How Rowhammer Could Be Used to Exploit Weaknesses in Computer Hardware

March 2016

Mark Lanteigne, CTO and Founder, Third I/O Inc.

FOREWORD

Gordon Moore, the father of Moore's Law, predicted that capacitor density in silicon would reach saturation within the next decade. One could argue that his prediction, based on current technologies, is optimistic. The miniaturization or "die shrink" of modern silicon has been slowing for the last several years. And one of the primary reasons for this slowdown is the simple fact that high density capacitor counts in silicon are more susceptible to crosstalk and transistor leakage. These are the types of problems that lead to the highest severity of computer failures, such as locked up or rebooting systems and lost or corrupted data.

As silicon capacitor density increases, there is a strong argument for more rigorous testing and validation of new technologies, but unfortunately computer and memory manufacturers seem headed in the opposite direction.

One of the best examples of substandard testing, validation, or screening is a widespread computer memory (DRAM) problem that is commonly referred to as Rowhammer. Rowhammer has been present on DDR3 silicon since 2010. The computer industry has been extremely tight lipped on this problem, but many have inferred or directly stated that the problem has been fixed or mitigated in DDR3. And furthermore that is has been fully resolved in DDR4.

Based on the analysis by Third I/O, we believe that this problem is significantly worse than what is being reported. And it is still visible on some DDR4 memory modules.

ROWHAMMER BACKGROUND

In June of 2014, Yoongu Kim and researchers from Carnegie Mellon released a groundbreaking paper detailing the Rowhammer phenomenon. In this paper, they revealed that DDR3 sampled with a manufacturing date of 2010 to 2014 had an 85% failure rate when exposed to Rowhammer testing. These failures were observable as "bit flips" or data corruption. A non-technical explanation of Rowhammer is fairly easy to understand. Kim's research showed that repeatedly opening (also known as ACTIVATE or activating), reading, and closing a specific memory address could cause electrical disturbances that could lead to data corruption in nearby memory locations. In other words, they showed that multiple accesses to a specific memory address have the potential to cause bit flips or data corruption in nearby or adjacent memory. Sounds like crosstalk or leakage, right?

In regards to computer quality expectations, data corruption is often considered to be the worst defect that a computer system can exhibit. Simply put, it is when you store specific data and then access it later, you can observe that it has been altered in an unintended manner. Have you ever saved a file, tried to open it later, and received an error message stating that the file was damaged or corrupt? That's one potential example of data corruption. Some extreme scenarios for data corruption could be the unexpected malfunctioning or halting of systems, changing of critical (financial, medical, military, airline or personal) data, incorrect values being reported on equipment and hacking exploits that can

take control of computer systems. **Kim's paper was essentially a roadmap for hackers as he gave detailed information on how to exploit this previously unknown phenomenon.**

Rowhammer as a security exploit was initially proven by Mark Seaborn, Thomas Dullien, and researchers at Google on March 9, 2015. Using Rowhammer strategies, this team was able to cause data corruption and use it as an exploit to gain access to the entire memory subsystem of vulnerable laptops. Google's research was extraordinary in the fact that they were able to exploit the Rowhammer anomaly using accessible user-space processes. They also revealed some new details of how to make Rowhammer more effective. Specifically, they revealed that:

1) Linux hugepages (large OS allocated memory pages) allow for user space access to contiguous areas of physical memory up to 2 MB in size.
2) That "double-sided hammering" could be a catalyst to revealing more Rowhammer data corruptions even on systems that seemed immune to Kim's single-sided techniques.
3) That random memory hammering could be an effective exploit. For example, allocating 1 GB of memory and randomly guessing which rows to hammer could also cause data corruption.

To show another interesting exploit, Daniel Gruss and team of the Graz University of Technology, Austria released a paper and exploit in August of 2015 detailing how they successfully were able to create Rowhammer data corruptions using JavaScript. As JavaScript is a fundamental technology of all modern web browsers, Daniel showed that vulnerable systems could be exploited by simply visiting the wrong website. In January, 2016, Gruss claimed to have reproduced his findings on DDR4 using native code.

THIRD I/O'S MEMESIS MEETS ROWHAMMER

In 2006, Third I/O released Iris; a high speed Fibre Channel solid state disk that used servers as the platform and computer memory as a storage media. From 2006 to 2010, Iris was regularly benchmarked as the fastest external storage device available as it allowed for extremely high bandwidth and I/O operations per second. Unfortunately, Iris may have been a little too fast for the industry as running it often found critical bugs on enterprise servers. It found so many problems, in fact, that we started to brainstorm on how to create a diagnostic to see if a system's CPU, memory, and chipset could handle extreme DMA (direct memory accesses) as was required by Iris.

This brainstorming led us to innovate and create Memesis; a Linux kernel embedded enterprise memory test. We designed Memesis to push extreme levels of stress and bandwidth between the processors and memory while looking for data corruptions and ECC events. It was our goal to create a test that could perform operations similar to high performance DMA without using specialized peripherals. Since 2009, Memesis has been responsible for finding hundreds of unique hardware anomalies at the CPU, memory, chipset, power, and cooling subsystems of computer systems.

Prior to reading Kim's paper, we had never heard the term Rowhammer used in a publicly available document. However, we believe that we had observed several manifestations of this anomaly since approximately 2011. Normally, these problems could be seen as data corruption or ECC events on

shipping systems that we used as test platforms. They generally occurred during memory walk or common memory testing where we focused multiple CPU cores to read, write, and compare data patterns into small regions of system memory. We encountered this problem most often on multiple processor servers with high density memory configurations.

Kim's paper led us to start thinking about how to ACTIVATE memory more aggressively. Our early research led us to some papers published by Barb Aichinger of FuturePlus Systems. Aichinger's knowledge is at the DDR3 protocol level, so her insight into this problem helped us learn rapidly. It was her belief that semaphores are the key to creating Rowhammer type events. And based on our experience with NUMA server systems, we believed that her idea might be an effective manner to hammer multiprocessor systems such as dual processor servers; a staple in most datacenters.

SUCCESSFUL MEMESIS EXPERIMENTS AND REVELATIONS

We owe a great deal of thanks to Kim, Seaborn, Dullien, Aichinger, Gruss, and all of the engineers that supported their research and writings. To say the least, they helped open our eyes to a very easy to exploit problem once we began to understand it. Using their collective findings and theories, we crafted a Memesis test case that focused on creating excessive ACTIVATEs by simply reading from a memory address or addresses repeatedly and flushing cache after each operation.

It should be noted that Memesis does not attempt to determine CPU and CPU cache information. Our tests work at a black box level. The scriptable and flexible nature of Memesis allows us to use it as a brute force attack that could be applied to any computer system, from portables to enterprise servers.

We tested across a variety of x86-64 laptops, desktops, and dual processor ECC-protected server systems. Through our experiments, we have learned the following:

Multithreading

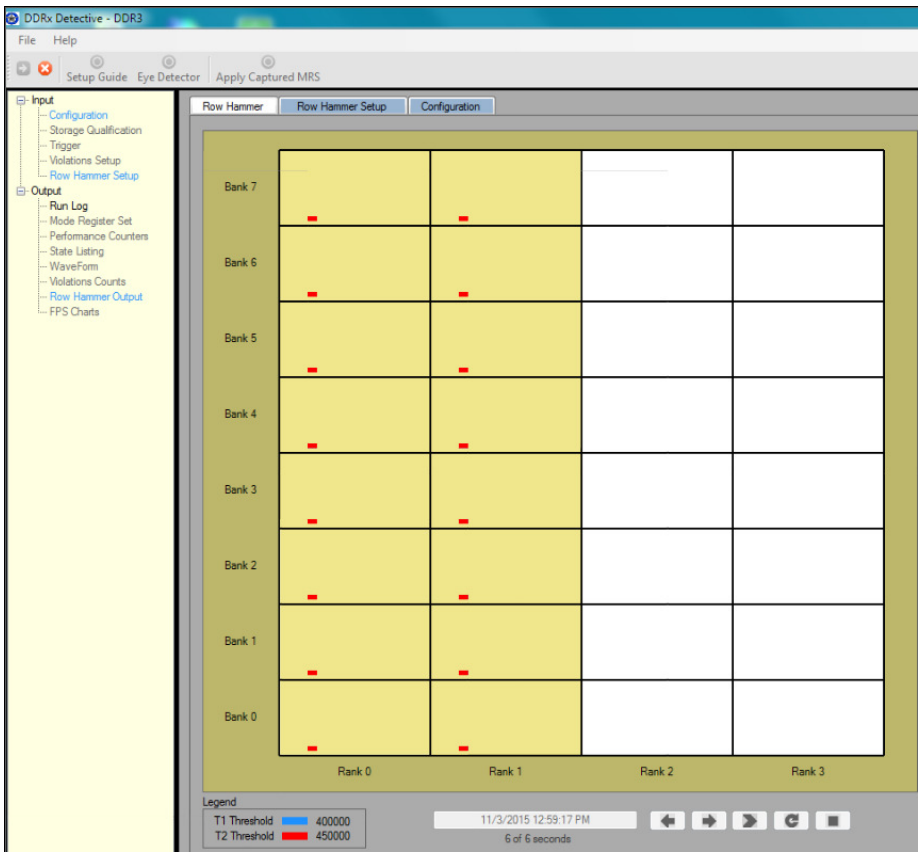
Multithreading Rowhammer attacks is a catalyst for finding significantly more data corruption. When using between 2 and the maximum CPU core count of threads, we found that multithreaded attacks will always induce more bit flips than a single threaded attack. Oddly enough, we also learned that using the maximum core count was not always the most effective test. From our experiments, we learned that threading beyond the total physical and virtual CPU core count would be counterproductive as well.

Regional Rowhammer

Multithreaded hammering inside of small and contiguous regions of memory would always yield the highest number of bit flips. In our experiments, we found that the 2 MB region size (such as a Linux hugepage) was generally the best memory size to test. We investigated this finding on dual-core to hexa-core processors and found this test case to be highly effective. It was so effective, in fact, that Third I/O has coined the term "Regional Rowhammer" to give a name to our most successful test case. A regional Rowhammer attack is simply allocating a small, contiguous region of memory and hammering it with 2 or more threads.

We sent a copy of Memesis to FuturePlus Systems for analysis. Their DDR Detective software performs real time analysis of DDR protocol traces and it includes a running count of ACTIVATE commands at the rank and bank level. According to FuturePlus, the Memesis 2 MB regional Rowhammer attack is more effective than other available tests because it creates excessive ACTIVATE commands in all 8 banks across both ranks of an individual DDR3 module. In fact, the Memesis regional Rowhammer showed ACTIVATE counts of 450,000+ in every bank on the DIMM during standard 64 ms retention time cycles. We believe that this is the catalyst for our success in reproducing Rowhammer bit flips.

According to FuturePlus, our software occasionally hits 600,000+ ACTIVATE commands in a 64 ms time period, the retention time of the DRAM, which should be of major concern to all computer system and memory vendors. Our experiments have shown that the results of this level of Rowhammer activity can be catastrophic in nature on susceptible systems.



FuturePlus' DDR Detective Showing Memesis Regional Rowhammer Stressing all Banks

Data Patterns on x86 Platforms

Data patterns DO matter and can be a catalyst for finding more bit flips; sometimes by an order of magnitude. In Kim's research, he showed that using specific data patterns on an FPGA test could influence the number of bit flips. Kim's x86 test using Memtest, however, only utilized solid one and zero data patterns. He speculated that modern memory controllers would scramble or swizzle any

predictable data pattern into the equivalent of a random data pattern. Third I/O's analysis shows that this is most certainly not the case on many systems. We find that seeding with a random data pattern is always much more effective than solid zeroes or ones. We have also identified specific "killer" data patterns that can cause increased bit flips or ECC errors on specific systems.

Data Pattern	Bit flips Discovered
Random	2055
All Zero	80
All Ones	113
Memesis Killer #1	2329

As you can see from the above example, using a random data pattern is much better than using a static pattern such as all zeroes or ones. Third I/O generally believes that using a random data pattern should be your default test case. In the above example, we reference a killer data pattern that can show as much as 50% more bit flips versus a random data pattern. However, we have found that this pattern is not always superior to random on every system.

This killer data pattern is actually an archaic pattern that was once used in the early days of Gigabit Ethernet testing. The pattern is: 492492492492492492492492492492492492492492492492 in hex or in binary it looks like this:

010010010010010010010010010010010010010010010010010010010010010010
100100100100100100100100100100100100100100100100100100100100100001
001001001001001001001001001001001001001001001001001001001001010010

We believe that there is likely a killer data pattern specific and unique to every system, but that discovering a single pattern to find more issues on all systems is highly unlikely due to architectural differences. That is why we endorse the use of random data patterns as a default hammering strategy.

Advanced Rowhammer Techniques Require Less Hammering

By using multithreading, regional Rowhammer, and different data patterns we have been able to greatly reduce the number of hammer operations necessary to cause bit flips. Kim suggested 2.7 million hammers per address, whereas our current defaults are only 800,000 for double sided and 1.5 million for single sided hammering to be most efficient. This allows us to investigate systems for bit flips or other Rowhammer induced anomalies faster than other diagnostics.

ROWHAMMER HAS BEEN MITIGATED. IT HAS NOT BEEN FIXED

Ultimately, Rowhammer is a hardware problem that is based on the laws of physics. It appears as though the industry has been racing to mitigate the potential damage of Rowhammer, but it is rarely heard that it has been fixed or resolved definitively in hardware. We have observed multiple systems that still fail with the most common mitigations, which are doubling memory refresh rates and using ECC protection.

The reason why these mitigations fail appears obvious. Simply put, on some systems that are vulnerable to Rowhammer one can observe hundreds or even thousands of bit flips per minute. These are not always single bit errors. On vulnerable systems, we have seen mitigations only slow down the frequency of occurrences of bit flips and ECC events.

Example One: Rowhammer on a DDR3 Laptop

Most of the work performed by Seaborn's team at Google has been on laptop systems. From our experience, we have also confirmed that laptops are highly vulnerable to Rowhammer events. We are not certain why laptops seem more vulnerable, but one theory is simply that laptop memory (SO-DIMMs) may not be screened as rigorously as memory that is destined for use in high end desktop, workstation, or server systems.

It was on a 2013 manufactured laptop that we first started developing our Memesis Rowhammer test strategies. We chose this system for our development as Kim's original Rowhammer on Memtest would show approximately 35 errors every 15 minutes. Once we figured out how to optimize the effectiveness of Rowhammer, Memesis showed the following by simply testing 8 MB of total memory:

Cores Used	Double-Sided 2 MB Region Bit Flips	Single-Sided 2 MB Region Bit Flips
0	0	0
0,1	892	**2759**
0,1,2	2	2
0,1,2,3	425	454
0,2	2	**1097**
1,3	2	**960**

There are several interesting revelations from the test above. The first is that running solely on cores 0 and 1 showed the highest number of both double and single sided Rowhammer failures. It should be noted that these correspond to the first physical and virtual core, respectively. The second most interesting dynamic is that single sided hammering appears more effective on this system than dual sided, at least for a regional attack. It was also surprising to see single-sided hammering find 1,000 more bit flips over double-sided hammering in lower core count tests.

We need to re-emphasize that the above test was focused on 8 MB of total memory, so to see 1 bit flip or data corruption is bad news. The highest bit flip counts from the chart above were on tests that ran for less than 2 minutes; yielding over 1,000 bit flips per minute.

Example Two: Rowhammer on an Enterprise DDR3 Server Introduction

Prior to the writing of this paper, we had not seen any discussion of server vulnerability to Rowhammer. In our opinion, this is extremely short sighted as servers place significantly more demand on memory compared to consumer or portable devices. In fact, one of the biggest buzzwords in the x86 cloud is the coming inclusion of in-memory database servers. Many of these systems advertise as many as 72

physical and 72 virtual processing cores and can contain terabytes (TB) of DDR memory. From our perspective, this allows for 144 cores for Rowhammer attack capability.

For our baseline experiments, we decided to investigate a server chipset that became popular in late 2012 and was showcased in many datacenters and supercomputing facilities from 2012 to late 2014. Servers using this chipset are commonly populated with two physical processors with each processor having its own local DDR memory. Each processor and the respective local memory are commonly referred to as a NUMA (non-uniform memory access) node. When used efficiently, NUMA allows for CPU to memory performance to scale in a linear manner as more processors and memory are added. However, we believe that NUMA also allows for greater opportunities to exploit Rowhammer.

Before we begin to discuss our results, we must first discuss ECC protected systems and the fact that there are no standards on what constitutes ECC protection or ECC event reporting. At its base level, ECC protection simply means that a server can handle or correct single bit errors, although some systems advertise the ability to correct multiple bit errors. Generally, it is our belief that ECC events should be reported to the operating system so that savvy users can gauge the health of their infrastructure.

Unfortunately, server vendors routinely use a technique called ECC threshold or the 'leaky bucket' algorithm where they count ECC errors for a period of time and report them only if they reach certain levels of failure. From what we understand, this threshold is commonly above 100 per hour, but this remains a trade secret and varies based on the server vendor. So, to see ECC errors (MCE in Linux or WHEA in Windows), there generally needs to be 100 bit flips per hour or greater. This makes “seeing” Rowhammer on server error logs more difficult.

In addition, we have observed some server vendors will NEVER report ECC events back to the OS, although they might get logged into IPMI. Typically, users expect to see correctable ECC errors logged directly to the OS or that halt the system when they cannot be corrected. During our investigation into this phenomenon, we even encountered one server that neither reported ECC events to the OS nor halted when bit flips were not correctable. The end result was data corruption at the application level. This is something, in our opinion, that should never happen on an ECC protected server system.

ROWHAMMER IN A NUMA OPTIMIZED MANNER

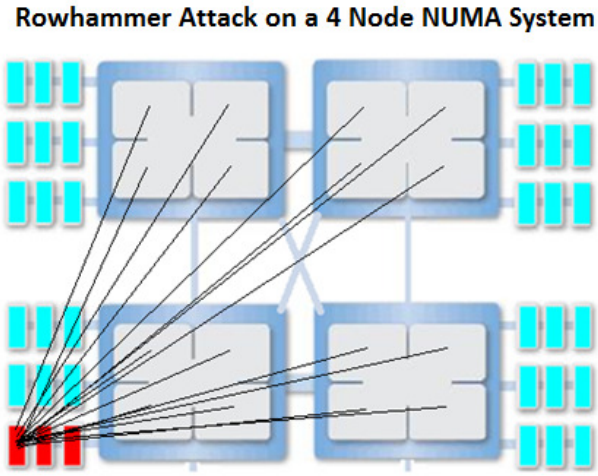
On modern operating systems, users can allocate memory and can set a flag that asks for NUMA optimized memory. What this means is that programs can allocate memory that is specific to a NUMA node. It is then up to the program to use local or remote processing cores. When we use the term “NUMA optimized” we are simply stating that we ran tests where processing cores only communicated with local or high performance memory.

As stated earlier, finding Rowhammer vulnerabilities is not always easy on server systems due to ECC thresholds or lack of ECC reporting. However, we found a system that was highly vulnerable to Rowhammer on our 3rd attempt. When hammered, this system would quickly report “Hardware Errors” or “CMCI Storms” (Corrected Machine Check Interrupt) and it would generally lockup or reboot within 3

minutes. After reboot, the system would report that uncorrectable errors occurred about half of the time. The other half of the time, the system never completed POST and it was essentially rendered useless from a Rowhammer attack until it was manually power cycled.

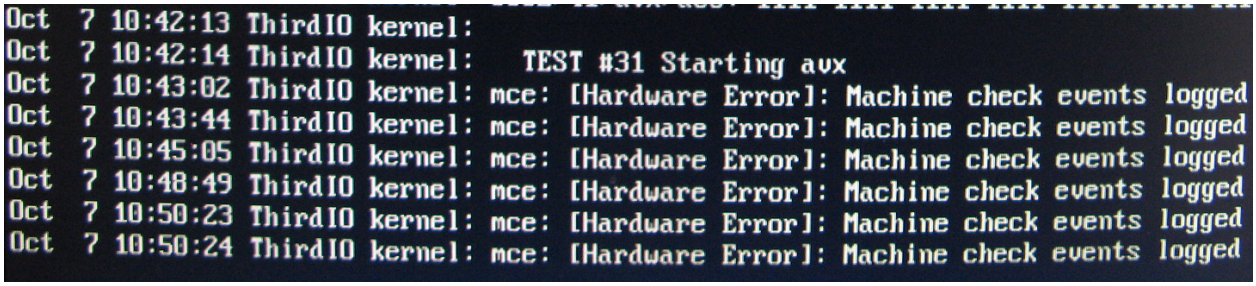
This ECC protected server only had a single settable Rowhammer mitigation which was doubling memory refresh rates. By doubling the memory refresh rate, we did notice that the system ran much better in a NUMA optimized mode. It ran, and only reported Hardware Errors that were listed as "memory read" and "scrubbing" errors. Although there was a tangible performance reduction when errors were reported, this system stopped locking up or rebooting. That was until we attempted to run a Rowhammer NUMA node attack.

We use the expression "NUMA node attack" to describe all processing cores in a multi-socket NUMA server reading or writing to a memory region located on a specific node. This attack uses local and remote processing cores simultaneously. It is our belief that cross-socket cache coherency naturally creates an environment that is more conducive to cache flushing and potential excessive ACTIVATE commands in main memory. By adding in the concept of a multithreaded, regional Rowhammer test, it is our belief that this type of testing represents a worst case scenario for enterprise servers. Operationally, this type of test would also mimic the types of behavior that one might observe if using a multithreaded semaphore where all threads share a common, small region of memory.



The above graphic is representative of a proposed in-memory database server with 4 NUMA nodes. In this illustration, all processing cores are focusing on a node specific Regional Rowhammer attack.

Using these advanced techniques, we were able to observe ECC events within the first 3 minutes of test time. Generally, this system would lockup or reboot within 30 minutes. Once again, this was on a Rowhammer mitigated system using both ECC and double refresh. So it follows that dual mitigations, on some systems, appear to be flawed and can be exploited as a denial of service.



Rowhammer on a Mitigated NUMA Platform – Each Hardware Error Represents Multiple Bit Flips

ROWHAMMER ON DDR4

Although the computer industry appeared to be caught off guard with Rowhammer on DDR3, there is plenty of evidence showing that mitigation has been researched and implemented in both hardware and software since 2012. Some companies have even boldly stated that DDR4 would be immune to Rowhammer. In one such presentation, Samsung claimed that their DDR4 would be "Row Hammer Free" due to Targeted Row Refresh (TRR) technology. And Micron has expressed that: "Target row refresh (TTR) mode is not required to be used, and in some cases has been rendered inoperable. Micron's DDR4 devices automatically perform TRR mode in the background." NOTE: The use of the acronym TTR was taken directly from Micron datasheets, and it should read as TRR.

In late October, 2015, Rowhammer on DDR4 sightings began to appear in public forums. Many of these sightings were vague or lacking in full configuration details. However, there was a curious pattern showing DDR4 memory manufactured by Crucial as being one of the components in Rowhammer failing systems.

In January of 2016, we received an email from Daniel Gruss, the researcher pursuing Rowhammer in Javascript. In his message, Gruss informed us that he found DDR4 that was rapidly failing his native code Rowhammer test. In addition, he sent us links for a German IT website that had a recommended system configuration for performance computing. According to Gruss, these systems seemed prone to DDR4 Rowhammer failures. And this system was populated with Crucial memory.

The combination of the public reports and Gruss' success was the catalyst for us to determine if Rowhammer still exists on DDR4. In addition to purchasing a fast Intel Skylake based system, we also acquired four Crucial Ballistix Sport 2400 MHz, two Crucial Ballistix Elite 2666 MHZ, two Geil Super Luce 2400 MHz, two G.Skill Ripjaws 4 3200 MHz, and two Micron branded 2133 MHz DDR4 memory modules for testing.

We created a comprehensive script of short duration (90 second) Rowhammer tests to scan the memory for any problems running system defaults. These early tests did not quickly find any bit flips. We then extended the runtime of the script to one hour per test and began to test each DIMM individually. The third DIMM that we tested was a Crucial Sport and it showed the first bit flip running a double-sided regional Rowhammer attack and the default random data pattern.

Oddly enough, we later learned that this test was even more effective when we ran the 492 killer data pattern as referenced above. We found that the killer data pattern was finding approximately 50% more errors than our default random pattern. We found this remarkable as this system even has a setting labeled “Memory Scrambling Enabled”. Because this system contains the latest processor and memory technologies, we were fully expecting random data patterns to be our best case scenario.

Once we honed in on the exact parameters for achieving the highest number of bit flips in the shortest period of time, we began a focused, four hour scan of each DIMM to see how sensitive they might be to a more aggressive attack. The initial results are in the middle column of the chart:

Rowhammer on DDR4 Test Results

DDR4 Module	Default Refresh Rate Bit Flips	25% Reduced Refresh Rate Bit Flips (5 minute scan)
Crucial Ballistix Sport #1	46	9260
Crucial Ballistix Sport #2	27	5741
Crucial Ballistix Sport #3	12	6623
Crucial Ballistix Sport #4	31	4723
Crucial Ballistix Elite #1	10	4922
Crucial Ballistix Elite #2	30	5186
Geil Super Luce #1	0	0
Geil Super Luce #2	0	0
G.Skill Ripjaws 4 #1	0	2
G.Skill Ripjaws 4 #2	0	1
Micron Branded #1	43	3127
Micron Branded #2	54	3836

Of the twelve memory modules we tested, eight showed bit flips during our 4-hour experiment. And of these eight failures, every memory module that failed at default settings was on DDR4 silicon manufactured by Micron. The Geil branded modules contained SK Hynix and the G.Skill modules contained Samsung silicon.

Our base system was highly configurable. After seeing errors using “optimized defaults” or XMP defaults, we decided to reduce our memory refresh rates (less frequent refreshing) in order to spur more bit flips. And as you can see from the right hand side of the graph, this increased the number of bit flips on the failing modules by orders of magnitude, whereas it only encourages 1 or 2 bit flips on the otherwise stable memory modules.

Although the sample size of memory we tested was very small, we can definitively say that Rowhammer bit flips are most certainly reproducible on DDR4.

FUTURE EXPERIMENTS

This paper has focused primarily on innovations that we created using publicly available information on the Rowhammer anomaly. However, we believe that current research is only scratching the surface of a potentially much larger issue. In the coming months, we will be investigating:

1) According to the original paper from Yoongu Kim and CMU, it appears as though Intel processors can show 200 times the number of bit flips versus comparable AMD processors. We would like to experiment and determine why a memory error is more exploitable on specific processor types.
2) Rowhammer on ARM. We intend to create an ARM processor Rowhammer attack so we can start to investigate consumer devices and/or future server platforms.
3) External Rowhammer attacks using DMA and RDMA. We believe that high speed DMA might be able to trigger Rowhammer events on vulnerable systems. This could possibly allow for outside attackers to cause bit flips or ECC events external to Rowhammer failing systems.
4) Can modern storage arrays or platforms be attacked externally in a Rowhammer fashion? Many storage devices use memory as cache, so it might be possible to attack them in this manner.
5) Rowhammer on GPU. We intend to investigate enterprise co-processors and GPU's such as the Nvidia Tesla with 4992 cores or the Intel Xeon Phi with 61 cores
6) We will begin to research semaphore usage on NUMA platforms. Some ideas include using Read-Modify-Write and Compare-and-Swap atomic operations on multiprocessor systems.

We believe that our findings on DDR4 are simply the tip of the iceberg. We intend to aggressively pursue new test methods and will update this paper as we acquire more memory samples to test.

AUTHOR'S NOTE

This is the first paper that we have ever written publicly in regards to a computer defect. Since our founding eleven years ago, we have always worked directly with computer manufacturers and attempted to help them develop more reliable products and promote their positive attributes. However, even modern enterprise computers are now viewed as a commodity. In that regard, quality and reliability have been taking a backseat to revenue and profits.

Currently, there are researchers and hackers all over the world trying to determine better ways to exploit Rowhammer and related problems. The computer industry does not wish to talk about Rowhammer because it is a liability for them. They have known about this issue and they rolled the dice in shipping defective products.

For computer companies, there is a tremendous loss of both reputation and profit when major hardware problems are found in the field. Fixing problems after a product is released are not only costly; this can also be perceived as wasting precious time that could be utilized developing new technologies. Many times, computer defects are only resolved when consumers demand a solution.

We have written this paper to show that this problem is very easy to exploit and we believe that there are many alternative ways for this problem to occur either naturally or through more aggressive exploits. If the industry does not step up and fix problems such as Rowhammer, then hardware problems have the potential to destabilize our entire computing infrastructure. This is especially true if we continue to see hacking attempts that focus on hardware defects.

ROWHAMMER RESEARCH THAT INSPIRED US

The original paper by Yoongu Kim and researchers from Carnegie Mellon University:

https://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

Barb Aichinger and team at FuturePlus Systems's paper that discusses Rowhammer and semaphores:

http://www.futureplus.com/images/FS2800/The%20Known%20Failure%20Mechanism%20in%20DDR3%20memory%20called%20Row%20Hammer.pdf

Mark Seaborn, Thomas Dullien, and Google's Project Zero Rowhammer research:

http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html

Daniel Gruss and Graz University of Technology's paper on the Javascript Rowhammer exploit:

http://arxiv.org/pdf/1507.06955v2.pdf

A Blog Stating that Rowhammer on DDR4 Might be a thing of the past. Also, this blog has links to Samsung stating they are "Row Hammer Free" and Micron stating that their DDR4 performs TRR in the background.

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/

About the Author and Research:

Mark Lanteigne is the CTO and founder of Third I/O Inc. He has been involved in enterprise test and test tool development since 1996. He was previously co-founder and the Director of Test and Test Tool development at Medusa Labs. And prior to Medusa, Mark was the lead test engineer for the Dell Poweredge product test team.

This paper would not have been possible without the technical expertise and assistance of Dr. David Schinke and George Pee, both of Georgetown, Texas. David has always been a brilliant coder; able to transform complex ideas into beautifully clean, functional, and highly effective code. And George has always helped out David and Mark when they lose hope at finding solutions to complex problems. We would like to give a special shout out to George's "inefficient" code that eventually became our most aggressive Rowhammer test case.