Storage Side Channel Attacks in Modern OS and Networking Stacks

--- How to break isolation in OS?

Zhiyun Qian
University of California, Riverside
Outline

- Background and methodology
- Android UI state inference
- Off-path TCP sequence number inference
  - Firewall-middlebox-enabled attacks
  - Host-based attacks
- Summary
Outline

- Background and methodology
- Android UI state inference
- Off-path TCP sequence number inference
  - Firewall-middlebox-enabled attacks
  - Host-based attacks
- Summary
Side channels - Real world example
Mafia game
Another example

Anyone at home?

???
OS Security Mechanism -- Isolation

- Memory isolation
OS Security Mechanism -- Isolation

- File system isolation

```
/home
  /home/alice
    drwx----- alice
  /home/bob
    drwx----- bob
```
OS Security Mechanism -- Isolation

- Android File system isolation

```
/data
 /data/app1
  drwx------ app1
 /data/app2
  drwx------ app2
```
OS Security Mechanism -- Isolation

- Exceptions
  - `/proc/[pid]/statm`
  - `/proc/net/netstat`
  - Etc.
Anyone at home?
What is a side channel attack?

- Information gained from the physical implementation of a cryptosystem, rather than brute force or theoretical weaknesses [1]
  - Timing, Power monitoring, Acoustic, Electromagnetic, etc.
  - Used as early as World War II.

Modern side channel attacks

- Information gained from the physical design and implementation of a crypto system, rather than brute force or theoretical weaknesses
  - Keystrokes (e.g., password) inference [Song01, Zhang09, Vuagnoux09, Chen10]
    - Timing, IPID, Power, Electromagnetic waves
  - Crypto key extraction through VM co-residency [Zhang12]
    - CPU cache

Passive

Clear input/output
Password authentication

```java
for(i = 0; i < len; i++) {
    if(input[i] != password[i]) {
        failed = true;
        break;
    }
}
```
Memory allocation

```c
secret_func() {
    malloc(1000KB);
    // ... computation
    malloc(1000KB);
    // ... computation
    malloc(1000KB);
    // ... computation
}
```
Research contributions

- Uncover a new class of storage side channel attacks against OS and networking stacks.
- Real-world security impact caused by OS design, firewall middleboxes, and network stacks.

- Google
- Linux kernel
- FreeBSD kernel

Check Point response to "Off-Path TCP Sequence Number Inference Attack"

**Symptoms**

- Researchers at the University of Michigan have published a paper "Off-Path TCP Sequence Number Inference Attack How Firewall Middleboxes Reduce Security".
- This attack identifies the current sequence range of a TCP connection, by exploiting the fact that firewalls drop out-of-window TCP packets. After the sequence range is identified, an off-path attacker may inject data or hijack the TCP connection.
- Client applications that use cleartext connections (e.g., HTTP and not HTTPS) are potential targets for these attacks.
Research methodology

Measurement-based characterization

OS component, Network policy, protocol behavior

Identification of sensitive state

Secret

Identification of side channels

Secret and side channels

Attack discovery

Vulnerability

Attack defense

Android GUI framework

UI State

Known side-channels

Source code analysis

Reverse engineering

Source code analysis

Principle-driven
Outline

- Background and methodology
- Android UI state inference
  - [USENIX SECURITY 14]
- Off-path TCP sequence number inference
  - Firewall-middlebox-enabled attacks
  - Host-based attacks
- Summary
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
Android OS

- App no root privilege
- App can request limited permissions (users have to agree)
- Apps isolated from each other
Android Security Mechanism -- Isolation

- Memory isolation
Android Security Mechanism -- Isolation

- File system isolation

```
/data/data
  /data/data/app1
    drwx------ app1
  /data/data/app2
    drwx------ app2
```
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

Use UI state info for best timing

Serious security implications!
Enabled Attack: UI State Hijacking

- Hijack sensitive UI state to steal private input
- No glitches as we disable the animation
- Precise attack timing
- 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.
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**

**A single bit of information**

- Activity transition detection
- Activity inference

**Newly-discovered Shared-memory side channel**

**Other side channels** (e.g., CPU, network activity)

**UI state based attacks:**
- UI state hijacking
- Camera peaking
**Finding**: shared virtual memory size changes are correlated with Android window events.

![Diagram showing the correlation between shared virtual memory size and Android window events.](image-url)
Shared-Memory Side Channel

- Root cause for this correlation

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

- Android
- iOS
- MacOS
- Windows

The changed size is the off-screen buffer size.

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

The root cause is here.
Activity Transition Detection

- Detect shared-memory size change pattern
  - **Nice properties:**

  1. **Clean channel**
  2. **Unique pattern**

  + Buffer allocation for the new Activity

  - Fixed (Full screen)

  - Buffer deallocation for the previous Activity
Activity Signature Design

- Consists of various features

Activity 1 → Activity 2

- Content Provider feature
- Network event feature
- Input method feature
- CPU utilization time feature
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:**
  - WebMD
  - Instagram
  - Amazon
  - Newegg
  - Gmail
  - H&R Block
  - Google Chrome
Evaluation Results

- **Activity transition detection**, for all apps
  - Detection accuracy $\geq 96.5\%$
  - FP and FN rates both $\leq 4\%$

- **Activity inference accuracy**
  - 80–90% for 6 out of 7 popular apps
    - **Important features**: CPU, network, transition model

- **Inference computation & delay**
  - Inference computation time: $\leq 10\text{ ms}$
  - Delay (Activity transition $\rightarrow$ inference result): $\leq 1.3\text{ sec}$
    - Improved to $\leq 500\text{ ms}$ for faster and more seamless Activity hijacking

- **Power overhead**
  - 2.2–6.0%

- **Status**
  - Working with Google now to fix the problem