文章编号:1671-4229(2021)03-0030-14

内存 RowHammer 攻击与防御综述

杨莹莹, 侯 锐*
(中国科学院信息工程研究所 信息安全国家重点实验室, 北京 100093)

摘 要: RowHammer 攻击利用 DRAM 的固有特性,可对不具有访问权限的内存数据进行修改。通过攻击可导致提升特权、拒绝服务,甚至数据泄露等严重后果,破坏了内存数据的完整性和机密性。低成本且高效的攻击手段给用户和企业的数据保护带来了很大的威胁和挑战。针对 RowHammer 的攻击和防御已经成为该领域的一个研究热点。文章首先简要介绍内存的基本架构,对 RowHammer 攻击进行分析和总结;然后从硬件和软件的角度对现有的防御方法进行梳理,讨论了现有缓解技术的可行性和实用性;最后指出了未来值得研究的发展方向。

关键词: RowHammer 攻击; 位翻转; 硬件安全; RowHammer 防御
中图分类号: TP 302.1;TP 309.2 **文献标志码**: A

Overview of memory RowHammer attacks and defense technology

*YANG Ying-ying, HOU Rui**
(Institute of Information Engineering of CAS, State Key Laboratory of Information Security, Beijing 100093, China)

Abstract: Using the inherent characteristics of DRAM, the RowHammer attack can modify memory data that does not have access rights, destroy the integrity and confidentiality of memory data, and may lead to serious consequences such as privilege escalation, confidential data leakage, and denial of service. Low-cost and efficient attack methods have brought great threats and challenges to the data protection of users and enterprises. Attack and defense against memory security have become a research hotspot in this field. This article first briefly introduces the basic architecture of memory, analyzes and summarizes the RowHammer attack; then combs the existing defense methods from the perspective of hardware and software, discusses the feasibility and practicality of existing mitigation technologies; and finally points out the development direction worth studying in the future.

Key words: RowHammer attack; bit flip; hardware security; RowHammer defense

动态随机存储器(Dynamic Random Access Memory, DRAM)作为最常用的主存储器,存储着计算机运行所必需的敏感数据、程序代码等内容。内存安全一直是公众最为关注的问题之一。研究证明,目前系统中较常用的 DRAM 芯片普遍存在 RowHammer 漏洞,它是一种由软件引起的影响 DRAM 芯片安全的硬件故障。它利用 DRAM 存储单元密集的特性,在单个刷新周期内重复访问内存的某一行,可加快存储单元的电荷释放,最终导致相邻行数据发生位翻转错误。RowHammer 漏洞暴露了当前 DRAM 芯片存在的设计缺陷,利用此漏洞可以破坏 DRAM 数据的完整性和机密性。通过对 DRAM 单元中的数据进行未经授权的修改,可导致特权提升,拒绝服务甚至数据泄露等严重后果。目前针对 RowHammer 的攻击和防御已成为计算机安全领域关注的重点话题。

作者简介: 杨莹莹(1996—),女,硕士研究生. E-mail: yangyingying@iie.ac.cn
* 通信作者. E-mail: hourui@iie.ac.cn

引文格式: 杨莹莹, 侯锐. 内存 RowHammer 攻击与防御综述[J]. 广州大学学报(自然科学版),2021,20(3):30-43.

自 2014 年 Kim 等[1]发现现有的 DDR3 模块中存在 RowHammer 漏洞以来,RowHammer 引起了大量学者的关注和研究。2015 年,谷歌安全团队[2]发布了 2 个具体的漏洞利用,即利用漏洞获取权限,从浏览器沙箱中逃脱和提升 Linux 上内存的访问权限。证明了 RowHammer 的存在能够引发严重的安全问题。Gruss 等[3]证明 RowHammer 攻击不仅可以通过运行本地代码进行攻击,还可以利用浏览器中 JavaScript 的特点实现远程攻击。

从攻击的范围来看,目前 RowHammer 漏洞波及到几乎所有类型的计算系统,包括个人计算机[2-5]、虚拟机[6-7]、移动设备[8-11]和云服务器[5-6,8,12-16]等。2016 年,Xiao 等[7]主要对 Xen 半虚拟化的 VM 进行攻击。这种攻击可使来宾 VM 拥有对共享计算机上任意物理页面的读写访问权限,打破了 Xen 半虚拟化的内存隔离;同年,Veen 等[8]提出了对移动设备的攻击,并在真实的系统上得到验证;2018 年,Frigo 等[11]指出可以利用移动设备中配置的 GPU 进行远程攻击。2020 年,Nethammer[15]首次提出可远程攻击服务器,处理网络请求的软件如果使用未缓存的内存、非时间指令或刷新指令,那么攻击会导致位翻转。

从攻击的目标来看,位翻转目标包括页表[2,8]、对象指针[4,11]、操作码[5]和加密密钥[6]等。执行攻击的环境包括本地执行代码[2,5-6,8-11,13,16]、利用浏览器的 JavaScript 执行代码[2-4,17]和远程客户端[12,15]等。内核权限提升主要是通过破坏页表达到攻击目的,2019 年,Cheng 等[18]指出操作系统中存在内核和用户共享的缓存区(double-owned buffers),这种缓存区的存在使 CATT[19]内核隔离失效,提出了 CATTmew 攻击。CATTmew 利用内存伏击技术将页表翻转,有效地破解了内核隔离防御,获得内核权限。2020 年,Zhang 等[20]指出内存访问时会从内存中获取页表条目(PTE),这相当于对 PTE 进行了一次攻击,因此,提出了针对 PTE 的 PThammer 攻击。

从造成的攻击结果来看,大部分攻击实现了权限提升[2-3,5-6,8-9]。针对跨 VM 攻击,主要是打破了虚拟机之间的内存隔离,获取到对任意内存的访问权限[6-7];针对浏览器的攻击,利用了 JavaScript 的相关特性,可在浏览器中获得任意内存的读写访问权限[4];针对服务器的攻击主要是为了破坏数据,引起系统崩溃,并最终导致拒绝服务[5,15,17];RAMBleed 攻击通过提前获知受害者进程的内存分配策略,且短时间内数据不会发生变化的前提下,可利用位翻转推测同一硬件上运行的其他进程的敏感数据的值[14]。可以看出,RowHammer 漏洞的危害性很大,对云服务提供商、个人电脑和移动设备等的数据安全构成严重威胁。

针对 RowHammer 攻击,研究者提出了许多防御方案。例如,联想和苹果等制造商通过缩短刷新间隔来缓解防御;Google 的沙箱中禁止使用 clflush 指令;Linux 内核中也对 pagemap 接口的访问权限进行限制;pTRR 最早在 2014 年就出现在英特尔 CPU 的内存控制器中,用来保护易受攻击的 DRAM 模块。目前实施的防御方法提升了攻击难度,但并未从根本上解决问题。

近几年,RowHammer 攻击和防御已成为热门研究领域,许多新的攻击方法和防御方案不断被提出。本文首先对 RowHammer 漏洞的背景知识进行介绍,探讨漏洞产生的原因和所需条件;然后系统地分析和归类现有的攻击方法;接着从硬件和软件方面对现有的防御方法进行归纳和总结;枚举针对攻击的检测技术;最后基于现有的攻击和防御技术的优缺点,讨论未来攻击和防御的主要挑战和潜在研究方向。

1　背景知识

DRAM 单元密度的增加使得存储单元更容易受到相互干扰,这种干扰通常会降低内存的可靠性,甚至会导致内存损坏。因此,本节中首先对内存系统进行剖析,介绍产生 RowHammer 漏洞的背景知识,最后简要介绍 RowHammer 攻击成功所需的条件。

1.1　内存系统

内存系统主要由 3 部分组成:①存储数据的 DRAM 模块;②负责读写调度和命令转换的内存控制器;③连接 DRAM 和控制器的相关总线。

1.1.1　DRAM 模块

内存模块通常由多个通道(channel)组成,这些通道作为 DRAM 和内存控制器之间的数据传输路径,每个通道对应一个内存控制器。一个通道包含多个 rank,一个 rank 由多个 DRAM 芯片(chip)组成。DRAM 芯片内包含多个 bank,bank 之间可并行传输数据。bank 是由行、列、行缓冲区

和存储单元组成,bank 中的全部行共享一个行缓冲区。图1和图2分别展示了内存系统的整体架构和 bank 的内部结构。

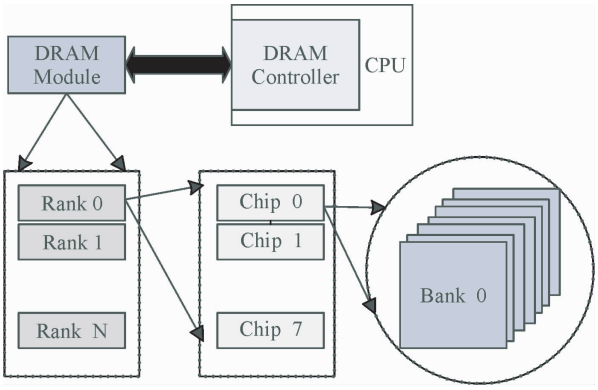

图1 内存系统架构

Fig. 1 Memory system architecture

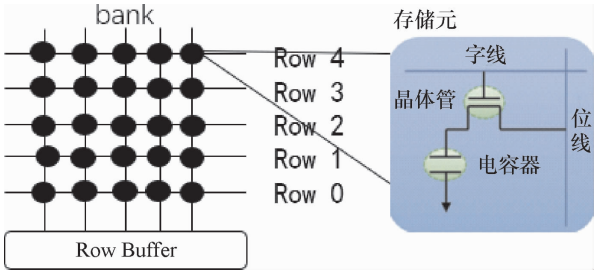

图2 Bank 结构

Fig. 2 Bank structure

存储单元由2个组件组成:电容器和晶体管。当电容器充电时,数据1存储在存储单元中。由于电容存在漏电现象,电容里的电荷会随着数据读取或时间的变化产生电荷泄漏问题,反复的读取操作对存储单元具有破坏性。如果在特定的时间内没有刷新,数据就有可能丢失。

近年来随着内核数量的增加,内存系统逐渐成为整体性能的瓶颈。为了增大容量,存储单元紧密的排在一起,导致相邻的存储单元之间容易相互干扰。RowHammer 攻击正是利用相邻行之间的干扰特性,通过反复激活特定行,导致其相邻行发生位翻转,从而达到获取权限等目的。

1.1.2 内存控制器

内存控制器控制主存和处理器之间的数据和信号传输。它接收 CPU 的读写请求,对接收到的请求进行处理后发送访存信号到内存。它的主要功能是根据命令的优先级和调度算法从请求缓存区中调度请求,并将系统请求转换为内存系统可识别的命令。内存根据命令和地址激活数据所在的行,整行读出到行缓冲区中,内存控制器一次只能激活一行数据,并按列地址操作数据。当进行内存访问时,请求行通常有3种状态:

(1)行命中:访问的行已经在行缓冲区中,可直接对行缓冲区进行操作,这种情况下访问延迟最小。

(2)行关闭:行缓冲区中没有缓存行,需进行行激活操作。

(3)行冲突:访问的行和行缓冲区中的行不同,需要先将行缓冲区中的行写入内存,激活要访问的行,这种情况下延迟最高。

以上3种状态与打开页策略(open-page)或关闭页策略(closed-page)息息相关。打开页策略是指在内存访问结束后继续保持当前行打开状态,延迟预充电操作。而关闭页策略指当前访问结束后关闭当前行,避免了行缓冲区管理的复杂性,提供了一致的访问延迟。根据数据访问通常具有局部性的特点,打开页策略是目前大多数系统的常用配置。

由于内存控制器负责接收所有来自 CPU 的内存请求,RowHammer 攻击与行激活的数量有关,因此,在发生 RowHammer 攻击时,内存控制器可以主动监视对 DRAM 行的激活次数、对地址进行重新映射,最终采取措施来保证数据的正确性。

1.1.3 访存分析

CPU 读写数据时,首先会从缓存中读取数据,缓存未命中,则会发送相应的读写请求到内存控制器中。当多个请求映射到同一缓存集或同一缓存行时,会引发冲突,从而导致缓存驱逐现象。要确保每次从内存中直接访问数据则需要绕过缓存。例如使用 clflush 指令、缓存驱逐和非时间指令等。

1.2 RowHammer 攻击所需条件

目前攻击能够成功执行通常需要以下几个条件:①将目标数据存放在可控区域的邻接行;②绕过缓存,直接访问内存;③清除行缓冲区,重复打开攻击行。

1.2.1 确定目标数据位置

RowHammer 攻击可以导致内存单元位翻转,但成功利用位翻转达到攻击目的较难,因为内存中无关位的翻转可能会导致内存崩溃。为了提高攻击成功的可能性,在攻击前,需要先将目标数据

放置到攻击者可控的内存行的邻接行中。目前主要通过以下几种方法达到目的:

(1)内存重复数据删除技术[6,21]。它旨在减少虚拟机上内存占用,现在也是 Windows 8.1/10 操作系统中的默认启用功能。它通过将物理页面的多个相同副本映射到单个共享副本,降低了内存的总占用量。攻击者创建一个已知的内存页面,通过控制物理内存布局将目标物理页面反向映射到攻击者控制下的内存页面[4],这使得攻击者能够以完全受控的方式在任意物理内存中引发位翻转。

(2)内存喷射技术(memory spraying)[2]。它是指通过大量分配内容可控内存,使特定位置填入预期的内容。利用它可以将大量页面放入内存中,其中一些页面会被放置在与攻击者控制的行的相邻行上,这是一种概率攻击。

(3)Victor 等提出了 Phys Feng Shui[8],它利用物理内存分配器,依次请求从大到小的物理连续块,以可预测的方式重用和划分内存,通过耗尽不同大小的可用内存块来驱动物理内存分配器,将目标数据放置在物理内存中特定的位置。

(4)内存路由技术(memory waylaying)[5]。它不使用 pagemap 接口而是利用预取侧通道来确定用户空间中虚拟地址对应的物理地址,利用 Linux 页面缓存(page cache)的替换算法,有效地从页面缓存中驱逐选定的页面。这些页面由于随时可以被收回,不计入总的内存利用率,因此,内存伏击技术不会导致系统内存耗尽,具有一定的隐蔽性。通过对页面连续驱逐,页面最终被放置在易受攻击的物理位置上。

(5)内存伏击技术(memory ambush)[18]。利用 Linux 的伙伴分配器和 mmap[22],重复调用 mmap 函数将同一用户文件映射到地址空间的不同部分。与 Phys Feng shui 不同,它请求固定大小的处于内核中内存块。

总结可知,内存喷射和 Phys Feng Shui 为了将内核空间的页面放置在攻击者控制的页面旁边,会造成系统内存耗尽问题;内存路由技术和伏击技术有着占用内存小、不会导致系统内存耗尽的优点。但前者建立在 page cache 基础上,通过耗尽页面缓存来影响目标页面的物理位置;后者使用伙伴分配器,即可达到将目标数据放置在易受攻击位置。

1.2.2　提高访存率

攻击的关键是重复激活行而不是重复发送对某一行请求。行激活的频率是导致位翻转的重要因素。对于给定的 DRAM 单元,它仅在一个方向上经历数据丢失:“1”到“0”或“0”到“1”。触发位翻转的核心在于,在单个刷新间隔内,尽可能提高攻击行的行激活率,降低缓存对内存行激活频率的影响,因此,攻击者需要了解绕过缓存的方法。

一般来讲,在 x86 和 ARM 中绕过缓存的方法并不相同。在 x86 中主要有 3 种方法:①利用 clflush 指令[1-2];②使用缓存驱逐清除缓存中的数据[3,23]。Gruss 等[3]通过对计算机中的驱逐策略进行探索,确定较优的驱逐策略。Tadesse 等利用英特尔的缓存分配技术(CAT),减少了缓存冲突所需的访问次数,通过设计导致缓存集冲突的数据来替代利用 clflush 清除缓存数据[24];③非时间指令。Qiao 等[17]提出了一种基于非时间指令(一段时间内仅使用一次,不具时间局部性)的 RowHammer 攻击。

上述 3 种方法在 ARM 中并不实用,原因在于移动平台中的刷新指令具有特权,使用缓存驱逐访问内存行的速度太慢,非时间指令在移动平台中不可用,内存重复数据删除技术主要布置在 Win8.1/10 上。ARM 中主要通过利用用户可访问的 DMA 缓冲区来绕过 CPU 缓存[10],或利用 GPU 的缓存策略来实现对主存的快速访问[11]。

1.2.3　地址映射分析

一般来讲,程序请求地址为虚拟地址,即 CPU 发送到内存管理单元(Memory management unit,MMU)的地址为虚拟地址,MMU 将虚拟地址映射到物理地址,物理地址需经由内存控制器映射成 DRAM 相应的地址位,最终发往 DRAM 进行数据读写。通常物理地址与 DRAM 中地址位的映射关系非公开,逻辑上相邻的行位置,并不意味着物理上相邻。确定 DRAM 中物理地址的映射关系,这对执行双面攻击至关重要。

由于 DRAM 芯片中每个 bank 都有一个行缓冲器(Row buffer)来缓存最近使用的行,因此,交替访问一个 bank 中的不同行要比访问不同 bank 的访问时间长。通过交替访问 2 个只有特定位不同的物理地址,便可以推断出这些物理地址中特

定位的映射关系。

DRAMA[24]是在 Intel 机器上使用的通用逆向工程工具,它通过 2 种方法对内存地址进行逆向分析,一种是使用机器探测内存总线,但探测机器比较昂贵,不具有通用性;另一种是枚举地址的所有可能组合在软件上自动运行,这种方法效率低,通常无法确定具体的 DRAM 地址映射。

Xiao 等[7]根据公开的信息发现 Intel 的地址映射方案是 XOR 方案,提出了基于图的位检测方法,可以在一两分钟内可靠地确定行位和 bank 位的异或方案。

百度安全团队设计的地址映射逆向工具 DRAMDig[25],利用已公开的信息辅助逆向,精心设计物理地址对,先根据时间差对行和列的地址位进行划分,分析 bank 的寻址函数,最后分析剩余位的对应关系。这种工具能够在几分钟内快速可靠地逆向得出 DRAM 地址映射关系。

1.2.4 RowHammer 攻击类型

2014 年,Kim 等[1]通过运行表 1 所示的程序,对大量的 DDR3 芯片进行研究,发现实验中约 85% 的芯片容易受到 RowHammer 攻击,且容易受到攻击的芯片自 2010 年之后开始产生,这主要是由于工艺升级和芯片密度的增加,使得访问存储数据时电荷更容易泄露到相邻的单元而产生干扰。2020 年,Kim 等[26]对不同类型的 DRAM 进行测试发现,随着尺寸的缩小和芯片内部更多的共享结构,较新的芯片中位翻转所需的激活次数在不断减少,同时攻击也影响了更多的行。

表 1 RowHammer 攻击代码

Table 1 RowHammer attack code

序号	代码
1	Code:mov (X), % eax // read values of address X and Y
2	mov (Y),% ebx
3	clflush(X) // evict data in the cache
4	clflush(Y)
5	jmp code //repeat above stage

在表 1 的程序中,每次数据访问时生成对 DRAM 的读取请求,首先分别从 DRAM 的地址 X 和 Y 读取数据(X,Y 地址属于同 bank 不同行),将数据取到上层的缓存或寄存器中,其次,2 个 clflush 指令将缓存中的数据进行清除,最后跳回第 1 个指令,以上操作可不断重复打开和关闭对应的内存行。

目前大部分的攻击主要是通过单面攻击、双面攻击和单位攻击这 3 种攻击方式达到目的。本文将反复激活实施攻击的行称为攻击行,目标数据所在的行称为目标行/受害行。在单个刷新周期内,随机选择多个地址交替激活,其中一行与目标行相邻的攻击称为单面 RowHammer 攻击;双面 RowHammer 攻击指访问位于目标行相邻的 2 行来达到攻击目的;单位攻击是指通过利用部分内存控制器倾向于行关闭的调度策略,反复激活一个内存行的攻击方式,这种攻击无需了解内存映射关系。图 3 展示了 3 种攻击的不同效果,其中蓝色代表攻击行,黄色代表受害行。图 3(a)单面攻击是指当目标数据位于 row5 时,通过交替激活 row4,row0,可导致 row5 的单元发生位翻转;图 3(b)双面攻击,通过连续激活 row5 和 row3 可导致 row4 的单元位更快发生翻转。图 3(c)单位攻击仅适用于内存控制器使用行关闭策略或自适应页策略[27],通过单独激活一行获得攻击效果。在未配置缓解方案的芯片中,DDR4 芯片中的单元比 DDR3 芯片上的更容易产生位翻转。双面攻击相对于单面攻击只需不到一半的激活次数即可翻转,但双面攻击需要花费更多时间获取内存地址映射关系。

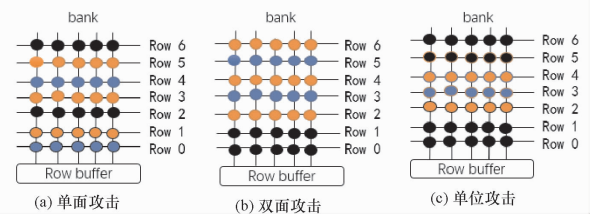

图 3 攻击类型

Fig. 3 Hammering type

2 RowHammer 攻击技术

目前攻击的手段是通过打破内存管理单元(MMU)的隔离,如进程内的隔离、进程间的隔离、内核和用户的隔离及虚拟机之间的隔离等。以下对攻击手段进行分类介绍,同时在表 2 中对常见的攻击方法进行总结。

表 2　常见的攻击方法

Table 2　Summary of mainstream attack methods

攻击分类	攻击方法	攻击所需条件/技术	效果	缺点
进程内隔离攻击	Bosman 等[4]	JavaScript 对象分配策略/易受攻击位置，内存重复数据删除技术	浏览器中获得任意内存的读写权限	操作系统中需配置内存重复数据删除技术
	Glitch[11]	GPU 缓存驱逐策略	远程攻击、浏览器任意读写权限	攻击原语复杂,需对 GPU 架构进行逆向
	SMASH[28]	切片着色策略、大页面	浏览器任意读写权限	禁用大页面、保护指针的完整性可阻止攻击
针对进程间隔离的攻击	Gruss 等[5]	行关闭策略或自适应策略、SGX 技术，Memory waylaying	特权升级、拒绝服务	系统需配置 SGX 技术、特殊的内存控制器调度算法
	Bhattacharya 等[29]	pagemap,标识缓存集、clflush	恢复密钥、root 权限	需要目标数据在攻击过程中持续存在于特定位置
	Throwhammer[12]	RDMA,memory massaging	升级特权	网络中需配置 RDMA,成功率低。不具有通用性
	RAMBleed[14]	内存伙伴分配器	获取机密数据	要求内存分配方式已知,在攻击过程中数据存在于内存中,攻击时间长
	Nethammer[15]	Intel CAT/uncached memory/clflush	远程攻击、拒绝服务	需要较高的网络吞吐量
针对内核和用户隔离的攻击	Drammer[8] Fraile et al[9]	ION 分配器	root 权限	可通过禁用 ION 和改进内存管理提高攻击难度
	CATTmew[18]	共享的缓存区 Memory ambush	root 权限	需要符合特定要求的内存
针对虚拟机之间的隔离攻击	Razavi 等[6]	内存重复数据删除技术	位翻转绕过身份验证等	操作系统中需配置内存重复数据删除技术
	Xiao 等[7]	VM 页表替换	绕过身份验证	适用于 Xen 半虚拟化的 VM
	Aga 等[23]	缓存驱逐算法/CAT	位翻转	了解缓存驱逐算法,适用于配置 CAT 的英特尔处理器
其他攻击	PThammer[20]	缓存驱逐、TLB 驱逐集、内存喷涂技术	内核特权	可通过缓存分区提高攻击难度
	TRRespass[30]	–	特权升级、破坏 RSA 公钥、操作码翻转	真实系统重现困难,不同芯片的有效攻击模式不等

2.1　针对进程内隔离的攻击

Bosman 等[4]提出了一种针对 Windows10 Microsoft Edge 浏览器的基于 JavaScript 的端到端攻击。将重复数据删除技术与 RowHammer 攻击相结合,通过利用 Edge 中的对齐方式获取它的代码指针,最终获得了在浏览器中任意内存的读写访问权限。

Glitch 攻击证明可以远程对安卓手机进行 RowHammer 攻击[11],它利用了个人电脑和移动设备的处理器中配置的 GPU 微体系结构,通过构造驱逐策略,绕过图形处理单元(GPU)的缓存策略来直接访问内存。利用 JavaScript 的特性,在进程虚拟空间的任何映射区域上获得读写权限。

针对配置了 TRR 的 DDR4 的系统,Ridder 等[28]提出了基于 JavaScript 的 SMASH 攻击。它利用切片着色策略(slice coloring)划分双面攻击对,利用缓存替换策略设置最佳访问模式,并通过将内存请求与内存控制器发送到内存的刷新命令同步,更精确地控制 TRR 可以识别的行。执行 SMASH 攻击只需要目标用户访问恶意网站,就可以控制用户的浏览器。

2.2 针对进程间隔离的攻击

攻击针对进程间的隔离可以窃取私人信息,破坏数据完整性。Gruss 等[5]提出了触发操作码翻转的单位攻击。攻击可在英特尔 SGX 安全区内执行,这种方式能绕过性能计数器的检测,同时由于 SGX 的安全检查特性,攻击可导致云服务器拒绝服务。

Bhattacharya 等[29]利用 Prime + Probe 攻击探测解密算法的执行,通过缓存冲突来确保解密进程的每个访问请求从内存中进行数据访问。对时序进行分析,识别密钥所在行,通过攻击产生的位翻转来产生错误的 RSA 签名,从而恢复密钥。

Tatar 等[12]提出了针对局域网计算机的 Throwhammer 攻击。他们指出,由于配置 RDMA(Remote direct memory access, RDMA)的设备,为了实现高性能,会将网络数据包直接传送到应用程序。攻击者仅发送网络数据包到启用了 RDMA 的网络计算机上,就可以直接触发内存行位翻转。

Nethammer 攻击不需要目标系统运行由攻击者控制的代码行。Nethammer 首次提出可远程攻击服务器,如 DNS 和 OCSP 服务器[15],通过向受害者发送特定的针对同一组内存位置的大量内存访问的网络包进行攻击。处理网络请求的软件如果使用未缓存的内存、非时间指令或刷新指令,那么攻击会导致位翻转,进而导致系统崩溃,造成持续的拒绝服务。

大部分攻击主要目的是获取访问权限,修改无法访问的内存内容。RAMBleed 攻击[14]指出位翻转不仅取决于位的翻转方向,而且和相邻位的值息息相关,因此,利用 Linux 伙伴分配器分配连续内存块,将目标数据放置在特定位置。通过检查位翻转情况,推断位翻转位置的值,获取攻击者禁止访问的内存区域中的敏感数据。

2.3 针对内核和用户隔离的攻击

Veen 等[8]提出了针对移动设备上的 Drammer 攻击。该攻击通过使用 Phys Feng Shui 技术控制物理内存的分布,将目标数据放置在攻击者选择的易受攻击的物理内存位置。Drammer 通过分配大量未缓存的,且物理上连续的内存来隔离易受攻击的行,并通过为页面表分配易受攻击的行进行双面攻击。利用攻击可破坏 Linux 进程的页表(Page table, PT)。该攻击已在 ARM v7/v8 64 位的设备上得到验证。攻击可在几分钟内完全控制受害者的手机。

Fraile 等[9]在真实的手机系统上通过对 Drammer 技术进行优化实现攻击,指出通过消耗内存块将目标数据放置在易受攻击位置的方法有时会不成功,在内存管理中可通过对内存分配进行中止并进行重试的方法,确保实现目标页表落入易受攻击的内存位置,提高成功的概率。

CATT[19]通过将用户空间和内核空间进行隔离来保护内核免受攻击。Cheng 等[18]指出操作系统中存在内核和用户共享的缓存区(double-owned buffers),这种缓存区的存在使 CATT 内核隔离失效,提出了 CATTmew 攻击。CATTmew 利用内存伏击技术将页表翻转,有效地破解了内核隔离防御,获得内核权限。

2.4 针对虚拟机之间的隔离攻击

Xiao 等[7]主要对 Xen 半虚拟化的 VM 进行攻击的研究。这种攻击可使来宾 VM 拥有对共享计算机上任意物理页面的读写访问权限,打破了 Xen 半虚拟化的内存隔离。此外,与文献[1]中分配大量内存、使用页表项填充,并进行随机攻击不同,此攻击利用 VM 的页表特性,提出了页表替换攻击,使得攻击引起的位翻转不会使系统崩溃。

Razavi 等[6]利用 Linux 中的内存重复数据删除技术,使攻击者能够以完全受控的方式在任意物理内存中引发位翻转。最终可以未经授权访问正在运行的受害 VM 的 OpenSSH,破坏公钥身份验证,从受信任的密钥中伪造签名攻击,甚至破坏共同托管的受害 VM 的更新系统。

2.5 其他攻击

大部分攻击主要是打破基于 MMU 的内存隔离,反复访问攻击行达到攻击目的。Zhang 等[20]指出内存访问时会从内存中获取页表条目(PTE),这相当于对 PTE 进行一次攻击,因此,提出了 PThammer 攻击。此攻击不通过对攻击行进行访问,而是利用缓存逐出和 TLB 逐出来达到对 PTE 的访问和位翻转。这种攻击可绕过 CATT 等对内存进行隔离的防御技术。

Frigo 等[30]提出使用 TRRespass 来识别现在系统上配置的目标行刷新(TRR)的信息,过程中不依赖内存控制器或 DRAM 芯片的任何实现细节。TRReapass 对 3 大 DRAM 制造商的部分模块进行分析,发现攻击者可通过识别 TRR 配置中的弱点建立有针对性的访问模式,进而在配备了 TRR 的系统上构建有效的攻击模块。

3　RowHammer 防御方法

Kim 等[1]不仅证明了 RowHammer 的存在而且提出了 7 种系统级缓解方案:①通过改进电路设计在芯片级解决这个问题;②增加 ECC 模块来纠正错误;③缩短刷新频率,如将刷新频率从 64 ms 缩短到 32 ms;④制造商提前识别易出错单元,重新映射到备用单元;⑤通过用户来处理可靠性问题;⑥标识易受攻击的行,刷新其邻近行;⑦概率刷新相邻行 PARA。但以上这些方法在实际配置过程中会存在工艺升级继续导致问题出现或者配置成本高、操作难等问题。

RowHammer 漏洞是由于内存单元密度大引起的电荷干扰导致的,从技术角度来讲短时间内很难通过升级解决问题。为了应对 RowHammer 攻击,研究者提出了许多方案来防御 RowHammer 攻击。由于硬件解决方案需要对 DRAM 模块或内存控制器等进行修改,无法应用于已有的 DRAM 模块,因此,目前已实施的缓解方案主要从软件层面上进行缓解,硬件方面的缓解方案是制造商在内存控制器或 DRAM 中加入 TRR 进行防御。图 4 中列举出了针对不同设备的首次攻击时间和已实施的缓解方案。本节主要从硬件和软件 2 个角度对防御技术进行概述,同时表 3 中对常见的防御方法进行了总结。

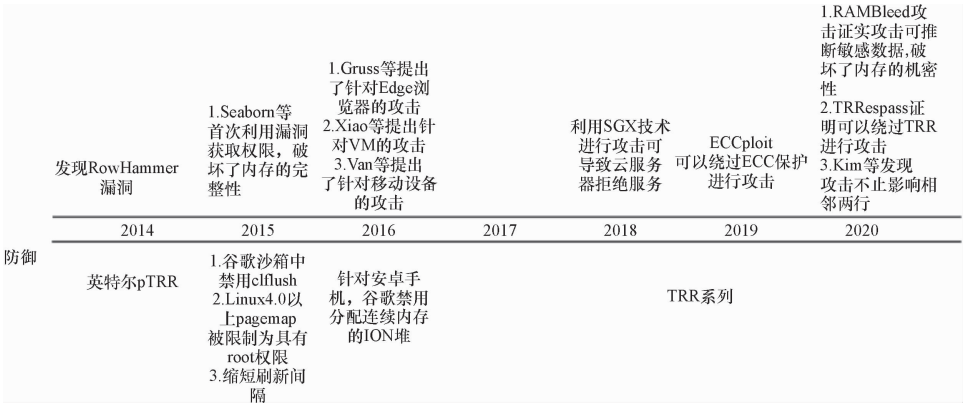

图 4　针对不同系统的首次攻击时间和已实施的防御方案

Fig. 4　Time to first attack against different systems and implemented defense plans

表 3　防御方法总结

Table 3　Summary of defense methods

缓解方法		防御措施	劣势
软件	用户级别禁用 clflush 等相关指令和 pagemap 接口[2,31]	限制虚拟到物理的映射访问,禁用缓存刷新等相关指令	可绕过进行攻击
	重复数据删除技术	禁用重复数据删除技术	内存利用率降低,性能降低
	ANVIL[32]	利用性能计数器监视最后一级缓存未命中情况来防御攻击	要求处理器具有性能计数器。需更改内核
	物理隔离页面[10,12,19,33-34]	需要额外的内存来隔离敏感数据	隔离成本高,可绕过进行攻击
	SoftTRR[35]	监视和跟踪包含页表的所有相邻行的内存访问	需提前获取地址映射信息
硬件	缩短刷新间隔[1,36]	如 64 ms 缩短为 32 ms	性能降低和能耗增大。可绕过进行攻击
	ECC 内存[1]	纠正部分位	有限的纠错能力,成本较高,可绕过进行攻击
	概率刷新[1,37-38]	打开和关闭一行时,按概率刷新相邻行	执行不必要刷新数与概率值相关。不能完全阻止攻击
	计数器[39-43]	行激活数量高于阈值时刷新相邻行	开销较大,组计数时易造成不必要刷新
	TRR 系列[30]	目标行的激活量达到阈值,则刷新相邻行	具体方案未公开。可绕过进行攻击
	动态偏斜树[44]	使用滑动窗口和哈希树来检测位翻转	哈希计算需要时间
	地址重映射[45]	内存控制器中进行两级地址映射	地址映射表较大

3.1 基于硬件的防御方法

制造商已经实施的硬件防御方案主要是缩短刷新间隔[36],配置目标行刷新(TRR)来保护DRAM免受攻击。基于硬件的防御方法主要从4个方面进行:①在每次激活行时,以低概率对其相邻行进行刷新;②对DRAM中行的激活次数进行检测,对达到激活阈值的行的相邻行进行刷新以达到防御效果;③访问数据时,通过事后检查的方式,降低内存数据发生位翻转带来的影响;④限制行激活频率,最大限度地降低内存行达到激活阈值的数量。

3.1.1 基于概率的防御措施

概率相邻行激活(Probabilistic adjacent row activation, PARA)[1],在每次打开或关闭一行时,它以低概率刷新其相邻行,这种方案开销较小。

ProHIT[39]对PARA进行改进,通过管理冷表和热表对受害行进行排序。当某一行成为受害行时首先插入冷表中,如果发现此行已经在冷表中,则控制器将其放入热表中,热表中的行根据访问次数进行优先级提升,拥有最高优先级的行在每个行刷新周期被刷新。

MRLoc[40]利用内存访问局部性来优化攻击产生的额外刷新问题。通过管理一个存储受害行地址循环队列,基于每行的访问历史动态计算加权概率。具有较高局部性的受害行拥有较短的队列命中距离,基于此调整每次激活的受害行地址,使用它们来优化每个队列命中时的刷新概率。性能比PARA要好,且降低了额外的刷新次数。

这种方法的理念很简单,配置时可选择在内存控制器或DRAM芯片中实现。如果在内存控制器中实现,内存控制器需要知道哪些行在物理上相邻,如果没有此信息,就无法有选择地刷新行。如果在DRAM芯片中实现,则DRAM芯片需添加刷新接口,对DRAM内部进行额外的刷新操作。

3.1.2 基于计数器的防御措施

基于计数器的防御方案主要从行激活的角度出发,因此,可以对攻击实施完备性的保护。CRA方案跟踪每行的激活次数[41]。当行激活次数达到阈值时,刷新与此行相邻的行。这种方法存在计数器存储空间大,检索时间长,且需要额外的操作管理CRA的缺点。

CAT[42-43]是基于计数器的自适应树,它固定计数器数量,以组为单位计算行的总激活数量。当组激活数量超过子阈值时,进一步将组划分为子组。因此,更频繁激活的组被拆分为较小的组,从而更可能以细粒度方式发现攻击的行。但是如果计数器用完,或者它不能将组划分为足够小的组,则必须同时刷新组内所有行。其刷新数量比PARA高出几倍,同时也存在最佳子阈值确定问题。

由于DRAM单个刷新周期内的激活次数受DRAM固有的时间参数限制,因此,时间窗口计数器TWiCe[37]通过定期修剪激活频率不够的行来减少计数器总数。这种方法与CBT相比,它需要更大的表。

CAT-TWO对CAT进行优化[38],它根据单个刷新周期中rank内的总激活数量要小于所有bank内的总激活数量的特点,以rank为单位配置计数器。同时增加根节点数量,使得树的叶节点对应于单个DRAM行,确保受害行总是在树的最后一级刷新。与CAT相比,它进一步降低了额外的激活次数。

硬件制造商目前已采用目标行刷新(TRR)来保护DRAM免受攻击。TRR不是固定的方案,具体TRR配置是由相应的制造商决定。它可以在内存控制器中实现,也可以在DRAM芯片中实现。由于TRR只能刷新有限数量的DRAM行,因此,文献[22]中提出了使用TRRespass利用多面攻击的方式识别TRR配置中的弱点,建立有针对性的访问模式,进而在配备了TRR的系统上构建有效的攻击模块。Nethammer[15]同样证明可绕过TRR进行攻击。

3.1.3 事后检查的防御措施

ECC内存可以检测和纠正位错误,因此,配备ECC可有效降低攻击成功的可能性。但ECC通常每64 bit纠正一位错,检测两位错。Kim等[1]证明RowHammer攻击可导致一行中多位翻转,且ECC内存配置成本较高,已被证明同样可受到RowHammer攻击[5,13]。

Vig等[44]提出了基于滑动窗口和哈希算法SHA-3的动态完整性树,以快速检测RowHammer攻击引起的位翻转。滑动窗口以较短的间隔监视对同一存储库的频繁访问,识别易受攻击的行,滑动窗口的大小限制了树的高度。每当内存数据发送请求给任何易受攻击的行时,对数据执行验证,从而能够快速检测到位翻转。过程中需递归计算哈希树,并与内存控制器存储的哈希根比较才能

确定是否发生位翻转。

Kim 等[45]提出了基于 DRAM 两级地址重新映射的新技术,有效地将错误分布到不同的行和列上。第 1 级在芯片级别重新映射地址,以使每个芯片的地址不同。第 2 层重新映射芯片内部的地址,按字节内部的每个位进行分配映射,从而允许 ECC 更有效地更正错误。

3.1.4　限制行激活率

苹果等制造商[36]通过提高刷新率,降低在单个刷新周期内整体的行激活总数,达到防御目的。通过 BIOS 更新以提高 DRAM 刷新速率的防御方法,仅增加了实施攻击的难度。文献[23]表明此方法并没有消除攻击,还影响了系统的吞吐量。

BlockHammer[46]主要通过使用 Bloom 过滤器来跟踪并限制行激活率,当它观察到某一行在给定时间间隔内的激活数量达到预定阈值时,通过将该行设定为潜在攻击行,并限制该行的进一步激活,来确保没有行被快速激活达到翻转阈值。它不需要对 DRAM 芯片进行修改,完全可以在内存控制器中实现。

3.2　基于软件的防御方法

制造商已经实施的软件缓解措施包括删除对页面映射的无特权访问[2,31],禁用 clflush 指令[2],默认情况下禁用页面重复删除[4,6]。禁用 clflush 并不能阻止 RowHammer 攻击。同样地,禁止在用户空间中使用 pagemap 接口[2],默认情况下禁用页面重复数据删除只能防止利用此功能的攻击,并不能阻止所有的 RowHammer 攻击。目前攻击者已证明可以绕过这几种缓解措施实施攻击[5,7,32]。其他基于软件的防御方案主要分为缓解和检测 2 个方面。具体是通过对内存进行隔离、在处理器中监视缓存未命中情况或通过监视内存分配情况达到防御目的。

3.2.1　缓解技术

CATT[19]通过修改系统的内存分配器或缓存管理策略,以物理方式将内核和用户域隔开来抵御 RowHammer 攻击。

Veen 等[10]为缓解在 ARM 上的 Drammer 攻击,提出了 GuardION,它主要是通过强制隔离 DMA 缓冲区来防止攻击。目前谷歌禁用了负责分配连续内存的 ION 堆(kmalloc),并通过隔离内核内存和用户内存来进一步提高移动设备的安全性。

ZebRAM[33]是针对 VM 的方法,将物理内存交错拆分成安全行和不安全行。针对安全行的攻击,只会导致不安全行无效翻转,因为访问不安全行数据时,默认执行完整性检查和错误更正。非特权用户无法访问不安全行。

以上 3 种防御措施可用于防止页表受到攻击。

石培涛等[34]针对 VM 的攻击提出了 RDXA 防御机制。它通过在虚拟机监视器层实现的内存分配机制来保证任意 2 个虚拟设备对应的物理内存不会分配到相同 bank 的相邻行上,从而防止不同区域的相互影响。

3.2.2　检测技术

ANVIL 技术[32]通过使用 CPU 性能计数器监视缓存未命中率,并根据缓存未命中情况来判断特定行的访问行为,有选择地执行刷新操作来缓解攻击。这种方案需要系统中存在性能计数器,而且容易出现误报和漏报问题。文献[10]中指出可通过 DMA 技术绕过 ANVIL 中对最后一级缓存的未命中判断。

由于 RowHammer 需要分配大量内存来保证在特定的位置放置目标数据,故存在内存耗尽的情况,如内存喷射等方法很可能耗尽整个内存,因此,可通过优化内存分配器来防止内存耗尽等现象的发生,这将导致攻击者无法将目标数据放置到特定的内存位置[3,8]。

Lee 等[47]提出了一种使用静态分析预先检测攻击的技术,研究使用 IDA Python 对 RowHammer 攻击进行静态分析和检测技术。指出所有攻击文件中使用的通用操作码和接口包括 mov, clflush, mmap 等相关指令或函数。此方法通过提前对平台上的操作码等进行分析,可以准确检测 x86 – 64 指令中执行的所有 RowHammer 攻击文件。

SoftTRR[35]保护 x86 上所有针对页表的 RowHammer 攻击,如内存喷射、CATTmew 和 TRRespass 等涉及页表的攻击。它利用虚拟内存子系统监视和跟踪所有相邻行的内存访问,在检测到可疑的攻击行为时刷新页表占用的行。此防御方法可作为内核附加模块,无需对内核进行修改。

4　RowHammer 攻击分析技术

目前的技术多停留在对 RowHammer 攻击和防御上,但开发可靠的检测和分析工具至关重要,

它有助于开发更好的可靠芯片,同时也可以主动预防攻击的发生。

4.1 辅助分析工具

Tatar 等[48]开发了一个开源的可用于测试、分析和模拟 RowHammer 漏洞的模拟器,即 Hammertime。这个模拟器包含了大量的 DRAM 芯片的位翻转模式,可允许研究人员在软件中模拟硬件位翻转。

Lou 等[49]根据攻击的来源、意图和方法提出一个用于分析 RowHammer 攻击的统一参考框架,提出表达式 RH 攻击,用于分析现有攻击的组成和相应的攻击对策。通过分析现有的攻击提出了在造成伤害之前进行主动预防。

Hong 等[16]提出了漏洞分析框架,该框架可以翻转给定模型的参数中的每一位并测量验证集上的错误分类率。此框架还允许通过检查各种因素的影响来表征漏洞的能力:位位置、位翻转方向、参数符号、层宽度、激活函数、规范化和模型体系结构。主要发现包括漏洞的诱发原因、DNN 参数设置对发现漏洞的正负影响性,由此指出可以通过大幅增加或减小模型参数的值来轻易造成严重的不加区分的损害。

由于现实环境中验证攻击需要有缺陷的芯片,有缺陷的 DRAM 单元位置因芯片而异,且重现攻击的成本较高。Yim[50]提出了一种识别、验证和评估 RowHammer 攻击状态的方法,即 RowHammer 攻击注入(RowHammer attack injection, RAI)。RAI 通过对 RowHammer 基本操作进行建模,并使用软件机制将错误注入到目标 DRAM 行的逻辑相邻行的某些单元中,因此,RAI 可以验证各种目标状态,不需要有缺陷的芯片。

4.2 检测工具

Cojocar 等[51]指出以往针对 DRAM 的测试并未达到最坏情况下的测试条件,提出了一种端到端的方法来确定云服务器是否容易受到攻击,即构造了一个新的指令序列,可生成对 DRAM 的最高速率的行激活命令,且通过此方法云供应商可以为 DRAM 构建最坏的测试条件。

5 未来研究方向

DRAM 作为最常用的主存储器,它的安全必将受到越来越多的关注。针对 RowHammer 攻击与缓解的未来发展方向,还有很多工作值得深入研究。具体来讲,可以分为以下 7 个方面:

(1)提高目标数据放置到易受攻击位置的有效性和准确率。对于未知的 DRAM 模块,如何定位易受攻击行,在较短时间内通过技术将目标数据放到易受攻击的行而不引起内存崩溃可作为未来的一个研究热点。

(2)提高攻击的准确性和有效性。攻击存在对修改位的不确定,可能创建可修正错误,导致攻击无效;攻击也可能造成内存数据损坏,从而导致内存崩溃。内存地址的映射关系可通过反向工程获得,但 DRAM 制造商通常会配备大约 2% 的备用行[14],这些备用行可通过物理方式替换故障行,这意味着物理上相邻的行,由于备用行的存在,导致攻击难度升级。未来提高攻击的准确度和有效性依然是研究的热点。

(3)升级现有技术,减弱其对 RowHammer 攻击成功率的影响。目前攻击涉及的技术包括内存分配器、SGX 技术和 CAT 技术等。这些技术暴露了其设计之初留下的各种弊端。对这些技术进行规范和升级,减弱其在 RowHammer 攻击中的辅助作用,也是未来的一个重要研究方向。在部署新技术时,也须考虑其对 RowHammer 攻击的影响。

(4)在保证效率的同时保障内存数据的完整性和机密性。目前为止没有任何防御方案同时兼顾性能和功耗友好,已有的解决方案会产生大量的功耗和性能开销,且会对系统的吞吐量造成影响。文献[26]中指出 RowHammer 漏洞不仅影响直接相邻的存储单元,而且影响相邻行旁边的存储单元。如何保证内存数据的整体安全和制定更有效同时又兼顾性能和功耗友好的防御方案,在未来的研究中依然重要。

(5)针对不同架构的防御。配置 FPGA、GPU 等混合的系统可能比单独的 CPU 更容易受到硬件攻击,未来针对配置在系统中的很多设备的安全防御也是下一步的研究热点。

(6)攻击检测分析。由于针对真实硬件的攻击通常需要分析有缺陷的芯片,进而进行攻击和防御研究。因为芯片差异和复现攻击的成本较高,如何深入理解芯片结构,开发出高效的检测工具和模拟漏洞的模拟器也是很好的研究热点。

(7)新兴的内存系统的安全。目前 NAND 中也被指出可以执行类似的攻击[52],因此,新兴的内

存架构中的数据安全也需纳入研究之中。

6 结　语

内存是存储计算机运行数据的重要载体,面向内存的攻击和防御也将继续成为硬件安全领域的研究热点。本文整理和总结了 RowHammer 攻击和防御方法,进行了系统的分类。总结发现,大部分攻击建立在分析和利用现有的系统配置,且对内存映射进行逆向分析的基础上,对于攻击结果具有不确定性,即成功的概率较低。大部分防御技术在真实系统上配置非常困难,很多研究人员提出的防御方法,通常处于一个理想状态,且仅从其中一个方面对漏洞产生的错误进行缓解。相应的 RowHammer 分析工具也不完善。因此,在未来的工作中,可以针对不同的系统配置,对 DRAM 芯片、内存控制器、操作系统和处理器等进行综合研究,并将其作为解决攻击的主要研究方向。

参考文献:

[1] Kim Y, Daly R, Kim J, et al. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors[C]//ACM/IEEE 41st International Symposium on Computer Architecture (ISCA). Piscataway: IEEE, 2014: 361-372.

[2] Seaborn M, Flake H. Exploiting the DRAM RowHammer bug to gain kernel privileges[EB/OL]. (2015-03-09) http://googleprojectzero. blogspot. com/2015/03/exploiting-dram-rowhammer-bug-to-gain. html 2015.

[3] Gruss D, Maurice C, Mangard S. RowHammer. Js: A remote software-induced fault attack in JavaScript[C]//In Detection of Intrusions and Malware Vulnerability Assessment (DIMVA). Berlin: Springer,2016:300-321.

[4] Bosman E, Razavi K, Bos H, et al. Dedup est machina: Memory deduplication as an advanced exploitation vector[C]// IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE, 2016:987-1004.

[5] Gruss D, Lipp M, Schwarz M, et al. Another flip in the wall of Rowhammer defenses[C]//IEEE Symposium on Security and Privacy (SP). New York: IEEE, 2018:245-261.

[6] Razavi K, Gras B, Bosman E, et al. Flip feng shui: Hammering a needle in the software stack[C]//Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16). Berkeley: USENIX Association,2016:1-18.

[7] Xiao Y, Zhang X K, Zhang Y Q, et al. One bit flips, one cloud flops: Cross-VM RowHammer attacks and privilege escalation[C]//Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16). Berkeley: USENIX Association, 2016:19-35.

[8] Veen V, Fratantonio Y, Lindorfer M, et al. Drammer: Deterministic RowHammer attacks on mobile platforms[C]//The 2016 ACM SIGSAC Conference. New York:ACM, 2016:1675-1689.

[9] Fraile L P, Fournaris A P, Koufopavlou O. Revisiting RowHammer attacks in embedded systems[C]//14th International Conference on Design & Technology of Integrated Systems In Nanoscale Era (DTIS). Piscataway: IEEE, 2019:1-6.

[10] Veen V D, Victor, Lindorfer, et al. GuardION: Practical mitigation of DMA-based RowHammer attacks on ARM[C]//Giuffrida C, Bardin S, Blanc G. In Detection of Intrusions and Malware Vulnerability Assessment (DIMVA). Berlin: Springer,2018:92-113.

[11] Frigo P, Giuffrida C, Bos H, et al. Grand pwning unit: Accelerating microarchitectural attacks with the GPU[C]//2018 IEEE Symposium on Security and Privacy (SP). Piscataway: IEEE, 2018:195-210.

[12] Tatar A, Konoth R K, Athanasopoulos E, et al. Throwhammer: RowHammer attacks over the network and defenses[C]// Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference (USENIX ATC '18). Berkeley: USENIX Association, 2018:213-225.

[13] Cojocar L, Razavi K, Giuffrida C, et al. Exploiting correcting codes: On the effectiveness of ECC memory against RowHammer attacks[C]//2019 IEEE Symposium on Security and Privacy (SP). Los Alamitos: IEEE Computer Society, 2019:55-71.

[14] Kwong A, Genkin D, Gruss D, et al. RAMBleed: Reading bits in memory without accessing them[C]//2020 IEEE Symposium on Security and Privacy (SP). Los Alamitos: IEEE Computer Society, 2020:695-711.

[15] Lipp M, Schwarz M, Raab L, et al. Nethammer: Inducing RowHammer faults through network requests[C]//2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW). Los Alamitos: IEEE Computer Society,2020:710-719.

[16] Hong S, Frigo P, Kaya Y, et al. Terminal brain damage: Exposing the graceless degradation in deep neural networks under hardware fault attacks[C]//Proceedings of the 28th USENIX Conference on Security Symposium (SEC'19). Berkeley: USENIX Association, 2019:497-514.

[17] Qiao R, Seaborn M. A new approach for RowHammer attacks[C]//The 9th IEEE International Symposium on Hardware Oriented Security and Trust (HOST). Piscataway: IEEE, 2016: 161-166.

[18] Cheng Y, Zhi Z, Nepal S, et al. CATTmew: Defeating software-only physical kernel isolation[J]. IEEE Transactions on Dependable and Secure Computing,2019,18(4): 1989-2004.

[19] Brasser F, Davi L, Gens D, et al. CAn't touch this: Software-only mitigation against RowHammer attacks targeting kernel memory[C]//In 26th USENIX Security Symposium (USENIX Security 2017). Berkeley: USENIX Association, 2017:117-130.

[20] Zhang Z, Cheng Y, Liu V, et al. PThammer: Cross-user-kernel-boundary RowHammer through implicit accesses[C]// 2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Los Alamitos: IEEE Computer Society,2020:28-41.

[21] 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). Piscataway: IEEE, 2013: 1-12.

[22] Gorman M. Understanding the linux virtual memory manager[M]. Upper Saddle River: Prentice Hall PTR,2007.

[23] Aga M T, Aweke Z B, Austin T. When good protections go bad: Exploiting anti-DoS measures to accelerate rowhammer attacks[C]//2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST). Los Alamitos: IEEE Computer Society,2017: 8-13.

[24] Pessl P, Gruss D, Maurice C, et al. DRAMA: Exploiting dram addressing for cross-cpu attacks[C]//Proceedings of the 25th USENIX Conference on Security Symposium (SEC'16). Berkeley: USENIX Association,2016:565-581.

[25] Wang M H, Zhang Z, Cheng Y Q, et al. DRAMDig: A knowledge-assisted tool to uncover DRAM address mapping[C]// Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference (DAC'20). Piscataway: IEEE, 2020:1-6.

[26] Kim J S, Patel M, Giray A, et al. Revisiting RowHammer: An experimental analysis of modern DRAM devices and mitigation techniques[C]//ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE,2020:638-651.

[27] Kaseridis D, Stuecheli J, John L K. Minimalist open-page: A DRAM page-mode scheduling policy for the many-core era [C]//44th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO). Los Alamitos: IEEE Computer Society, 2011:24-35.

[28] Ridder F D, Frigo P, Vannacci E, et al. SMASH: Synchronized many-sided RowHammer attacks from JavaScript[C]//30th USENIX Security Symposium(SEC'21). Berkeley: USENIX Association,2021:1001-1018.

[29] Bhattacharya S, Mukhopadhyay D. Curious case of RowHammer: Flipping secret exponent bits using timing analysis[C]// Gierlichs B, Poschmann A Y. International Conference on Cryptographic Hardware and Embedded Systems. Berlin: Springer-Verlag Berlin,2016:602-624.

[30] Frigo P, Vannacc E, Hassan H, et al. TRRespass: Exploiting the many sides of target row refresh[C]//IEEE Symposium on Security and Privacy (SP). Los Alamitos: IEEE Computer Society, 2020:747-762.

[31] Shutemov K A, Pagemap: Do not leak physical addresses to non-privileged userspace[EB/OL]. (2015-03-09) https://git.kernel.org/cgit/Linux/kernel/git/torvalds/Linux.git/commit/?id=ab676b7d6fbf4b294bf198fb27ade5b0e865c7ce.

[32] Aweke Z B, Yitbarek S F, Rui Q, et al. ANVIL: Software-based protection against next-generation RowHammer attacks [J]. ACM SIGOPS Operating Systems Review, 2016, 50(2):743-755.

[33] Konoth R K, Oliverio M, Tatar A, et al. ZebRAM: Comprehensive and compatible software protection against RowHammer attacks[C]//Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation (OSDI'18). Berkeley: USENIX Association, 2018:697-710.

[34] 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(4):5-16.

[35] Zhang Z, Cheng Y Q, Wang M H, et al. SoftTRR: Protect page tables against RowHammer attack using software-only target row refresh[J/OL]. 2021: ArXiv e-prints ArXiv:2102.10269.

[36] Apple Product Security. About the security content of Mac EFI security update 2015-001[EB/OL]. [2017-01-23]. https://support.apple.com/en-us/HT204934.

[37] Lee E, Lee S, Suh G E, et al. TWiCe: Time window counter based row refresh to prevent row-hammering[J]. IEEE Computer Architecture Letters, 2018, 17(1):96-99.

[38] Kang I, Lee E, Ahn J H. CAT-TWO: Counter-based adaptive tree, time window optimized for DRAM row-hammer prevention[J]. IEEE Access, 2020, 8:17366-17377.

[39] Son M, Park H, Ahn J, et al. Making DRAM stronger against row hammering[C]//54th ACM/EDAC/IEEE Design Automation Conference (DAC). Piscataway: IEEE, 2017:1-6.

[40] You J M, Yang J. MRLoc: Mitigating row-hammering based on memory Locality[C]//56th ACM/IEEE Design Automation Conference (DAC). New York: ACM, 2019:1-6.

[41] Kim D, Nair P J, Qureshi M K. Architectural support for mitigating row hammering in DRAM memories[J]. IEEE Computer Architecture Letters, 2015,14(1):9-12.

[42] Seyedzadeh S M, Jones A K, Melhem R. Counter-based tree structure for row hammering mitigation in DRAM[J]. IEEE Computer Architecture Letters, 2017,16(1):18-21.

[43] Seyedzadeh S M, Jones A K, Melhem R. Mitigating wordline crosstalk using adaptive trees of counters[C]//2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA). Piscataway: IEEE,2018: 612-623.

[44] Vig S, Bhattacharya S, Mukhopadhyay D, et al. Rapid detection of RowHammer attacks using dynamic skewed hash tree [C]// Proceedings of the 7th International Workshop on Hardware and Architectural Support for Security and Privacy (HASP'18). New York: ACM, 2018:1-8.

[45] Kim M, Jungwoo C, Kim H, et al. An effective DRAM address remapping for mitigating RowHammer errors[J]. IEEE Transactions on Computer, 2019, 68(10):1428-1441.

[46] Yaglikci A G, Patel M, Jeremie S, et al. BlockHammer:Preventing RowHammer at low cost by blacklisting rapidly-accessed DRAM rows[C]//27th IEEE International Symposium on High Performance Computer Architecture, HPCA 2021. Los Alamitos:IEEE Computer Society, 2021:345-358.

[47] Lee M, Kwak J. Detection technique of software-induced RowHammer attacks[J]. Computers Materials & Continua, 2021, 67(1):349-367

[48] Tatar A, Giuffrida C, Bos H, et al. Defeating software mitigations against RowHammer: A surgical precision hammer[C]// 21st International Symposium on Research in Attacks, Intrusions and Defenses (RAID). Berlin: Springer, 2018:47-66.

[49] Lou X, Zhang F, Chua Z L, et al. Understanding RowHammer attacks through the lens of a unified reference framework[J/OL]. 2019: ArXiv e-prints ArXiv:1901.03538v1.

[50] Yim K S. The RowHammer attack injection methodology[C]// IEEE 35th Symposium on Reliable Distributed Systems (SRDS). Piscataway: IEEE,2016:1-10.

[51] Cojocar L, Kim J, Patel M, et al. Are we susceptible to Rowhammer? An end-to-end methodology for cloud providers[C]// IEEE Symposium on Security and Privacy (SP). Los Alamitos: IEEE Computer Society, 2020:712-728.

[52] Anil K, Nikolas I, Matthias N, et al. From random block corruption to privilege escalation: a filesystem attack vector for rowhammer-like attacks[C]//Proceedings of the 11th USENIX Conference on Offensive Technologies (WOOT'17). Berkeley: USENIX Association, 2017:4-12.

【责任编辑: 周　全】