacks on Android Through Interrupt Timing Analysis

May 24, 2016

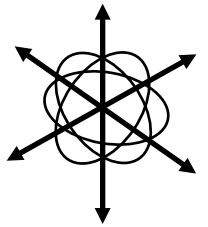
Wenrui Diao, Xiangyu Liu, Zhou Li, and Kehuan Zhang

香港中文大學
The Chinese University of Hong Kong

Motivation -- Hardware and Kernel

- Mobile platform – mobility and usability
- New specialized hardware components



- Previous research → particular hardware components
  → reading data directly from sensors

Q: What about the security implications of the integration of specialized **hardware** and tailored **kernel**?

Main Idea -- Hardware Interrupt

- Android inherits the **interrupt mechanism** from Linux.
- Efficient communication method between CPU and external devices.

- **Public** interrupt statistical information: **`/proc/interrupts`**

```
shell@shamu:/ $ ls -l /proc/interrupts
-r--r--r-- root     root            0 2016-04-13 14:39 interrupts
```

- Reflect the **real-time running status** of devices
  - Inference attack!
  - New attack surface!

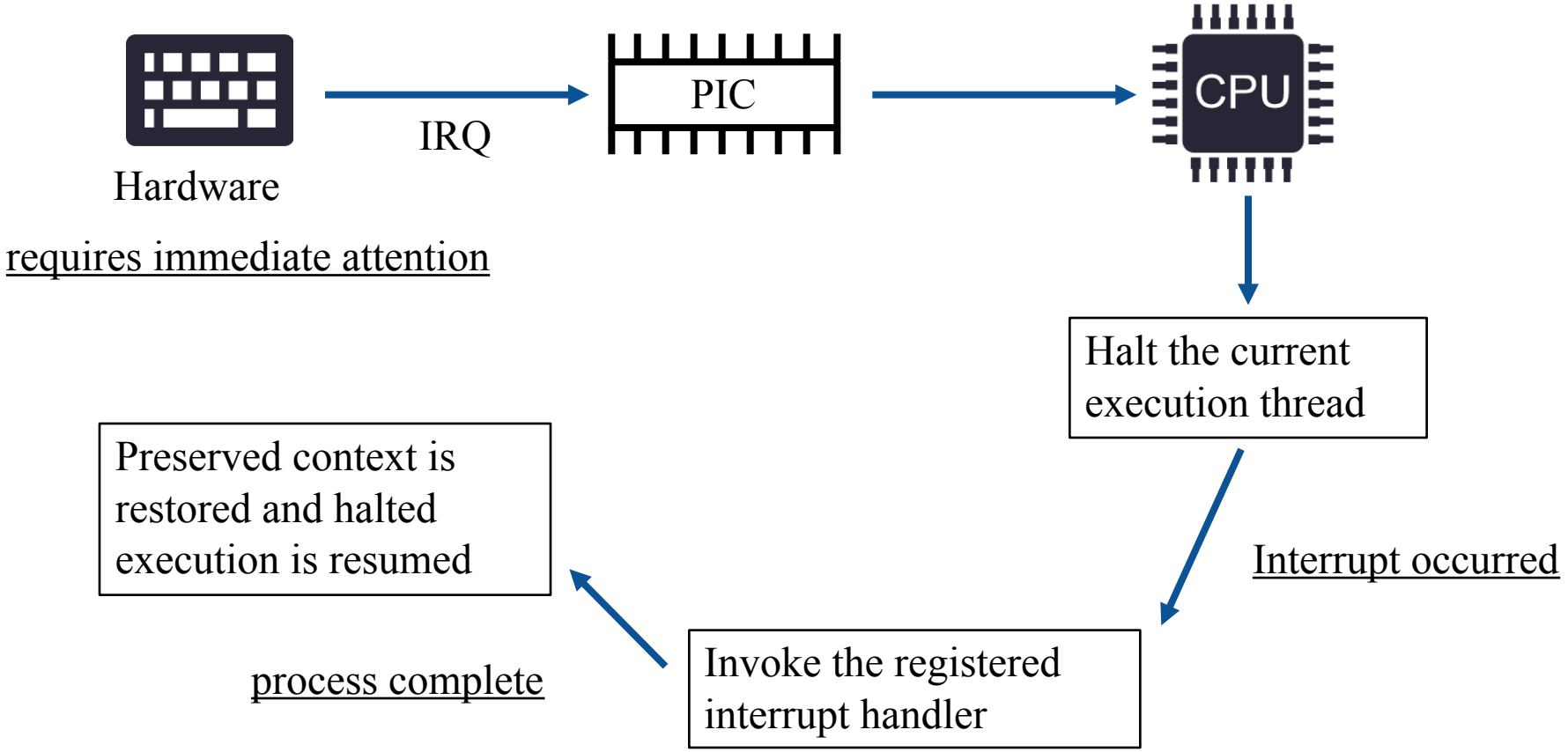
Main Idea -- Interrupt Timing Analysis

A: Through analyzing the **time series of interrupts** occurred for a particular device, user's **sensitive information** could be inferred.

- Root Cause: ill-conceived integration of specialized hardware components and tailored kernel.
- Gifts from mobile platform → new hardware components
  → interact with user directly
- Related work: Zhang et al. Usenix'09, Jana et al. S&P'12

Background -- Hardware Interrupt Mechanism

- Enable timely event management



Public /proc/interrupts on Linux

```
shell@shamu:/ $ cat /proc/interrupts
           CPU0       CPU1       CPU2       CPU3
 20:      41662      16103      14960      14905       GIC  arch_timer
 25:          0          0          0          0       GIC  MSM_L1
 33:       2007          0          0          0       GIC  bw_hwmon
 34:          0          0          0          0       GIC  MSM_L2
 35:          0          0          0          0       GIC  apps_wdog_bark
 39:       2162       1375       1354        819       GIC  arch_mem_timer
 61:         85          0          0          0       GIC  mxhci_hsic_pwr_evt
 64:       6022          0          0          0       GIC  xhci-hcd:usb1
 65:       5434          0          0          0       GIC  kgsl-3d0
 74:          0          0          0          0       GIC  msm_iommu_nonsecure_irq
 75:          0          0          0          0       GIC  msm_iommu_secure_irq, msm_iommu_secure_irq
 76:        616          0          0          0       GIC  msm_vidc
 78:          0          0          0          0       GIC  msm_iommu_secure_irq, msm_iommu_secure_irq
 79:          0          0          0          0       GIC  msm_iommu_nonsecure_irq
 81:          2          0          0          0       GIC
```

The amount of interrupts occurred

- Counter update → Interrupt occurred → Event coming

Concrete Attack Showcases

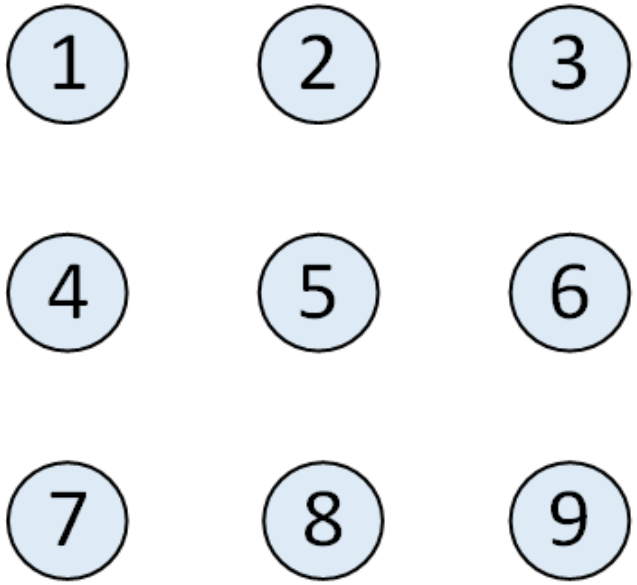
- **General Approach: Interrupt Timing Analysis**

- Inferring unlock pattern -- Touchscreen Controller

- Inferring foreground app -- Display Sub-System (DSS)

Attack Case 1 -- Touchscreen and Unlock Pattern

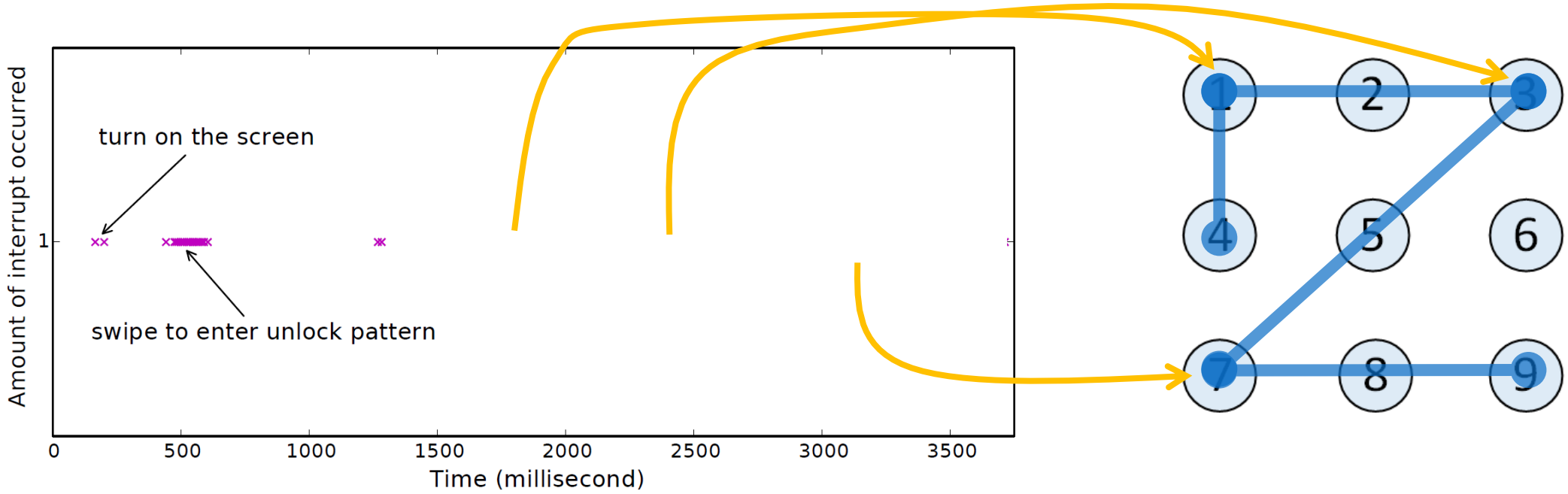
- Touchscreen: A large amount of user's sensitive information pass through.



- Unlock pattern
- Overcome the usability
- 3 × 3 matrix
- Connect dots in a certain order

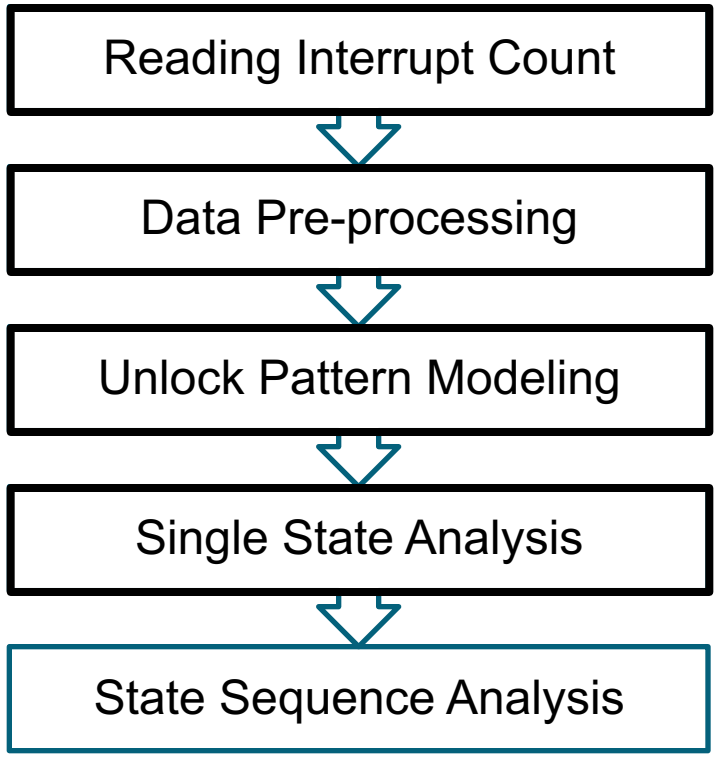
Touchscreen Controller and Interrupt

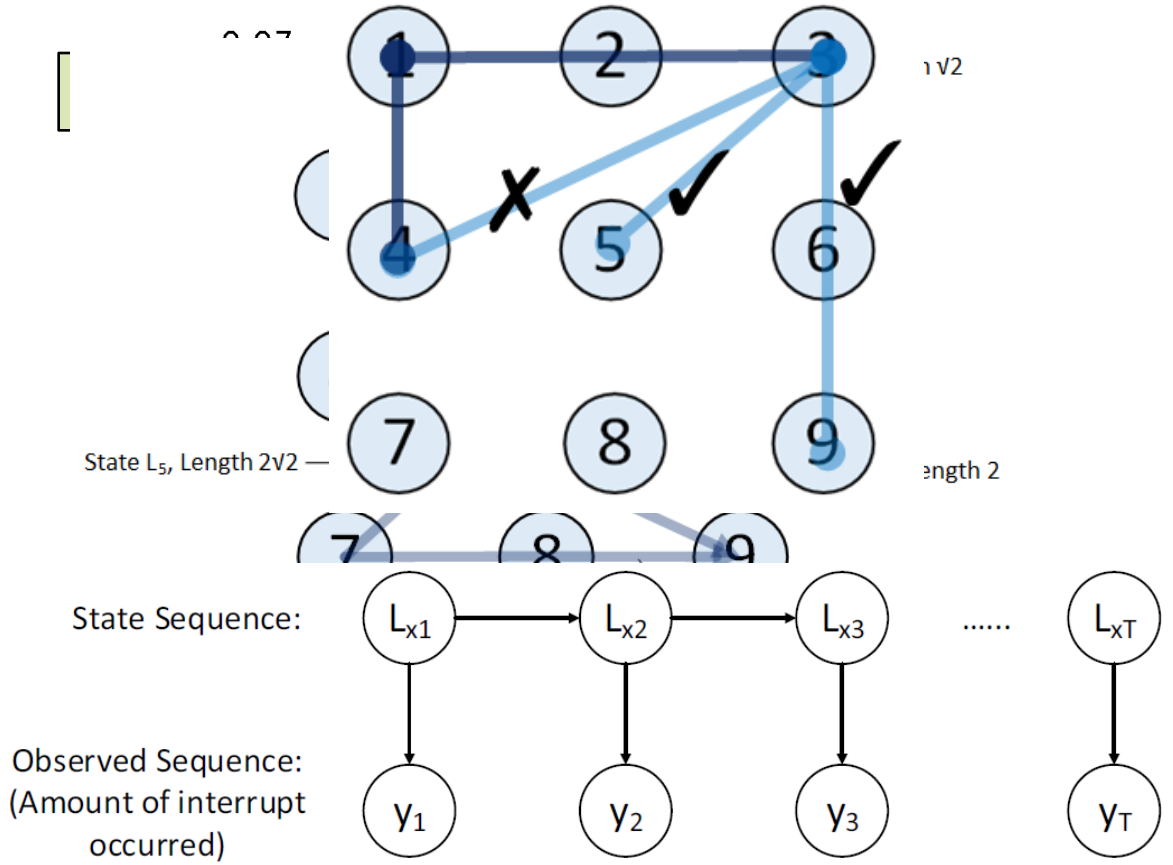
- Touch/Leave the touchscreen -- Interrupt



- Different lines could result in different interrupt sequences and a gap could be observed between lines' interrupts.

Inferring Unlock Pattern -- Work Flow





Derive the state sequence, solve HMM

Inferring Unlock Pattern -- Experiment

- Target **all 389,112** patterns, without training specific pattern in advance.
  - Cai et al. HotSec'11 → 1 pattern, Aviv et al. ACSAC'12 → 50 patterns

- Five users to get the length-interrupt relationship (Gaussian-like model).
- Another two users joined the testing phase.

- In total, obtain 160 password patterns from each user
  - Draw each generated pattern two times.
  - Consider 2-gram, 3-gram, 4-gram and 5-gram types.
  - Randomly generated 20 patterns for each type.

Inferring Unlock Pattern -- Result

Success Rate for Gram Segmenting (Gap Searching)

Pattern	Search Space Reduction	Success Rate
2-gram	389,112 → 168	98.75%
3-gram	389,112 → 2,544	92.5%
4-gram	389,112 → 11,048	97.5%
5-gram	389,112 → 37,160	97.5%

Search space has be substantially reduced.

Inferring Unlock Pattern -- Result

Success Rate for State Sequence Inference

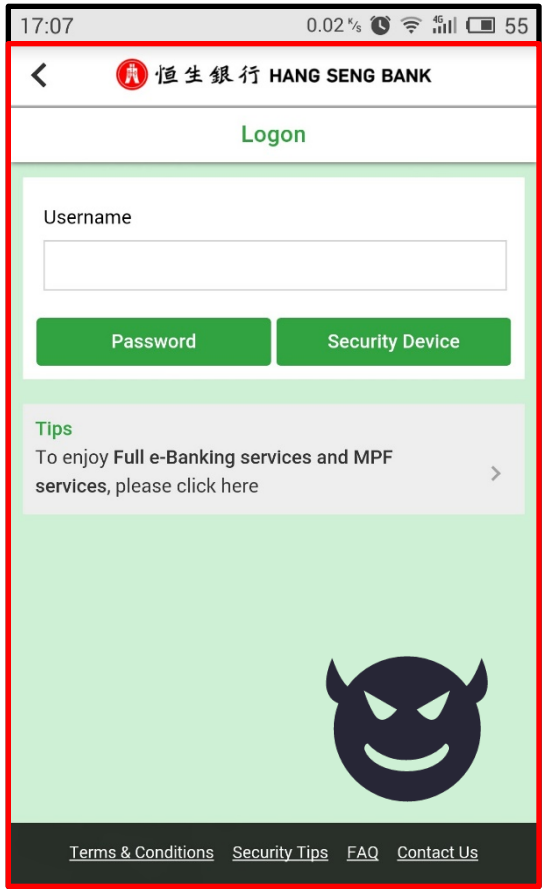
User #	Top N	2-gram	3-gram	4-gram	5-gram
User 1	*Top 3*	50%	25%	7.5%	0
	Top 5	80%	27.5%	10%	0
	Top 10	97.5%	40%	20%	2.5%
	Top 20	97.5%	60%	37.5%	12.5%
	Top 40	97.5%	90%		
User 2	*Top 3*	45%	20%		
	Top 5	62.5	22.5		
	Top 10	95	35		
	Top 20	100	50	40	20
	Top 40	100	70	57.5	22.5

Random guess: 0.0157% (guessing 3 times)

Improve up to thousands of times

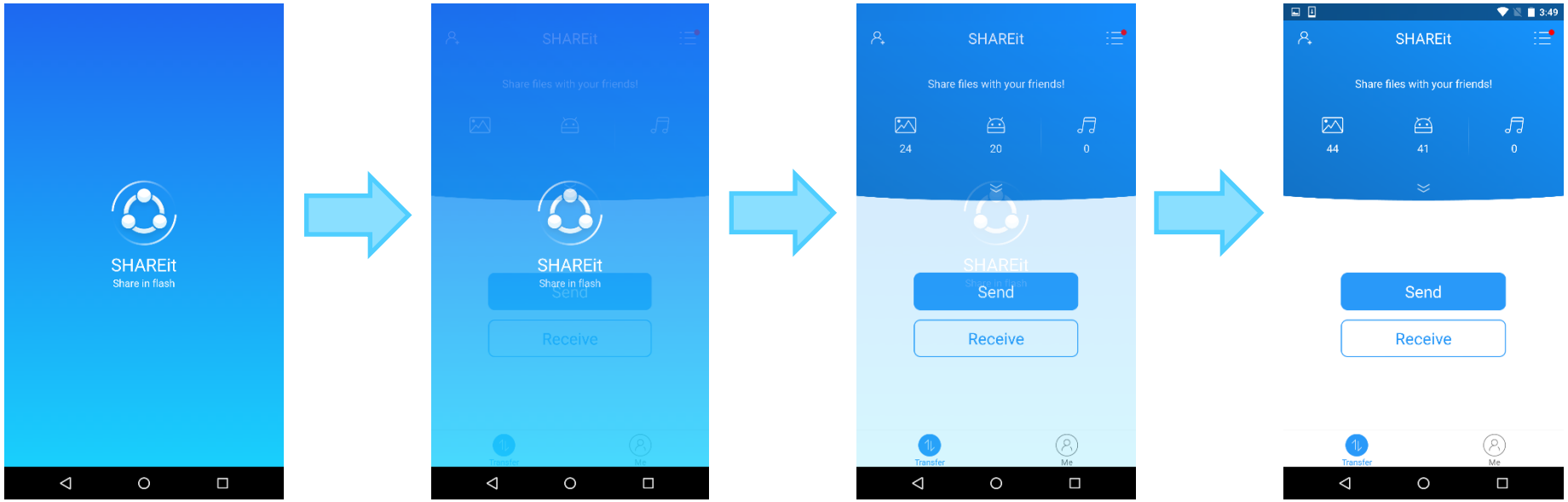
Attack Case 2 -- App Running in the Foreground

- Phishing attacks



UI Refreshing and Interrupts

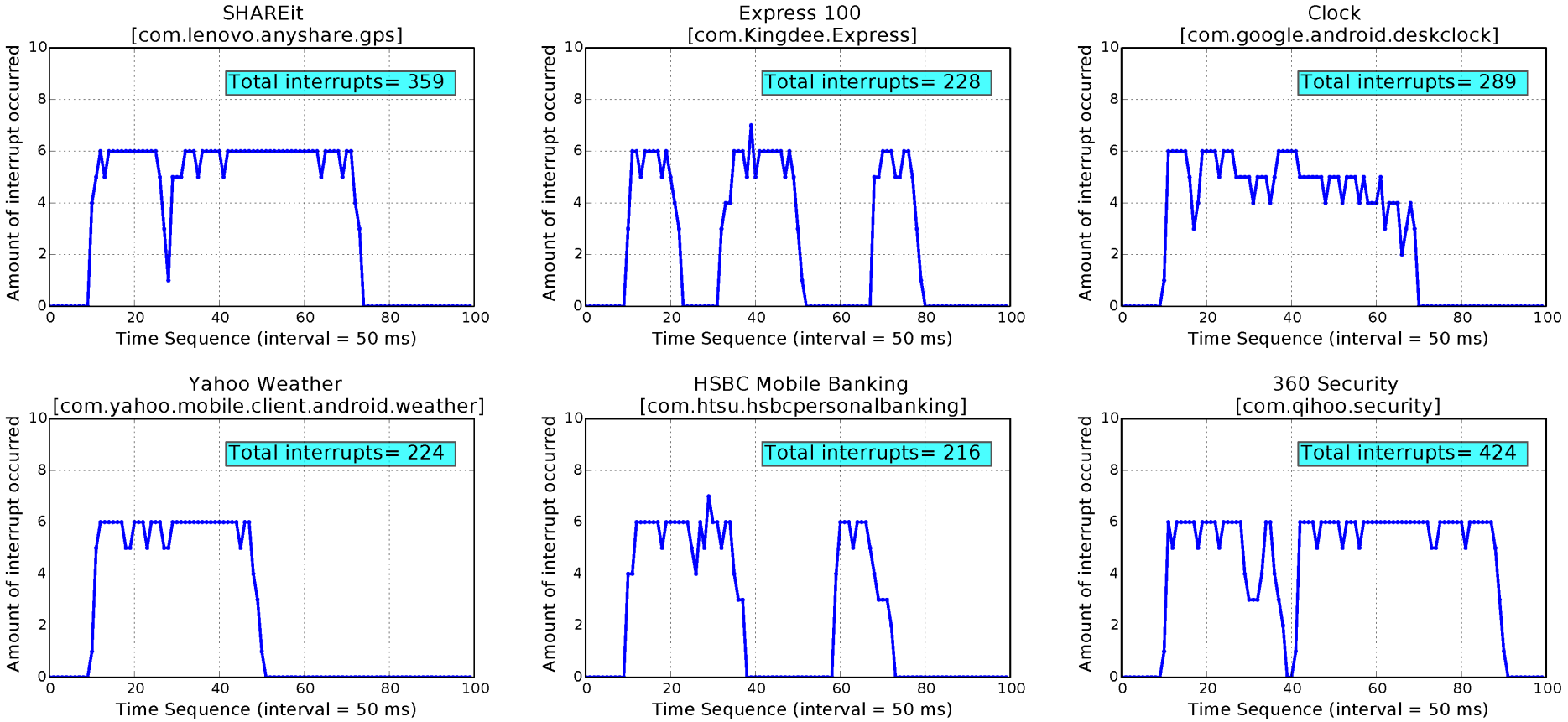
- Foreground UI is continuously refreshed.



- UI Refreshing -- Display Sub-System (DSS) → Interrupt request (vsync)
- Different UI layout and refreshing strategies – different interrupt time series

UI Refreshing and Interrupts

Interrupt patterns of 6 apps' launching processes



One-page Take-away

- New attack surface in the interrupt handling mechanism: public /proc/interrupts
- Counter update → Interrupt occurred → Event coming
- General approach: interrupt timing analysis

- Concrete cases:
  - Touchscreen controller -- unlock pattern inference
  - Display Sub-System -- foreground app inference

- Defense: fine-grained access control, decreasing the resolution

Q&A

- Contacts:

- Wenrui Diao
- The Chinese University of Hong Kong
- Email: dw013@ie.cuhk.edu.hk
- Homepage: http://home.ie.cuhk.edu.hk/~dw013/

Backup: Inferring Foreground App -- Experiment

- Select 100 popular apps from Google Play to build the training set.
- Each app is launched 10 times, and 1,000 fingerprints are recorded in total.

- Testing set, we randomly select 10 apps from these 100 apps in the training set, run each one 10 times -- 100 fingerprints in total.

Backup: Inferring Foreground App -- Result

Success Rate for App Identification under different k (k-NN)

k	k=3	k=5	k=7	k=9
Top 1	77%	87%	83%	82%
Top 2	85%	91%	88%	90%
Top 5	93%	95%	94%	93%
Top 10	94%	96%	96%	98%

Backup: Inferring Foreground App -- Result

Success Rate for App Identification k=5

App Name	Top 1	Top 2	Top 5
tv.danmaku.bili	100 %	100 %	100 %
com.baidu.search	80 %	90 %	90 %
com.icoolme.android.weather	90 %	90 %	90 %
com.scb.breezebanking.hk	80 %	90 %	100 %
ctrip.android.view	50 %	50 %	60 %
com.lenovo.anyshare.gps	100%	100 %	100 %
com.sometimeswefly.littlealchemy	100 %	100 %	100 %
io.silvrr.silvrrwallet.hk	90 %	100 %	100 %
com.cleanmaster.mguard	100 %	100 %	100 %
com.ted.android	80 %	90 %	100 %