All.Net Analyst Report and Newsletter

Welcome to our Analyst Report and Newsletter

Error-induced misoperation - rowhammer

Defined long ago as "Errors caused by the attacker induce incorrect operations." Examples include the creation of a faulty network connection to deny network services, the intentional introduction of incorrect data resulting in incorrect output (i.e., garbage in - garbage out), and the use of a scratched and bent diskette in a disk drive to cause the drive to permanently fail.[1]

> See also: **hardware failure - system flaw exploitation** - "Known hardware or system flaws are exploited by the attacker. Examples include a hardware flaw permitting a power-down instruction to be executed by a non-privileged user, causing an operating system to use results of a known calculation error in a particular microprocessor for a key decision, and sending a packet with a parameter that is improperly handled by a network component."[2]
>
> See also: **induced stress failures** - "Stresses induced on a system cause it to fail. Examples include paging monsters that result in excessive paging and reduced performance, process viruses that consume various system resources, and large numbers of network packets per unit time which tie up systems by forcing excessive high-priority network interrupt processing."[3]

Rowhammer[4] is the latest invocation of this concept, and it is indeed worthy of your thoughts and considerations.

> "Rowhammer" is a problem with some recent DRAM devices in which repeatedly accessing a row of memory can cause bit flips in adjacent rows. … by repeatedly accessing two "aggressor" memory locations within the process's virtual address space, they can cause bit flips in a third, "victim" location. The victim location is potentially outside the virtual address space of the process — it is in a different DRAM row from the aggressor locations, and hence in a different 4k page (since rows are larger than 4k in modern systems)."[5]

I love this stuff

When I was in graduate school in the early 1980s I studied testing, and testing sometimes revealed such pattern sensitivity faults in components. Of course the idea was to test and not use these components because they would produce such failures, or to use redundancy to cover those faults so they would not produce an externally visible failure. But the inevitable tradeoff between cost, performance, and reliability ultimately leads to exploration by malicious actors, and here we see a classic example realized in the modern world.

---

1 F. Cohen, et. al. "A Preliminary Classification Scheme for Information System Threats, Attacks, and Defenses; A Cause and Effect Model; and Some Analysis Based on That Model", 1998, Computers and Security and http://all.net/journal/ntb/cause-and-effect.html (see http://all.net/CID/Attack/Attack67.html)
2 I.b.i.d., at http://all.net/CID/Attack/Attack67.html
3 I.b.i.d., at http://all.net/CID/Attack/Attack69.html
4 http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html
5 For technical details see: http://users.ece.cmu.edu/~yoonguk/papers/kim-isca14.pdf

> "This works because DRAM cells have been getting smaller and closer together. As DRAM manufacturing scales down chip features to smaller physical dimensions, to fit more memory capacity onto a chip, it has become harder to prevent DRAM cells from interacting electrically with each other. As a result, accessing one location in memory can disturb neighbouring locations, causing charge to leak into or out of neighbouring cells. With enough accesses, this can change a cell's value from 1 to 0 or vice versa."

Not the first time

One of the nice things about well done research is that it stands on the shoulders of giants. In this case, the report references a 2003 paper[6] using bit flips to escape the Java virtual machine. The various papers also cite (between them) about 90 prior papers starting in the 1970s, a few in the '80s, several in the '90s, and the rest from this century.

Not the only thing

Exploiting this fault to produce a desired security failure is a non-trivial exercise.

- First you have to figure out what part of memory to attack:
  - An intelligence process is involved, which is made easier in some operating environments which provide helpful details on memory mapping and locations of processes. Protection against access to such information seems reasonable.
- Then you have to be able to get physical proximity for processed under your control:
  - This is largely a matter of luck, but by creating and destroying processes one after another, eventually, you may get memory alignment. A paging monster may also be used to force other memory out and gain access to the desired memory areas.
- Then you have to invoke behaviors in the right part of memory to induce bit flips:
  - This just involves traveling ones and zeros in the right portion of memory – it's easy enough to do once you know where things are physically. Of course you need to be careful you flip the right bit(s) and don't flip them back again...
- Then you have to take advantage of them in a timely fashion.
  - If you wait too long, they may get corrected through a refresh process or other similar overwriting activity.
- These don't work each time, so you need to have extensive loops to try again and again, and it takes time to win.
  - This makes them fairly detectable depending on how many times you need to try and what sorts of detection are present in the system. In most cases today, this will go unnoticed indefinitely.

Summary

If you are going to really understand security, you need to do your research. And it's a joy to see those that take basic concepts, exploit them to their logical conclusion, chain them together, and then tell the rest of us about it. What we knew long ago is proven to be true again and again, and that means that science is working. We just need to use it more.

6 https://www.cs.princeton.edu/~appel/papers/memerr.pdf

