RAMBleed: Reading Bits in Memory Without Accessing Them

Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom

Presented by Erik Saathoff

11/16/2020
Motivation

• Rowhammer has previously only been demonstrated as a threat to DRAM integrity.
• Flipping the roles of attacker and victim make it possible to use Rowhammer as a read channel.
• ECC RAM can be exploited as a timing channel.
Threat Model

• Attacker runs unprivileged software on same OS and victim program.
• OS maintains isolation between attacker and victim programs.
• Attacker cannot exploit microarchitectural side channel leakage from victim.
• The machine is vulnerable to Rowhammer, but programs can only access their own private memory.
• Attacker can trigger the victim to perform allocation of secret data.
Background – DRAM Configuration
Background – DRAM Configuration

• DRAM cells are accessed at the granularity of the entire row
• Two pages exist in one row.
• Hammering one page will automatically cause the other page on the same row to be hammered.
Bit Flips

• The three adjacent bits in a column can be represented by x-y-z.
• 0-1-0 and 1-0-1 are stripe patterns and are likely to flip.
• 0-0-0 and 1-1-1 are uniform patterns and aren’t likely to flip.
• 1-1-0, 1-0-0, 0-1-1, and 0-0-1 are neither and the outcome is unknown.

```
1 1 0 0 0 0 1 1
0 1 0 1 0 1 0 1
1 1 0 0 1 1 0 0
```

```
1 1 0 0 0 0 1 1
1 1 0 0 ? ? ? ?
1 1 0 0 1 1 0 0
```

```
1 1 0 0 0 0 1 1
1 1 0 0 ? ? ? ?
1 1 0 0 1 1 0 0
```
Overall Technique

- Allocate a consecutive block of DRAM and check for cell susceptible to Rowhammer.
- Strategically deallocate memory to trick the victim into placing a secret value in the rows above and below an attacker controlled sampling page.
- Access the other pages on the same rows as the secrets to leak the data into the middle attacker row.
- Combine bits recovered by placing the secret in various locations in the allocated DRAM block.
- Use math to recover all missing components of the RSA key.
Thoughts?
Strengths/Weaknesses

• Strengths
  • Novel usage of Rowhammer to convert from a write to a read channel.
  • Works on Ubuntu Linux in standard configuration (no huge pages, page map access, memory deduplication).
  • Clever circumventions of ECC, memory scrambling, and physical address unalignment.
  • New mechanism (Frame Feng Shui) used to place victim pages at desired locations.

• Weaknesses
  • Capability to recover random data is not shown.
  • Relies heavily on *a priori* knowledge (key location, allocation patterns).
  • Technique seems much easier to mitigate than authors indicate.
  • A detailed study of the DRAM templating is not provided.
• Double-sided Rowhammer is preferred to maximize the likelihood of a bit flip.

• The secret (S) is placed above and below the sampling page (A1) in the same rows as A0 and A2.

• Accessing A0 and A2 hammers data into A1 without accessing S.
Memory Massaging – Obtain DRAM Block

• Attack Linux buddy allocator
  • Exhaust small blocks with \textit{mmap} and monitor available block sizes in kernel free lists until less than 2 MB of free space is available in blocks smaller than order 10 (4 MB).
  • Request two 2 MB blocks. A 4 MB block will be split and the second request will be physically consecutive memory.

![Diagrams showing the process of memory massaging](image-url)
Memory Massaging – Offsets and Templating

• Address differences between co-banked pages uniquely identifies unaligned block offset based on memory controller addressing design.
  • Address bits $a_0$ - $a_{20}$ are known once the offset is known.
  • Timing channels are used to identify co-banked pages.
• Get $a_{21}$ using $a_{17}^0 \oplus a_{21}^0 = a_{17}^1 \oplus a_{21}^1$ on consecutive rows.
• Template by imposing 1-0-1 and 0-1-0 stripes in consecutive row and checking for bit flips.
Key Extraction

- From templating, bits that flip in the same location as key data are considered useful (3/16).
- Bit flips occurring at the same offset in multiple rows are redundant and not useful (4/15).
- Out of 84K recovered bit flips, 4.2K will provide useful information for key extraction.
- Can achieve 3-4 bit/second.

<table>
<thead>
<tr>
<th>Type</th>
<th>Read Accuracy Percentages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Overall</td>
</tr>
<tr>
<td>Double-sided</td>
<td>90%</td>
</tr>
<tr>
<td>Single-sided</td>
<td>74%</td>
</tr>
</tbody>
</table>

Key info location
Frame Fung Shui

• Given a known victim DRAM allocation pattern, devise a situation such that the victim places the secret in T0 or T1.

Step 1: Dummy Allocations

Step 2: Deallocation

Step 3: Trigger Victim
SSH Attack

• 4,200 bits (68%) recovered at 0.31 bits/second and 82% accuracy. (~4 hours)
• Full key was successfully recovered with Heninger-Shacham algorithm.

<table>
<thead>
<tr>
<th>Type</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-sided RAMBleed</td>
<td>68.89%</td>
</tr>
<tr>
<td>Single-sided RAMBleed</td>
<td>28.22%</td>
</tr>
<tr>
<td>Unable to place victim</td>
<td>2.39%</td>
</tr>
</tbody>
</table>
Poll Question

• Which countermeasure would provide the greatest difficulty in performing the RAMBleed attack?
  • PARA (Probabilistic Adjacent Row Activation)
  • Using ECC RAM.
  • Randomly changing key location within secret page (S) during SSH child spawn.
  • Using DDR4 instead of DDR3.
  • Memory scrambling.
  • Flushing key from memory when done.
ECC Modifications

• In ECC DRAM, the data and check bits are 64 and 8 bits respectively.
• During a read, if the memory controller detects
  • One errors: A large read latency is observed and the unflipped data is read out.
  • Two errors: The machine crashes
• Templating now works using a binary search in each 64 bit word and looking for increased read latency.
• Use increased read latency to ‘read’ the sampling page.
• Achieves 0.64 bits/second and 73% accuracy.
Mitigations

• Probabilistic Adjacent Row Activation (PARA)
  • Not widely adopted and probabilistic security.

• Targeted Row Refresh (TRR)
  • Some papers have induced Rowhammer bit flips even with TRR.

• More frequent refresh (from 64ms to 32 ms)
  • Some papers can flip bits even with this change; not practical for mobile use.

• Using ECC
  • This paper demonstrates how ECC can be used as a vulnerability.

• Memory Encryption
  • This works. Bit flips can cause SGX to halt due to failed integrity check.

• Flush keys from memory
  • Not practical for data that must be stored for long durations.

• Probabilistic memory allocator
  • Cannot defeat RAMBleed with probabilistic memory spraying techniques.
  • Attacker can use row-buffer timing side channel to detect correct configuration.
Discussion Questions

• Is it feasible to use ECC mechanisms which don’t have a discernable latency on correction events?

• How would the attacker handle random placement of the keys within the secret page? If the key were continuously moved?

• The attack's execution leaned heavily on the determinism of Linux's buddy allocator. Would it still be possible to pull off this exploit with a randomized memory allocator?

• This paper (along with other exploits we have discussed) demonstrate how dangerous it can be to share memory mappings/physical memory layouts with user space programs. Is there work demonstrating memory controller based randomized physical layouts?

• Can the ECC RAMBleed attack work if each 64 bit word has more than one bit flip?