HammerScope：利用行冲突攻击观察 DRAM 功耗

Yaakov Cohen*
内盖夫本-古里安大学和英特尔公司
以色列贝尔谢巴
yaakoc@post.bgu.ac.il（此邮件地址无需翻译）

Kevin Sam Tharayil*
美国佐治亚州亚特兰大市 佐治亚理工学院
kevinsam@佐治亚理工学院.edu

Arie Haenel*
耶路撒冷理工学院和英特尔公司
以色列耶路撒冷
arie.haenel@jct.ac.il（此邮件地址无需翻译，直接照搬即可。）

Daniel Genkin
佐治亚理工学院
美国佐治亚州亚特兰大市
genkin@gatech.edu（此邮件地址无需翻译）

Angelos D. Keromytis
美国佐治亚州亚特兰大市 佐治亚理工学院
angelos@gatech.edu

Yossi Oren
内盖夫本-古里安大学和英特尔公司
以色列贝尔谢巴
yos@bgu.ac.il（此邮件地址无需翻译，直接保留原样即可。）

Yuval Yarom
澳大利亚阿德莱德大学 阿德莱德，澳大利亚
yval@cs.adelaide.edu.au（此邮件地址无需翻译）

摘要

存储单元尺寸的不断缩小提高了存储密度并降低了功耗，但也影响了其可靠性。行攻击（Rowhammer 攻击）正是利用这种降低的可靠性，在不直接访问这些位的情况下，诱导内存中的位翻转。大多数行攻击针对的是软件完整性，但最近的一些攻击展示了其用于破坏机密性的用途。

延续这一趋势，在本文中我们观察到，行攻击（Rowhammer 攻击）与内存瞬时功耗存在很强的相关性。我们利用这一观察结果设计了 HammerScope，这是一种基于行攻击的攻击技术，用于测量内存单元的功耗。由于功耗与内存的活跃程度相关，HammerScope 使攻击者能够推断出内存的活动情况。

为了展示 HammerScope 的攻击能力，我们用它来实施三次信息泄露攻击。首先，我们证明 HammerScope 可用于破解内核地址空间布局随机化（KASLR）。我们的第二次攻击利用内存活动作为 Spectre 攻击的隐蔽通道，从而从操作系统内核中泄露信息。最后，我们展示了 HammerScope 在执行网站指纹识别方面的应用，从而侵犯用户隐私。我们的工作表明，找到系统性的解决方案来应对行锤击攻击至关重要。

CCS 概念

· 硬件 → 动态内存； · 安全与隐私 → 侧信道分析及防护措施。

关键词

侧信道攻击，行击打攻击

ACM 参考格式：

雅科夫·科恩、凯文·萨姆·塔拉伊尔、阿里·海内尔、丹尼尔·根金、安杰洛斯·D·克罗米蒂斯、约西·奥伦和尤瓦尔·亚龙。2022 年。《HammerScope：利用行冲突观察 DRAM 功耗》。载于 2022 年 ACM SIGSAC 计算机与通信安全会议（CCS '22）会议录，2022 年 11 月 7 日至 11 日，美国加利福尼亚州洛杉矶。ACM 出版，纽约，美国，共 15 页。https://doi.org/10.1145/3548606.3560688

1 Introduction

“行攻击”是一种硬件漏洞，几乎影响所有配备动态随机存取存储器（DRAM）的现代系统。从宏观层面来看，攻击者通过执行某些精心策划的内存访问模式，能够在不直接写入这些位的情况下翻转内存中的位。从安全角度来看，“行攻击”通常被视为一种（受限的）写入原语，攻击者能够修改他们无权访问的内存内容。

自 2014 年首次公开披露 [30] 以及 2015 年引入安全研究领域 [50] 以来，行锤击（Rowhammer）故障攻击一直是学术界关注的焦点。特别是行锤击能够可靠地跨越安全边界翻转位，这一特性已被用于沙箱逃逸 [24, 50]、操作系统和虚拟机管理程序的权限提升攻击 [22, 24, 47, 50, 56, 61, 65]、拒绝服务攻击 [22, 26]，甚至用于加密协议中的故障注入 [4]。最近，行锤击攻击已被证实能够突破基于硬件的行锤击防护措施，例如定向行刷新 [17] 和纠错码（ECC-RAM）[10]。

虽然这些攻击手段中的每一种从安全角度来看都可能造成危害，但一个共同的趋势依然存在。在几乎所有基于“行攻击”的漏洞利用中，攻击者都在积极尝试通过翻转那些由于软件隔离机制而无法直接访问的位来破坏系统的数据完整性。而对于其他方面的情况，人们了解得则要少得多。

*All student authors contributed equally to this paper

CCS ’22, November 7–11, 2022, Los Angeles, CA, USA © 2022 Association for Computing Machinery. ACM ISBN 978-1-4503-9450-5/22/11. . . $15. 00 https://doi. org/10. 1145/3548606.3560688

“行攻击”（Rowhammer）的安全影响，例如数据保密性和真实性。

RAMBleed [34] 首次表明，行攻击（Rowhammer）的影响不仅限于完整性，还发现数据相关的位翻转可用于读取内存单元的内容，而非仅仅翻转其中的位。最近，SpecHammer [54] 证明行攻击可增强 Spectre 攻击，再次导致跨安全边界的敏感数据泄露。鉴于数据保密性成为行攻击研究的新领域，在本文中我们提出以下问题：

行击锤攻击能否用于其他保密应用？攻击者要实施此类攻击需要具备哪些条件？

它们会导致信息泄露吗？

1.1 我们的贡献

我们提出了一种名为 HammerScope 的新技术，该技术利用行攻击（Rowhammer）来测量内存模块的功耗。

变化的阈值。要利用行冲突攻击翻转特定的受害位，攻击者必须在短时间内快速激活附近的攻击行若干次。我们研究了成功利用行冲突攻击翻转位所需的激活阈值，发现其与被攻击内存模块的功耗相关。具体而言，功耗越高，翻转一位所需的行激活次数就越多。

构建 HammerScope。基于这一观察结果，我们构建了 HammerScope，这是一种仅使用无特权软件即可测量直接内联内存模块（DIMM）功耗的侧信道攻击。从高层次来看，HammerScope 持续对目标位进行行锤击，记录成功翻转该位所需的激活次数。由于激活阈值与 DIMM 的功耗相关，HammerScope 可以追踪其随时间变化的功耗情况。因此，HammerScope 实质上将行锤击转化为一种工具，使攻击者能够通过无特权软件监测 DIMM 的功耗，而无需任何额外设备。

对 HammerScope 进行基准测试和分析。我们在六种系统上对 HammerScope 进行基准测试，结果表明它能够追踪 DDR3 和 DDR4 内存的 DIMM 功耗。令人意外的是，我们发现将现代 DDR4 模块的行刷新周期延长一倍，HammerScope 的采样率也会相应翻倍，从而能够更精确地测量功耗。最后，我们对 HammerScope 的工作原理进行了假设，并在寄生电容模型下提出了几种可能的解释。

基于“行攻击”的功耗分析攻击的不断涌现。作为最后的贡献，我们表明“行攻击”对隐私的影响不仅限于故障攻击。我们展示了如何通过记录双列直插式内存模块（DIMM）的功耗来实现精确的指令定时、消除内核地址空间布局随机化（KASLR）的随机性、实施基于功耗分析的“幽灵”（Spectre）攻击，甚至用于网站指纹识别。我们通过 HammerScope 证明了所有这些用例都是可行的，并表明 HammerScope 为仅通过软件手段实施功耗分析攻击提供了一种途径。

贡献总结。总之，在本文中，我们做出了以下贡献：

我们对不同电源条件下行冲撞故障攻击的行为进行了形式化描述和特征分析，并表明动态随机存取存储器（DRAM）对行冲撞位翻转的易感性与其功耗之间存在明显的关联（第 4.1 至 4.4 节）。

我们介绍了一种新的攻击技术，称为“HammerScope”，并证明它可以应用于来自多个供应商的多台台式机和笔记本电脑的 DDR3 和 DDR4 内存（第 4.5 节）。

我们应用 HammerScope 来实施一系列端到端攻击，这些攻击能够破坏内核地址空间布局随机化（KASLR）、从内核泄漏内存以及进行网站指纹识别（第 5 节）。

1.2 漏洞利用工具的可用性与负责任的披露

我们将把与该项目相关的所有成果作为开源发布，包括同步行攻击的代码、基于反馈和基于阈值的 HammerScope 攻击方法的实现，以及针对 KASLR 的端到端攻击代码。此外，我们还将把通过 HammerScope 和 RAPL 收集的网站指纹追踪数据集，以及用于分类的机器学习流程发布到我们的网站上。部分成果已可在 https://github.com/hammerscope/artifacts 所在的代码库中获取。

我们向英特尔、AMD、谷歌和惠普的相关联系人提供了本文的早期版本。我们还与包含来自各 DRAM 制造商及其他硬件供应商代表的 JEDEC JC-42 委员会分享了我们的论文。

2 Background

在本节中，我们将解释一些与 HammerScope 相关的关键概念以及之前的相关研究成果。

2.1 动态随机存取存储器

动态随机存取存储器（DRAM）被用作大多数台式机、服务器和移动设备的主要易失性内存。DRAM 中的基本存储单元是 DRAM 单元，它使用电容器存储一位数据。

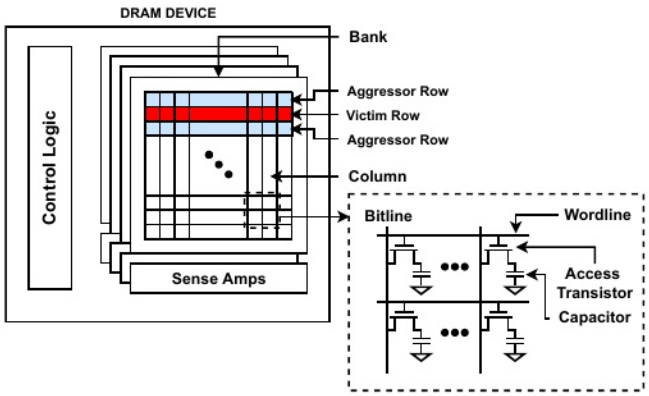


图 1：动态随机存取存储器（DRAM）芯片的结构。取自 [3]

如图 1 所示，这些电池通常排列成行，每行由多个并联的电池组成。每个电池都与

通过访问晶体管 [28] 的垂直位线。这些位线中的每一条都由多行 DRAM 单元共享。访问晶体管本身通过水平排列且由多列共享的字线来控制。

从动态随机存取存储器（DRAM）中读取数据。中央处理器（CPU）无法直接访问存储在电容器中的数据。相反，它通过 DRAM 上的一个称为行缓冲器的元件与存储单元进行交互，使用三步协议：要从 DRAM 单元读取数据，CPU 的 DRAM 控制器首先发出激活命令。此命令切换行的字线，激活该行中的所有访问晶体管，并将存储单元的存储电容器连接到位线。这将该行中包含单元的电容器的内容转移到行缓冲器，在那里使用感测放大器电路将其转换为离散的数字值。接下来，DRAM 控制器发出读取命令，将数据从感测放大器移动到 DRAM 总线。最后，DRAM 控制器发出预充电命令，将行缓冲器中的值恢复到 DRAM 单元中。

刷新命令。为防止电容因长时间放电而导致数据丢失，动态随机存取存储器（DRAM）芯片必须定期刷新。在刷新操作期间，DRAM 将单行数据读入行缓冲区，然后立即写回存储单元。中央处理器（CPU）以及在某些情况下 DRAM 控制器本身，每隔几微秒就会执行一次刷新操作，按照固定的模式遍历所有行。台式机级别的 DRAM 芯片的标称刷新周期为 64 毫秒，而典型的 DRAM 芯片有 8192 行，这意味着大约每 8 微秒就会安排一次行刷新操作。

电荷泄漏。向 DRAM 单元写入值会将单元电容中存储的电荷设置为固定值，要么为 DRAM 电源电压 V_{DD}or，要么为 0。为便于理解，以下讨论均假设写入 $V_{DD}$to 单元的情况。该电荷会随时间衰减，这是由于单元电容的固有泄漏所致。如果电荷衰减到距离 $V_{DD}/2$ 一定范围内，感测放大器将无法正确恢复 DRAM 中最初存储的值，从而可能导致位翻转。虽然固有泄漏只是由时间的推移引起，但还有一种泄漏类型，称为耦合泄漏，它发生在目标单元的相邻行被激活时。这种耦合泄漏是行锤击效应的基础。

2.2 行冲突攻击

行冲突攻击是一种针对动态随机存取存储器（DRAM）单元的故障攻击，最初由 Kim 等人[30]提出，随后 Jiang 等人[28]对其进行了进一步研究。行冲突攻击基于这样一个观察：在 DRAM 中激活一行位会因耦合效应而耗尽相邻行的电荷。因此，恶意攻击者若以可控方式访问 DRAM，使某一行在单个刷新周期内多次激活和失活，就能导致相邻行中的位发生翻转。我们将易受行冲突攻击影响的位称为“易翻转位”，将用于翻转目标位而被激活的行称为“攻击行”。行冲突攻击方法。典型的行冲突攻击包含几个步骤。攻击者首先逆向工程其地址空间中的虚拟地址到物理内存地址的映射，并进一步将这些地址映射到 DRAM 的行、存储体和通道位置。然后，攻击者对内存进行分析以发现易翻转位。接下来，攻击者通过操作内存，使易受攻击的目标内存位置映射到易翻转位上。

最后，攻击者反复激活攻击行以翻转目标位。这最后一步颇具挑战性，因为攻击者需要绕过缓存层次结构，以确保内存访问是从动态随机存取存储器（DRAM）中提供的。

行冲攻击的安全影响 过往的研究提出了多种利用这种攻击造成安全影响的方法。例如，Razavi 等人[47]利用行冲攻击破坏 OpenSSH 公钥认证并伪造 GPG 签名，Xiao 等人[61]展示了它如何破坏 Xen 中的内存隔离，并在共享机器上访问任意物理内存，Kwong 等人[34]展示了如何从 OpenSSH 守护进程中提取 RSA 密钥。行冲攻击已在多种环境中得到验证：从 JavaScript [12, 24]，通过网络[39, 53]，借助异构 FPGA 系统[60]，甚至利用 GPU [16]。

行冲撞防御。多篇文献提出利用硬件和软件机制相结合的方式来检测和防范行冲撞攻击[1, 5, 29, 35, 51]。动态随机存取存储器（DRAM）供应商也在其模块中实施了针对行冲撞的防护措施。大多数供应商通常采用的目标行刷新（TRR）机制，即 DRAM 模块会监测哪些行被激活的次数最多，并对相邻行提前发出额外的行刷新指令。然而，许多 TRR 的实现方式都容易受到专门设计的激活模式的攻击，从而被绕过[12, 17, 25, 27, 32]。

最小激活次数。行攻击的一个重要性能指标是 $_{ACmin}$（最小激活次数）——攻击者必须激活攻击行多少次才能使特定位翻转。虽然最初公布的攻击需要数百万次激活，但针对某些内存变体的新攻击仅需 6000 次激活即可[25]。正如我们在工作中所展示的，同一 DRAM 模块中的各个单元的最小激活次数各不相同，而且它具有明显的阈值效应——当正确配置时，执行的行激活次数少于 $_{ACmin}$ 的行攻击总是会失败，而激活次数略多于 $_{ACmin}$ 则会使位翻转的概率非常高。

2.3 功率分析

功耗分析是侧信道攻击中最为成熟的一种形式[15, 31]。它利用了 CMOS 设备瞬时功耗与这些设备内部状态随时间变化之间的依赖关系。这些状态变化取决于正在处理的指令和数据[41]。因此，能够监测设备功耗的攻击者可以了解其内部状态。要实施功耗分析攻击，攻击者通常会在待测设备（DUT）与其电源之间放置测量探头，并使用深存储示波器收集流经探头的瞬时电流。由于需要物理接触，此类攻击只有在攻击者对 DUT 具有物理控制权时才被视为实际威胁。

降低接触要求。用近场电磁探头替代电源探头[18]可以放宽对物理接触的要求，使攻击能够在距离被测设备几毫米的范围内进行。通过测量远场电磁波中的功耗调制，可以进一步放宽要求，将攻击范围扩大到几米[6, 45]。

Genkin 等人[20]利用了笔记本电脑地线中与功耗相关的波动，攻击者可以通过触摸电脑机箱上暴露的金属部分或以太网、VGA 或 USB 线缆的远端来获取这些波动。尽管攻击范围有所增加，但这些设置本质上仍与接触式攻击类似，因为它们要求攻击者必须与目标设备物理上相邻。

消除近距离要求。通过让被测设备自行测量其功耗并将其报告给攻击者，可实现攻击距离的大幅增加。例如，可利用板载模拟 - 数字转换器（ADC）电路来测量功耗，从而使不受信任的代码能够从 TrustZone 安全元件中提取密钥[43]。对于基于 FPGA 的设备，Zick 等人[66]展示了如何利用可重新配置逻辑使 FPGA 测量其自身的供电电压，而 Schellenberg 等人[48]则展示了如何将此基本功能构建到一个特洛伊木马中，从而实现远程功耗分析攻击。Genkin 等人[19]展示了 PC 内置麦克风如何拾取与功耗相关的辐射，从而使拥有远程麦克风访问权限的对手（例如，在电话会议期间）能够恢复机密信息。在个人计算机领域，Platypus 攻击展示了拥有对英特尔[38]和 AMD[37]处理器运行平均功耗（RAPL）寄存器访问权限的对手如何执行功耗分析攻击。自鸭嘴兽（Platypus）首次被发现以来，对 RAPL 接口的访问一直仅限于特权软件，从而减轻了这一攻击途径。

⊠Attack 模型与实验设置

我们假设存在一个用户级的攻击者，其能够对目标机器反复实施行攻击（Rowhammer），并且能够观察攻击是否成功。我们明确指出，我们假设不存在额外的 CPU 或操作系统支持来报告功耗、电池电量、射频辐射或任何其他功耗相关特征。我们也不需要 root 级权限，仅使用用户级权限即可执行 HammerScope 测量。最后，与许多行攻击不同，我们不对受害者的内存位置做出任何假设，也不依赖于“内存按摩”技术。

实验设置。我们在多种机器上评估了我们的攻击，如表 1 所示，这些机器涵盖了多个 CPU 世代以及 DDR3 和 DDR4 模块。对于 DDR3 机器，我们使用 HammerTime 软件套件 [52] 中的代码来恢复物理内存与 DRAM 布局之间的映射关系，然后应用标准的双面行攻击来翻转易受攻击的位。对于 DDR4 机器，我们使用 TRRespass 工具包 [17] 中的代码来恢复映射关系，然后使用九面行攻击来翻转位。为了将 [17] 中的代码转换为用户模式下运行，我们采用了 Kwong 等人 [34] 和 Tobah 等人 [54] 的方法，强制 Linux 分配器分配一个连续的 2MB 块，然后应用 Pessl 等人 [46] 的时间测量方法来查找该块中的连续 DRAM 行。

4 HammerScope

如上所述，HammerScope 攻击基于观察 DRAM 功耗与实现位翻转所需 Rowhammer 次数之间的关联。更具体地说，我们发现随着 DIMM 功耗的增加，DIMM 中的位更难翻转，需要更多的

中央处理器型号	动态随机存取存储器类型	动态随机存取存储器供应商	内核版本	操作系统版本
酷睿 i7-4770	DDR3-1	三星	5.13.0 版本	Ubuntu 20.04.1 版本
酷睿 i7-4790（1）	DDR3-2	金斯顿	5.9.0 版本	Debian 5.9.1 系统
酷睿 i7-4790（2）	DDR3-3	金斯顿	5.9.0 版本	Debian 5.9.1 版本
酷睿 i7-6600U	DDR4-1	三星	5.4.1	Ubuntu 18.04.6 版本
酷睿 i7-770	DDR4-2	三星	5.4.1	Ubuntu 18.04.6 版本
酷睿 i7-10700K	DDR4-320	三星	5.13.0 版本	Ubuntu 20.04.3 版本
酷睿 i7-8700K	DDR4-400	三星	5.15.0 版本	Ubuntu 20.04.2 版本

表 1：测试机器规格

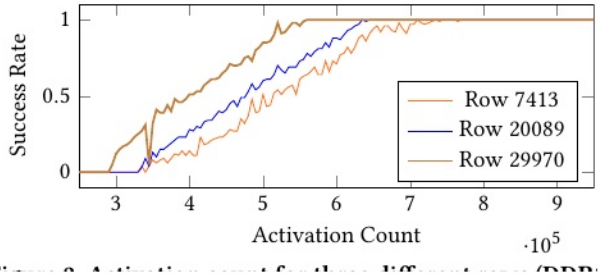


图 2：三条不同行（DDR3）的激活计数

行冲刷攻击尝试。最后，由于双列直插式内存模块（DIMM）的功耗与系统活动相关，我们可以利用行冲刷攻击作为基于功耗的侧信道来监测各种系统事件。

4.1 设定激活阈值

由于 HammerScope 依赖于利用行翻转攻击来观察 DRAM 的功耗，我们的首要任务是确定翻转一位所需的行激活次数的基准。为此，我们首先在 DDR3 内存模块（表 1 中的 DDR3-2）上进行了行翻转分析步骤，确定了包含易翻转位的几行。对于每一行，我们测量了其发生位翻转的概率与行激活次数之间的关系。我们测量了 150 种不同的激活次数，范围在 25 万到 99.5 万之间，对于每个激活次数，都尝试翻转一位 100 次。

图 2 展示了实验结果。具体而言，当激活次数低于某个最小阈值 $_{\text{ACmin}}$ 时，不会出现行翻转位翻转现象。此外，该阈值取决于所选的易受攻击行。最后，我们观察到当激活计数接近激活阈值时，行翻转位翻转的概率较低，这给我们的测量带来了噪声。

4.2 观察系统活动

在确定了空闲系统中不同行的 $_{\text{ACmin}}$ 值之后，我们现在展示如何利用翻转一个位所需的行激活次数来监测系统的内存活动情况。为此，我们运行一个程序，该程序在 30 秒内反复向 DRAM 写入数据，然后让内存空闲 10 秒。在运行我们的程序的同时，我们反复计算翻转一个预先选定的易受行攻击影响的位所需的行激活次数。图 3 展示了所需行激活次数随时间的变化情况。可以看出，在空闲期间（白色背景），翻转目标位所需的行激活次数较低，而在内存活动期间（灰色背景）则较高。

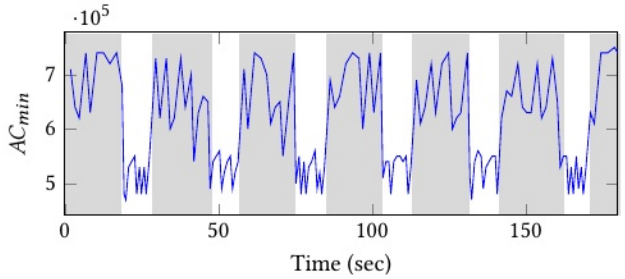


图 3：内存闲置时和访问时 ACmin 的变化情况。

4.3 降低噪音

尽管图 3 中所描绘的信号让我们能够区分内存活动的高低区域，但所观察到的信号仍然存在噪声。这是由于仅使用 $_{\text{ACmin}}$ 激活时 Rowhammer 的成功概率较低（见图 2），这使得 HammerScope 无法准确测量 DIMM 的功耗。在本节中，我们将展示如何通过确定针对给定易受攻击位启动 Rowhammer 尝试的最佳时间来对 HammerScope 观测到的信号进行去噪处理。

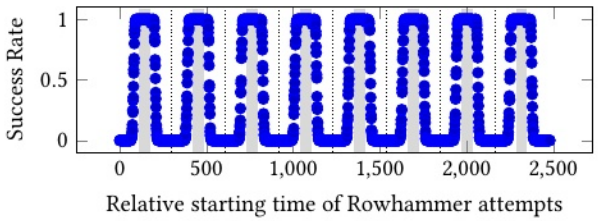
$_{\text{行激活}}$的$_{\text{机制}}$。首先回顾一下，动态随机存取存储器（DRAM）单元是电容器，其会随时间漏电，从而导致位错误。为避免数据丢失，内存控制器会定期发出刷新命令，为同一行中所有电容器补充电荷。我们用刷新周期来指代对同一行连续两次刷新命令之间的时间间隔。

提高行冲突攻击的成功概率。由于行冲突攻击会增加相邻行的电荷耗尽率，我们假设在一行的刷新周期内进行的行冲突攻击序列，且未被刷新命令中断，将更有可能翻转该行中的易受攻击位。换句话说，每行都有一个易受攻击的时间窗口，在该窗口内启动的行冲突攻击比在窗口外启动的攻击更有可能成功，这大概是因为攻击能够在后续的刷新命令之前翻转位。

为了找到这个易受攻击的时间窗口，我们首先对选定的行进行多次行锤击尝试，每次尝试包含固定数量的行激活操作。对于每次尝试 I，我们记录其开始时间 T_I 以及该尝试是否成功（即是否导致位翻转）。然后，我们利用收集到的信息来确定刷新周期，并由此得出该行的易受攻击时间窗口。

确定刷新周期。为了确定刷新周期，我们在 20 毫秒到 65 毫秒之间以 1 纳秒的分辨率搜索所有可能的刷新周期，从而形成一个包含 4500 万种可能性的搜索空间。对于每个候选值 T_R，我们测试其起始时间 $T_I$of 是否落在相对于候选刷新周期的固定窗口内，具体如下所述。

对于每次以 T_I 为起始时间的行攻击尝试 I，我们通过 $T_I' = T_I \bmod T_R$ 计算其相对于候选时间 $T_R$period 的起始时间。如果选择了正确的 T_R 值，不同刷新窗口的 T_I's 将会对齐，我们将会看到一个成功率很高的明确区域。



指令。（a）正确设置刷新周期。阴影区域表示易受攻击的时间窗口，虚线表示（推测的）刷新位置。

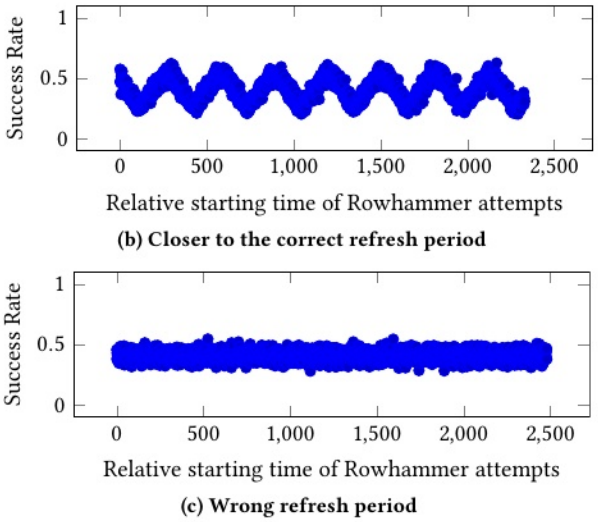


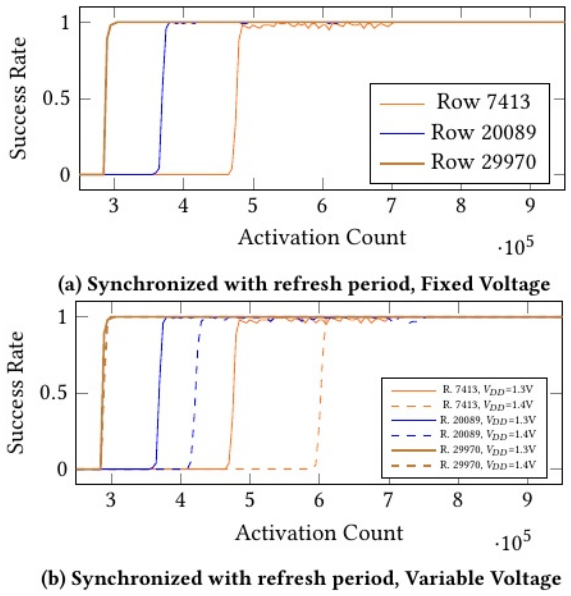
图 4：在 DDR3 上查找易受行攻击影响的窗口

观察刷新周期。图 4 展示了这一现象，为了便于说明，重复了 8 个连续的刷新周期。对于正确的刷新周期值（图 4a），在连续刷新命令之间定义明确的窗口内开始的行攻击尝试通常会成功，而窗口之外的尝试通常会失败。对于不正确的刷新周期值（图 4b 和 4c），这种现象不会出现，因为成功行攻击尝试的起始时间相对于（错误猜测的）刷新周期不再处于相同的位置。因此，产生行攻击成功率最高区域最窄的 T_R 值即为正确的刷新周期。

识别行冲突漏洞窗口。除了计算刷新率之外，图 4a 还确定了易受攻击的时间窗口，这些窗口对应的是尽管仅使用了 $_{\text{ACmin}}$ 激活，但行冲突攻击成功率仍较高的时间段。更具体地说，查看图 4a 中的灰色阴影区域，我们发现相对于刷新命令（虚线）存在一个区间 (a, b)，在此区间内开始的行冲突攻击有很高的成功概率。换句话说，行的易受攻击窗口涵盖了所有满足 $JT_R+A < X < JT_R+B$（对于某个 $j \in \{0, 1, 2, ...\}$）的时间 x，使得在此窗口内执行的行冲突攻击极有可能成功。

$_{\text{一种同步行攻击}}$。掌握了易受行攻击影响的内存窗口的位置后，我们现在可以重新审视

不同值的 $_{ACmin}$ 在不同可翻转位上的成功概率。实际上，图 5a 展示了重复图 2 中实验的结果，但这次将行翻转攻击尝试与目标行的行翻转易受攻击窗口同步。可以看出，与易受攻击窗口同步会导致行翻转攻击尝试在失败与成功之间出现更明显的转变，这将显著降低 HammerScope 测量到的功率信号中的噪声。



(a) Synchronized with refresh period, Fixed Voltage

(b) Synchronized with refresh period, Variable Voltage

图 5：DDR3 锤击成功率与激活次数的关系

处理 DDR4 系统。我们用于识别 DIMM 刷新周期和行易受攻击窗口的技术不仅适用于 DDR3，也适用于更新的 DDR4 系统。图 6 是图 4a 的 DDR4-3 对应图。可以看出，正确的 T_R still 值会产生高行冲突成功率的明显区域，尽管总体成功率较低（约 50%），与 DDR3 系统相比。我们推测这是因为我们用于攻击此系统的 9 边形攻击方法仅部分成功地克服了模块的 TRR 缓解措施。

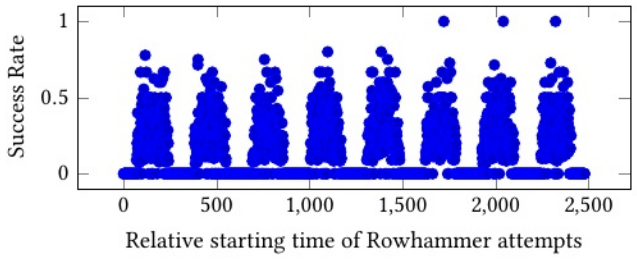


图 6：在 DDR4 上查找易受行攻击影响的窗口

观察刷新周期。我们将用于查找刷新周期和易受攻击窗口的方法应用于我们的 DDR3 和 DDR4 实验设置，总结结果如表 2 所示。

如表所示，所测得的刷新周期从未达到 64 毫秒，并且在相同机器的不同实验中有所变化，这取决于 DRAM 的时钟频率。最后，与 Hassan 等人[25]的研究结果一致，我们发现所有 DDR4 模块的刷新周期均低于 32 毫秒，这可能是为了防范行攻击。

动态随机存取存储器模块	动态随机存取存储器频率	刷新周期（毫秒）	锤击测试采样率（赫兹）
DDR3-1	1066 兆赫	63.897672	15.65
DDR3-1	1333 兆赫	42.598448	23.47
DDR3-2	1333 兆赫	64.045484	15.61
DDR3-3	1400 兆赫	42.614029	23.47
DDR3-3	1600 兆赫兹	42.598448	23.47
DDR4-1	2133 兆赫兹	30.529955	32.75
DDR4-2	2400 兆赫兹	30.628924	32.65
DDR4-320	2400 兆赫兹	30.611017	32.67
DDR4-400	2400 兆赫兹	30.558066	32.72

表 2：不同 DRAM 配置下的实测刷新周期及对应的采样率。

4.4 电压对交流最小值的影响

在获得了准确评估目标位翻转的 $_{ACmin}$ 值的方法之后，我们接下来观察 $_{ACmin}$ 值如何随 DIMM 电源电压的变化而变化。为此，我们重复了图 5a 中的实验，并有意将系统的 DRAM 电源电压从 1.4V 降至 1.3V。对于每个电压水平，我们随后在 3 行中测量 $_{ACmin}$ 值，并将结果绘制在图 5b 中。

如图 5b 所示，对于给定的行，$_{ACmin}$ 的值与 DIMM 的供电电压成正比，这意味着在较低电压下，要成功翻转位，所需的行激活次数更少。这不仅表明 DIMM 在欠压时更容易翻转，而且 $_{ACmin}$ 与电压之间的关系还表明，我们可以通过行冲突攻击来获取有关 DIMM 供电电压的信息。例如，从图 5b 可以看出，对行 7413 进行 500,000 次激活，在 V_{DD}=1.3V 情况下成功率为 100%，而在 V_{DD}=1.4V 情况下则为 0%。因此，通过测量对行 7413 进行行冲突攻击时的激活次数，攻击者能够区分 V_{DD}=1.4V 和 V_{DD}=1.3V。

4.5 构建 HammerScope

在观察到 DIMM 供电电压与成功触发行冲突攻击导致位翻转所需的激活阈值 $_{ACmin}$ 之间存在关联之后，我们现在将描述如何利用行冲突攻击来测量 DIMM 的功耗。

一次简单的尝试。理论上，安装 HammerScope 最基本的方法是等待下一个易受攻击的窗口出现，然后开始对目标位进行“锤击”，记录激活次数，直到该位翻转。虽然这种方法听起来

[1]We note here that both 1.3V and 1.4V are technically below the recommended V_{DD} rating for DDR3, which is 1.425V to 1.575V. However, this undervolted condition was used only in this section for visual clarity, and all subsequent measurements and attacks used the recommended voltage range.

具有吸引力的是，读取位的当前值这一操作也会刷新其电容器，从而使得进一步的激活尝试失效。猜测与记录。相反，攻击者可以先选择一个略高于空闲状态下 $_{ACmin}$ 的激活计数。然后，攻击者执行与易受攻击窗口同步的行攻击，检查在达到选定的激活计数后目标位是否已翻转。攻击者随后记录结果，并继续进行下一轮攻击。

这种方法的缺点在于它会产生二进制值。位翻转成功表明系统处于空闲状态，而位翻转失败则表明 DRAM 处于活动状态。

闭环反馈。为了进一步提高 HammerScope 的测量分辨率，我们借鉴了控制理论中的一种技术，称为闭环反馈[14]。其主要思想是仍然执行猜测和记录，但根据当前攻击迭代翻转目标位的成功（或失败）情况，自适应地选择下一次攻击迭代的 $_{ACmin}$ 值。由于我们实质上是在跟踪 $_{ACmin}$ 随时间的变化值，因此这种方法使我们能够获得更高灵敏度的测量结果。

更具体地说，图 7 展示了我们方法的伪代码。在第 2 行循环的每次迭代中，我们首先将目标受害行设置为已知的行攻击状态（第 3 行），并将其从 CPU 缓存中清除。然后暂停，等待下一个易受攻击的时间窗口（第 4 行）。当易受攻击的时间窗口到达时，我们使用可控的激活计数（第 5 行）执行行攻击，并在跟踪中记录激活计数。最后，我们检查行攻击是否成功，并相应地更新下一次攻击迭代的激活计数（第 7 至 10 行）。

通过这种方法，我们实质上是追踪 $_{ACmin}$ 值随时间的变化情况，并将其作为 DIMM 功耗水平的替代指标。尤其值得一提的是，这种方法的输出结果并非二进制形式，与前一种方法不同，而且与猜测并记录的方法相比，其分辨率要精细得多。

```
1  let activation_count = AC_START
2  while true:
3    reset_row()
4    wait_for_vulnerable_window() //As explained in Sec. 4.3
5    perform_rowhammer(activation_counts)
6    trace.append(activation_counts)
7    if bit_flipped():
8      activation_counts -= DELTA
9    else:
10     activation_counts += DELTA
```

图 7：HammerScope 控制回路算法。

4.6 锤击作用范围的特征描述

我们现在将 HammerScope 与通过 RAPL 和示波器获得的功耗测量值进行比较，并讨论 HammerScope 的采样率、带宽和信噪比。

将 HammerScope 与 RAPL 进行比较。为了展示我们方法在测量 DRAM 功耗方面的有效性，我们将 HammerScope 的结果与 RAPL 接口报告的 DIMM 功耗进行了比较。RAPL 是一种用于测量和控制基于英特尔和 ARM 的机器中各个组件功耗的机制。它提供了一些功能，其中包括基于向 DRAM 供电的集成电压调节器的高分辨率采样，对 DRAM 子系统的功耗进行实时直接测量 [11, 13]。在本次实验中，我们运行一个程序，该程序在 30 秒内反复向 DRAM 写入数据，然后闲置 10 秒。我们使用英特尔 RAPL 接口和 HammerScope 控制循环实现，在我们的 DDR3-1 设备上以 16Hz 的采样率对 DRAM 功耗进行采样。见图 8a。

如图所示，RAPL 和 HammerScope 测量都能准确捕捉到 DRAM 活动。RAPL 和 HammerScope 测量结果高度相关（$\rho = .97$，$P < p < .001$）。在我们的 DDR3-3 和 DDR4-2 设备上也得到了类似的结果。分别见图 8b 和图 8c。

将 HammerScope 与示波器测量结果进行比较。

除了将 HammerScope 与通过 RAPL 接口获取的测量结果进行比较之外，我们还将其与在我们的 DDR4-4 设备上使用示波器对 DRAM V_{PP}supply 线进行的直接物理测量进行了对比。为此，我们将一个无源探头连接到 DRAM 的 ☒ 线上，并使用 Picotech PicoScope 3425 深存储示波器以 100 kHz 的采样率对电压水平进行采样，同时进行 HammerScope 和 RAPL 测量。计算机执行了一个脚本，该脚本每两秒重复访问内存，然后闲置两秒。由于我们的探头连接是与电压供应并联，而非串联，因此这种物理测量实际上测量的是由功耗变化引起的瞬态电压下降，而非直接测量功耗。为了恢复功耗曲线，我们应用了一个以 33 kHz 为中心的数字带通滤波器，接着是一个整流器和一个低通滤波器。此评估的结果如图 9 所示。示波器读数以相对比例显示。如图所示，Rowhammer 测量值与 RAPL 测量值和示波器测量值非常准确地吻合，尽管采样率较低。

计算 HammerScope 的采样率。为了执行 HammerScope，攻击者在先前确定的易受攻击窗口内，每 DIMM 刷新周期进行一次测量。因此，HammerScope 的采样率取决于机器特定的内存模块及其配置，见表 2。在我们测试的不同内存配置中，我们观察到最高的采样率为 32.67 赫兹（对应于 DDR4 上 32 毫秒的刷新周期），最低的采样率为 15.65（DDR3 上 64 毫秒的刷新间隔）。特别是，DDR4 机器上 DIMM 刷新率的翻倍似乎提供了更高的 HammerScope 采样率，从而提高了现代机器上的测量分辨率。最后，我们注意到，尽管 32.67 赫兹与其他功耗分析原语相比采样率相当低，但对于对现实目标发起攻击而言，它仍然足够（见第 5 节）。

评估 HammerScope 的带宽。为了表征 HammerScope 的带宽（即水平精度），我们执行了一个程序，该程序反复访问 DRAM，然后休眠一段预定时间，从而在 DRAM 的功耗中生成一个方波信号。接下来，我们测量了 DRAM 的 RAPL 接口和 HammerScope 是否能够跟踪此信号。我们的结果表明，这两种方法都能够跟踪频率高达 4 赫兹的周期性信号。在更高的频率下，DRAM 无法

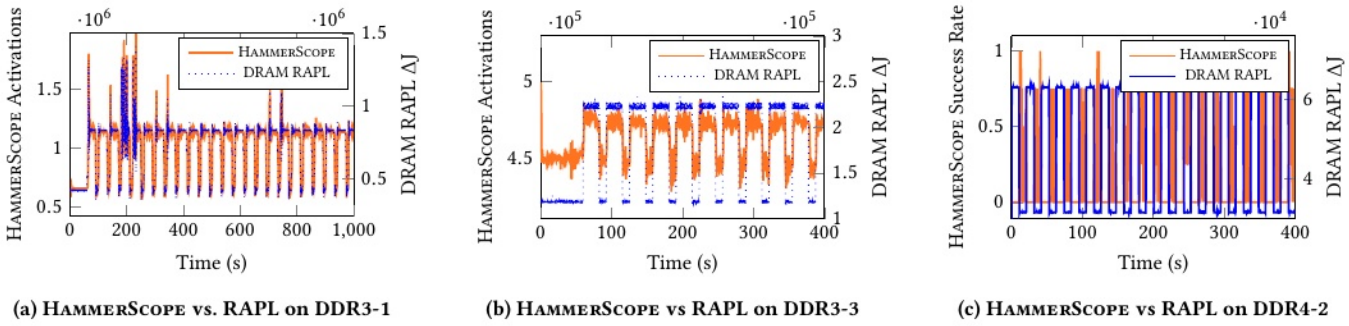


图 8：HammerScope 与 RAPL 的比较

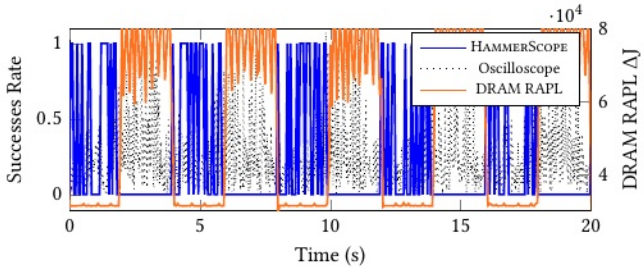


图 9：HammerScope 与 RAPL 与示波器在 DDR4-4 上的对比

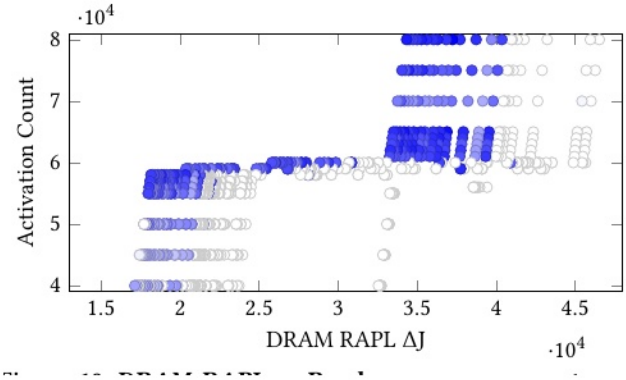


图 10：DDR4 内存上 DRAM RAPL 与行攻击（Rowhammer）成功率对比（深色圆圈表示成功）

由于没有足够的时间从高功率状态恢复过来，这使得 RAPL 和 HammerScope 的测量结果都不准确。

表征 HammerScope 的信噪比。为了确定 HammerScope 的信噪比（即垂直精度），我们执行了一个不同的测试程序，该程序在一个紧密循环中交替进行内存访问和可控数量的空操作指令。这有效地通过脉冲宽度调制（PWM）创建了可控的功耗，其中调制的占空比由内存访问和空操作指令的数量之比控制。接下来，我们改变激活次数，同时测量行冲突攻击的成功率以及 DRAM 的实际功耗（使用 RAPL）。该实验的结果如图 10 所示。

在该图中，每个圆圈代表总共 1200 次实验中的单次实验，圆圈的颜色表示锤击成功率，颜色越深表示成功率越高。如图所示，对于给定的 DRAM RAPL 值，激活计数存在一个非常明显的阈值，如果激活计数高于此阈值，则锤击成功率较高（深色圆圈），而低于该阈值时则几乎无成功（白色圆圈）。在实际的 HammerScope 攻击中，攻击者可以控制激活计数（相当于在图中垂直移动），并需要确定 DRAM 当前的功耗水平（相当于确定其在图中的水平位置）。在这种情况下，只有当多个功耗水平共享相同的锤击阈值时，才会发生测量错误事件。在我们的测量中未观察到此类情况。

4.7 解释 HammerScope 的工作原理

我们的实验结果表明，动态随机存取存储器（DRAM）功耗的增加会降低行攻击（Rowhammer）的有效性，这使得 HammerScope 能够通过行攻击间接测量 DRAM 的功耗。在本节中，我们基于 Kim 等人最初提出的电容串扰模型[30]，对观察到的现象提出了一些可能的解释。

存储器组织。我们首先回顾一下，动态随机存取存储器（DRAM）单元是以矩阵结构排列的，每个 DRAM 单元包含一个存储电容C_{S}connected，通过一个访问晶体管与一条位线（垂直排列）相连。访问晶体管通过字线（水平排列）进行控制，字线根据要访问的单元来决定何时打开哪些晶体管，如图 1 所示。当字线被激活时，访问晶体管导通，将存储电容与位线连接起来。存储的值由一个检测放大器通过将电容上存储的电荷与参考电压进行比较来检测。

一种行冲突攻击的物理模型。在可靠性研究领域，针对行冲突攻击提出了多种理论解释，包括电子注入与捕获、电容串扰，或者这两种情况的某种组合[59]。根据电容串扰模型，动态随机存取存储器（DRAM）单元阵列的矩阵结构会在相邻字线之间产生寄生耦合电容，如图 11 所示。当攻击行的字线被激活时，它

在电压水平从低到高快速转换时，由于电容串扰，这种转换会导致相邻行的字线出现短暂的电压尖峰。该尖峰反过来会短暂激活受害行的访问晶体管，无意中导致该行的存储单元从其存储电容中泄漏电荷。攻击行的持续和重复激活会放大这种来自受害行的无意电荷泄漏的影响。特别是，在经过 $_{ACmin}$ 次激活后，一个或多个存储电容上的电压会接近 $V_{DD}/2$，从而导致位翻转。通过这种视角审视我们的结果，我们可以提出几种模型来解释 HammerScope 背后的机制。

斜率变化率解释。如图 12 所示，寄生激活在受害字线上的持续时间取决于攻击字线的变化率，即斜率$_{变化率}$。更具体地说，攻击信号变化越快，在受害字线上形成的峰值就越窄，但幅度相对较高；攻击信号变化越慢，在受害字线上形成的峰值就越宽，但幅度相对较低[58]。如果 DRAM 功耗增加导致字线斜率变化率加快，这或许可以解释我们的实验观察结果。Chang 等人在[9，图 5]中提供了一些相关证据，作者通过模拟表明，当 DRAM 模块欠压时，其控制线的斜率变化率会变慢。基于这种解释，我们推测 DIMM 功耗的变化会导致其瞬时电压水平发生变化，从而影响其控制线的斜率$_{变化率}$，进而影响行冲突攻击的有效性。这反过来又导致了 HammerScope 观察到的与功耗相关的信号。

电荷泵耗尽解释。假设上升沿速率不受 DRAM 活动的影响，那么我们观察到的现象的另一种可能解释是考虑 DRAM 字线的瞬时电压水平。在 DDR3 模块中，此电压轨由内部电荷泵电路生成，而在 DDR4 模块中，它通过专用输入外部提供，V_{PP}。如果 DRAM 功耗增加导致电荷泵输出端出现瞬时电压下降，那么攻击者字线上的电压水平就会降低。如果上升沿速率保持不变，这反过来会导致受害者字线上的峰值变小，从而使受害者单元对行冲突攻击的敏感度降低。对此解释的支持可能来自 Yaglikci 等人的一项独立的同期研究[62]。在那里，[62] 使用 FPGA 测试板测量了多个 DDR4 DRAM 芯片，在保持激活计数为 300K 不变的同时，改变字线提升电压轨 V_{PP}，观察行冲突攻击的成功率。他们发现，在大多数评估的芯片上，将 V PP 从其标称值降低会使芯片对行冲突攻击更具抵抗力。根据这种解释，我们推测 DIMM 功耗的变化会导致其内部字线提升信号的电平发生变化，从而影响 DIMM 控制线的$_{幅度}$，进而影响行冲突攻击的效果。

关于 DRAM 竞争的解释。最后一种解释假定，无论是上升沿速率还是电压水平都不受 DRAM 功耗的影响。相反，有可能的是，对原本闲置的 DRAM 系统进行的行冲突访问与对繁忙的 DRAM 进行的访问在时间布局上有所不同。

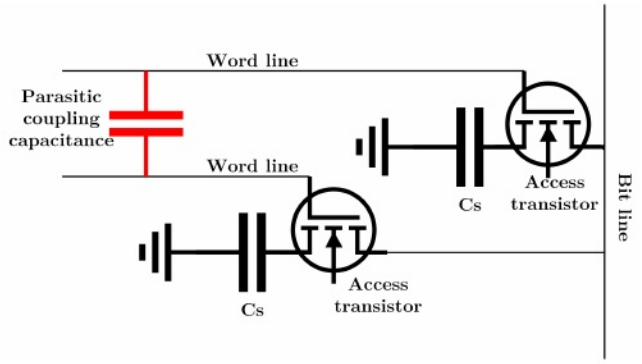


图 11：两个相邻 DRAM 单元之间的寄生耦合电容

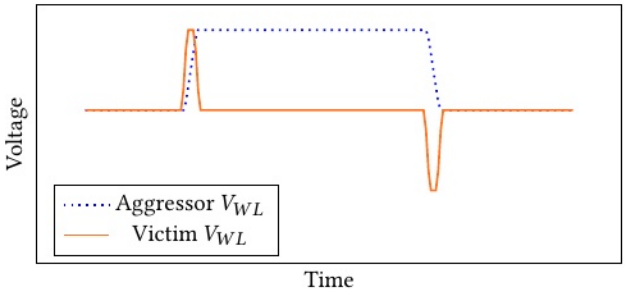


图 12：寄生耦合与行激活

在同时处理行攻击（Rowhammer）和其他工作负载的情况下，根据这种解释，引发行攻击的对攻击行的访问与非行攻击相关的内存访问相互交错，从而降低了行攻击的威力，使行攻击更难实施。在这种解释下，我们推测，高 DRAM 活动区域（这与 DIMM 的功耗间接相关）会导致 DRAM 控制总线上的竞争，进而使行攻击的效果降低。

鉴于上述所有解释都同样合理，我们把系统探究 HammerScope 和 Rowhammer 背后的物理原因这一任务留待未来的研究来完成。

4.8 利用 HammerScope 区分指令

在本节中，我们将展示如何使用 HammerScope 来测量执行不同指令所需的时间，从而对它们加以区分。

测量技术。我们首先注意到，如果将激活 DRAM 的指令与不激活 DRAM 的指令交错执行，那么整体平均 DRAM 功耗将取决于 DRAM 处于激活状态的时间比例，即 DRAM 占空比。我们利用这一特性，通过基于行冲突的内存测量来观察核心执行指令的运行时间。更具体地说，我们执行一个程序，该程序通过访问未缓存的地址反复激活 DRAM，然后接着

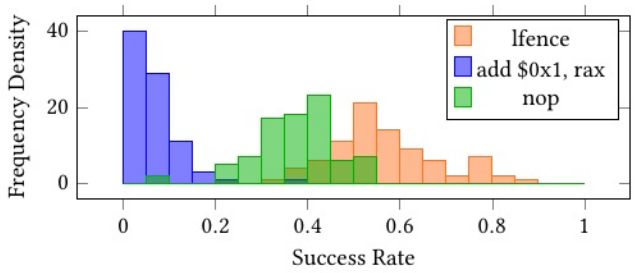


Figure 13: Differentiating instructions using HammerScope

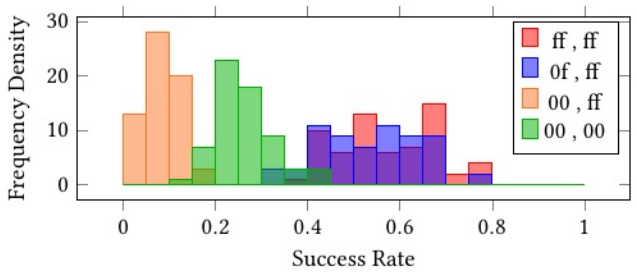


图 14：使用 HammerScope 观察 DIV 操作数

执行目标指令。同时，在另一个物理核心上，我们使用 HammerScope 持续测量 DIMM 的功耗。执行时间较长的指令会导致 DRAM 的平均占空比降低，从而降低 DRAM 功耗。相反，执行时间较短的指令会导致占空比升高，从而增加 DRAM 功耗。因此，通过观察执行各种指令时 DRAM 的功耗，我们能够区分这些指令。

区分指令类型。为了展示 Hammer-Scope 区分不同指令类型的能力，我们使用了 DDR4-3 设置，并反复执行 8 次 q 字写操作，随后执行 16 次待测指令。同时，在另一个物理核心上，我们使用 Hammer-Scope 持续采集 DIMM 的功耗，共采集 128 个样本。图 13 总结了我们的发现。可以看出，由于执行时间的不同，我们能够区分这三种不同的指令。

观察除法操作数。我们不再仅仅区分不同的指令，而是进一步展示 HammerScope 区分不同可变时间指令操作数的能力。为此，我们关注除法运算，已知其执行时间可变[55]，且存在安全影响[33]。为了展示区分除法操作数的能力，我们再次使用 DDR4-3 设置，并执行 8 次 8 字节写操作，随后使用不同操作数执行 16 次除法指令。具体来说，我们使用了四个不同的 128 位被除数和一个固定的除数 0xffffffffffffffff，见图 14。如图所示，我们能够区分不同操作数的除法运算。

预取指令。预取指令[49]允许程序向 CPU 表明内存中的某个位置可能很快会被访问，从而让 CPU 选择性地将所引用的地址加载到缓存中。正如先前的研究[21, 23, 37]所观察到的，如果预取指令引用的是无效或未映射的地址，那么其执行速度会更慢。这是因为无效地址的转换信息不会存储在转换后备缓冲区（TLB）中，导致对这些地址的预取需要遍历页表。相反，有效地址的转换信息即使页面不可访问也会存储在 TLB 中。因此，对有效地址的预取比对无效地址的预取要快，这可以通过 HammerScope 观察到。接下来我们将展示如何利用 HammerScope 作为测量预取指令执行时间的另一种方法，从而推断出地址的有效性。

通过 HammerScope 定时预取指令。如上所述，预取指令在无效或未映射地址上执行的速度比在正常地址上慢。为了利用这一点，我们运行一个程序，该程序通过从几个连续的未缓存地址读取来反复激活 DRAM，然后对目标地址执行预取指令。同时，我们在另一个物理核心上运行 HammerScope 来测量 DIMM 的功耗，这与之前的实验类似。由于预取指令仅在第一次迭代中访问内存（以实际获取目标），而在后续迭代中不再访问，因此预取指令执行时间越长，DRAM 的平均占空比就越低，从而导致 DRAM 功耗降低。相反，预取指令执行时间越短，DRAM 的占空比就越高，从而导致 DRAM 功耗增加。因此，攻击者通过测量 DRAM 的功耗，可以推断出预取指令的执行时间，从而推断出预取地址的有效性。

通过 HammerScope 测量预取指令。为了实证地展示这种影响，我们使用 DDR4-2 设置来捕获 DRAM RAPL 和 HammerScope 测量值，同时系统执行一个由 8 次 DRAM 内存访问和可变数量的重复执行预取指令组成的循环。对于预取地址，我们针对已映射和未映射的内存地址进行了测试。我们的结果如图 15 所示。Y 轴表示在 32 次尝试中，固定激活次数为 60,000 的 9 边同步行攻击的平均锤击成功概率，X 轴显示预取指令计数。如图所示，已映射和未映射地址的 HammerScope 行为存在显著差异。

⊠Attacks 使用 HammerScope

在证实了 HammerScope 能够区分指令类型和操作数的可行性之后，本节将展示如何利用 HammerScope 发动端到端攻击。

5.1 区分已映射地址和未映射地址

我们首先回顾一下 HammerScope 能够测量 Prefetch 命令的执行时间，而且 Prefetch 命令的执行时间取决于地址是已映射还是未映射。在本节中，我们将展示如何利用这一点来区分已映射地址和未映射地址。如图 15 所示

结果表明，当在每次循环迭代中执行少量预取命令时，系统大部分时间都花在执行 DRAM 访问命令上，因此映射内存和未映射内存情况下的功耗测量值相似。然而，当每次迭代中执行的预取命令次数足够多时，针对映射地址快速终止的预取命令与针对未映射地址较慢的预取命令之间的运行时间差异就会通过 DRAM 功耗表现出来。

更具体地说，由于预取映射地址时的平均功率水平较高，所以在预取映射内存期间执行的行攻击尝试往往会失败。另一方面，只要连续发出足够多的预取指令，预取未映射内存期间的行攻击尝试就会有非零的成功率。如图 15 所示，当 8 个 DRAM 访问命令与 64 个预取命令交错时发出的行攻击，只有在预取的地址是无效的未映射地址时才会成功。这一特性可被用于侧信道攻击，使攻击者能够绕过 SMAP 和 ASLR，前提是攻击者能够访问高分辨率定时器。

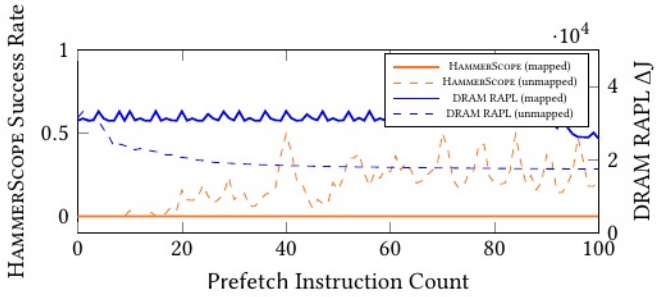


图 15：使用 HammerScope 的定时指令

5.2 攻击内核地址空间布局随机化（KASLR）

地址空间布局随机化（ASLR）是一种通过随机化敏感代码和数据的内存位置来防范代码重用攻击的机制[42]。内核地址空间布局随机化（KASLR）是 Linux 中 ASLR 的实现，旨在通过随机化内核代码段的基地址来保护内核，该代码段包含内核及其相关内核模块的代码。内核的代码段映射到 0xffffffff80800000 至 0xffffffffc0800000 范围内的某个地址，最大大小为 1GB[7]。正如[8]所观察到的，内核代码段对 2MB 边界进行对齐，这使得内核代码有 512 种可能的基地址。

利用 HammerScope 破解 KASLR。借助 HammerScope 利用“预取”命令区分已映射地址和未映射地址的能力，我们能够从无特权进程攻击 KASLR，无需访问（如今已受保护的）RAPL 接口或进行时间测量。

在攻击过程中，我们使用 PREFETCHNTA 指令反复预取内核文本段所有可能基地址的首个字节。我们执行 512 次测量，在 0xffffffff80800000 到 0xffffffffc0800000 范围内以 2MB 为间隔从各个地址预取数据。最后，为了区分映射地址和未映射地址的预取命令，我们采用第 5.1 节中的方法，将预取命令与内存读取命令交错执行，并观察成功翻转位的激活阈值。

攻击 DDR3 系统。图 16 展示了我们使用 DDR3-3 设置进行实验的结果。X 轴表示文本段的候选地址。为了安装 HammerScope，我们对候选地址连续使用了 8 个 Prefetch 命令，并在其中穿插了 8 个连续的 DRAM 访问。在访问的同时，我们执行了双面行冲突攻击，并使用第 4.5 节中介绍的控制循环方法记录了 AC_{min} 的值。如图 16 所示，当访问未映射内存时，Prefetch 命令较慢，因此阈值激活计数较低。然而，当预取映射到内核的地址时，Prefetch 命令完成得更快，从而导致激活计数更高。

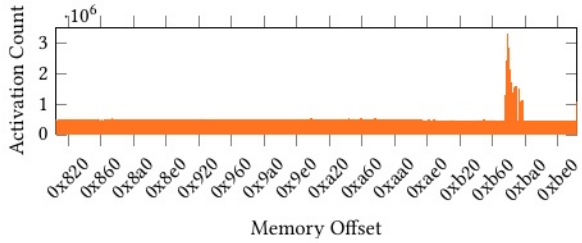


图 16：利用 HammerScope 攻击 DDR3 上的 KASLR

攻击 DDR4 系统。从 DDR3 机器转向 DDR4，图 17 展示了我们对 DDR4-3 系统的攻击结果。在此，我们对候选地址连续使用 64 次预取命令，并在其中穿插 8 次连续的 DRAM 访问。最后，我们使用 9 边形行锤击 [17] 来克服目标行刷新防护措施，实验中将激活计数阈值设为 65000。我们采用第 4.5 节中描述的猜测与记录方法，使用 HammerScope 攻击，并在 Y 轴上绘制 50 次 HammerScope 尝试的平均成功概率。如图 17 所示，预取未映射地址会导致较高的行锤击成功率，而预取已映射地址则成功率极低。总体而言，与 DDR3 的情况一样，我们能够清晰地识别出内核基地址的值，从而破坏了内核地址空间布局随机化（KASLR）。

攻击演示。在论文的附录资源库中可找到一段视频，展示了针对配备 DDR4 内存的英特尔 i7-7700 系统的端到端无特权攻击。整个攻击耗时为

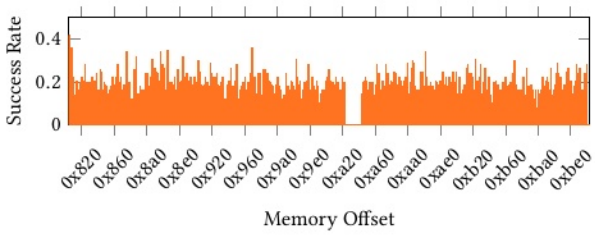


图 17：利用 HammerScope 攻击 DDR4 上的 KASLR

总共 12 分 30 秒，其中 15 秒用于内存分配和模板处理，3 分 45 秒用于定位易受攻击的行并计算刷新周期和相位，8 分 30 秒用于攻击本身。

5.3 将 HammerScope 和 Spectre 相结合

幽灵漏洞（Spectre）使得攻击者能够利用内核中的推测执行漏洞，通过基于缓存的隐蔽通道读取内核内存。攻击者首先在内核中定位一个幽灵漏洞 1（Spectre v1）的漏洞利用程序（gadget）。该漏洞利用程序执行一个有边界检查的数组访问（见图 18 中的第 2 行），获取值 secret。最后，该漏洞利用程序执行另一个访问（见图 18 中的第 3 行），访问地址取决于 secret 的值，从而通过基于缓存的隐蔽通道泄露 secret。

```
1  if (offset < array1_size)
2    secret = array1[offset]
3    temp &= array2[secret * 512];
```

图 18：幽灵 v1 漏洞利用程序

使用 HammerScope 进行 Spectre 攻击。我们实现了一个自定义的内核模块，其中包含一个类似于图 18 中的 Spectre v1 漏洞利用程序，将其加载到 Linux 内核中，并允许用户模式应用程序通过 ioctl 调用对其进行控制。首先，我们使用较小的偏移值调用该漏洞利用程序，旨在误导分支预测器。接下来，我们使用超出范围的偏移值调用 Spectre 漏洞利用程序，导致 CPU 越过图 18 中第 1 行的检查进行推测执行。这种推测执行会从攻击者控制的地址将一个值加载到 secret 中（图 18 中第 2 行）。然后，当推测执行到达图 18 中第 3 行时，会将一个依赖于 secret 的 array2 偏移量加载到 CPU 的缓存中。最后，CPU 发现发生了错误的推测执行，尝试撤销结果，并使用正确的控制流恢复执行。

一种基于行攻击的隐蔽通道。我们并非通过测量数组 array2 中元素的访问延迟来恢复秘密值，而是使用 HammerScope 来监测该装置瞬态内存访问所消耗的功率。我们依次刷新数组 array2 中的元素，然后使用 HammerScope 测量在随后对 array2 元素进行瞬态访问时 DIMM 的功率。当我们正确猜出数组索引时，对与秘密值对应的缓存的 array2 元素进行推测性访问，其功率消耗模式与未缓存的 array2 元素相比会有显著差异。

实验结果。我们在 DDR4-2 设备上对这种攻击进行了评估，使用了固定激活次数为 65,000 次的 9 边形行攻击，并采用了第 4.5 节中概述的 Hammer-Scope 的“猜测与记录”变体。该实验的结果如图 19 所示。如图所示，当访问与秘密值（本例中的 s）对应的 array2 中的（缓存）元素时，行攻击的成功率明显低于访问 array2 中的其他（未缓存）元素。最后，我们承认，将 HammerScope 用作 Spectre 隐蔽通道会导致读取速度显著降低，每访问 array2 中的一个元素需要 8 秒，每读取一个字节大约需要 30 分钟。

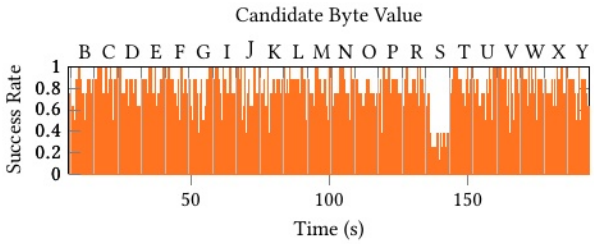


图 19：利用 Spectre 和 HammerScope 读取内核内存

5.4 网站指纹识别

作为 HammerScope 的最后一个用例，我们展示了记录 DIMM 功耗的能力如何被用于发起网站指纹攻击，即攻击者试图识别用户正在浏览的网页。现有研究已经表明功耗可用于网站指纹识别，但这些研究要么假设存在供应商提供的 API（如 JavaScript 电池 API），要么假设存在某些外部硬件，例如恶意电池或充电器 [36, 44, 63]。

实验设置。我们使用配备 i7-10700K 处理器的 DDR4-3 系统，并运行 Linux 版本的 Firefox 96.0 浏览器来进行实验。我们在封闭世界模型中收集数据，该模型假设用户访问预先确定的 25 个网站中的一个。在这种情况下，一个简单的分类器的基准准确率为 4%。我们为每个页面收集了 100 条轨迹，每条轨迹由 Firefox 加载页面时连续 40 秒的 HammerScope 测量值组成，采样率为 18 赫兹。每个生成的侧信道轨迹都是长度为 721 的二进制向量。作为参考，我们还以相同的采样率收集了 DRAM RAPL 轨迹，作为我们分类器性能的上限。我们的数据处理流程基于 Python 3.9 的 SKLearn。我们将未经任何预处理的侧信道轨迹输入到一个深度学习网络中，该网络由 3 个卷积和池化层组成，后面跟着两个密集层和一个最终的 Softmax 输出层。我们对所有层都使用了 ReLU 激活函数，并使用 Adam 优化器和稀疏分类交叉熵损失来优化网络。然后，使用分层 5 折交叉验证将数据分为训练集和测试集。

分类结果。我们的分类器准确率为 40%，远高于 4% 的基准率。基于参考 RAPL 跟踪数据训练的分类器准确率为 59%，这表明 Hammer-Scope 的信号质量并不比 RAPL 差很多。

☒Limitations 及应对措施

6.1 局限性

低采样率。HammerScope 的主要局限之一在于其采样率约为 15 - 32 赫兹（见表 2），这远低于低端示波器的采样率。HammerScope 的采样率确实给针对抗侧信道攻击实现的更高级攻击（如差分功率分析 [31]）的实施带来了重大挑战。因此，我们把针对抗攻击实现的低采样率攻击的开发工作留待未来研究。

垂直精度低。除了采样率低之外，HammerScope 测量结果的准确性也远低于使用专用模数转换器所获取的数据。这意味着能量级之间的间隔必须足够大，才能让我们通过 HammerScope 区分它们。这需要攻击者付出努力，因为被观察的指令必须以某种方式与那些引起大量 DRAM 活动的指令交错，详情见第 4.8 节。

无法攻击恒定时间代码。虽然从技术上讲属于功耗测量攻击，但 HammerScope 目前无法从诸如 $_{\text{aes-ni}}$ 或整数乘法运算之类的恒定时间指令中提取信息。我们承认这一局限性，将其归因于 HammerScope 在垂直精度和采样率方面的不足，并将从恒定时间操作中提取信息的任务留待未来的工作。

代码执行的需求。HammerScope 的核心是一种行攻击，因此继承了行攻击的威胁模型，即在目标机器上实现（无特权）代码执行。虽然在某些情况下（例如远程服务器）这种场景是自然存在的，但对于某些设备（如智能卡、硬件安全模块（HSM）和专有微控制器）来说，这确实带来了挑战。这与更传统的功耗分析攻击形成了对比，后者仅需要使用外部测量设备对目标进行被动观察。

6.2 应对措施

由于 HammerScope 测量的是物理现象，并非直接依赖制造商提供的机器状态寄存器，因此要缓解其影响并非易事。要防范 HammerScope 攻击，我们必须观察使其成为可能的一系列事件：首先，机器的机密行为被调制到其功耗上；其次，这种功耗影响了 DRAM 行对行攻击的易感性；最后，攻击者触发行攻击。要缓解 HammerScope 攻击，必须通过软件和底层硬件的非平凡更改来打破这一链条。

$_{\text{恒定功率}}$代码。为防止机密行为通过系统功耗泄露，编写安全敏感代码的人员可以考虑编写恒定功率代码，其功耗不依赖于正在处理的指令或数据。虽然此类代码已应用于被认为存在功耗分析风险的编程设备，如支付卡[41]，但我们注意到，由于 HammerScope 目前的局限性，对数据的攻击仍具挑战性。特别是，要实现基于 HammerScope 的数据提取，似乎还需要进一步改进。

添加噪声。另一种广为人知但效果稍逊的方法是随机噪声防护措施，其目的在于降低攻击者可获取的侧信道轨迹的信噪比，从而将攻击时间延长至不切实际的程度。在这种情况下，敏感代码可以考虑在执行敏感操作期间有意访问动态随机存取存储器（DRAM）。

缓解行攻击。下一个干预点可能是帮助预防或检测行攻击。Kim 等人在[29]中对基于软件和基于硬件的行攻击缓解机制进行了大量研究。Loughlin 等人[40]提出了一种分类法，将缓解措施分为三类：以隔离为中心、以频率为中心和以刷新为中心。Kim 等人在[30]中建议提高刷新率，直到在刷新窗口内无法进行足够的激活操作来翻转位。随着微型化和功耗优化使得 DRAM 硬件越来越容易受到行攻击，这变得越来越困难。Kim 等人还提出了 PARA，它在每次刷新一行时，以低概率刷新一个或多个相邻行。Kim 等人在[29]中实现了一种理想的刷新机制，该机制跟踪行激活情况，并在行即将翻转位之前向其发出有针对性的刷新命令。ANVIL [1] 通过使用硬件计数器跟踪 DRAM 访问的延迟来检测行攻击，以识别频繁访问的行。Brasser 等人实现了 CATT [5] 机制，该机制通过使用操作系统物理内存分配器在内核内存和用户内存之间进行物理隔离，从而防止攻击者利用行攻击从用户模式破坏内核内存。ProHIT [51] 通过使用低开销的概率表来维护 DRAM 激活历史记录来缓解行攻击。同样，TWiCe [35] 使用对性能影响较小的计数器来检测行攻击。Panopticon [2] 提出了一种完整的在 DRAM 内缓解行攻击的方法，通过维护一个带有激活阈值的计数器表来实现。GuardION [57] 通过使用保护行隔离 DMA 缓冲区来缓解 ARM 上的行攻击利用。You 等人提出根据每行的访问历史动态调整刷新受害行的概率来缓解行攻击 [64]。

7 Conclusions

在这项工作中，我们介绍了 HammerScope，这是一种利用行攻击进行基于软件的功耗分析的方法，并展示了在某些情况下如何利用它来泄露机密信息。行攻击在故障攻击中独树一帜，因为触发它的操作完全在数字域中进行，而其影响则在模拟域中显现。这项工作表明，这种关系是双向的——系统的模拟条件可以通过行攻击感知，并重新引入数字域。

虽然我们展示了 Hammer-Scope 的一些安全影响，但我们承认，本文中所呈现的所有攻击也都可以通过其他渠道实施。我们把进一步研究 Hammer-Scope 的其他应用以及系统分析其根本原因的任务留待未来的工作。

致谢

作者们衷心感谢阿纳托利·舒斯特曼、利亚德·奥兹、埃拉姆·加尔、卢西安·科约卡、罗恩·安德森和大卫·布兰肯贝克勒给予的帮助和建议。

本研究得到了澳大利亚研究理事会（ARC）早期职业研究人员发现奖 DE200101577；ARC 发现项目编号 DP210102670；特拉维夫大学布拉瓦尼克国际计算机研究中心；美国空军科学研究办公室（AFOSR）项目编号 FA9550-20-1-0425；美国国家科学基金会 CNS-1954712 项目；以及 AMD、英特尔和高通公司的资助。

参考文献

[1] Zelalem Birhanu Aweke, Salessawi Ferede Yitbarek, Rui Qiao, Reetuparna Das, Matthew Hicks, Yossi Oren, and Todd M. Austin. 2016. ANVIL: Software-Based

Protection Against Next-Generation Rowhammer Attacks. In ASPLOS. 743–755. https://doi.org/10.1145/2872362.2872390

[2] Tanj Bennett, Stefan Saroiu, Alec Wolman, and Lucian Cojocar. 2021. Panopticon: A Complete In-DRAM Rowhammer Mitigation. In DRAMSec. https://dramsec.ethz.ch/papers/panopticon.pdf

[3] Ishwar Bhati, Mu-Tien Chang, Zeshan Chishti, Shih-Lien Lu, and Bruce L. Ja-cob. 2016. DRAM Refresh Mechanisms, Penalties, and Trade-Offs. IEEE Trans. Computers 65, 1 (2016), 108–121. https://doi.org/10.1109/TC.2015.2417540

[4] Sarani Bhattacharya and Debdeep Mukhopadhyay. 2016. Curious Case of Rowhammer: Flipping Secret Exponent Bits Using Timing Analysis. In CHES. 602–624. https://doi.org/10.1007/978-3-662-53140-2_29

[5] Ferdinand Brasser, Lucas Davi, David Gens, Christopher Liebchen, and Ahmad-Reza Sadeghi. 2017. CAn't Touch This: Software-only Mitigation against Rowhammer Attacks targeting Kernel Memory. In USENIX Security. 117– 130. https://www. usenix. org/conference/usenixsecurity17/technical-sessions/presentation/brasser

[6] Giovanni Camurati, Sebastian Poeplau, Marius Muench, Tom Hayes, and Aurélien Francillon. 2018. Screaming Channels: When Electromagnetic Side Channels Meet Radio Transceivers. In CCS. 163–177. https://doi.org/10.1145/3243734.3243802

[7] Claudio Canella, Daniel Genkin, Lukas Giner, Daniel Gruss, Moritz Lipp, Ma-rina Minkin, Daniel Moghimi, Frank Piessens, Michael Schwarz, Berk Sunar, Jo Van Bulck, and Yuval Yarom. 2019. Fallout: Leaking Data on Meltdown-resistant CPUs. In CCS. 769–784. https://doi.org/10.1145/3319535.3363219

[8] Claudio Canella, Michael Schwarz, Martin Haubenwallner, Martin Schwarzl, and Daniel Gruss. 2020. KASLR: Break It, Fix It, Repeat. In AsiaCCS. 481–493. https://doi.org/10.1145/3320269.3384747

[9] Kevin K. Chang, Abdullah Giray Yaglikçi, Saugata Ghose, Aditya Agrawal, Ni-ladrish Chatterjee, Abhijith Kashyap, Donghyuk Lee, Mike O'Connor, Hasan Hassan, and Onur Mutlu. 2017. Understanding Reduced-Voltage Operation in Modern DRAM Devices: Experimental Characterization, Analysis, and Mech-anisms. Proc. ACM Meas. Anal. Comput. Syst. 1, 1 (2017), 10:1–10:42. https: //doi.org/10.1145/3084447

[10] Lucian Cojocar, Kaveh Razavi, Cristiano Giuffrida, and Herbert Bos. 2019. Exploit-ing Correcting Codes: On the Effectiveness of ECC Memory Against Rowhammer Attacks. In IEEE SP. 55–71. https://doi.org/10.1109/SP.2019.00089

[11] Howard David, Eugene Gorbatov, Ulf R. Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: memory power estimation and capping. In ISLPED. 189–194. https://doi.org/10.1145/1840845.1840883

[12] Finn de Ridder, Pietro Frigo, Emanuele Vannacci, Herbert Bos, Cristiano Giuf-frida, and Kaveh Razavi. 2021. SMASH: Synchronized Many-sided Rowhammer Attacks from JavaScript. In USENIX Security. 1001–1018. https://www.usenix.org/conference/usenixsecurity21/presentation/ridder

[13] Spencer Desrochers, Chad Paradis, and Vincent M. Weaver. 2016. A Validation of DRAM RAPL Power Measurements. In MEMSYS. 455–470. https://doi.org/10.1145/2989081.2989088

[14] Steven A. Frank. 2018. Control Theory Tutorial. Springer International Publishing.

[15] Jeffrey Friedman. 1972. Tempest: A signal problem. NSA Cryptologic Spectrum 2, 3 (1972). https://www.nsa.gov/portals/75/documents/news-features/declassified-documents/cryptologic-spectrum/tempest.pdf

[16] Pietro Frigo, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Grand Pwning Unit: Accelerating Microarchitectural Attacks with the GPU. In IEEE SP. 195–210. https://doi.org/10.1109/SP.2018.00022

[17] Pietro Frigo, Emanuele Vannacci, Hasan Hassan, Victor van der Veen, Onur Mutlu, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2020. TRRespass: Exploiting the Many Sides of Target Row Refresh. In IEEE SP. 747–762. https: //doi.org/10.1109/SP46214.2022.9833664

[18] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. 2001. Electromagnetic Analysis: Concrete Results. In CHES. 251–261. https://doi.org/10.1007/3-540-44709-1_21

[19] Daniel Genkin, Noam Nissan, Roei Schuster, and Eran Tromer. 2022. Lend Me Your Ear: Passive Remote Physical Side Channels on PCs. In USENIX Security. 4437–4454. https://outflux.net/slides/2013/lss/kaslr.pdf

[20] Daniel Genkin, Itamar Pipman, and Eran Tromer. 2015. Get your hands off my laptop: physical side-channel key-extraction attacks on PCs - Extended version. J. Cryptogr. Eng. 5, 2 (2015), 95–112. https://doi.org/10.1007/s13389-015-0100-7

[21] Daniel Gruss, Moritz Lipp, Michael Schwarz, Richard Fellner, Clémentine Maurice, and Stefan Mangard. 2017. KASLR is Dead: Long Live KASLR. In ESSoS. 161–176. https://doi.org/10.1007/978-3-319-62105-0_11

[22] Daniel Gruss, Moritz Lipp, Michael Schwarz, Daniel Genkin, Jonas Juffinger, Sioli O'Connell, Wolfgang Schoechl, and Yuval Yarom. 2018. Another Flip in the Wall of Rowhammer Defenses. In IEEE SP. 245–261. https://doi.org/10.1109/SP.2018.00031

[23] Daniel Gruss, Clémentine Maurice, Anders Fogh, Moritz Lipp, and Stefan Man-gard. 2016. Prefetch Side-Channel Attacks: Bypassing SMAP and Kernel ASLR. In CCS. 368–379. https://doi.org/10.1145/2976749.2978356

[24] Daniel Gruss, Clémentine Maurice, and Stefan Mangard. 2016. Rowhammer.js: A Remote Software-Induced Fault Attack in JavaScript. In DIMVA. 300–321. https: //doi.org/10.1007/978-3-319-40667-1_15

[25] Hasan Hassan, Yahya Can Tugrul, Jeremie S. Kim, Victor van der Veen, Kaveh Razavi, and Onur Mutlu. 2021. Uncovering In-DRAM RowHammer Protection Mechanisms: A New Methodology, Custom RowHammer Patterns, and Implica-tions. In MICRO. 1198–1213. https://doi.org/10.1145/3466752.3480110

[26] Yeongjin Jang, Jaehyuk Lee, Sangho Lee, and Taesoo Kim. 2017. SGX-Bomb: Locking Down the Processor via Rowhammer Attack. In SysTEX. 5:1–5:6. https: //doi.org/10.1145/3152701.3152709

[27] Patrick Jattke, Victor van der Veen, Pietro Frigo, Stijn Gunter, and Kaveh Razavi. 2022. BLACKSMITH: Scalable Rowhammering in the Frequency Domain. In IEEE SP. 716–734. https://doi.org/10.1109/SP46214.2022.9833772

[28] Yichen Jiang, Huifeng Zhu, Dean Sullivan, Xiaolong Guo, Xuan Zhang, and Yier Jin. 2021. Quantifying Rowhammer Vulnerability for DRAM Security. In DAC. 73–78. https://doi.org/10.1109/DAC18074.2021.9586119

[29] Jeremie S. Kim, Minesh Patel, Abdullah Giray Yaglikçi, Hasan Hassan, Roknoddin Azizi, Lois Orosa, and Onur Mutlu. 2020. Revisiting RowHammer: An Experi-mental Analysis of Modern DRAM Devices and Mitigation Techniques. In ISCA. 638–651. https://doi.org/10.1109/ISCA45697.2020.00059

[30] Yoongu Kim, Ross Daly, Jeremie S. Kim, Chris Fallin, Ji-Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. 2014. Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors. In ISCA. 361–372. https://doi.org/10.1109/ISCA.2014.6853210

[31] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. 1999. Differential Power Analysis. In CRYPTO. 388–397. https://doi.org/10.1007/3-540-48405-1_25

[32] Andreas Kogler, Jonas Juffinger, Salman Qazi, Yoongu Kim, Moritz Lipp, Nicolas Boichat, Eric Shiu, Mattias Nissler, and Daniel Gruss. 2022. Half-Double: Ham-mering from the Next Row Over. In USENIX Security. 3807–3824. https://www.usenix.org/conference/usenixsecurity22/presentation/kogler-half-double

[33] Robert Kotcher, Yutong Pei, Pranjal Jumde, and Collin Jackson. 2013. Cross-Origin Pixel Stealing: Timing Attacks Using CSS Filters. In CCS. 1055–1062. https://doi.org/10.1145/2508859.2516712

[34] Andrew Kwong, Daniel Genkin, Daniel Gruss, and Yuval Yarom. 2020. RAMBleed: Reading Bits in Memory Without Accessing Them. In IEEE SP. 695–711. https: //doi.org/10.1109/SP40000.2020.00020

[35] Eojin Lee, Ingab Kang, Sukhan Lee, G. Edward Suh, and Jung Ho Ahn. 2019. TWiCe: Preventing Row-hammering by Exploiting Time Window Counters. In ISCA. 385–396. https://doi.org/10.1145/3307650.3322232

[36] Pavel Lifshits, Roni Forte, Yedid Hoshen, Matt Halpern, Manuel Philipose, Mohit Tiwari, and Mark Silberstein. 2018. Power to peep-all: Inference Attacks by Malicious Batteries on Mobile Devices. PoPETs 2018, 4 (2018), 141–158. https: //doi.org/10.1515/popets-2018-0036

[37] Moritz Lipp, Daniel Gruss, and Michael Schwarz. 2022. AMD Prefetch Attacks through Power and Time. In USENIX Security. 643–660. https://www.usenix.org/conference/usenixsecurity22/presentation/lipp

[38] Moritz Lipp, Andreas Kogler, David F. Oswald, Michael Schwarz, Catherine Easdon, Claudio Canella, and Daniel Gruss. 2021. PLATYPUS: Software-based Power Side-Channel Attacks on x86. In IEEE SP. 355–371. https://doi.org/10.1109/SP40001.2021.00063

[39] Moritz Lipp, Michael Schwarz, Lukas Raab, Lukas Lamster, Misiker Tadesse Aga, Clémentine Maurice, and Daniel Gruss. 2020. Nethammer: Inducing Rowhammer Faults through Network Requests. In EuroS&P Workshops. 710–719. https://doi.org/10.1109/EuroSPW51379.2020.00102

[40] Kevin Loughlin, Stefan Saroiu, Alec Wolman, and Baris Kasikci. 2021. Stop! Hammer Time: Rethinking our Approach to Rowhammer Mitigations. In HotOS. 88–95. https://doi.org/10.1145/3458336.3465295

[41] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. 2007. Power Analysis Attacks: Revealing the Secrets of Smart Cards. Springer.

[42] Hector Marco-Gisbert and Ismael Ripoll Ripoll. 2019. Address space layout randomization next generation. Applied Sciences 9, 14 (2019), 2928. https://doi.org/10.3390/app9142928

[43] Colin O'Flynn and Alex Dewar. 2019. On-Device Power Analysis Across Hard-ware Security Domains. Stop Hitting Yourself. IACR Trans. Cryptogr. Hardw. Em-bed. Syst. 2019, 4 (2019), 126–153. https://doi.org/10.13154/tches.v2019.i4.126-153

[44] Lukasz Olejnik, Gunes Acar, Claude Castelluccia, and Claudia Díaz. 2015. The Leaking Battery - A Privacy Analysis of the HTML5 Battery Status API. In DPM/QASA at ESORICS. 254–263. https://doi.org/10.1007/978-3-319-29883-2_18

[45] Yossef Oren and Adi Shamir. 2007. Remote Password Extraction from RFID Tags. IEEE Trans. Computers 56, 9 (2007), 1292–1296. https://doi.org/10.1109/TC.2007. 1050

[46] Peter Pessl, Daniel Gruss, Clémentine Maurice, Michael Schwarz, and Stefan Mangard. 2016. DRAMA: Exploiting DRAM Addressing for Cross-CPU Attacks. In USENIX Security. 565–581. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/pessl

[47] Kaveh Razavi, Ben Gras, Erik Bosman, Bart Preneel, Cristiano Giuffrida, and Herbert Bos. 2016. Flip Feng Shui: Hammering a Needle in the Software Stack. In USENIX Security. 1–18. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/razavi

[48] Falk Schellenberg, Dennis R. E. Gnad, Amir Moradi, and Mehdi Baradaran Tahoori. 2018. An inside job: Remote power analysis attacks on FPGAs. In DATE. 1111– 1116. https://doi.org/10.23919/DATE.2018.8342177

[49] Martin Schwarzl, Thomas Schuster, Michael Schwarz, and Daniel Gruss. 2021. Speculative dereferencing of registers: Reviving Foreshadow. In FC. 311–330. https://doi.org/10.1007/978-3-662-64322-8_15

[50] Mark Seaborn and Thomas Dullien. 2015. Exploiting the DRAM rowhammer bug to gain kernel privileges. https://googleprojectzero.blogspot.com/2015/03/exploiting-dram-rowhammer-bug-to-gain.html. (2015).

[51] Mungyu Son, Hyunsun Park, Junwhan Ahn, and Sungjoo Yoo. 2017. Making DRAM Stronger Against Row Hammering. In DAC. 55:1–55:6. https://doi.org 10. 1145/3061639.3062281

[52] Andrei Tatar, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Defeating Software Mitigations Against Rowhammer: A Surgical Precision Hammer. In RAID. 47–66. https://doi.org/10.1007/978-3-030-00470-5_3

[53] Andrei Tatar, Radhesh Krishnan Konoth, Elias Athanasopoulos, Cristiano Giuffrida, Herbert Bos, and Kaveh Razavi. 2018. Throwhammer: Rowhammer Attacks over the Network and Defenses. In USENIX ATC. 213–226. https: //www.usenix.org/conference/atc18/presentation/tatar

[54] Youssef Tobah, Andrew Kwong, Ingab Kang, Daniel Genkin, and Kang G. Shin. 2022. SpecHammer: Combining Spectre and Rowhammer for New Speculative Attacks. In IEEE SP. 681–698. https://doi.org/10.1109/SP46214.2022.9833802

[55] Jo Van Bulck, Frank Piessens, and Raoul Strackx. 2018. Nemesis: Studying Microarchitectural Timing Leaks in Rudimentary CPU Interrupt Logic. In CCS. 178–195. https://doi.org/10.1145/3243734.3243822

[56] Victor Van Der Veen, Yanick Fratantonio, Martina Lindorfer, Daniel Gruss, Clémentine Maurice, Giovanni Vigna, Herbert Bos, Kaveh Razavi, and Cristiano Giuffrida. 2016. Drammer: Deterministic Rowhammer Attacks on Mobile Plat-forms. In CCS. 1675–1689. https://doi.org/10.1145/2976749.2978406

[57] Victor Van der Veen, Martina Lindorfer, Yanick Fratantonio, Harikrishnan Padmanabha Pillai, Giovanni Vigna, Christopher Kruegel, Herbert Bos, and Kaveh Razavi. 2018. GuardION: Practical Mitigation of DMA-Based Rowhammer Attacks on ARM. In DIMVA. 92–113. https://doi.org/10.1007/978-3-319-93411-2_5

[58] VLSI System Design Corporation. 2017. Effect of Coupling Capacitance. (2017). https://www.vlsisystemdesign.com/effect-of-coupling-capacitance/

[59] Andrew J. Walker, Sungkwon Lee, and Dafna Beery. 2021. On DRAM Rowhammer and the Physics of Insecurity. IEEE Transactions on Electron Devices 68, 4 (2021), 1400–1410. https://doi.org/10.1109/TED.2021.3060362

[60] Zane Weissman, Thore Tiemann, Daniel Moghimi, Evan Custodio, Thomas Eisenbarth, and Berk Sunar. 2020. JackHammer: Efficient Rowhammer on Heterogeneous FPGA-CPU Platforms. IACR Trans. Cryptogr. Hardw. Embed. Syst. 2020, 3 (2020), 169–195. https://doi.org/10.13154/tches.v2020.i3.169-195

[61] Yuan Xiao, Xiaokuan Zhang, Yinqian Zhang, and Radu Teodorescu. 2016. One Bit Flips, One Cloud Flops: Cross-VM Row Hammer Attacks and Privilege Escalation. In USENIX Security. 19–35. https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/xiao

[62] Abdullah Giray Yaglikçi, Haocong Luo, Geraldo F. de Oliviera, Ataberk Olgun, Minesh Patel, Jisung Park, Hasan Hassan, Jeremie S. Kim, Lois Orosa, and Onur Mutlu. 2022. Understanding RowHammer Under Reduced Wordline Voltage: An Experimental Study Using Real DRAM Devices. In DSN. 475–487. https: //doi.org/10.1109/DSN53405.2022.00054

[63] Qing Yang, Paolo Gasti, Gang Zhou, Aydin Farajidavar, and Kiran S. Balagani. 2017. On Inferring Browsing Activity on Smartphones via USB Power Analysis Side-Channel. IEEE Trans. Inf. Forensics Secur. 12, 5 (2017), 1056–1066. https: //doi.org/10.1109/TIFS.2016.2639446

[64] Jung Min You and Joon-Sung Yang. 2019. MRLoc: Mitigating Row-hammering Based on Memory Locality. In DAC. 19. https://doi.org/10.1145/3316781.3317866

[65] Zhi Zhang, Yueqiang Cheng, Dongxi Liu, Surya Nepal, Zhi Wang, and Yuval Yarom. 2020. PThammer: Cross-User-Kernel-Boundary Rowhammer through Implicit Accesses. In MICRO. 28–41. https://doi.org/10.1109/MICRO50266 2020. 00016

[66] Kenneth M. Zick, Meeta Srivastav, Wei Zhang, and Matthew French. 2013. Sensing Nanosecond-Scale Voltage Attacks and Natural Transients in FPGAs. In FPGA. 101–104. https://doi.org/10.1145/2435264.2435283