Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Summary

- ARM based devices can be vulnerable to Drammer as well.
- **Drammer:**
  - Memory templating
    - Scan memory for vulnerable bits
  - Land sensitive data
  - Reproduce the bit flip
- Root access exploitation possible with high reliability.
  - By modifying entries in Page Table Pages (PTP)
- Severe consequences for numerous devices that are currently in use.
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Problem

- Rowhammer failure mechanism only exploitable in a probabilistic way
  - Can it be done deterministically?
- Not clear if Rowhammer attacks are possible on ARM
  - Some researcher thought that it might be impossible on ARM
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Rowhammer Failure Mechanism

- Access adjacent rows at a high frequency (hammer)
- This causes voltage leakage in the victim row
- Bit flips are induced
Primitives to exploit Rowhammer Bug – x86

- P1: Fast uncached memory access
  - Explicit cache flush (clflush instruction)
  - Cache eviction sets
  - Non-temporal access instructions

- P2: Physical memory massaging
  - Page-table spraying (probabilistic)

- P3: Physical memory addressing
  - Pagemap interface
  - Huge pages
Rowhammer Bug on x86

- Probabilistic
  - Page-table spraying used because we don’t know exactly where bitflips will occur.

- Countermeasures
  - Disable clflush
  - Error Correcting Codes (ECC)
  - Probabilistic Adjacent Row Activation (PARA)
  - Many more
Primitives to exploit Rowhammer Bug – ARM

- **P1: Fast uncached memory access**
  - Explicit cache flush (\texttt{clflush} instruction) \textit{Privileged instruction}
  - Cache eviction sets \textit{Too slow}
  - Non-temporal access instructions \textit{Only suggests to not cache it}

- **P2: Physical memory massaging**
  - Page-table spraying (probabilistic) \textit{Can crash the system}

- **P3: Physical memory addressing**
  - Pagemap interface \textit{No unprivileged access anymore}
  - Huge pages \textit{Disabled on stock Android}
Rowhammer Failure Mechanism

- x86 Architectures are known to be vulnerable if the DRAM is modern enough.
  - Are ARM architectures vulnerable as well?

Before this paper:
- Probabilistic Rowhammer attacks on x86 based devices
  - Low reliability
  - Limited impact in practice

After this paper:
- Deterministic Rowhammer attacks
  - High reliability
  - Allows to completely subvert any vulnerable system
  - Requires to trick the OS to put a page table in a known and vulnerable memory location.
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Goal

"Deterministic Rowhammer Attack on Mobile Platforms”

- Deterministic:
  - Memory templating
  - Land sensitive Data → Phys Feng Shui
  - Reproduce the bit flip
  - *Note: This approach is not completely deterministic but much more reliable than the probabilistic approach that we know from Rowhammer attacks on x86.*

- Rowhammer Attack:
  - Vulnerable system required

- Mobile Platforms:
  - ARMv7, ARMv8 running Android
Primitives to exploit Rowhammer Bug - ARM

- **P1: Fast uncached memory access**
  - Provided by DMA buffer management APIs (ION)

- **P2: Physical memory massaging**
  - Memory Templating
  - Physical Memory Allocator
    - Buddy allocator
  - Phys Feng Shui

- **P3: Physical memory addressing**
  - Provided by DMA buffer management APIs (ION)
Outline

- Summary
- Problem
- Background
- Goal
- **Mechanisms**
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Phys Feng Shui

- Variant of Flip Feng Shui
  - FFS was used to mount attacks against other guest OSes running on the same hypervisor.

- Goal
  - Force the OS to place a page-table in a vulnerable memory location such that we can modify an entry in a deterministic way.

- Some size definitions
  - S chunk = chunk of the size of a page (typically 4KB)
  - M chunk = chunk of the size of a row (has to be determined)
  - L chunk = largest possible contiguous chunk
Determining Row Size

- Access time for page pairs
- If access time increases then the pages are on different rows
  - Therefore we can determine the row size

(0,1) (0,2) (0,3) ... (0,16)

Figure 3: Heatmap representing the time required to access a given pair of pages on a LG Nexus 5. The diagonal pattern clearly indicates that the row size is 16 pages = 64K.
Phys Feng Shui

- **Step 1:** Fill in as many L chunks as possible and create templates
- **Step 2:** Fill in as many M chunks as possible
- **Step 3:** Free one L chunk where we want to launch the attack

Free(L*)
Phys Feng Shui

- Step 4: Fill L* with M chunks
- Step 5: Free M* and all L chunks (M* is where we launch the attack)
- Step 6: Fill in S chunks until the first one falls into M*
- Step 7: Add padding to align victim page table
- Step 8: Launch attack on the victim page table that then points to a page table which we created in L*

\[ \text{Map}(M) \]
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
## Empirical Results

<table>
<thead>
<tr>
<th>Device</th>
<th>ARMv7</th>
<th>ARMv8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nexus 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv8+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv8+</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 5</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nexus 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARMv8+</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Vulnerable* | *Resilient*
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Threat Model

- **Set up:**
  - Attacker has control over an unprivileged app running on an ARM based Android device.
  - No permissions at all.
  - Latest stock version of Android with all the latest security updates installed.
  - No special features enabled.

- **Goal of the attacker:**
  - To mount an privilege escalation attack to acquire root privileges.
Threat Model

Attacker
- wants root privileges

controls

Application
- +unprivileged

Device
- +ARM based
- +Latest stock Android
- +No special features enabled

runs on
Novelty, Key Approach and Ideas

- First deterministic Rowhammer attack on ARM architecture
- Generalization of deterministic Rowhammer attacks on x86 architecture
- Mounting a Drammer attack using Direct Memory Access (DMA) bypasses existing defenses (i.e. disabling clflush) on x86 architectures
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- **Strengths**
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Strengths

- Shows a severe problem current mobile devices have.
  - First paper that shows a Rowhammer attack on ARM architecture based devices.
- The attack is deterministic.
  - Much more serious consequences in practice.
- Very detailed description of the work that was done
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths

**Weaknesses**

- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Weaknesses

- All countermeasures have an overhead
- ARMv8 sample size too small
  - No representative conclusion possible
  - No explanation why they seem more resilient
- Tested only Android smartphones (i.e. no Smartwatches, no IPhones, etc.)
- Considering that the occurrence of bitflips also depends on environmental aspects, the sample size is clearly too small.
- Not so easy to understand (Phys Feng Shui)
Thoughts and Ideas

- Future research needs to test the influence factors like temperature, age of the components, etc. have on the number of vulnerable bits.

- A broader range of devices needs to be tested (also running other OSes than Android).

- More effective and efficient countermeasures need to be found.
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
Takeaways

- Rowhammer on ARM
- Deterministic exploitation
- Practical impact
Outline

- Summary
- Problem
- Background
- Goal
- Mechanisms
- Key Results: Methodology and Evaluation
- Novelty, Key Approach and Ideas
- Strengths
- Weaknesses
- Thoughts and Ideas
- Takeaways
- Questions/Open Discussion
QUESTIONS?
Discussion Question

What do you consider being more valuable regarding security?

- Opensource Android
  - Known that Drammer is possible
  - Countermeasures proposed by third parties

- Proprietary iOS
  - Unsure if Drammer is possible
  - If possible then there are not yet any countermeasures
Discussion Question

Opensource Android

Positive:
- Huge research community doing research on Android
- More vulnerabilities get found and can be patched

Negative:
- Attacker can learn more about the OS
- Practical attacks happen faster

Proprietary iOS

Positive:
- Attackers have it harder to mount a practical attack since they know less about the OS

Negative:
- Once an attacker is successful, he can be active for a very long time
- Fewer security researchers
Discussion Question

- Do you think it is okay that the researchers published the paper before Google was able to patch its devices?
Discussion Question

- Researchers reported the attack to Google 91 days before the release of the paper at CCS 2016
  - Including some mitigation techniques
- Google asked them to delay the release of the paper
- The researchers refused
- Google asked them to obfuscate parts of the paper
- The researchers refused again
- → Did the researchers act responsibly?
Discussion Question

- Can you think of countermeasures that have little to no overhead?
Discussion Question

- **Hardware based:**
  - Probabilistic Adjacent Row Activation (PARA) looks promising
  - Better isolation between rows in DRAM?

- **Software based:**
  - Disallow features that can be used to satisfy the primitives P1-P3
  - Detect access patterns that could imply that an attack is happening
Supplementary Material

- [Video] CCS 2016 – Drammer: Deterministic Rowhammer Attacks on Mobile Platforms
  - https://www.youtube.com/watch?v=ITaMvBN1PoA

- [Video] Computer Architecture – Lecture 2: RowHammer and Beyond (ETH Zürich, Fall 2018)
  - https://www.youtube.com/watch?v=560JzQ-oeLE