Unveiling your keystrokes: A Cache-based Side-channel Attack on Graphics Libraries

Daimeng Wang, Ajaya Neupane, Zhiyun Qian, Nael Abu–Ghazaleh, Srikanth V. Krishnamurthy, Edward J. M. Colbert†, Paul Yu‡
University of California Riverside, Virginia Tech†, U. S. Army Research Lab‡
Introduction

- Graphics is essential
- Graphic rendering is complex
  - gdk, gtk, pixman, freetype, cairo, skia, hwui, …
Introduction

- Graphics is essential
- Graphic rendering is complex
  - gdk, gtk, pixman, freetype, cairo, skia, hwui, …
Example

static void D32_LCD32_Opaque(...) {
    ...
    do {
        blit_lcd32_opaque_row(dstRow, srcRow, color, width);
        dstRow = (SkPMColor*)((char*)dstRow + dstRB);
        srcRow = (const SkPMColor*)((const char*)srcRow + maskRB);
    } while (--height != 0);
}

static void blit_lcd32_opaque_row(dst, src, color, width) {
    ...
    for (int i = 0; i < width; i++) {
        if (0 == src[i]) {
            continue;
        }...
    }
}
Example

static void D32_LCD32_Opaque(...) {
    ...
    do {
        blit_lcd32_opaque_row(dstRow, srcRow, color, width);
        dstRow = (SkPMColor*)((char*)dstRow + dstRB);
        srcRow = (const SkPMColor*)((const char*)srcRow + maskRB);
    } while (--height != 0);
}

static void blit_lcd32_opaque_row(dst, src, color, width) {
    ...
    for (int i = 0; i < width; i++) {
        if (0 == src[i]) {
            continue;
        }
        ...
    }
}
Exploiting the Side-channel

- If an attacker can know when …
  - Text rendering procedure starts
  - Text rendering procedure ends
Potentially Vulnerable Apps
Example: Onboard

- Onscreen keyboard
  - Ubuntu 16.04

6. While using third party computers, use an on-screen keyboard while entering important details
Example: Onboard

- Onscreen keyboard
  - Ubuntu 16.04

6. While using third party computers, use an on-screen keyboard while entering important details.
Attack: Onboard

Measured Execution Time

Time (ns)
Attack: Onboard

Seems reasonable, but...
Challenges

› How to perform measurement?
  › Unprivileged attacker
Challenges

- How to find start/end of rendering?
  - Millions of code. Multiple libraries.
  - Varies from application to application.
Challenges

› Noise?
  › Victim’s noise
    › CPU cache, TLB, branch prediction, …
  › Attacker’s noise
Challenges

› Perform measurement without privilege

› Discover side-channel in graphic libraries

› Noise-resistant key prediction
Threat Model

▷ Attacker’s goal
  ▷ Eavesdrop sensitive text input that will be rendered on screen
    ▶ PIN, passwd, etc

▷ Attacker’s capabilities
  ▷ Access to same model/version of victim’s hardware and graphic libraries
    ▶ Offline profiling
  ▷ Launch unprivileged process alongside victim process
    ▶ Online attack
Performing Measurement

...
Performing Measurement

Physical Memory

libA.so
renderStart()
renderEnd()
...

renderStart()
renderEnd()

libA.so
renderStart()
renderEnd()
Performing Measurement

Physical Memory

libA.so
renderStart()
renderEnd()

CPU LLC

libA.so
renderStart()
renderEnd()
Measurement Resolution

- Onboard
  - ~ 600ns per round
  - ~ 350,000ns rendering
Side-channel Discovery

- Instrument graphic libraries & collect victim trace

\[ a_0 \quad a_1 \quad a_2 \quad a_3 \quad a_4 \quad a_5 \]
Side-channel Discovery

- Instrument graphic libraries & collect victim trace

![Diagram of side-channel discovery process with two timelines labeled 'a' and 'b', each containing five segments labeled $a_0$, $a_1$, $a_2$, $a_3$, $a_4$, and $a_5$. The timelines are vertically aligned and arrows indicate the flow of data.]
Side-channel Discovery

► Instrument graphic libraries & collect victim trace
Side-channel Discovery

Instrument graphic libraries & collect victim trace
Side-channel Discovery

Instrument graphic libraries & collect victim trace
Side-channel Discovery

Instrument graphic libraries & collect victim trace
Side-channel Discovery

- Select pairs of addresses \((a_x, a_y)\)
  - \(\text{dist}(a_x, a_y)\) is long enough for flush+reload to measure
  - \(\text{dist}(a_x, a_y)\) is stable across the same input
  - \(\text{dist}(a_x, a_y)\) has high information gain
  - \(a_x, a_y\) not affected by cache architecture

> 6 million pairs

~1000 pairs
Side-channel Discovery

- Select pairs of addresses \((a_x, a_y)\)
  - \(\text{dist}(a_x, a_y)\) is long enough for flush+reload to measure
  - \(\text{dist}(a_x, a_y)\) is stable across the same input
  - \(\text{dist}(a_x, a_y)\) has high information gain
  - \(a_x, a_y\) not affected by cache architecture

- Sanity Check: Run attack offline
  - Filter out addressed affected system noise
  - Filter out addressed affected by instrumentation

> 6 million pairs

~1000 pairs

<100 pairs
Keypress Prediction

- Machine Learning Model Construction
  - Collect measurements for all different user inputs
  - Random Forest with 100 estimators

- Keypress Prediction
  - Classify measurement with confidence

s:0.70, a:0.15
n:0.12, m:0.03
Challenges

- Perform measurement without privilege
- Discover side-channel in graphic libraries
- Noise-resistant key prediction
# Attack 1: Onboard

- **Ubuntu 16.04, Intel Core i7-4770**

<table>
<thead>
<tr>
<th>#</th>
<th>Address</th>
<th>Library</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0x75a40</td>
<td>libcairo.so</td>
<td>_cairo_surface_create_scratch</td>
</tr>
<tr>
<td></td>
<td>0x69e40</td>
<td>libcairo.so</td>
<td>_cairo_scaled_font_map_lock</td>
</tr>
<tr>
<td>2</td>
<td>0x69e40</td>
<td>libcairo.so</td>
<td>_cairo_scaled_font_map_lock</td>
</tr>
<tr>
<td></td>
<td>0x41f40</td>
<td>libcairo.so</td>
<td>_cairo_intern_string</td>
</tr>
<tr>
<td>3</td>
<td>0x24440</td>
<td>libcairo.so</td>
<td>_cairo_clip_copy_with_translation</td>
</tr>
<tr>
<td></td>
<td>0xbe000</td>
<td>libcairo.so</td>
<td>_cairo_ft_unscaled_font_lock_face</td>
</tr>
<tr>
<td>4</td>
<td>0x6b900</td>
<td>libcairo.so</td>
<td>_cairo_path_fixed_approximate_stroke_extents</td>
</tr>
<tr>
<td></td>
<td>0x41700</td>
<td>libcairo.so</td>
<td>_intern_string_pluc</td>
</tr>
<tr>
<td>5</td>
<td>0x6a5c0</td>
<td>libcairo.so</td>
<td>_cairo_scaled_font_thaw_cache</td>
</tr>
<tr>
<td></td>
<td>0x41700</td>
<td>libcairo.so</td>
<td>_intern_string_pluc</td>
</tr>
</tbody>
</table>
Attack 1: Onboard

- Single keypress prediction
Attack 1: Onboard

- Single keypress prediction
- Augment 1: Multiple logins
Attack 1: Onboard

- Single keypress prediction
- Augment 1: Multiple logins
Attack 1: Onboard

- Single keypress prediction
- Augment 1: Multiple logins
- Augment 2: Dictionary
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
  - Resolution 20x lower than Intel
Attack 2: Capital One

- Huawei Nexus 6P. Android 8.0
- Flush+reload => Evict+reload
- Resolution 20x lower than Intel
- Pre-render triggered only once

<table>
<thead>
<tr>
<th>#</th>
<th>Library</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>libskia.so</td>
<td>SkScalerContext_FreeType_Base::generateGlyphImage</td>
</tr>
<tr>
<td></td>
<td></td>
<td>libskia.so SkMask::getAddr</td>
</tr>
<tr>
<td>2</td>
<td>libskia.so</td>
<td>SkGlyph::computeImageSize</td>
</tr>
<tr>
<td></td>
<td></td>
<td>libskia.so SkAAClipBlitter::~SkAAClipBlitter</td>
</tr>
</tbody>
</table>
Attack 2: CapitalOne

Augment 1: 10 logins
Attack 2: CapitalOne

- Augment 1: 10 logins
Attack 2: CapitalOne

- Augment 1: 10 logins
- Augment 2: Dictionary
Other Apps
Discussion

» Measurement challenge
  » Measurement resolution

» Extensions

» Mitigations
Discussion

- Measurement challenge
- Extensions
  - Inter-keystroke timing
  - Combining with other side-channels
  - Examine other libraries
- Mitigations
Discussion

- Measurement challenge
- Extensions
- Mitigations
  - Prevent flush+reload
  - Constant-time rendering
Conclusion

- Effective execution time measurement
- Exposing side-channels in graphics libraries
- Evaluations on real-world applications

Acknowledgement

- U.S. Army Research Laboratory Cyber Security Collaborative Research Alliance
Thanks!

Q & A