Rowhammer:
From DRAM faults to escalating privileges

Authors
Jan Kalbantner, MSc (Royal Holloway, 2019)
Konstantinos Markantonakis, ISG, Royal Holloway

Abstract

In 2014, researchers from Carnegie Mellon University and Intel Labs found that due to the increased density within dynamic random-access memory (DRAM) technology, it gets challenging to prevent cell charges from interacting with adjacent cells. Through rapidly accessing the same row in DRAM, data in adjacent rows can be corrupted. This evolves to the vulnerability "Rowhammer" which can be used to exploit memory management techniques in different environments, inject errors in cryptographic protocols and perform privilege escalation attacks.

In this article, we provide an overview of Rowhammer attacks and the threat it poses to current computer devices. Further, we describe an implementation of the Phys Feng Shui exploitation technique on an LG Nexus 5 mobile device and show that it is a real threat against mobile devices. We also discuss countermeasure and future research directions. [a]

[a] This article is published online by Computer Weekly as part of the 2020 Royal Holloway information security thesis series `https://www.computerweekly.com/ehandbook/Royal-Holloway-Rowhammer-from-DRAM-faults-to-escalating-privileges`. It is based on an MSc dissertation written as part of the MSc in Information Security at the ISG, Royal Holloway, University of London. The full thesis is published on the ISG's website at `https://www.royalholloway.ac.uk/research-and-teaching/departments-and-schools/information-security/research/explore-our-research/isg-technical-reports/`.

Introduction

In 2020, about 7.26 billion mobile users are estimated worldwide. We use mobile devices to communicate on a day to day basis, play games and even do mobile banking. To cope with the steadily increasing requirements to performance and availability, mobile devices must get more powerful, contain more memory and become more reliable. As the industry has yet to find a way to satisfy all three criteria in one device, the growing demand forced manufacturers to optimise their products, which accidently created security vulnerabilities. New zero-day exploits on smartphones are published more and more frequently. One widespread attack based on a hardware vulnerability first detected in 2014[1] is known as Rowhammer. It is related to the growing demand for increasing memory density of dynamic random-access memory (DRAM) modules. Due to the high density in those modules, electromagnetic interference can occur in memory, which can corrupt stored data. Researchers showed that this bug could be triggered on purpose, thus resulting in bits to flip. Researchers found out that the Rowhammer vulnerability is a widespread problem that affects every DRAM produced after 2011.

In this article, we provided an overview of Rowhammer attacks and the threat it poses to current computer devices. We structure the Rowhammer attacks into four procedures: (1) preparation, (2) hammering, (3) verification and (4) exploitation. Further, we implement the Phys Feng Shui exploitation technique and evaluate it with an LG Nexus 5 mobile device to show the possibility of Rowhammer attacks. This implementation based on direct memory accesses (DMA) showed that it is a practical concern. On average, we found

Rowhammer attack

(1) Preparation.

(2) Hammering.

(3) Verification.

(4) Exploitation.

[1] "Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors", Y. Kim, R. Daly, J. Kim, C. Fallin, J. H. Lee, D. Lee, C. Wilkerson, K. Lai and O. Mutlu, 2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA), Minneapolis, MN, 2014, pp. 361-372.

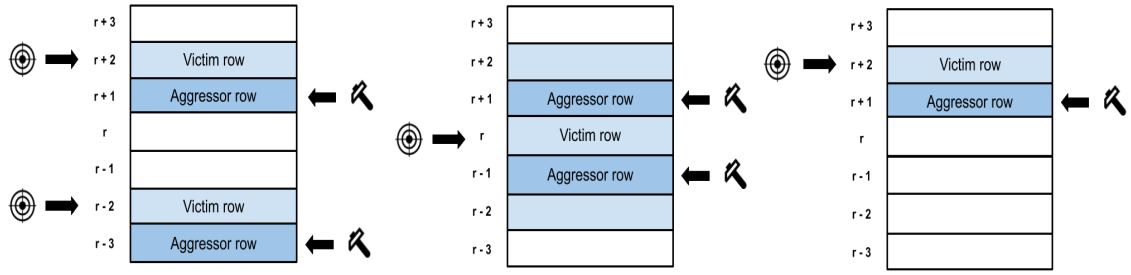

Figure 1: Schemata of (1) single-sided, (2) double-sided and (3) one-location Rowhammer.

exploitable unique bit flips after 473 seconds and it showed that the Phys Feng Shui technique is a real threat against mobile devices.

Microarchitectural attacks

Microarchitectural attacks are a kind of attack that either aims to (1) compromise confidentiality and data integrity, or (2) damage data or systems. They were developed to elude defences protecting cryptographic algorithms. The foundation of the attack primitives lies on hardware properties. While there is a wide range of definitions for microarchitectural attacks, we will categorise them into (1) side-channel attacks and (2) fault attacks.

Microarchitectural side-channel attacks exploit timing and behaviour differences that are (partially) caused through optimisations. Timing attacks aim at recovering data that is dependent on the timing behaviour of an application. These side-channel attacks allow secret information used in a computation to be derived from inadvertent influences the computation has on its environment.

Fault attacks exploit hardware and software to corrupt data. These attacks aim to bring hardware into an undocumented state where the behaviour is undefined (e.g., out of range voltage). Attackers are thus able to modify data that should not have been accessible. Rowhammer, the focus of our article, is a software-based microarchitectural fault attack.

What is Rowhammer?

Kim et al. observed in their paper “Flipping Bits in Memory Without Accessing Them: An Experimental Study of DRAM Disturbance Errors”, published in 2014, that the increasing density of modern DRAM memory modules has made them predisposed to disturbance errors due to charge leakage into adjacent cells while memory is accessed. The authors showed that repeated toggling of the word line of a row (the aggressor row) could accelerate the likelihood of leakage of charge from nearby rows (the victim rows). These disturbances can cause bits to flip. The triggering of bit flips through repeatedly accessing, i.e. hammering, of a row, is known as Rowhammer. As shown in Figure 1, there are multiple versions of Rowhammer. When the Rowhammer technique relies on the use of one aggressor row to attack an adjacent row, it is called single-sided Rowhammer (1). If an attacker uses two rows, one lying above and one below the victim row, it is called a double-sided Rowhammer (2). Hammering only one selected memory address is a relatively new technique and has been named one-location Rowhammer (3). One-location Rowhammer is also able to bypass security techniques such as error-correcting code (ECC) and table row refresh (TRR) which were seen as insurmountable.

We structured our software-based microarchitectural fault attacks into four procedures. First, the attacker selects suitable vulnerable memory positions for locating security-sensitive objects. Second, when the hammer-able positions are found, the attacker continues and hammers the DRAM to produce bit flips. Here it is notable that when a DRAM cell produces bit flips at one location, at these same locations the bit flips can be reproduced. Third, the attacker verifies that exploitable bit errors

have been produced and, finally, exploits them.

Utilising Rowhammer

After the discovery of the Rowhammer error, researchers worldwide tried to develop exploits using the vulnerability. Several mechanisms have been used for that, but all of them had similarities and relied on a probabilistic element. Researchers from Google, for example, used a memory massaging technique to trigger the Rowhammer bug. By massaging the memory, they forced the kernel to an 'out of memory' error. Through probabilistic reuse of already released physical memory pages they were able to perform a privilege escalation.

Probabilistic Rowhammer (PR) attacks, which have some non-deterministic and non-foreseeable elements, always depend on techniques such as memory spraying. Memory spraying techniques allocate a considerable number of objects in the memory to predict the layout of the memory and exploit it. The downside of PR attacking mechanisms is that they only offer weak reliability of exploiting a victim object. Due to the spraying, it is not guaranteed that a page table will be located in a security-sensitive area such as the kernel. (The page table is a key component for accessing data in memory. We need a controllable page table in the kernel to be able to exploit it.) When the wrong rows are hammered, it could permanently damage data or lead to a system failure.

As opposed to probabilistic variants, deterministic Rowhammer techniques depend on the expected behaviour, for example, of the physical memory allocator, and memory pattern. In order to place a memory page table into a foreseen location deterministically, the attacker must be able to control the layout of the physical memory predictably.

Because we wanted to control the exploitation, we used a technique called Flip Feng Shui (FFS). This procedure is based on specific memory management utilities and comprises three attack primitives.

- **Templating**: The attacker templates the physical memory in order to locate the cells prone to flips through hardware bugs (such as Rowhammer). We recall that when one location is prone to flipping bits, the flip can be reproduced at the same location.
- **Massaging**: After a victim row was found, an attacker places an appropriate physical memory page into a vulnerable position.
- **Exploitation**: The last step is exploitation; when the memory page is placed, the attacker triggers the hardware bug to cause a bit flip and corrupt chosen data.

Originally, Flip Feng Shui was developed for servers. By using deduplication mechanisms researchers were able to copy controlled memory pages into vulnerable positions. However, we wanted to exploit ARM platforms running Android and therefore we needed to use an advanced technique called Phys Feng Shui (PFS). It is utilising the same three primitives but instead of memory deduplication we used the Linux Buddy Allocator.

Without going into too many technical details, the buddy allocator is a memory allocation technique that divides one memory block into two smaller equal blocks to satisfy an allocation request. For the PFS technique, we can use this behaviour to control where we want to place our Rowhammer vulnerable memory positions. The process for memory massaging is shown in Figure 2 and involves the following steps:

Initial situation. Through the predictable behaviour of the Android buddy allocator, it is possible to acquire contiguous memory which is necessary to perform Rowhammer. To get a response in a predictable way, there are three chunk sizes required: Small (S), medium (M) and large (L). The small size is fixed at the size of one memory page (4 KB), M is set to the size of one row, and L is the size of the largest possible chunk.

Step 1. First, it is necessary that all of the memory is exhausted. To do this, we need to use the buddy allocator to allocate all chunks of considerable size (L) and examine them for later exploitation.

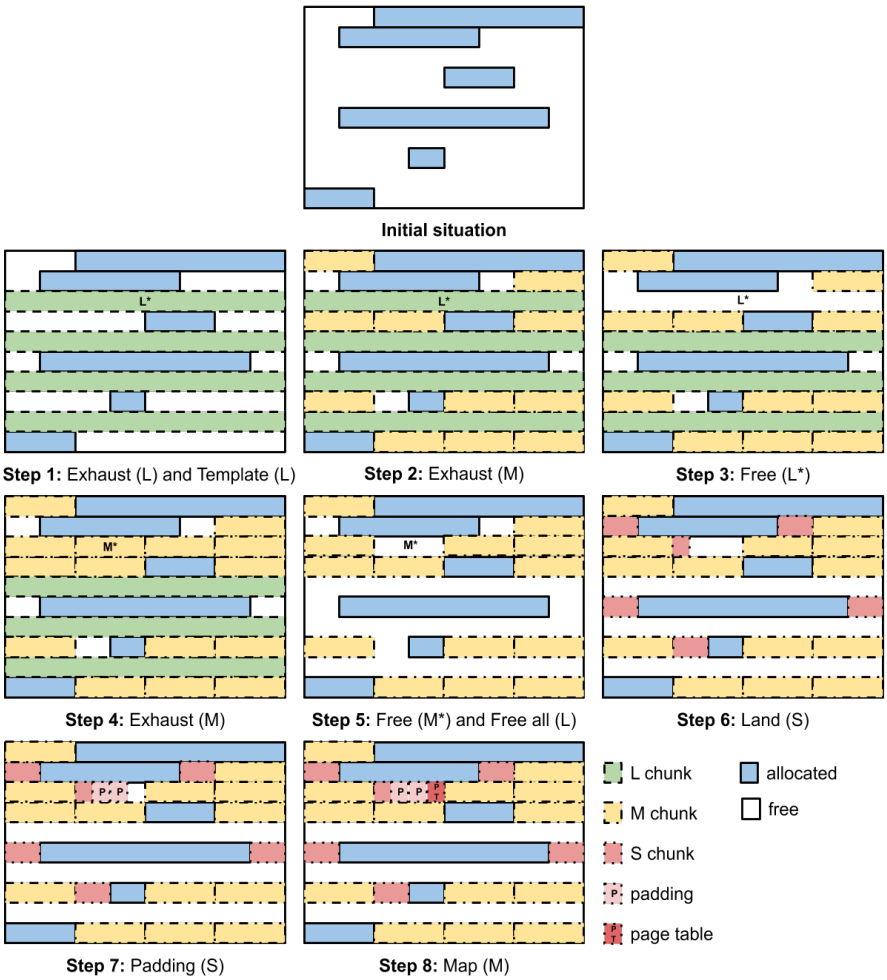

Figure 2: Layout of physical memory with effects before and after each step of the memory messaging.

Step 2. The next step is to exhaust all M chunks so that there is no space left for large or medium chunk allocations.

Step 3. Third, we select the vulnerable large chunk (L*) which we acquire knowledge about in the previous templating phase and release it.

Step 4. After releasing L*, we need to allocate M chunks again. Because the rest of the memory is already full, the allocator needs to fill the just released L* chunk space with M chunks. This means that one of the M chunks will become a vulnerable one, i.e. M*.

Step 5. Before we can actually place a page table (PT) in M*, we need to release the given chunk and also release all large chunks. PFS creates a lot of pressure in the RAM, and before causing an out of memory (OOM) issue, we take precautions by releasing L chunks. Running into an OOM situation would force the system to clean the memory or cause the system to crash.

Step 6. After releasing M*, we are able to place a small chunk in the same vulnerable region. To reliably guarantee that subsequent S land in M*, we need to map 4 KB sized memory repeatedly. In order to determine whether the allocations are placed in the vulnerable regions, we can use commands to gain insight into the allocated and available page tables, memory nodes and zones.

Step 7. The last step is mapping a page in the released L* chunk but beforehand we need to align the victim page table page (PTP). We must make sure that we later can flip bits in the page table entry (PTE), and in order to do so, we allocate padding page tables. The number of padding PTP depends on the location of the victim PTP.

Step 8. Lastly, we are able to map a page p either on the left (if we flip a 0 to a 1), or on the right side (if we flip a 1 to a 0). The chosen virtual memory address is fixed to allocate a new page table page. Therefore, the position of the page table entry solely depends on the virtual address picked. Moreover, the vulnerable PTP (in M*) must be $2n$ pages apart from page p, to be able to flip the n lowest bit in the victim PTE. The corresponding bit changes the PTE in a deterministic manner and points to the vulnerable page table page.

Privilege Escalation

At this point, we want to focus on privilege escalation. To be able to escalate privileges in any system the following requirements need to be met:

1. The ability to scan the whole memory for specific data.
2. Bypassing established security mechanisms.
3. Access to the kernel memory.
4. Execution of attack primitives in security critical areas.

Through templating, we are able to acquire knowledge of the whereabouts of these vulnerable positions. We use the Phys Feng Shui technique to predictably shift vulnerable memory positions. The next steps are then to execute Rowhammer. With rapid hammering, we flip the bits at vulnerable positions and gain access through a controlled change in the code structure to acquire root privileges.

Analysis

The results of our implementation show that we are able to produce bit flips on ARMv7 devices and therefore we are able to use Rowhammer on Android to attack the integrity of those devices.

Table 1 discusses the results of the previously described phases. In total, there were six attempts and twelve rounds within those attempts, each with another amount of utilised memory chunks and every time with 5920 hammered pairs. The average time spent for each round is described in the second column, followed by the flipped bits found, and the unique (i.e. hammerable) bit flips found.

Attempt	Avg. Time (in s)	Flips	Unique Flips
1	456	208	127
2	503	5	2
3	507	5	2
4	456	231	143
5	462	167	161
6	458	305	185

Table 1: Results of Rowhammer experiments on Android.

The average time used for one round in the overall attempts was between 456 and 507 seconds. The amount of flips found varies from 0 to 395 flips. The first attempt shows, on average an amount of 208 flips with 127 unique flips that can be exploited. Hence, we saw varying results but were able to find at least one exploitable bit. We concluded that the amount of bit flips depending on the type of phone use and the utilisation of the phone.

How to protect?

The State-of-the-Art for the industry was to use ECC and TRR to negate all kinds of microarchitectural attacks. As shown by multiple security researchers worldwide, microarchitectural fault attacks such as Rowhammer are not always mitigated. Because some of the devices cannot exchange the hardware for new resistant DRAM modules, the research has to establish a robust software-based mitigation.

As part of an MSc dissertation, we analysed current countermeasures according to some specific criteria to see if they are reliable, practical, secure and usable. The results showed that the overall majority of them are neither reliable nor practical and only two of the analysed countermeasures were secure. Moreover, some of them are only usable against a specific sort of Rowhammer attack. This is an issue, and future research is needed to find an effective, efficient, secure and lightweight mechanism to secure devices against Rowhammer and other types of microarchitectural attacks.

A defence mechanism must be able to counter one of the attack primitives: preparation, hammering, verification. Most of the countermeasures now concentrate on hammering, but none of the existing defences is currently practical and usable due to the low security they provide. In order to find a secure mechanism, we should focus on the other attack primitives. A protection against preparation would prohibit an attacker from finding any contiguous areas in physical memory. This is a particularly tricky part as it affects memory. Current techniques use amongst other things blacklisting of vulnerable memory positions as a defence, but research has shown that this prevention is not effective.

Another attempt to develop a secure countermeasure would be to thwart the verification primitive. The main idea of verification prevention is the isolation of memory areas into different domains to ensure that an attacker cannot attack security-sensitive areas. Some of the recent software-based countermeasures were successful with this approach but targeted particular Rowhammer attacks. To this point the authors are not in knowledge of one viable protection that provides efficient, practical, usable and secure solution for all variants of devices.

Future

Rowhammer on embedded devices

In the beginning, Rowhammer was limited to the usual computer systems. Later, it was proven that servers are also affected by the vulnerability. Recent work showed that not only DDR3 and DDR4

RAM but also LPDDR2, LPDDR3 and LPDDR4 (i.e. mobile devices) are affected. Yet, there is a lack of analysis of the execution of Rowhammer on devices such as Windows phones or iOS smartphones, IoT (Internet of Things) devices or embedded computing boards. The mobile device market separates into about 85.1% Android and 14.8% iOS devices. Therefore, a notable amount of devices are produced by Apple and must be analysed for security flaws. Moreover, there are over seven billion IoT devices worldwide forecast for 2020. Some of them have major security flaws which can be utilised to form massive botnets. We conclude that there is an urgency to focus on a proactive strategy when it comes to microarchitectural attacks on mobile, IoT and embedded devices.

Rowhammer over networks

Rowhammer over networks poses a significant threat to computer systems. They are practical and stealthy enough to infiltrate servers over a network connection without attracting the attention of any countermeasure. The Nethammer attack showed that the use of Intel Cache Allocation Technology (CAT), which is used as Denial of Service countermeasure within servers, accelerated their Rowhammer attack. Intel CAT does that by increasing the number of reading accesses within the RAM of the server. Moreover, through the execution of Nethammer researchers were able to break a system which would not boot anymore. At a closer look, they flipped a bit in an index node (inode) of the file system, which was responsible for the kernel getting corrupted. Rowhammer attacks over networks have a vast potential and could become the centre of future research.

Rowhammer as side-channel

An attack dubbed RAMbleed showed that attacking confidentiality with Rowhammer as a side-channel is possible. Through an exploitation technique called Frame Feng Shui, a variant of Flip Feng Shui, and the data dependency of RAM, bit flips in adjacent cells can be produced, which resulted in leaking an RSA key from OpenSSH. They were testing their attack with enabled ECC, but it did not mitigate RAMbleed. As a countermeasure for OpenSSH, software engineer Damien Miller added another layer of protection for RSA private keys which can be used against side-channel and speculative attacks like Spectre, Meltdown, RAMbleed, or other Rowhammer attacks. Private keys were encrypted with a symmetric key that was derived from a random prekey. To decrypt, any attacker must recover the prekey beforehand. However, the attacks are currently not sufficiently error-free, which makes the recovery of the prekey

highly unlikely. More attacks of this kind that use Rowhammer to read secrets from secured areas will likely follow in the future.

Rowhammer with GPU acceleration

In 2018, ‘GLitch’ was published as the first GPU accelerated Rowhammer attack which is completely based on a JavaScript implementation. Researchers from VU Amsterdam used a timing side-channel attack to gather internal information about the memory to then acquire contiguous memory. Then they used Rowhammer on the OpenGL implementation in Firefox and Chrome, WebGL 2.0, to execute the hammering primitive to finally exploit the system by breaking out of the web browser sandbox. As a reaction, Mozilla and Google disabled some modules in their browsers. However, this attack showed that GPU accelerated attacks are possible and the GPU can, as in many other areas, accelerate this particular attack. Future research should include the possibility that the GPU can be exploited and focus on countermeasures for the GPU. For example, when it comes to OpenGL, one can focus on the successor of OpenGL, the Vulkan API, which will be included in Android and Google’s new Operating System Fuchsia. An accelerated attack which can be operated on the upcoming generation of operating systems can pose an enormous threat and must be prevented before it arrives on the consumer market.

Biographies

Jan Kalbantner studied Applied Computer Science at the Cooperative State University Baden Württemberg (Germany) and later Information Security at Royal Holloway, University of London. He is currently member of the scientific staff of the Information Security Group at Royal Holloway, University of London where he is conducting his doctoral research. Mr. Kalbantner's main research focus is in the areas of cloud and edge computing, applications of blockchain technology and privacy-preserving algorithms.

Konstantinos Markantonakis is a Professor of Information Security at the Information Security Group at Royal Holloway, University of London and the Director of the Smart Card and IoT Security Centre. His main research areas include trusted execution environments, embedded devices and cyber physical system security, smart cards and RFIDs, avionics and drone security, automotive, payment and transport systems, mobile phone/NFC/secure element security, ambient sensors, Internet-of-Things (IoTs).

Series editor: S.- L. Ng