Row hammer

Row hammer (also written as **rowhammer**) is a security exploit that takes advantage of an unintended and undesirable side effect in dynamic random-access memory (DRAM) in which memory cells interact electrically between themselves by leaking their charges, possibly changing the contents of nearby memory rows that were not addressed in the original memory access. This circumvention of the isolation between DRAM memory cells results from the high cell density in modern DRAM, and can be triggered by specially crafted memory access patterns that rapidly activate the same memory rows numerous times.[1][2][3]

The row hammer effect has been used in some privilege escalation computer security exploits,[2][4][5][6] and network-based attacks are also theoretically possible.[7][8]

Different hardware-based techniques exist to prevent the row hammer effect from occurring, including required support in some processors and types of DRAM memory modules.[9][10]

Contents

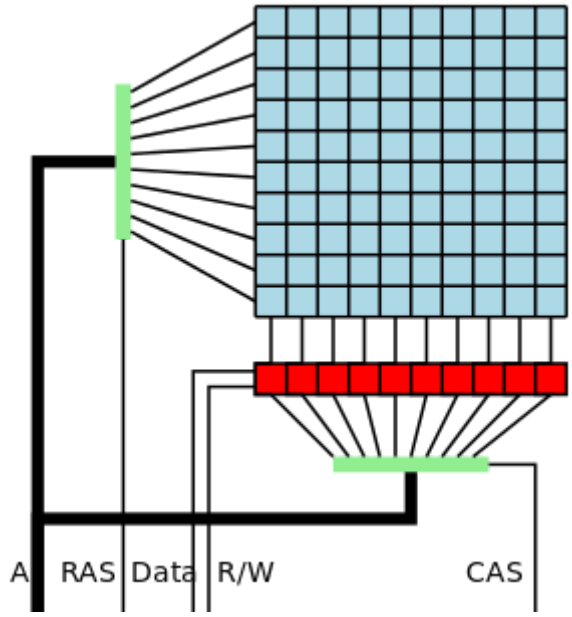
Background

In dynamic RAM (DRAM), each bit of stored data occupies a separate memory cell that is electrically implemented with one capacitor and one transistor. The charge state of a capacitor (charged or discharged) is what determines whether a DRAM cell stores "1" or "0" as a binary value. Huge numbers of DRAM memory cells are packed into integrated circuits, together with some additional logic that organizes the cells for the purposes of reading, writing, and refreshing the data.[11][12]

Memory cells (blue squares in both illustrations) are further organized into matrices and addressed through rows and columns. A memory address applied to a matrix is broken into the row address and column address, which are processed by the row and column address decoders (in both illustrations, vertical and horizontal green rectangles, respectively). After a row address selects the row for a read operation (the selection is also known as row activation), bits from all cells in the row are transferred into the sense amplifiers that form the row buffer (red squares in both illustrations), from which the exact bit is selected using the column address. Consequently, read operations are of a destructive nature because the design of DRAM requires memory cells to be rewritten after their values have been read by transferring the cell charges into the row buffer. Write

operations decode the addresses in a similar way, but as a result of the design entire rows must be rewritten for the value of a single bit to be changed.[1]:2–3[11][12][13]

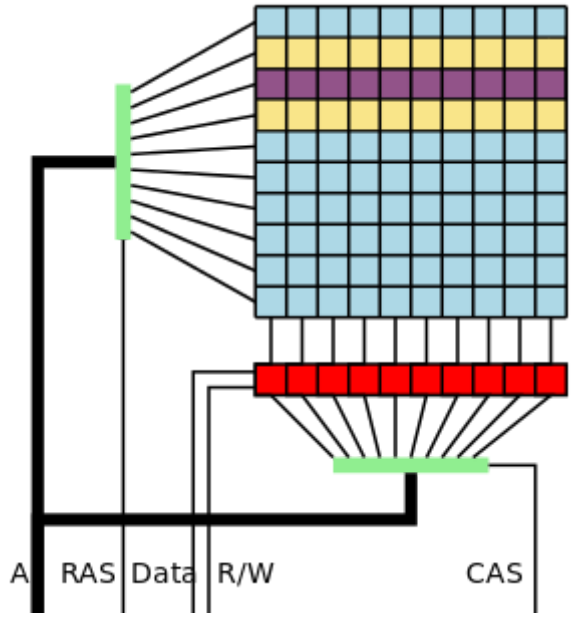
As a result of storing data bits using capacitors that have a natural discharge rate, DRAM memory cells lose their state over time and require periodic rewriting of all memory cells, which is a process known as refreshing.[1]:3[11] As another result of the design, DRAM memory is susceptible to random changes in stored data, which are known as soft memory errors and attributed to cosmic rays and other causes. There are different techniques that counteract soft memory errors and improve the reliability of DRAM, of which error-correcting code (ECC) memory and its advanced variants (such as lockstep memory) are most commonly used.[14]



A high-level illustration of DRAM organization, which includes memory cells (blue squares), address decoders (green rectangles), and sense amplifiers (red squares)

Overview

Increased densities of DRAM integrated circuits have led to physically smaller memory cells capable of storing smaller charges, resulting in lower operational noise margins, increased rates of electromagnetic interactions between memory cells, and greater possibility of data loss. As a result, *disturbance errors* have been observed, being caused by cells interfering with each other's operation and manifesting as random changes in the values of bits stored in affected memory cells. The awareness of disturbance errors dates back to the early 1970s and Intel 1103 as the first commercially available DRAM integrated circuits; since then, DRAM manufacturers have employed various mitigation techniques to counteract disturbance errors, such as improving the isolation between cells and performing production testing. However, researchers proved in a 2014 analysis that commercially available DDR3 SDRAM chips manufactured in 2012 and 2013 are susceptible to disturbance errors, while using the term *row hammer* to name the associated side effect that led to observed bit flips.[1][3][15]



Rapid row activations (yellow rows) may change the values of bits stored in victim row (purple row).[15]:2

The opportunity for the row hammer effect to occur in DDR3 memory[16] is primarily attributed to DDR3's high density of memory cells and the results of associated interactions between the cells, while rapid DRAM row activations have been determined as the primary cause. Frequent row activations cause voltage fluctuations on the associated row selection lines, which have been observed to induce higher-than-natural discharge rates in capacitors belonging to nearby (adjacent, in most cases) memory rows, which are called *victim rows*; if the affected memory cells are not refreshed before they lose too much charge, disturbance errors occur. Tests show that a disturbance error may be observed after performing around 139,000 subsequent memory row accesses (with cache flushes), and that up to one memory cell in every 1,700 cells may be susceptible. Those tests also show that the rate of disturbance errors is not substantially affected by increased environment temperature, while it depends on the actual contents of DRAM

because certain bit patterns result in significantly higher disturbance error rates.[1][2][15][17]

A variant called *double-sided hammering* involves targeted activations of two DRAM rows surrounding a victim row: in the illustration provided in this section, this variant would be activating both yellow rows with the aim of inducing bit flips in the purple row, which in this case would be the victim row. Tests show that this approach may result in a significantly higher rate of disturbance errors, compared to the variant that activates only one of the victim row's neighboring DRAM rows.[4][18]:19–20[19]

Mitigation

Different methods exist for more or less successful detection, prevention, correction or mitigation of the row hammer effect. Tests show that simple ECC solutions, providing single-error correction and double-error detection (SECDED) capabilities, are not able to correct or detect all observed disturbance errors because some of them include more than two flipped bits per memory word.[1]:8[15]:32 Furthermore, research shows that precisely targeted three-bit row hammer flips prevents ECC memory from noticing the modifications.[20][21]

A less effective solution is to introduce more frequent memory refreshing, with the refresh intervals shorter than the usual 64 ms,[a] but this technique results in higher power consumption and increased processing overhead; some vendors provide firmware updates that implement this type of mitigation.[22] One of the more complex prevention measures performs counter-based identification of frequently accessed memory rows and proactively refreshes their neighboring rows; another method issues additional infrequent random refreshes of memory rows neighboring the accessed rows regardless of their access frequency. Research shows that these two prevention measures cause negligible performance impacts.[1]:10–11[23]

Since the release of Ivy Bridge microarchitecture, Intel Xeon processors support the so-called *pseudo target row refresh* (pTRR) that can be used in combination with pTRR-compliant DDR3 dual in-line memory modules (DIMMs) to mitigate the row hammer effect by automatically refreshing possible victim rows, with no negative impact on performance or power consumption. When used with DIMMs that are not pTRR-compliant, these Xeon processors by default fall back on performing DRAM refreshes at twice the usual frequency, which results in slightly higher memory access latency and may reduce the memory bandwidth by up to 2–4%.[9]

The LPDDR4 mobile memory standard published by JEDEC[24] includes optional hardware support for the so-called *target row refresh* (TRR) that prevents the row hammer effect without negatively impacting performance or power consumption.[10][25][26] Additionally, some manufacturers implement TRR in their DDR4 products,[27][28] although it is not part of the DDR4 memory standard published by JEDEC.[29] Internally, TRR identifies possible victim rows, by counting the number of row activations and comparing it against predefined chip-specific *maximum activate count* (MAC) and *maximum activate window* (t_{MAW}) values, and refreshes these rows to prevent bit flips. The MAC value is the maximum total number of row activations that may be encountered on a particular DRAM row within a time interval that is equal or shorter than the t_{MAW} amount of time before its neighboring rows are identified as victim rows; TRR may also flag a row as a victim row if the sum of row activations for its two neighboring rows reaches the MAC limit within the t_{MAW} time window.[24][30]

Due to their necessity of huge numbers of rapidly performed DRAM row activations, row hammer exploits issue large numbers of uncached memory accesses that cause cache misses, which can be detected by monitoring the rate of cache misses for unusual peaks using hardware performance counters.[4][31]

Version 5.0 of the MemTest86 memory diagnostic software, released on December 3, 2013, added a row hammer test that checks whether computer RAM is susceptible to disturbance errors, but it only works if the computer boots UEFI; without UEFI, it boots an older version with no hammer test.[32]

Implications

Memory protection, as a way of preventing processes from accessing memory that has not been assigned to each of them, is one of the concepts behind most modern operating systems. By using memory protection in combination with other security-related mechanisms such as protection rings, it is possible to achieve privilege separation between processes, in which programs and computer systems in general are divided into parts limited to the specific privileges they require to perform a particular task. Using privilege separation can also reduce the extent of potential damage caused by computer security attacks by restricting their effects to specific parts of the system.[33][34]

Disturbance errors (explained in the section above) effectively defeat various layers of memory protection by "short circuiting" them at a very low hardware level, practically creating a unique attack vector type that allows processes to alter the contents of arbitrary parts of the main memory by directly manipulating the underlying memory hardware.[2][4][18][35] In comparison, "conventional" attack vectors such as buffer overflows aim at circumventing the protection mechanisms at the software level, by exploiting various programming mistakes to achieve alterations of otherwise inaccessible main memory contents.[36]

Exploits

The initial research into the row hammer effect, published in June 2014, described the nature of disturbance errors and indicated the potential for constructing an attack, but did not provide any examples of a working security exploit.[1] A subsequent October 2014 research paper did not imply the existence of any security-related issues arising from the row hammer effect.[16]

```
code1a:
  mov (X), %eax  // read from address X
  mov (Y), %ebx  // read from address Y
  clflush (X)    // flush cache for address X
  clflush (Y)    // flush cache for address Y
  mfence
  jmp code1a
```

A snippet of x86 assembly code that induces the row hammer effect (memory addresses X and Y must map to different DRAM rows in the same memory bank)[1]:3[4][18]:13–15

On March 9, 2015, Google's Project Zero revealed two working privilege escalation exploits based on the row hammer effect, establishing its exploitable nature on the x86-64 architecture. One of the revealed exploits targets the Google Native Client (NaCl) mechanism for running a limited subset of x86-64 machine instructions within a sandbox,[18]:27 exploiting the row hammer effect to escape from the sandbox and gain the ability to issue system calls directly. This NaCl vulnerability, tracked as CVE-2015-0565 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-0565), has been mitigated by modifying the NaCl so it does not allow execution of the `clflush` (cache line flush[37]) machine instruction, which was previously believed to be required for constructing an effective row hammer attack.[2][4][35]

The second exploit revealed by Project Zero runs as an unprivileged Linux process on the x86-64 architecture, exploiting the row hammer effect to gain unrestricted access to all physical memory installed in a computer. By combining the disturbance errors with memory spraying, this exploit is capable of altering page table entries[18]:35 used by the virtual memory system for mapping virtual addresses to physical addresses, which results in the exploit gaining unrestricted memory access.[18]:34,36–57 Due to its nature and the inability of the x86-64 architecture to make `clflush` a privileged machine instruction, this exploit can hardly be mitigated on computers that do not use hardware with built-in row hammer prevention mechanisms. While testing the

viability of exploits, Project Zero found that about half of the 29 tested laptops experienced disturbance errors, with some of them occurring on vulnerable laptops in less than five minutes of running row-hammer-inducing code; the tested laptops were manufactured between 2010 and 2014 and used non-ECC DDR3 memory.[2][4][35]

In July 2015, a group of security researchers published a paper that describes an architecture- and instruction-set-independent way for exploiting the row hammer effect. Instead of relying on the `clflush` instruction to perform cache flushes, this approach achieves uncached memory accesses by causing a very high rate of cache eviction using carefully selected memory access patterns. Although the cache replacement policies differ between processors, this approach overcomes the architectural differences by employing an adaptive cache eviction strategy algorithm.[18]:64–68 The proof of concept for this approach is provided both as a native code implementation, and as a pure JavaScript implementation that runs on Firefox 39. The JavaScript implementation, called *Rowhammer.js*,[38] uses large typed arrays and relies on their internal allocation using large pages; as a result, it demonstrates a very high-level exploit of a very low-level vulnerability.[39][40][41][42]

In October 2016, researchers published DRAMMER, an Android application that uses row hammer, together with other methods, to reliably gain root access on several popular smartphones.[43] The vulnerability was acknowledged as CVE-2016-6728 (https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2016-6728)[44] and a mitigation was released by Google within a month. However, due to the general nature of possible implementations of the attack, an effective software patch is difficult to be reliably implemented. As of June 2018, most patch proposals made by academia and industry were either impractical to deploy or insufficient in stopping all attacks. As a mitigation, researchers proposed a lightweight defense that prevents DMA-based attacks by isolating DMA buffers with guard rows.[45][46]

See also

- Memory scrambling – memory controller feature that turns user data written to the memory into pseudo-random patterns
- Radiation hardening – the act of making electronic components resistant to damage or malfunctions caused by ionizing radiation
- Single event upset – a change of state caused by ions or electromagnetic radiation striking a sensitive node in an electronic device
- Soft error – a type of error involving erroneous changes to signals or data but no changes to the underlying device or circuit

Notes

a. Research shows that the rate of disturbance errors in a selection of DDR3 memory modules closes to zero when the memory refresh interval becomes roughly seven times shorter than the default of 64 ms.[15]:17,26

References

1. Yoongu Kim; Ross Daly; Jeremie Kim; Chris Fallin; Ji Hye Lee; Donghyuk Lee; Chris Wilkerson; Konrad Lai; Onur Mutlu (June 24, 2014). "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors" (http://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf) (PDF). *ece.cmu.edu*. IEEE. Retrieved March 10, 2015.
2. Dan Goodin (March 10, 2015). "Cutting-edge hack gives super user status by exploiting DRAM weakness" (https://arstechnica.com/security/2015/03/cutting-edge-hack-gives-super-user-status-by-exploiting-dram-weakness/). Ars Technica. Retrieved March 10, 2015.

3. Paul Ducklin (March 12, 2015). " 'Row hammering' – how to exploit a computer by overworking its memory" (https://nakedsecurity.sophos.com/2015/03/12/row-hammering-how-to-exploit-a-computer-by-overworking-its-memory/). Sophos. Retrieved March 14, 2015.
4. Mark Seaborn; Thomas Dullien (March 9, 2015). "Exploiting the DRAM rowhammer bug to gain kernel privileges" (http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html). *googleprojectzero.blogspot.com*. Retrieved March 10, 2015.
5. "Using Rowhammer bitflips to root Android phones is now a thing" (https://arstechnica.com/security/2016/10/using-rowhammer-bitflips-to-root-android-phones-is-now-a-thing/). *Ars Technica*. Retrieved October 25, 2016.
6. Swati Khandelwal (May 3, 2018). "GLitch: New 'Rowhammer' Attack Can Remotely Hijack Android Phones" (https://thehackernews.com/2018/05/rowhammer-android-hacking.html). *The Hacker News*. Retrieved May 21, 2018.
7. Mohit Kumar (May 10, 2018). "New Rowhammer Attack Can Hijack Computers Remotely Over the Network" (https://thehackernews.com/2018/05/rowhammer-attack-exploit.html). *The Hacker News*. Retrieved May 21, 2018.
8. Swati Khandelwal (May 16, 2018). "Nethammer—Exploiting DRAM Rowhammer Bug Through Network Requests" (https://thehackernews.com/2018/05/remote-rowhammer-attack.html). *The Hacker News*. Retrieved May 21, 2018.
9. Marcin Kaczmarski (August 2014). "Thoughts on Intel Xeon E5-2600 v2 Product Family Performance Optimisation – Component selection guidelines" (http://infobazy.gda.pl/2014/pliki/prezentacje/d2s2e4-Kaczmarski-Optymalna.pdf) (PDF). Intel. p. 13. Retrieved March 11, 2015.
10. Marc Greenberg (October 15, 2014). "Reliability, Availability, and Serviceability (RAS) for DDR DRAM interfaces" (https://web.archive.org/web/20160705220034/http://www.memcon.com/pdfs/proceedings2014/NET105.pdf) (PDF). *memcon.com*. pp. 2, 7, 10, 20, 27. Archived from the original (http://www.memcon.com/pdfs/proceedings2014/NET105.pdf) (PDF) on July 5, 2016. Retrieved March 11, 2015.
11. "Lecture 12: DRAM Basics" (http://www.eng.utah.edu/~cs7810/pres/11-7810-12.pdf) (PDF). *utah.edu*. February 17, 2011. pp. 2–7. Retrieved March 10, 2015.
12. "Understanding DRAM Operation" (https://web.archive.org/web/20170829153054/http://www.ece.cmu.edu/~ece548/localcpy/dramop.pdf) (PDF). IBM. December 1996. Archived from the original (https://www.ece.cmu.edu/~ece548/localcpy/dramop.pdf) (PDF) on August 29, 2017. Retrieved March 10, 2015.
13. David August (November 23, 2004). "Lecture 20: Memory Technology" (https://web.archive.org/web/20050519185856/http://www.cs.princeton.edu/courses/archive/fall04/cos471/lectures/20-Memory.pdf) (PDF). *cs.princeton.edu*. pp. 3–5. Archived from the original (https://www.cs.princeton.edu/courses/archive/fall04/cos471/lectures/20-Memory.pdf) (PDF) on May 19, 2005. Retrieved March 10, 2015.
14. Bianca Schroeder; Eduardo Pinheiro; Wolf-Dietrich Weber (June 25, 2009). "DRAM Errors in the Wild: A Large-Scale Field Study" (http://www.cs.toronto.edu/~bianca/papers/sigmetrics09.pdf) (PDF). *cs.toronto.edu*. ACM. Retrieved March 10, 2015.
15. Yoongu Kim; Ross Daly; Jeremie Kim; Chris Fallin; Ji Hye Lee; Donghyuk Lee; Chris Wilkerson; Konrad Lai; Onur Mutlu (June 24, 2014). "Flipping Bits in Memory Without Accessing Them: DRAM Disturbance Errors" (http://users.ece.cmu.edu/~omutlu/pub/dram-row-hammer_kim_talk_isca14.pdf) (PDF). *ece.cmu.edu*. Retrieved March 10, 2015.
16. Kyungbae Park; Sanghyeon Baeg; ShiJie Wen; Richard Wong (October 2014). "Active-precharge hammering on a row induced failure in DDR3 SDRAMs under 3× nm technology". *Active-Precharge Hammering on a Row Induced Failure in DDR3 SDRAMs under 3x nm Technology*. IEEE. pp. 82–85. doi:10.1109/IIRW.2014.7049516 (https://doi.org/10.1109%2FIIRW.2014.7049516). ISBN 978-1-4799-7308-8.

17. Yoongu Kim; Ross Daly; Jeremie Kim; Chris Fallin; Ji Hye Lee; Donghyuk Lee; Chris Wilkerson; Konrad Lai; Onur Mutlu (July 30, 2015). "RowHammer: Reliability Analysis and Security Implications" (http://users.ece.cmu.edu/~omutlu/pub/rowhammer-summary.pdf) (PDF). *ece.cmu.edu*. Retrieved August 7, 2015.
18. Mark Seaborn; Thomas Dullien (August 6, 2015). "Exploiting the DRAM rowhammer bug to gain kernel privileges: How to cause and exploit single bit errors" (https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gain-Kernel-Privileges.pdf) (PDF). Black Hat. Retrieved August 7, 2015.
19. Andy Greenberg (March 10, 2015). "Googlers' Epic Hack Exploits How Memory Leaks Electricity" (https://www.wired.com/2015/03/google-hack-dram-memory-electric-leaks/). *Wired*. Retrieved March 17, 2015.
20. Shaun Nichols (November 21, 2018). "3 is the magic number (of bits): Flip 'em at once and your ECC protection can be Rowhammer'd" (https://www.theregister.co.uk/2018/11/21/rowhammer_ecc_server_protection/). *The Register*.
21. Dan Goodin (November 22, 2018). "Potentially disastrous Rowhammer bitflips can bypass ECC protections - ECCploit is the first Rowhammer attack to defeat error-correcting code" (https://arstechnica.com/information-technology/2018/11/potentially-disastrous-rowhammer-bitflips-can-bypass-ecc-protections/). *Ars Technica*. Retrieved January 17, 2021.
22. "Row Hammer Privilege Escalation (Lenovo Security Advisory LEN-2015-009)" (https://support.lenovo.com/us/en/product_security/row_hammer). Lenovo. August 5, 2015. Retrieved August 6, 2015.
23. Dae-Hyun Kim; Prashant J. Nair; Moinuddin K. Qureshi (October 9, 2014). "Architectural Support for Mitigating Row Hammering in DRAM Memories" (https://web.archive.org/web/20150311061310/http://users.ece.gatech.edu/~pnair6/rowhammer/rowhammer.pdf) (PDF). *ece.gatech.edu*. IEEE. Archived from the original (http://users.ece.gatech.edu/~pnair6/rowhammer/rowhammer.pdf) (PDF) on March 11, 2015. Retrieved March 11, 2015.
24. "JEDEC standard JESD209-4A: Low Power Double Data Rate (LPDDR4)" (http://www.jedec.org/standards-documents/docs/jesd209-4a) (PDF). JEDEC. November 2015. pp. 222–223. Retrieved January 10, 2016.
25. Kishore Kasamsetty (October 22, 2014). "DRAM scaling challenges and solutions in LPDDR4 context" (https://web.archive.org/web/20160603103733/http://www.memcon.com/pdfs/proceedings2014/MOB102.pdf#page=11) (PDF). *memcon.com*. p. 11. Archived from the original (http://www.memcon.com/pdfs/proceedings2014/MOB102.pdf#page=11) (PDF) on June 3, 2016. Retrieved January 10, 2016.
26. Omar Santos (March 9, 2015). "Mitigations Available for the DRAM Row Hammer Vulnerability" (http://blogs.cisco.com/security/mitigations-available-for-the-dram-row-hammer-vulnerability). *cisco.com*. Retrieved March 11, 2015.
27. Marc Greenber (March 9, 2015). "Row Hammering: What it is, and how hackers could use it to gain access to your system" (https://blogs.synopsys.com/committedtomemory/2015/03/09/row-hammering-what-it-is-and-how-hackers-could-use-it-to-gain-access-to-your-system/). *synopsys.com*. Retrieved January 10, 2016.
28. Jung-Bae Lee (November 7, 2014). "Green Memory Solution (Samsung Investors Forum 2014)" (http://aod.teletogether.com/sec/20140519/SAMSUNG_Investors_Forum_2014_session_1.pdf#page=15) (PDF). *teletogether.com*. Samsung Electronics. p. 15. Retrieved January 10, 2016.
29. "JEDEC standard JESD79-4A: DDR4 SDRAM" (https://www.jedec.org/standards-documents/docs/jesd79-4a) (PDF). JEDEC. November 2013. Retrieved January 10, 2016.
30. "Data Sheet: 4 Gb ×4, ×8 and ×16 DDR4 SDRAM Features" (https://web.archive.org/web/20180210062029/https://www.micron.com/~/media/documents/products/data-sheet/dram/ddr4/4gb_ddr4_sdram.pdf) (PDF). Micron Technology. November 20, 2015. pp. 48, 131. Archived from the original (http://www.micron.com/~/media/documents/products/data-sheet/dram/ddr4/4gb_ddr4_sdram.pdf) (PDF) on February 10, 2018. Retrieved January 10, 2016.

31. Nishad Herath; Anders Fogh (August 6, 2015). "These are Not Your Grand Daddy's CPU Performance Counters: CPU Hardware Performance Counters for Security" (https://www.blackhat.com/docs/us-15/materials/us-15-Herath-These-Are-Not-Your-Grand-Daddys-CPU-Performance-Counters-CPU-Hardware-Performance-Counters-For-Security.pdf) (PDF). Black Hat. pp. 29, 38–68. Retrieved January 9, 2016.
32. "PassMark MemTest86 – Version History" (https://www.memtest86.com/support/ver_history.htm). *memtest86.com*. February 13, 2015. Retrieved March 11, 2015.
33. Pehr Söderman (2011). "Memory Protection" (http://www.csc.kth.se/utbildning/kth/kurser/DD2395/dasakh10/Slides/dasakh10Lecture09memory.pdf) (PDF). *csc.kth.se*. Retrieved March 11, 2015.
34. Niels Provos; Markus Friedl; Peter Honeyman (August 10, 2003). "Preventing Privilege Escalation" (http://niels.xtdnet.nl/papers/privsep.pdf) (PDF). *niels.xtdnet.nl*. Retrieved March 11, 2015.
35. Liam Tung (March 10, 2015). " "Rowhammer" DRAM flaw could be widespread, says Google" (https://www.zdnet.com/article/rowhammer-dram-flaw-could-be-widespread-says-google/). ZDNet. Retrieved March 11, 2015.
36. Murat Balaban (June 6, 2009). "Buffer Overflows Demystified" (http://www.enderunix.org/docs/en/bof-eng.txt) (TXT). *enderunix.org*. Retrieved March 11, 2015.
37. "CLFLUSH: Flush Cache Line (x86 Instruction Set Reference)" (https://web.archive.org/web/20171203034552/http://x86.renejeschke.de/html/file_module_x86_id_30.html). *renejeschke.de*. March 3, 2013. Archived from the original (http://x86.renejeschke.de/html/file_module_x86_id_30.html) on December 3, 2017. Retrieved August 6, 2015.
38. Daniel Gruss; Clémentine Maurice (July 27, 2015). "IAIK/rowhammerjs: rowhammerjs/rowhammer.js at master" (https://github.com/IAIK/rowhammerjs/blob/master/javascript/rowhammer.js). *github.com*. Retrieved July 29, 2015.
39. Daniel Gruss; Clementine Maurice; Stefan Mengard (July 24, 2015). "Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript". arXiv:1507.06955 (https://arxiv.org/abs/1507.06955).
40. David Auerbach (July 28, 2015). "Rowhammer security exploit: Why a new security attack is truly terrifying" (http://www.slate.com/articles/technology/bitwise/2015/07/rowhammer_security_exploit_why_a_new_security_attack_is_truly_terrifying.html). *slate.com*. Retrieved July 29, 2015.
41. Alix Jean-Pharuns (July 30, 2015). "Rowhammer.js Is the Most Ingenious Hack I've Ever Seen" (https://motherboard.vice.com/en_us/article/9akpwz/rowhammerjs-is-the-most-ingenious-hack-ive-ever-seen). Motherboard.
42. Dan Goodin (August 4, 2015). "DRAM 'Bitflipping' exploit for attacking PCs: Just add JavaScript" (https://arstechnica.com/information-technology/2015/08/dram-bitflipping-exploit-for-attacking-pcs-just-add-javascript/). Ars Technica.
43. VUSec (October 2016). "DRAMMER: FLIP FENG SHUI GOES MOBILE" (https://www.vusec.net/projects/drammer/). Retrieved January 21, 2017.
44. NIST National Vulnerability Database (NVD). "CVE-2016-6728 Detail" (https://nvd.nist.gov/vuln/detail/CVE-2016-6728).
45. Victor van der Veen; Martina Lindor; Yanick Fratantonio; Harikrishnan Padmanabha Pillai; Giovanni Vigna; Christopher Kruegel; Herbert Bos; Kaveh Razavi (2018), "GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM" (https://research.vu.nl/en/publications/112a5465-aeb5-40fd-98ff-6f3b7c976676), *Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer International Publishing, pp. 92–113, doi:10.1007/978-3-319-93411-2_5 (https://doi.org/10.1007%2F978-3-319-93411-2_5), hdl:1871.1/112a5465-aeb5-40fd-98ff-6f3b7c976676 (https://hdl.handle.net/1871.1%2F112a5465-aeb5-40fd-98ff-6f3b7c976676), ISBN 9783319934105
46. "RAMPAGE AND GUARDION - Vulnerabilities in modern phones enable unauthorized access" (https://rampageattack.com/). Retrieved June 30, 2018.

External links

- Some notes on DRAM (#rowhammer) (http://blog.erratasec.com/2015/03/some-notes-on-dram-rowhammer.html), March 9, 2015, by Robert Graham
- Rowhammer hardware bug threatens to smash notebook security (http://www.infoworld.com/article/2894497/security/rowhammer-hardware-bug-threatens-to-smash-notebook-security.html), *InfoWorld*, March 9, 2015, by Serdar Yegulalp
- DDR3 Memory Known Failure Mechanism called "Row Hammer" (https://www.youtube.com/watch?v=7wIUQ04Vkes) on YouTube, July 17, 2014, by Barbara Aichinger
- Patent US 20140059287 A1: Row hammer refresh command (http://www.google.com/patents/US20140059287), February 27, 2014, by Kuljit Bains et al.
- Row Hammer Privilege Escalation Vulnerability (http://tools.cisco.com/security/center/content/CiscoSecurityAdvisory/cisco-sa-20150309-rowhammer), Cisco Systems security advisory, March 11, 2015
- ARMOR: A run-time memory hot-row detector (http://apt.cs.manchester.ac.uk/projects/ARMOR/RowHammer/armor.html), The University of Manchester, by Mohsen Ghasempour et al.
- Using Memory Errors to Attack a Virtual Machine (https://www.cs.princeton.edu/~appel/papers/memerr.pdf), March 6, 2003, by Sudhakar Govindavajhala and Andrew W. Appel
- A program for testing for the DRAM "rowhammer" problem (https://github.com/google/rowhammer-test), source code on GitHub