

Row Hammer 漏洞攻击研究

王文伟, 刘培顺

(中国海洋大学信息科学与工程学院, 山东 青岛 266100)

摘要: 对目前已出现的 Row Hammer 漏洞利用的相关技术和防御方法进行详尽的分析、总结和讨论, 指出其蕴含的安全问题及可能的防范方法。然后对现有国内外研究工作进行了总结, 指出了安全研究未来有价值的2个方向: ARM 架构下 Row Hammer 漏洞利用技术和防范方法, 以及 Row Hammer 漏洞的准确受控制的攻击方法。

关键词: Row Hammer 攻击; 提权; 动态随机访问存储器; 缓存驱逐; 页表喷射; 内存伏击

中图分类号: TP390

文献标识码: A

doi: 10.11959/j.issn.2096-109x.2018007

Study of Row Hammer attack

WANG Wenwei, LIU Peishun

College of Information Science and Engineering, Ocean University of China, Qingdao 266100, China

Abstract: The related technologies and defense methods of the current Row Hammer vulnerabilities were analyzed and summarized, and the security problems and possible precautions were pointed out. At the end, the current research work both at home and abroad is summarized. Two valuable directions for security research in the future are pointed out: Row Hammer exploit techniques and precautions in the ARM architecture, and accurate and controlled attacks on Row Hammer vulnerabilities.

Key words: Row Hammer, privilege escalation, DRAM, cache eviction, page table spraying, memory waylaying

1 引言

计算机中所有的程序都是运行在内存上的, 所以内存的安全关系着计算机系统能否正常运行。近年来的研究表明, 动态随机访问存储器(DRAM, dynamic random access memory)内存存在设计缺陷, 将该缺陷所产生的漏洞称为 Row Hammer。Row Hammer 漏洞在2014年被首次提出^[1], 该漏洞的主要表现是: 当攻击者反复敲打内存中特定的行时, 会导致

其相邻行的内存单元发生反转, 从而改变内存中的数据。随后在2015年, 谷歌安全研究小组提出了2个可以基于该漏洞进行提权的利用^[2,3], 一个是在Linux系统上利用该漏洞实现了提升访问整块内存区域的权限, 另一个是可以越过谷歌的Native Client (NaCl) 沙盒, 直接使用主机的系统调用。目前, 国外一些专家学者对该漏洞进行了深入研究, 并提出了独特的见解与利用, 但是国内对该漏洞的研究还比较少^[4]。

收稿日期: 2017-12-08; 修回日期: 2018-01-03

通信作者: 刘培顺, liups@ouc.edu.cn

基金项目: 国家重点研发计划基金资助项目 (No.2017YFC0806200)

Foundation Item: The National Key R&D Plan Program of China (No.2017YFC0806200)

2 背景知识

2.1 动态随机访问存储器

内存是由动态随机访问存储器和静态随机访问存储器 (SRAM, static random access memory) 两部分组成, 其中 DRAM 是由许多重复的存储单元组成, 每个存储单元又由一个电容和一个访问晶体管组成。电容可以存储 1 bit 数据, 对电容充放电后电容带电荷的多少分别对应二进制数据 0 和 1。其中访问晶体管和行地址线路相连, 电容和列地址线路相连。图 1 为内存单元组合缩略图和电气原理图。

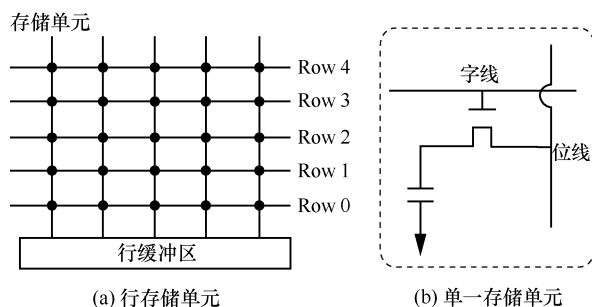


图 1 内存单元组合

图 1 中每个黑点为一个存储单元, 一系列的存储单元组成一个存储阵列, 同一存储阵列中除了行组成的存储单元外, 还包括一个行缓冲区。内存读写单位为行, 每次从指定行读取数据包括以下 3 个步骤。

激活: 字线被选中, 将该行的数据从位线传入行缓冲区。

访问: 包括读写操作, 针对行缓冲区进行操作。

关闭: 拉低字线, 同时行缓冲区中数据从位线写入到行, 并且清空行缓冲区。

2.2 动态随机访问存储器架构

DRAM 是由通道、颗粒阵列、存储阵列 3 个部分组成的层次结构, 其架构如图 2 所示。其中内存控制器和 DRAM 之间的物理连接模块称为通道。在通道内, 连接到主板上的物理内存模块称为双列直插存储器模块 (dual inline memory module), 其通常由一个或两个颗粒阵列组成。而一个颗粒阵列由多个存储阵列组成, 每一个存储阵列则是多个单元组成的二维空间集合, 通常一个存储阵列由 $2^{14} \sim 2^{17}$ 行和一个行缓冲区组成。

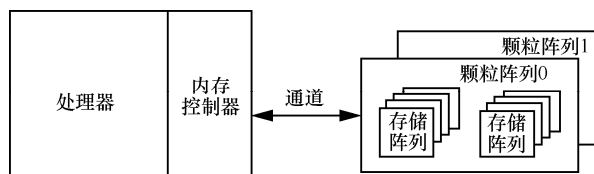


图 2 DRAM 架构

2.3 Cache 架构

在计算机系统中, 由于 CPU 和主存之间逐渐增大的差距, 系统设计者被迫在 CPU 寄存器文件和主存之间插入一个小的 SRAM 高速缓存存储器, 称为高速缓存。随着缓存等级的增大, 缓存的大小也随之增大, 同时访问时间也增加。其中 LLC 被多个处理器内核共享, 所以访问时间更长, 因此将其分为不同的片, 这样就能够被多个内核同时访问。

当处理器请求数据时, 操作系统首先为当前进程分配一个虚拟地址空间, 处理器请求的数据虚拟地址则位于这个虚拟地址空间。虚拟地址分为两部分, 低地址位表示虚拟页面偏移, 剩下的高位则表示虚拟页号, 虚拟页号作为页表中的索引。

2.4 Row Hammer 原理

随着动态随机访问存储器密度的不断增大, 内存单元越来越小, 从而能够存储更小的电荷。结果是内存单元之间噪声容限降低, 导致相互独立的 2 个内存单元之间的电荷相互影响。Row Hammer 攻击是针对内存硬件芯片设计上的这一缺陷, 其原理是反复读写 DRAM 内存单元中同行地址, 使相邻行发生电荷泄露, 从而导致相邻行产生位反转现象, 即 0 反转为 1, 1 反转为 0。这种反复高频率激活关闭 Row 的操作称为“捶打”, 即 Row-Hammer, Row-Hammer 分为 single-sided 和 double-sided 这 2 种。Single-sided 是反复高频率激活关闭同一行, 从而造成其相邻两行发生位反转。Double-sided 则是反复高频率激活关闭两行, 从而造成其中间行发生位反转。

3 漏洞利用技术

由前面部分内容可知, 现在计算机体系为了提高性能加入了缓存机制, 这导致了内存不能够快速激活行从而触发 Row Hammer 漏洞。同时由

于 Row Hammer 漏洞所产生的比特反转具有不可控性,所以要想利用漏洞实施端到端的可控攻击,必须解决以下 2 个问题。

1) 能够快速访问内存,即绕过缓存机制。

2) 物理内存信息,即欺骗受害者把敏感数据存放在攻击者受控制的物理内存页上。

3.1 缓存刷新技术

缓存刷新(cache flush)技术是由卡内基梅隆大学的 Yoongu 等^[1]于 2014 年提出的,主要原理是通过使用 clflush 指令刷新缓存,从而使访问内存的操作直接指向 DRAM,其形式如下。

Code-Hammer

```
{
    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
    Jmp Code-Hammer
}
```

首先,2 个 mov 指令从位于 DRAM 上的 X 和 Y 地址读取数据,并将数据放入寄存器和缓存中。然后 clflush 指令将刚放入缓存中的数据逐出,最后代码跳回第一个指令并继续从 DRAM 上读取数据。在无序处理器上,上述代码会产生多个 DRAM 读取请求,所有的这些请求再发送给 DRAM 之前都会在内存控制器中排队,即(req x, req y, req x, req y, …)。这时如果选取的 X 和 Y 位于同一存储阵列上的不同行,那么内存管理器会重复打开和关闭这两行,即(act X, read X, pre X, act Y, read Y, pre Y),从而导致比特反转。具体实验可以分为 2 种,即一次测试所有行和每次测试一行。其中一次测试所有行可以在每一行经过多次打开和关闭后,快速确定所有能够发生比特反转的内存单元,而一次测试一行则可以确定具体发生比特翻转的内存单元。而这 2 种方法都需要 3 个输入参数,即激活间隔、刷新闻隔及数据格式。首先激活间隔,确定行激活、关闭的频率,然后刷新闻隔确定实验中刷新频率,数据格式则是确定了在发生比特翻转之前的初始数据值。一次测试所有行首先是把数据格式写到整个内存中,然后在一个刷新闻隔时间段内以激活间隔为

速率来激活关闭一行,然后重复上面步骤,直到内存中所有行都执行完该步骤。最后读取整个内存中数据,并且确定所有发生比特翻转的内存单元。而一次测试一行过程与之类似,两者实现过程的伪代码如下。

一次测试所有行:

```
TestBulk(AI,RI,DP)
    setAI(AI)
    setRI(RI)
    N←(2xRI)/AI
    writeAll(DP)
    for r←0…ROWMAX
        for i←0…N
            ACT r…row
            READ 0th col
            PRE rth row
    readAll()
    findErrors()
```

一次测试一行:

```
TestBulk(AI,RI,DP)
    setAI(AI)
    setRI(RI)
    N←(2xRI)/AI
    for r←0…ROWMAX
        writeAll(DP)
        for i←0…N
            ACT rth row
            READ 0th col
            PRE rth row
    readAll()
    findErrors()
```

目前,谷歌安全团队“Project Zero”的两位安全研究人员 Mark Seaborn 和 Thomas Dullien 已经在 X86-64 平台上通过使用 Clflush 指令,成功地引起 Row Hammer 漏洞,并结合 PTE 的某一位的变化直接获得内核权限^[2,3]。

3.2 缓存驱逐技术

缓存驱逐(cache eviction)技术主要是重复访问位于同一缓存驱逐集中的内存地址,缓存驱逐集是一组相同地址的集合,其中只有当 2 个地址映射到同一缓冲行时,这 2 个地址相同。因此当

访问内存地址属于同一个缓存驱逐集时,在读取数据时会自动刷新缓存。缓存驱逐技术主要包括以下 3 个步骤。

1) 确定驱逐集:一个驱逐集由多个冲突地址和一个侵略者地址构成。

2) 确定 Cache 替换策略: Pseudo-LRU 算法是对 LRU 算法的改进,通过实验构造一个高缺失率的序列。

3) 构造访问序列:对于一个三级缓存,12 路组相连映射的系统,可以构造如下访问序列
 $A_0(\text{Row}_0, \text{setx}) \ X_1(\text{setx}) \ X_2(\text{setx}) \cdots X_9(\text{setx}) \ X_{10}(\text{setx})$
 $X_{11}(\text{setx}) \ X_1(\text{setx}) \ X_2(\text{setx}) \ \cdots \ X_9(\text{setx}) \ X_{10}(\text{setx})$
 $A_0(\text{Row}_0, \text{setx}) \ X_1(\text{setx}) \ X_2(\text{setx}) \ \cdots X_9(\text{setx}) \ X_{10}(\text{setx})$
 $X_{11}(\text{setx}) \ X_1(\text{setx}) \ X_2(\text{setx}) \ \cdots X_9(\text{setx}) \ X_{10}(\text{setx})$
 ...

其中, A_0 对应 Row_0 映射到组 X 中,然后访问同属于组 X 的冲突地址 $X_1, X_2, \cdots, X_{10}$,根据 Cache 替换策略, A_0 由于处于最近最少使用的位置,所以当访问 X_{11} 时, A_0 被驱逐出去,这样 Row_0 就会被访问一次。接下来继续访问同属于组 X 的冲突地址 $X_1, X_2, \cdots, X_{10}$,这时 X_{11} 处于最近最少使用的位置,当第二遍访问 A_0 时, X_{11} 被驱逐出去。这样按上述序列重复进行 N 次,则能够实现重复 N 次访问 Row_0 的效果。同理,对其他行进行相同的操作,即可实现对该行的 N 次访问。这样快速地对某一行进行重复驱逐与加载,可以触发 Row Hammer 漏洞。这种攻击技术相比 clflush 指令更具一般性,它适用于任意系统架构、编程语言及运行环境。Gruss 等^[5]利用在网页中广泛使用的 JavaScript 语言来实现缓存驱逐技术,这种基于 JavaScript 的 Row Hammer 能够远端触发该漏洞,而且影响范围广。

3.3 非缓存访问指令

在计算中,数据引用有不同的模式。一些具有时间特性,即数据在将来很快被再次访问。一些具有空间特性,即相邻位置的数据将被访问。同时也存在一些非时间特性的,这种数据只会被引用一次。由于大多数数据访问呈现时间和空间局部性,所以引入了高速缓存机制来提高性能,但是非时间特性数据的访问会污染缓存并降低性能。非缓存指令是为了解决非时间特性数据访问

可能造成污染缓存并降低性能这一问题而引入的,即一旦出现预期的非时间特性数据引用,则程序或编译器可以使用非缓存指令来最小化缓存污染。其中非缓存指令包括加载、存储和预取 3 种指令,这里使用非缓存指令,因为它们把目标内存视为无缓存的“写入组合”类型,即 CPU 直接访问内存而不需要通过缓存,如图 3 所示。

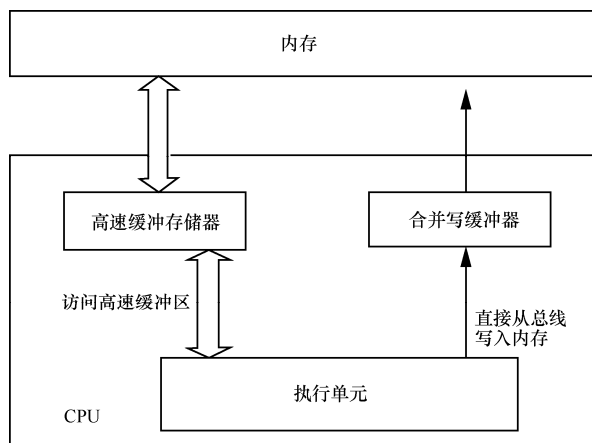


图 3 非缓存访问

由图 3 可以看出,使用非缓存指令可以绕过缓存机制,但是由于引入了写入组合缓冲区,使 CPU 访问 DRAM 速率降低。为此,Rui 和 Seaborn^[6]提出一种刷新写入组合缓冲区的方法,即通过缓存存储器访问非缓存写入数据地址来实现,从而能够快速地激活关闭行,继而触发 Row Hammer 漏洞,其实现原理如下。

Code-Hammer

```
{
    Movnti %eax, (X)
    Movnti %eax, (Y)
    Mov %eax, (X)
    Mov %eax, (Y)
    Jmp code-Hammer
}
```

由于非时态存储存在于重要的软件 C 库中,具体来说,它们会被 C 库中的 memset 和 memcpy 函数所调用,即当填充/复制的数据预计不会很快被访问时,它们是不会被存储到缓存。基于这一发现,Rui 和 Seaborn 分析了在不同库中,非缓存指令在 memset/memcpy 中的使用方法,以及在

什么情况下会被执行。他们发现在主流库中,大多数会使用非缓存指令。为了利用 `memset/memcpy` 函数触发 Row Hammer 漏洞,需要大量数据的移动从而执行非缓存指令。但是大量数据移动会导致较低的锤击速率,因此在实验中通过直接 `memset/memcpy` 的方式对不同的库进行锤击,并且成功地引发了比特翻转。

3.4 页表喷射

页表喷射技术不同于以往的关于 Row Hammer 漏洞的利用技术,以前的技术都是通过利用随机的 Row Hammer 漏洞来反转页表条目中的某些位来实现 root 权限提升。而页表喷射这种技术则具有概率性,首先通过在内存中部署大量的页表,以期望至少有一个能够出现在易受攻击的物理内存页上。然后使易受攻击的物理内存页发生比特反转,从而使受害者的页表指向一个任意物理内存位置。根据物理内存布局可知,这样的任意物理内存位置包含一个受攻击者控制的页表页,而这个页表页允许攻击者将他们的页表映射到受控制的地址空间上。基于这点,攻击者可以覆盖页表条目,并且能够访问任意物理内存中的页从而实现提权。其过程如下:首先在 `/dev/shm` 中创建一个文件,并在该文件的每 4 kB 内存开始处写入一个标记数,从而方便识别映射的内存页。同时为了避免内存页被分配到连续的物理内存地址,需要调用 `mmap` 从物理内存申请一块空间,随后调用 `madvise`,从该空间中释放一个内存页,从而系统重新申请一个 4 kB 内存页。之后反复调用 `mmap` 来映射前面生成的文件,在此过程中,

通过调用 `munmap` 来释放掉可以进行 bit 翻转的内存页,这时系统会有很高的可能性紧接着使用这篇物理内存保存页表,反复以上过程可以触发 Row Hammer 漏洞。Seaborn 和 Dullien^[2,3]利用该技术实现了 Linux 内核提权。

3.5 内存重复数据删除技术

内存重复数据删除技术^[7,8]是一种流行的减少运行系统内存的技术,其主要原理是,当多个内存页具有相同的内容时,系统管理程序会合并这些内存内容,然后将该页保留一份,其余的删除。当其中的一个重复数据删除的页被修改,则会用写时复制技术,即先将该页复制一份,然后在这个复制的页上进行写操作。当内存中有许多内容相同的页时,该技术可以节省大量的内存空间。然而该技术有安全漏洞,即当攻击者的虚拟机和受害者的虚拟机同时位于同一主机上时,攻击者可以获取其相邻的虚拟机内存信息。基于这一技术,Razavi 等^[9]提出一种新的攻击技术——反转风水 (FFS, flip feny shui),即利用内存重复数据删除技术来实施受控制的 Row Hammer 攻击。首先,攻击者确定内存中哪些单元存在 Row Hammer 漏洞,然后选择一个合适的反转位,攻击者伪造一个和他已知的在受害者上易受攻击的内存位置的内存页,内存重复数据删除引擎就会将受害者和攻击者的页合并(如图4所示),从而使受攻击者控制的虚拟内存页和受害者自己的内存页映射到同一个攻击者选择的易受攻击的物理内存页,进而攻击者可以通过触发 Row Hammer 漏洞来修改受害者的内存地址。

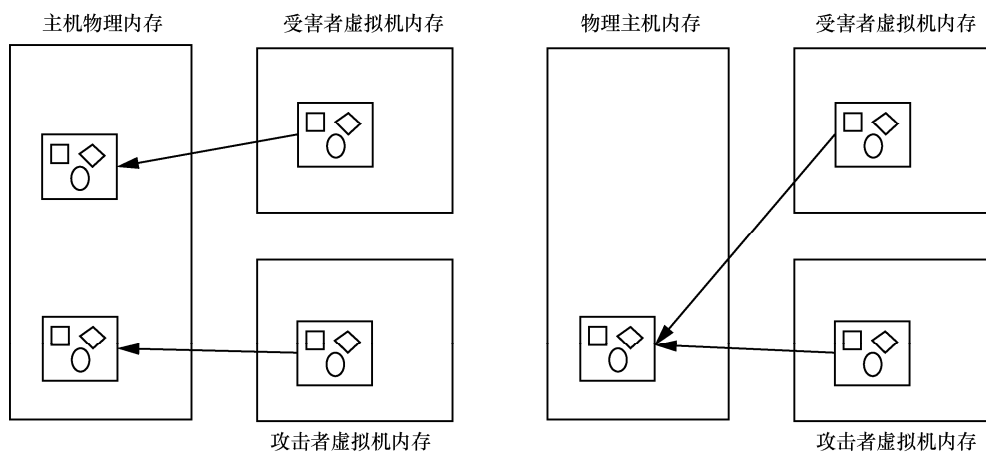


图4 内存重复数据删除

3.6 内存伏击技术

内存伏击技术会用到预取边道攻击和页缓存驱逐技术。其中预取边道攻击是用来检测虚拟内存中的数据是否映射到指定的物理内存中，而页缓存驱逐技术则是将文件页从页缓存中驱逐出去，使后续的文件访问不能从页缓存中找到与之匹配的页，从而使文件被映射到新的物理内存位置。内存伏击技术则是2种技术的结合，可以实现将目标数据映射到指定的物理内存位置上，其原理如图5所示。

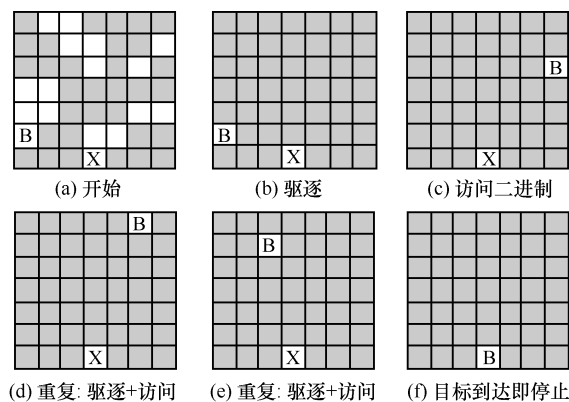


图5 内存伏击

其中空白方格代表空闲页面，浅色方格代表已分配给页缓存的页面，图5(a)表示刚开始内存的使用情况，内存中还有部分空闲页面，图5(b)将所有的空闲页面分配给页缓存，同时将目标页B从内存中驱逐出去，图5(c)是将目标页B重新加载到内存中，这时目标页B会被加载到不同位置的物理内存中，重复图5(b)和图5(c)，直到目标页B被加载到能够发生位反转的物理内存位置X上。

内存伏击技术是Gruss和Lipp等^[10]在2017年提出的，利用该技术，Daniel等实现了云端的DDoS攻击和本地的提权攻击。相比于之前的页喷射技术^[11]而言，内存伏击技术不需要分配整个内存空间，从而具有一定的隐秘性，其缺点是速度相对较慢。

4 防御技术

Row Hammer漏洞是硬件问题所导致的漏洞，所以首先想到的是从硬件方面来解决，现有的防御技术有以下几点。

1) ECC (error corrected code) 技术^[1]: ECC, 即错误更正码, 用来检验存储在DRAM中的整体数据, ECC在设计上比奇偶校验更精巧, 它不仅能够检测出多位数据错误, 并且能够同时指定出错位并改正。

2) PARA (probabilistic adjacent row activation) 技术^[1]: PARA技术的主要原理是每当一行关闭时, 内存控制器都会以一定的概率来刷新该行邻接行的值。

3) 提高刷新速率^[12], 到目前为止, 通过BIOS更新所提供的刷新速率仅提高了一倍, 这不足以防止所有针对DRAM的攻击。而且很大一部分用户没有更新BIOS, 同时这种方法会对系统性能造成显著影响。

4) TRR (target row refresh) 技术^[2]: TRR是应用在DDR4内存中的一种技术。这种技术采用了一种特殊模块, 从而可以跟踪记录内存中哪些行经常被激活, 并且刷新这些激活行相邻行的值。

从软件方面解决Row Hammer漏洞可以从以下几个方面进行。

1) 禁止使用cache相关指令, 由前面部分可知, 当使用Clflush指令时, 可以导致直接访问DRAM, 从而触发Row Hammer漏洞。例如, 目前谷歌的沙箱已经禁止使用Clflush指令^[13]。

2) 禁止一般用户访问物理地址接口pagemap, 如果一般用户访问pagemap时, 他能够获得所有物理地址和虚拟地址之间的映射关系, 有效地重复访问物理内存中的具体行, 从而触发Row Hammer漏洞。所以禁止一般用户访问物理地址接口pagemap也是一种防治措施, 目前linux内核已经禁止一般用户访问物理内存接口。

3) 基于虚拟化内存隔离技术^[4] (RDXA, RowHammer defense on Xen allocator) 系统, 该系统主要包括两部分, 第一部分是通过基于访问时间的旁路分析方法, 逆向出不同物理地址和实际内存存在芯片上的布局关系, 并将其存储在数据库中。第二部分是虚拟机监视器的物理内存分配, 在该过程中, 虚拟机监视器中的内存分配器会根据数据库中的内存物理布局数据, 将物理内存分为DRAM内存上相互不相邻

的若干块区域, 然后通过内存分配算法将这些区域分配给不同的安全实体, 其中每个安全实体间都隔离至少一行的距离, 这样保证不同安全实体间所对应的物理内存不会被物理内存放在同一个内存芯片的同一个存储阵列的相邻行上面, 从而可以有效阻止跨虚拟机的 Row Hammer 攻击。

5 结束语

Row Hammer 漏洞是由于内存设计缺陷导致的, 而内存又是当今计算机设备的基本硬件, 所以研究 Row Hammer 漏洞具有重要的现实意义。

从现有的研究成果来看, 未来对于 Row Hammer 漏洞的研究可分以下 2 个方向。

1) 以往的研究都是针对 X86 架构, 而已知的移动端则采用的是 ARM 架构, 这种架构和应用于 PC 端的 X86 架构完全不同, 所以针对 X86 架构的 Row Hammer 漏洞利用不适用于 ARM 架构。因此, 如何在移动端利用 Row Hammer 漏洞实施攻击是下一步研究的一个方向。

2) 以往的研究都是利用 Row Hammer 使页表条目中某一位发生反转, 而这种位反转是随机的、不可预测的, 从而使攻击不可靠、不受控制。因此, 如何利用该漏洞实施一个准确的受控制的攻击是下一步研究的方向。

参考文献:

- [1] KIM Y, DALY R, KIM J, FALLIN C. Flipping bits in memory without accessing them: an experimental study of DRAM disturbance errors[C]//The 41st International Symposium on Computer Architecture (ISCA).2014:361-372.
- [2] SEABORN M, DULLIEN T. Exploiting the DRAM RowHammer bug to gain kernel privileges[C]// Black Hat USA (BH-US). 2015.
- [3] SEABORN M, DULLIEN T. Exploiting the DRAM RowHammer bug to gain kernel privileges[EB/OL]. <http://googleprojectzero.blogspot.com/2015/03/exploiting-dram-RowHammer-bug-to-gain.html>.
- [4] 石培涛, 刘宇涛, 陈海波. 基于虚拟化内存隔离的 RowHammer 攻击防护机制[J]. 信息安全学报, 2017,2(10):1-12.
SHI P T, LIU Y T, CHEN H B. Defense against RowHammer attack with memory isolation in virtualized environments[J]. Journal of Cyber Security,2017,2(10):1-12.
- [5] GRUSS D, MANGARD S. Rowhammer.js: a remote software-induced fault attack in JavaScript[C]// International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment.2016:300-321.
- [6] QIAO R, SEABORN M. A new approach for RowHammer attacks[C]//The 9th IEEE International Symposium on Hardware Oriented Security and Trust (HOST). 2016:161-166.
- [7] BOSMAN E, RAZAVI K, BOS H, et al. Dedup est machina: memory deduplication as an advanced exploitation vector[C]//The 37th IEEE Symposium on Security and Privacy(S&P).2016: 987-1004.
- [8] XIAO J D, ZHANG X, HUANG H, et al. Security implications of memory deduplication in a virtualized environment[C]// International Conference on Dependable Systems and Networks (DSN). 2013: 1-12.
- [9] RAZAVI K, GRAS B, BOSMAN E, et al. Flip feng shui: hammering a needle in the software stack[C]// The 25th USENIX Security Symposium.2016.
- [10] GRUSS D, LIPP M, SCHWARZ M, et al..Another flip in the wall of row hammer defenses[EB/OL]. https://www.researchgate.net/publication/320179942_Another_Flip_in_the_Wall_of_Rowhammer_Defenses.
- [11] YIM K S. The RowHammer attack injection methodology[C]// The 35th Symposium on Reliable Distributed Systems . 2016:1-10.
- [12] Row hammer privilege escalation[EB/OL]. <https://support.lenovo.com/us/en/product-security/Row-Hammer>.
- [13] AWEKE Z B, YITBAREK S F, QIAO R, et al. ANVIL: software-based protection against next-generation RowHammer attacks[C]//The Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems.2016:743-755.

[作者简介]



王文伟 (1993-), 男, 陕西咸阳人, 中国海洋大学硕士生, 主要研究方向为信息安全。



刘培顺 (1975-), 男, 山东菏泽人, 中国海洋大学讲师, 主要研究方向为网络与信息安全。